TOÁN RỜI RẠC 2

Phạm Tiến Sơn

Đà Lạt - 2009

Mục lục

Lời nói đầu

Trong thực tế để miêu tả một số tình huống người ta thường biểu thị bằng một hình ảnh gồm các điểm (các đỉnh) - biểu diễn các thực thể - và vẽ các đoạn thẳng nối cặp các đỉnh biểu diễn mối quan hệ giữa chúng. Những hình như thế thường gọi là các *đồ thị*. Mục đích của giáo trình này cung cấp những kiến thức cơ bản để nghiên cứu các đồ thị. Các đồ thị xuất hiện trong nhiều lĩnh vực với các tên gọi khác nhau: "cấu trúc" trong công trình xây dựng, "mạch" trong điện tử, "lược đồ quan hệ", "cấu trúc truyền thông", "cấu trúc tổ chức" trong xã hội và kinh tế, "cấu trúc phân tử" trong hoá học, vân vân.

Do những ứng dụng rộng rãi của nó trong nhiều lĩnh vực, có rất nhiều nghiên cứu xung quanh lý thuyết đồ thị trong những năm gần đây; một nhân tố chủ yếu góp phần thúc đẩy sự phát triển đó là xuất hiện các máy tính lớn có thể thực hiện nhiều phép toán với tốc độ rất nhanh. Việc biểu diễn trực tiếp và chi tiết các hệ thống thực tế, chẳng hạn các mạng truyền thông, đã đưa đến những đồ thị có kích thước lớn và việc phân tích thành công hệ thống phụ thuộc rất nhiều vào các thuật toán "tốt" cũng như khả năng của máy tính. Theo đó, giáo trình này sẽ tập trung vào việc phát triển và trình bày các thuật toán để phân tích các đồ thị.

Các phương pháp phân tích và thiết kế các thuật toán trong giáo trình cho phép sinh viên có thể viết dễ dàng các chương trình minh họa. Giáo trình được biên soạn cho các đối tượng là sinh viên Toán-Tin và Tin học.

Giáo trình sử dụng ngôn ngữ C để minh họa, tuy nhiên có thể dễ dàng chuyển đổi sang các ngôn ngữ khác; và do đó, sinh viên cần có một số kiến thức về ngôn ngữ C. Ngoài ra, hầu hết các chương trình thao tác trên cấu trúc dữ liệu như danh sách liên

kết, nên đòi hỏi sinh viên phải có những kỹ năng lập trình tốt.

Giáo trình bao gồm bảy chương và một phần phụ lục với những nội dung chính như sau:

- Chương thứ nhất trình bày những khái niệm căn bản về đồ thị.
- Chương 2 trình bày những số cơ bản của đồ thị. Ý nghĩa thực tiễn của các số này.
- Chương 3 tìm hiểu bài toán tìm đường đi ngắn nhất.
- Chương 4 đề cập đến khái niệm về cây. Ứng dụng của cây Huffman trong nén dữ liệu. Ngoài ra xây dựng các thuật toán tìm cây bao trùm nhỏ nhất.
- Bài toán Euler và bài toán Hamilton và những mở rộng của chúng sẽ được nói đến trong Chương 5.
- Chương 6 nghiên cứu các tính chất phẳng của đồ thị; và cuối cùng
- Chương 7 tìm hiểu các bài toán trên mạng vận tải.

Ngoài ra, phần phụ lục trình bày các cấu trúc dữ liệu và những thủ tục cần thiết để đơn giản hoá các đoạn chương trình minh họa các thuật toán được trình bày.

Giáo trình được biên soạn lần đầu tiên nên không tránh khỏi khá nhiều thiếu sót. Tác giả mong có những đóng góp từ bạn đọc.

Tôi xin cảm ơn những giúp đỡ đã nhận được từ nhiều người mà không thể liệt kê hết, đặc biệt là các bạn sinh viên, trong quá trình biên soạn giáo trình này.

Đà Lạt, ngày 17 tháng 4 năm 2009

PHẠM Tiến Sơn

Chương 1

Đại cương về đồ thị

1.1 Định nghĩa và các khái niệm

1.1.1 Đồ thị có hướng

Đồ thị có hướng $G = (V, E)$ gồm một tập V các phần tử gọi là *đỉnh* (hay *nút*) và một tập E các *cung* sao cho mỗi cung $e \in E$ tương ứng với một cặp các đỉnh được sắp thứ tự. Nếu có đúng một cung e tương ứng các đỉnh được sắp thứ tự (a, b), ta sẽ viết $e := (a, b)$.

Chúng ta sẽ giả sử các đỉnh được đánh số là $v_1, v_2, \ldots, v_n$ hay giản tiện, $1, 2, \ldots, n$, trong đó $n = \#V$ là số các đỉnh của đồ thị.

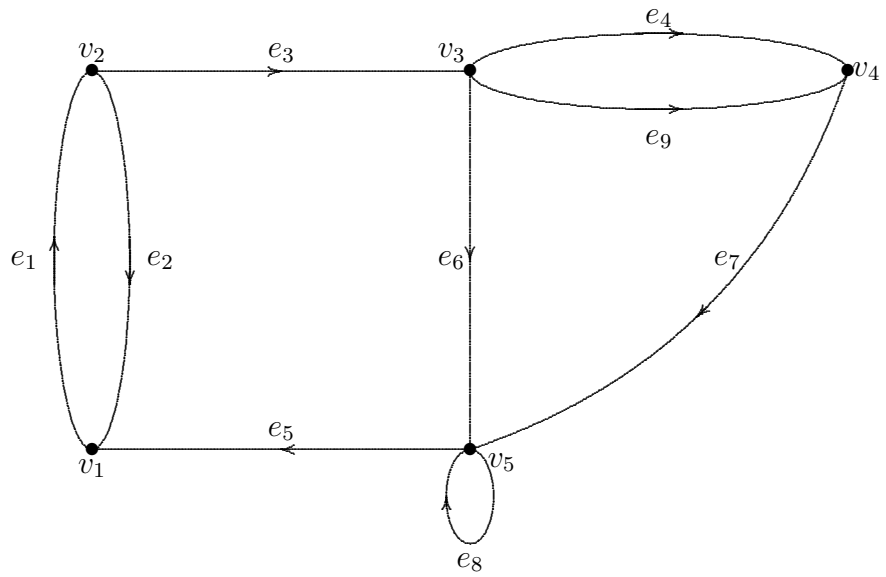
Nếu e là một cung tương ứng cặp các đỉnh được sắp thứ tự v_i và v_j thì đỉnh v_i gọi là *gốc* và đỉnh v_j gọi là *ngọn;* cung e gọi là *liên thuộc* hai đỉnh v_i và v_j. Chúng ta sẽ thường ký hiệu $m = \#E$ là số cung của đồ thị G. Các cung thường được đánh số là $e_1, e_2, \ldots, e_m$.

Một cách hình học, các đỉnh được biểu diễn bởi các điểm, và cung e được biểu diễn bởi một mũi tên từ điểm v_i đến điểm v_j.

Một cung có gốc trùng với ngọn gọi là *khuyên.*

Nếu có nhiều hơn một cung với gốc tại v_i và ngọn tại v_j thì G gọi là *đa đồ thị* và các cung tương ứng gọi là *song song*. *Đơn đồ thị* có hướng là đồ thị không khuyên trong đó hai đỉnh bất kỳ v_i và v_j có nhiều nhất một cung nối chúng.

Ví dụ 1.1.1. Đồ thị trong Hình 1.1 có cung e_8 là khuyên; các cung e_4 và e_9 là song song do cùng tương ứng cặp đỉnh v_3 và v_4. Cung e_5 liên thuộc các đỉnh v_1 và v_5.



Hình 1.1: Ví dụ của đồ thị có hướng.

1.1.2 Đồ thị và ánh xạ đa trị

Với mỗi $x \in V$, ký hiệu $\Gamma(x)$ là tập các đỉnh $y \in V$ sao cho tồn tại cung với gốc là đỉnh x và ngọn là đỉnh y. Khi đó ta có một ánh xạ *đa trị* $\Gamma\colon V \to 2^V, x \mapsto \Gamma(x)$. Ký hiệu Γ^{-1} là ánh xạ (đa trị) ngược của Γ.

Nếu G là đơn đồ thị, thì đồ thị này hoàn toàn được xác định bởi tập V và ánh xạ đa trị Γ từ V vào 2^V; trong trường hợp này ta có thể ký hiệu $G = (V, \Gamma)$.

Ví dụ 1.1.2. Nếu xóa cung e_9 trong Hình 1.1 ta nhận được đơn đồ thị có hướng và do đó có thể biểu diễn bởi ánh xạ đa trị Γ :

$$\Gamma(v_1) = \{v_2\}, \quad \Gamma(v_2) = \{v_1, v_3\}, \quad \Gamma(v_3) = \{v_4, v_5\}, \quad \Gamma(v_4) = \{v_5\}, \quad \Gamma(v_5) = \{v_1, v_5\}.$$

1.1.3 Đồ thị vô hướng

Khi nghiên cứu một số tính chất của các đồ thị, ta thấy rằng chúng không phụ thuộc vào hướng của các cung, tức là không cần phân biệt sự khác nhau giữa các điểm bắt đầu và kết thúc. Điều này đơn giản là mỗi khi có ít nhất một cung giữa hai đỉnh ta không quan tâm đến thứ tự của chúng.

Với mỗi cung, tức là mỗi cặp *có thứ tự* (v_i, v_j) ta cho tương ứng cặp *không có thứ tự* (v_i, v_j) gọi là các *cạnh*. Tương đương, ta nói rằng cạnh là một cung mà hướng đã bị *bỏ quên*. Về hình học, cạnh (v_i, v_j) được biểu diễn bởi các đoạn thẳng (hoặc cong) và không có mũi tên liên thuộc hai điểm tương ứng hai đỉnh v_i và v_j.

Nghiên cứu các tính chất *vô hướng* của đồ thị $G = (V, E)$ đưa về khảo sát tập E là tập các *cạnh*, tức là, một tập hữu hạn các phần tử mà mỗi phần tử là một cặp hai đỉnh phân biệt hay đồng nhất của V.

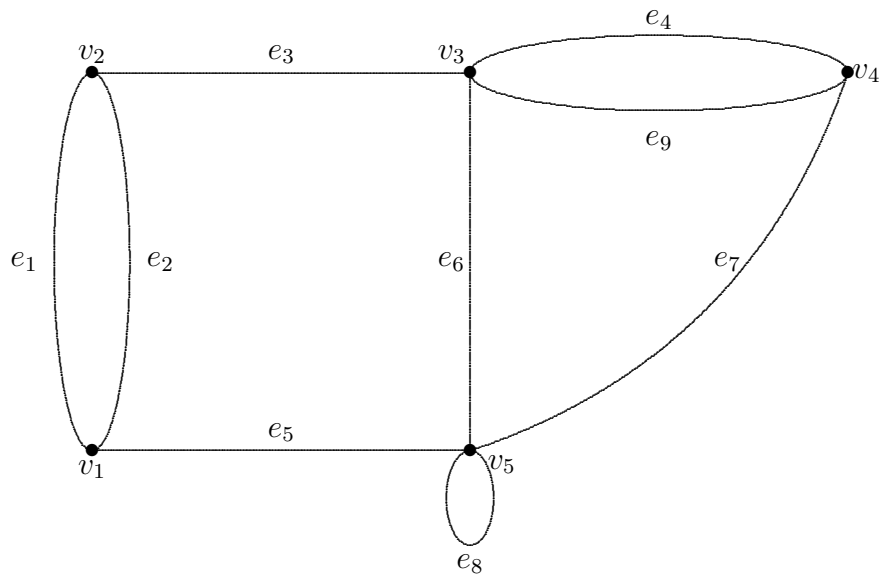
Đa đồ thị vô hướng là đồ thị mà có thể có nhiều hơn một cạnh liên thuộc hai đỉnh.

Đồ thị gọi là *đơn* nếu nó không có khuyên và hai đỉnh bất kỳ có nhiều nhất một cạnh liên thuộc chúng.

Ví dụ 1.1.3. Hình 1.2 là đồ thị vô hướng có 5 đỉnh và 9 cạnh. Cạnh e_1 song song với cạnh e_2. Cạnh e_8 là khuyên.

1.1.4 Các định nghĩa

Hai cung, hoặc hai cạnh gọi là *kề nhau* nếu chúng có ít nhất một đỉnh chung. Chẳng hạn, hai cạnh e_1 và e_3 trong Hình 1.2 là kề nhau. Hai đỉnh v_i và v_j gọi là *kề nhau* nếu tồn tại cạnh hoặc cung e_k liên thuộc chúng. Ví dụ trong Hình 1.2 hai đỉnh v_2 và v_3 là kề nhau (liên thuộc bởi cạnh e_3), nhưng đỉnh v_2 và v_5 không kề nhau.



Hình 1.2: Đồ thị vô hướng tương ứng đồ thị trong Hình 1.1.

Bậc

Bậc ngoài của đỉnh $v \in V$, ký hiệu $d_G^+(v)$ (hay $d^+(v)$ nếu không sợ nhầm lẫn) là số các cung có đỉnh v là gốc. *Bậc trong* của đỉnh $v \in V$, ký hiệu $d_G^-(v)$ (hay $d^-(v)$ nếu không sợ nhầm lẫn) là số các cung có đỉnh v là ngọn.

Chẳng hạn, đồ thị có hướng trong Hình 1.1 có $d^+(v_2) = 2, d^-(v_2) = 1$.

Hiển nhiên rằng, tổng các bậc ngoài của các đỉnh bằng tổng các bậc trong của các đỉnh và bằng tổng số cung của đồ thị G, tức là

$$\sum_{i=1}^{n} d^+(v_i) = \sum_{i=1}^{n} d^-(v_i) = m.$$

Nếu G là đồ thị vô hướng, *bậc* của đỉnh $v \in V$, ký hiệu $d_G(v)$ (hay $d(v)$ nếu không sợ nhầm lẫn) là số các cạnh liên thuộc đỉnh v với khuyên được đếm hai lần. Ví dụ đồ thị vô hướng trong Hình 1.2 có $d(v_2) = 3, d(v_5) = 5$.

Các cung (cạnh) liên thuộc tập $A \subset V$. Đối chu trình

Giả sử $A \subset V$. Ký hiệu $\omega^+(A)$ là tập tất cả các cung có đỉnh gốc thuộc A và đỉnh ngọn thuộc $A^c := V \setminus A$, và $\omega^-(A)$ là tập tất cả các cung có đỉnh ngọn thuộc A và đỉnh gốc thuộc A^c. Đặt

$$\omega(A) = \omega^+(A) \cup \omega^-(A).$$

Tập các cung hoặc cạnh có dạng $\omega(A)$ gọi là *đối chu trình* của đồ thị.

Đồ thị có trọng số

Đồ thị có trọng số nếu trên mỗi cung (hoặc cạnh) $e \in E$ có tương ứng một số thực $w(e)$ gọi là trọng lượng của cung e.

Đồ thị đối xứng

Đồ thị có hướng gọi là *đối xứng* nếu có bao nhiêu cung dạng (v_i, v_j) thì cũng có bấy nhiêu cung dạng (v_j, v_i).

Đồ thị phản đối xứng

Đồ thị có hướng gọi là *phản đối xứng* nếu có cung dạng (v_i, v_j) thì không có cung dạng (v_j, v_i).

Đồ thị đầy đủ

Đồ thị vô hướng gọi là *đầy đủ* nếu hai đỉnh bất kỳ v_i và v_j tồn tại một cạnh dạng (v_i, v_j). Đơn đồ thị vô hướng đầy đủ n đỉnh được ký hiệu là K_n.

Đồ thị con

Giả sử $A \subset V$. *Đồ thị con* được sinh bởi tập A là đồ thị $G_A := (A, E_A)$ trong đó các đỉnh là các phần tử của tập A và các cung trong E_A là các cung của G mà hai đỉnh nó liên thuộc thuộc tập A.

Nếu G là đồ thị biểu diễn bản đồ giao thông của nước Việt Nam thì đồ thị biểu diễn bản đồ giao thông của thành phố Đà Lạt là một đồ thị con.

Đồ thị bộ phận

Xét đồ thị $G = (V, E)$ và $U \subset E$. *Đồ thị bộ phận sinh bởi* tập U là đồ thị với tập đỉnh V và các cung thuộc U (các cung của $E \setminus U$ bị xóa khỏi G).

Đồ thị con bộ phận

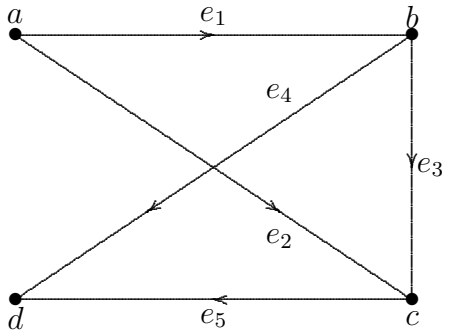
Xét đồ thị $G = (V, E)$ và $A \subset V, U \subset E$. *Đồ thị con bộ phận* sinh bởi tập A và U là đồ thị bộ phận của G_A sinh bởi U.

1.2 Ma trận biểu diễn đồ thị

1.2.1 Ma trận liên thuộc đỉnh-cung

Ma trận liên thuộc đỉnh-cung của đồ thị $G = (V, E)$ là ma trận $A = (a_{ij}), i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$, với các phần tử $0, 1$ và -1, trong đó mỗi cột biểu diễn một cung và mỗi hàng biểu diễn một đỉnh. Nếu $e_k = (v_i, v_j) \in E$ thì tất cả các phần tử của cột k bằng không ngoại trừ

$$a_{ik} = 1, \quad a_{jk} = -1.$$



Hình 1.3:

Ví dụ 1.2.1. Ma trận liên thuộc đỉnh-cung của đồ thị trong Hình 1.3 là

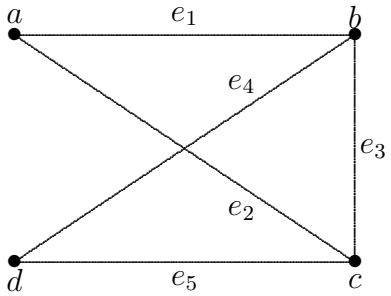
$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \end{array}.$$

Nhắc lại rằng, ma trận vuông gọi là *unimodular* nếu định thức của nó bằng 1 hoặc -1. Ma trận A cấp $m \times n$ gọi là *unimodular hoàn toàn* nếu tất cả các ma trận vuông con không suy biến của A là unimodular.

Mệnh đề 1.2.2. *Ma trận liên thuộc đỉnh-cung của đồ thị $G = (V, E)$ là unimodular hoàn toàn.*

Chứng minh. Chú ý rằng ma trận liên thuộc đỉnh-cung của đồ thị $G = (V, E)$ chứa đúng hai phần tử khác không trên mỗi cột, một bằng 1 và một bằng -1. Do đó ta có thể chứng minh theo quy nạp như sau: Hiển nhiên, tất cả các ma trận vuông con không suy biến cấp 1 của A là modular; giả sử khẳng định đúng cho mọi ma trận con không suy biến cấp $(k-1)$.

Xét ma trận vuông con A' cấp k của A. Nếu mỗi cột của A' chứa đúng hai phần tử khác không thì $\det(A') = 0$ (thật vậy, tổng tất cả các hàng của A' là vector không,



Hình 1.4:

do đó các hàng là độc lập tuyến tính). Nếu tồn tại một cột của A' không có phần tử khác không thì $\det(A') = 0$. Cuối cùng, nếu tồn tại cột j của A' sao cho có đúng một phần tử khác không a_{ij} (bằng 1, hay -1) thì $\det(A') = \pm\det(A'')$, trong đó A'' là ma trận vuông cấp $(k-1)$ nhận được từ A' bằng cách xóa hàng i và cột j. Theo giả thiết quy nạp, $\det(A')$ bằng 1, -1 hay 0 và do đó mệnh đề được chứng minh. ◁

1.2.2 Ma trận liên thuộc đỉnh-cạnh

Xét đồ thị vô hướng $G = (V, E)$. *Ma trận liên thuộc đỉnh-cạnh* của đồ thị G là ma trận $A = (a_{ij}), i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$, với các phần tử 0 và 1, trong đó mỗi cột biểu diễn một cạnh và mỗi hàng biểu diễn một đỉnh; ngoài ra, nếu cạnh e_k liên thuộc hai đỉnh v_i và v_j thì tất cả các phần tử của cột k bằng không ngoại trừ

$$a_{ik} = 1, \quad a_{jk} = 1.$$

Ví dụ 1.2.3. Ma trận liên thuộc đỉnh-cạnh của đồ thị trong Hình 1.4 là

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Trái với ma trận liên thuộc đỉnh-cung, nói chung ma trận liên thuộc đỉnh-cạnh

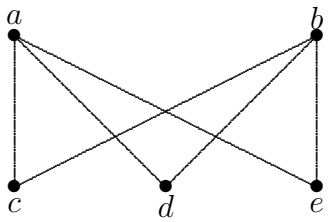
không unimodular hoàn toàn. Chẳng hạn, trong ví dụ trên, ma trận con

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

có định thức bằng -2.

Đồ thị vô hướng $G = (V, E)$ gọi là *hai phần* nếu có thể phân hoạch tập các đỉnh V thành hai tập con rời nhau V_1 và V_2 sao cho đối với mỗi cạnh $(v_i, v_j) \in E$ thì hoặc $v_i \in V_1, v_j \in V_2$ hoặc $v_j \in V_1, v_i \in V_2$.

Ví dụ 1.2.4. Dễ kiểm tra đồ thị $K_{2,3}$ trong Hình 1.5 là hai phần.



Hình 1.5: Đồ thị hai phần $K_{2,3}$.

Mệnh đề 1.2.5. *Ma trận liên thuộc đỉnh-cạnh của đồ thị vô hướng $G = (V, E)$ là unimodular hoàn toàn nếu và chỉ nếu G là đồ thị hai phần.*

Chứng minh. Điều kiện đủ. Giả sử G là đồ thị hai phần. Ta sẽ chứng minh theo quy nạp rằng mọi ma trận vuông con B của ma trận liên thuộc đỉnh-cạnh có định thức $\det(B) = 0, 1$ hoặc -1. Điều này đúng với các ma trận vuông con cấp 1; giả sử khẳng định đúng với các ma trận vuông con cấp $(k-1)$. Xét ma trận vuông con B cấp k.

Nếu mỗi cột B^j của B chứa đúng hai phần tử bằng 1 thì

$$\sum_{i \in I_1} B_i = \sum_{i \in I_2} B_i,$$

trong đó I_1 và I_2 là các tập chỉ số tương ứng hai phân hoạch của tập các đỉnh V và B_i là vector hàng của B. Các vector hàng phụ thuộc tuyến tính, nên $\det(B) = 0$.

Ngược lại, nếu tồn tại cột có đúng một phần tử bằng 1, chẳng hạn $b_{ij} = 1$, thì bằng quy nạp ta có

$$\det(B) = \pm \det(C) \quad (= 0, 1 \ \text{ hoặc } \ -1,$$

trong đó C là ma trận nhận được từ B bằng cách xóa hàng i và cột j.

Điều kiện cần. Dễ dàng chứng minh rằng ma trận liên thuộc đỉnh-cạnh của đồ thị là một chu trình độ dài lẻ (tức là số cạnh trên chu trình là lẻ-xem Phần 1.3) có định thức bằng ± 2. Do đó G không chứa chu trình độ dài lẻ và vì vậy nó là hai phần theo bổ đề sau. $\triangleleft$

Bổ đề 1.2.6. *Đồ thị vô hướng G là hai phần nếu và chỉ nếu G không chứa chu trình có độ dài lẻ.*

Chứng minh. Điều kiện cần. Do V được phân hoạch thành V_1 và V_2 :

$$V = V_2 \cup V_2, \quad V_1 \cap V_2 = \emptyset.$$

Giả thiết tồn tại một chu trình có độ dài lẻ

$$\mu = \{v_{i_1}, v_{i_2}, \ldots, v_{i_q}, v_{i_1}\}$$

và không mất tính tổng quát, lấy $v_{i_1} \in V_1$. Do G là hai phần, nên hai đỉnh liên tiếp trên chu trình μ phải có một đỉnh thuộc V_1 và đỉnh kia thuộc V_2. Do đó $v_{i_2} \in V_2, v_{i_3} \in V_1, \ldots,$ và tổng quát, $v_{i_k} \in V_1$ nếu k lẻ và $v_{i_k} \in V_2$ nếu k chẵn. Mà chu trình μ có độ dài lẻ nên $v_{i_q} \in V_1$ và bởi vậy $v_{i_1} \in V_2$. Điều này mâu thuẫn với $V_1 \cap V_2 = \emptyset$.

Điều kiện đủ. Không mất tính tổng quát giả thiết đồ thị G liên thông. Giả sử không tồn tại chu trình có độ dài lẻ.

Chọn đỉnh bất kỳ, chẳng hạn v_i và gán nhãn cho nó là " + ". Sau đó lặp lại các phép toán sau:

Chọn đỉnh đã được gán nhãn v_j và gán nhãn ngược với nhãn của v_j cho tất cả các đỉnh kề với đỉnh v_j.

Tiếp tục quá trình này cho đến khi xảy ra một trong hai trường hợp:

(a) Tất cả các đỉnh đã được gán nhãn và hai đỉnh bất kỳ kề nhau có nhãn khác nhau (một mang dấu + và một mang dấu −); hoặc

(b) Tồn tại đỉnh, chẳng hạn v_{j_k}, được gán hai nhãn khác nhau.

Trong Trường hợp (a), đặt V_1 là tập tất cả các đỉnh được gán nhãn "+" và V_2 là tập tất cả các đỉnh được gán nhãn "−". Do tất cả các cạnh liên thuộc giữa các cặp đỉnh có nhãn khác nhau nên đồ thị G là hai phần.

Trong Trường hợp (b), đỉnh v_{j_k} được gán nhãn "+" dọc theo một dây chuyền μ_1 nào đó, với các đỉnh được gán nhãn "+" và "−" xen kẽ nhau xuất phát từ v_i và kết thúc tại v_{j_k}. Tương tự, đỉnh v_{j_k} được gán nhãn "−" dọc theo một dây chuyền μ_2 nào đó, với các đỉnh được gán nhãn "+" và "−" xen kẽ nhau xuất phát từ v_i và kết thúc tại v_{j_k}. Nhưng như thế chu trình đi dọc theo μ_1 từ đỉnh v_i đến đỉnh v_{j_k} sau đó đi ngược lại dọc theo μ_2 về lại v_i có độ dài lẻ. Điều này mâu thuẫn với giả thiết, và do đó không thể xảy ra Trường hợp (b). Định lý được chứng minh. ◁

1.2.3 Ma trận kề

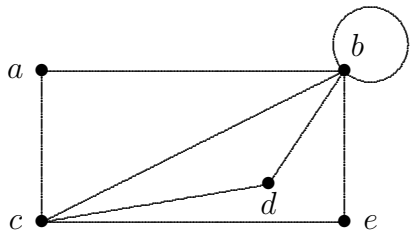
Giả sử $G = (V, E)$ là đồ thị sao cho có nhiều nhất một cung liên thuộc hai đỉnh bất kỳ v_i và v_j. *Ma trận kề* hay *ma trận liên thuộc đỉnh-đỉnh* là ma trận vuông $A = (a_{ij})$ cấp $n \times n$ với các phần tử 0 hoặc 1:

$$a_{ij} := \begin{cases} 1 & \text{nếu } (v_i, v_j) \in E, \\ 0 & \text{nếu ngược lại.} \end{cases}$$

Ví dụ 1.2.7. Đồ thị G trong Hình 1.6 có ma trận kề là

$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

Từ định nghĩa, dễ dàng suy ra



Hình 1.6:

Tính chất 1.2.8. *Giả sử $A = (a_{ij})_{n\times n}$ là ma trận kề của đồ thị vô hướng G. Khi đó*

(a) $a_{ii} = 1$ nếu và chỉ nếu G có khuyên tại đỉnh i.

(b) A là ma trận đối xứng.

(c) Nếu G không khuyên (và không có cạnh song song) thì bậc của một đỉnh bằng số các số 1 trong hàng hay cột tương ứng.

(d) Hoán vị các hàng và các cột tương ứng tương đương với việc đánh số lại các đỉnh. Tuy nhiên cần chú ý rằng, các hàng và các cột cần được sắp xếp theo cùng một thứ tự. Do đó nếu hai hàng thay đổi thì hai cột tương ứng cũng cần phải thay đổi.

Trong trường hợp đồ thị vô hướng, ma trận kề của đơn đồ thị cũng có thể được định nghĩa bằng cách xem mỗi cạnh (v_i, v_j) tương ứng hai cung (v_i, v_j) và (v_j, v_i). Trong trường hợp này, ma trận kề là đối xứng.

1.2.4 Các cấu trúc dữ liệu biểu diễn đồ thị

Để mô tả một đồ thị, ta có thể sử dụng một số cách biểu diễn khác nhau. Với quan điểm lập trình, nói chung các biểu diễn này không tương đương theo khía cạnh hiệu quả của thuật toán.

Có hai cách biểu diễn chính: Thứ nhất, sử dụng ma trận kề hoặc các dẫn xuất của nó; thứ hai, sử dụng ma trận liên thuộc hoặc các dẫn xuất của nó.

Sử dụng ma trận kề

Chúng ta biết rằng các ma trận kề cho phép miêu tả các đồ thị không có cạnh (cung) song song. Với cách biểu diễn đồ thị qua ma trận kề, ta thấy số lượng thông tin, gồm các bit 0 và 1, cần lưu trữ là n^2. Các bit có thể được kết hợp trong các từ. Ký hiệu w là độ dài của từ (tức là số các bit trong một từ máy tính). Khi đó mỗi hàng của ma trận kề có thể được viết như một dãy n bit trong $\lceil n/w \rceil$ từ [1]. Do đó số các từ để lưu trữ ma trận kề là $n\lceil n/w \rceil$.

Ma trận kề của đồ thị vô hướng là đối xứng, nên ta chỉ cần lưu trữ nửa tam giác trên của nó, và do đó chỉ cần $n(n-1)/2$ bit. Tuy nhiên, với cách lưu trữ này, sẽ tăng độ phức tạp và thời gian tính toán trong một số bài toán.

Trong trường hợp các ma trận thưa ($m \ll n^2$ với đồ thị có hướng; $m \ll \frac{1}{2}n(n+1)$ đối với đồ thị vô hướng) cách biểu diễn này là lãng phí. Do đó ta sẽ tìm cách biểu diễn chỉ các phần tử khác không.

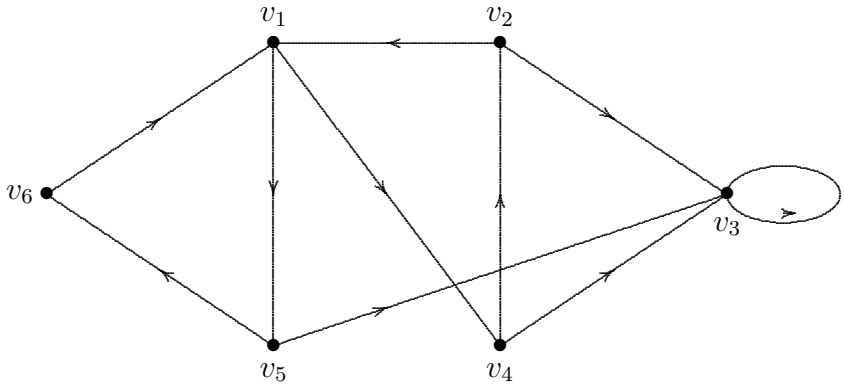
Vì mục đích này ta sẽ sử dụng một *mảng danh sách kề* cho đồ thị có hướng. Đồ thị có hướng được biểu diễn bởi một mảng các con trỏ $V\_out[1], V\_out[2], \ldots, V\_out[n]$, trong đó mỗi con trỏ tương ứng với một đỉnh trong đồ thị có hướng. Mỗi phần tử của mảng $V\_out[i]$ chỉ đến một nút đầu lưu trữ mục dữ liệu của nút tương ứng đỉnh v_i và chứa một con trỏ chỉ đến một danh sách liên kết của các đỉnh kề (đỉnh được nối với v_i theo hướng từ v_i ra). Mỗi nút kề có hai trường:

1. *Trường số nguyên*: lưu trữ số hiệu của đỉnh kề; và
2. *Trường liên kết* chỉ đến nút kế tiếp trong danh sách kề này.

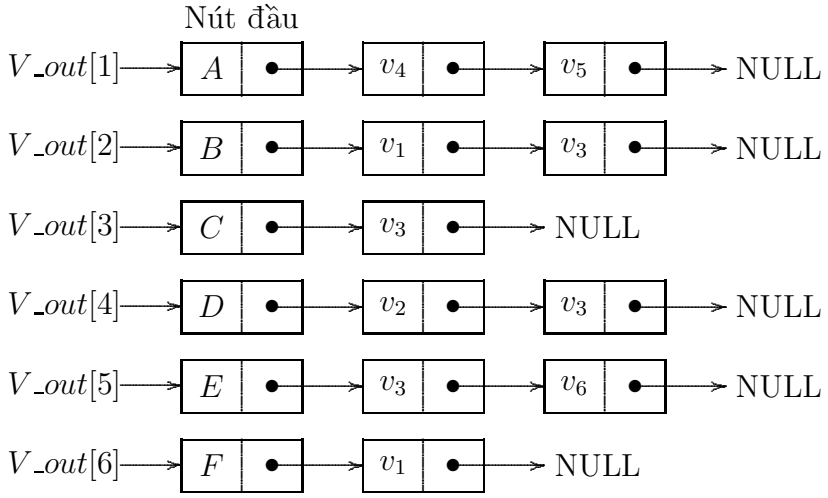
Cách biểu diễn mảng danh sách kề $V\_out[]$ của đồ thị có hướng trong Hình 1.7 được cho tương ứng trong Hình 1.8 (giả sử các mục dữ liệu tương ứng các đỉnh theo thứ tự là A, B, C, D, E, F).

Thay vì con trỏ chỉ đến một danh sách các đỉnh từ v_i *đi ra* trong $V\_out[i]$, ta trỏ đến danh sách các đỉnh *đi đến* v_i và do đó có thể lưu trữ đồ thị thông qua mảng các

[1] Ký hiệu $\lceil x \rceil$ là số nguyên nhỏ nhất không bé hơn x.



Hình 1.7:



Hình 1.8: Danh sách kề $V\_out[]$ tương ứng đồ thị trong Hình 1.7.

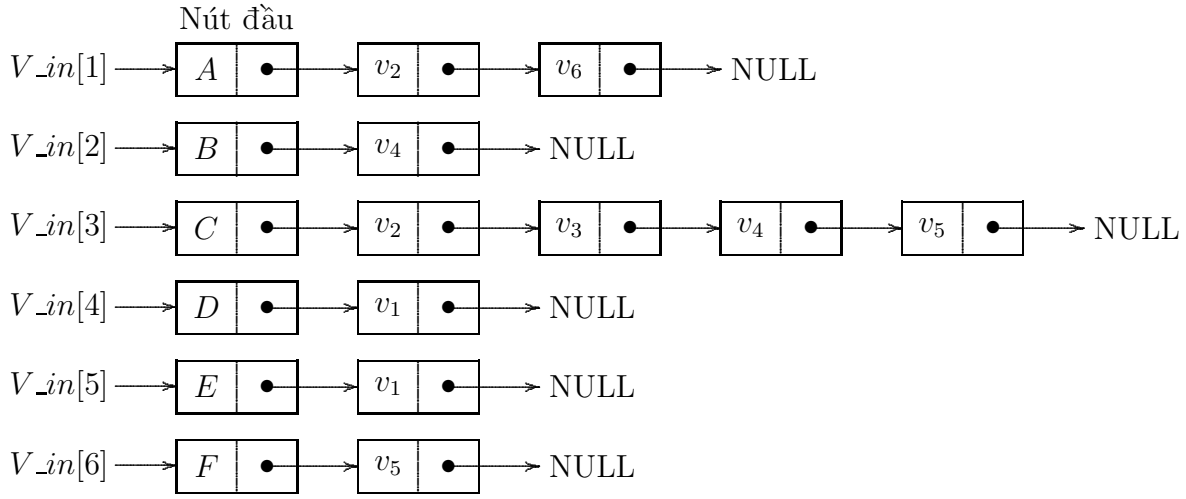
danh sách kề $V\_in[i]$. Hình 1.9 minh họa mảng các danh sách kề $V\_in[]$ của đồ thị có hướng trong Hình 1.7.

Để ý rằng, các số trong nút kề của $V\_out[]$ (tương ứng, $V\_in[]$) là những chỉ số cột (tương ứng, hàng) trong ma trận kề của đồ thị mà ở đó số 1 xuất hiện. Ngoài ra, trong trường hợp đồ thị vô hướng, hai danh sách kề này là trùng nhau.

Khi đồ thị có *trọng số*, tức là nếu mỗi cung hoặc cạnh $e \in E$ có một trọng lượng $w(e)$, ta chỉ cần mở rộng cấu trúc của mỗi nút trong danh sách kề bằng cách thêm một trường lưu trữ trọng lượng của cung.

Cách biểu diễn bằng danh sách kề của đồ thị có hướng có thể được cài đặt trong

ngôn ngữ lập trình C với các khai báo trong thư viện Graph.h (xem Phụ lục A). Để xây dựng mảng các danh sách kề $V\_out[]$ và $V\_in[]$ cho một đồ thị, ta có thể sử dụng các thủ tục MakeV_out() và MakeV_in() tương ứng.



Hình 1.9: Danh sách kề $V\_in[]$ tương ứng đồ thị trong Hình 1.7.

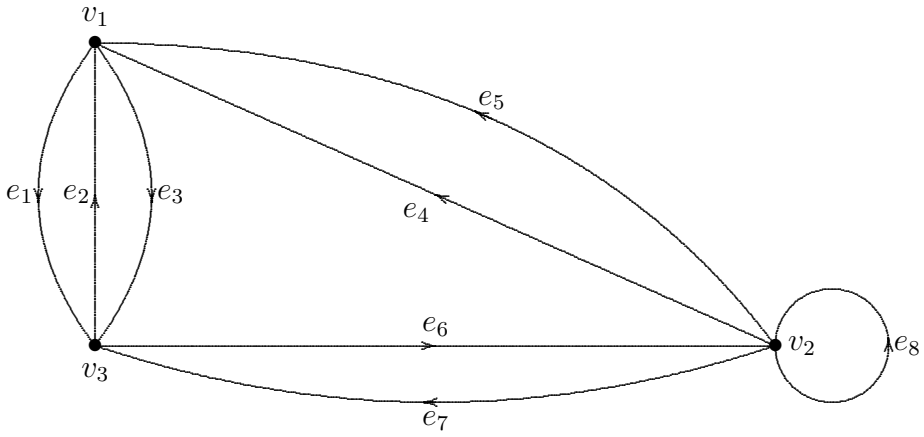
Sử dụng các ma trận liên thuộc đỉnh-cung hoặc đỉnh-cạnh

Ma trận liên thuộc đỉnh-cung hoặc đỉnh-cạnh cho phép chúng ta mô tả đầy đủ cấu trúc của một đa đồ thị *không có khuyên*. Tuy nhiên, do chỉ có hai phần tử khác không trong mỗi cột, nên có thể biểu diễn thông tin ở dạng thích hợp hơn.

Chúng ta định nghĩa hai *mảng tuyến tính* $\alpha[]$ và $\beta[]$ chiều m trong đó với mỗi cung hoặc cạnh $e_k, k = 1, 2, \ldots, m$, các giá trị $\alpha[k]$ và $\beta[k]$ là các chỉ số của các đỉnh mà e_k liên thuộc. Trong trường hợp có hướng, chúng ta quyết định $\alpha[k]$ là đỉnh gốc và $\beta[k]$ là đỉnh ngọn của cung e_k.

Chú ý rằng, trái với ma trận kề, cách biểu diễn này cũng có thể đặc trưng cho các đa đồ thị có khuyên.

Chẳng hạn, đa đồ thị của Hình 1.10 trong đó các cung được đánh số, ta nhận



Hình 1.10:

được

k	1	2	3	4	5	6	7	8
α	1	3	1	2	2	3	2	2
β	3	1	3	1	1	2	3	2

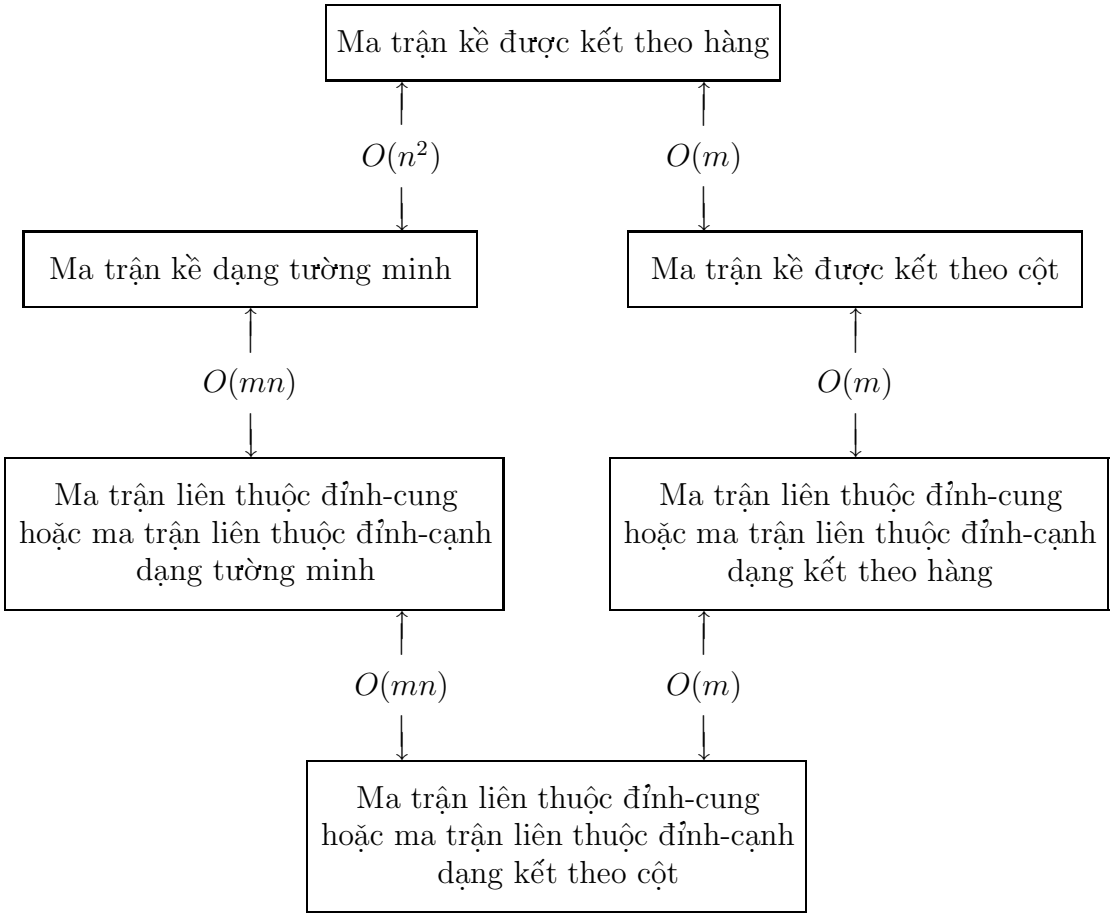
Trong trường hợp đồ thị có trọng số, ta chỉ cần thêm một mảng $w[]$ kích thước m lưu trữ trọng lượng của mỗi cạnh hoặc cung với tương ứng một-một các mảng $\alpha[]$ và $\beta[]$.

Về cách khác biểu diễn hiệu quả hơn của đồ thị vô hướng sử dụng danh sách các cạnh xem [43].

Mối liên hệ giữa các biểu diễn

Dễ dàng thấy rằng tồn tại các thuật toán đa thức để chuyển đổi giữa các kiểu dữ liệu trên đồ thị. Hình 1.11 minh họa các khả năng có thể có.

Để chuyển đổi giữa các kiểu dữ liệu, cần các chương trình thực hiện điều này (bài tập). Các biểu diễn này có thể cải biên cho phù hợp với yêu cầu. Chẳng hạn, đồ thị có trọng số có thể được biểu diễn bởi một *ma trận trọng lượng* (còn gọi là *ma trận chi phí* hay *khoảng cách*) cấp $n \times n$ trong đó phần tử (i, j) bằng trọng lượng của cạnh hay cung (v_i, v_j). Tuy nhiên, cần chú ý rằng, tính hiệu quả của nhiều vấn đề phụ thuộc vào



Hình 1.11: Mối liên hệ và độ phức tạp tính toán khi chuyển đổi giữa các biểu diễn khác nhau trong đồ thị.

biểu diễn của đồ thị. Do đó, việc chọn lựa các cấu trúc dữ liệu một cách thích hợp là quan trọng.

1.3 Liên thông

1.3.1 Dây chuyền và chu trình

Giả sử v_0, v_k là các đỉnh của đồ thị vô hướng $G := (V, E)$. *Dây chuyền* μ từ v_0 đến v_k độ dài k là một dãy xen kẽ $(k+1)$ đỉnh và k cạnh bắt đầu từ v_0 và kết thúc tại v_k,

$$\mu := \{v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k\},$$

trong đó cạnh e_i liên thuộc các đỉnh v_{i-1} và $v_i, i = 1, 2, \ldots, k$. Để giản tiện, ta thường viết

$$\mu := \{e_1, e_2, \ldots, e_k\}.$$

Dây chuyền được gọi là *đơn giản* (tương ứng, *sơ cấp*) nếu nó không đi hai lần qua cùng một cạnh (tương ứng, đỉnh).

Chu trình là một dây chuyền trong đó đỉnh đầu trùng với đỉnh cuối. Chu trình qua mỗi cạnh đúng một lần gọi là *đơn giản.* Chu trình là *sơ cấp* nếu nó đi qua mỗi đỉnh đúng một lần trừ đỉnh đầu tiên hai lần (một lần lúc xuất phát và một lúc trở về).

Đồ thị trong Hình 1.12 có

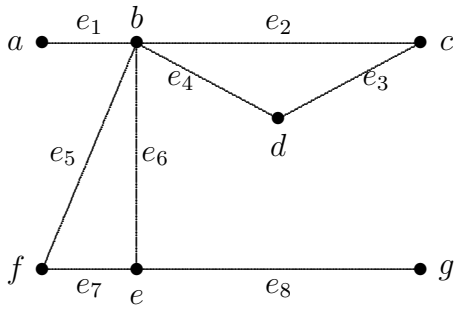
$$(a, e_1, b, e_2, c, e_3, d, e_4, b)$$

là dây chuyền từ đỉnh a đến đỉnh b có độ dài bốn. Các chu trình sau là sơ cấp

$$(b, e_2, c, e_3, d, e_4, b), \quad \text{và} \quad (b, e_5, f, e_7, e, e_6, b).$$

Trong trường hợp đồ thị không có *cạnh song song* (tức là hai đỉnh có nhiều nhất một cạnh liên thuộc chúng), để đơn giản dây chuyền μ được viết lại

$$\mu = \{v_0, v_1, v_2, \ldots, v_k\}.$$



Hình 1.12:

1.3.2 Đường đi và mạch

Giả sử v_0, v_k là các đỉnh của đồ thị có hướng $G := (V, E)$. *Đường đi* μ từ v_0 đến v_k độ dài k là một dãy xen kẽ $(k+1)$ đỉnh và k cung bắt đầu từ v_0 và kết thúc tại v_k,

$$\mu := \{v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k\},$$

trong đó cung e_i liên thuộc các đỉnh v_{i-1} và $v_i, i = 1, 2, \ldots, k$. Để giản tiện, ta có thể ký hiệu đường đi μ là $\{e_1, e_2, \ldots, e_k\}$.

Do đó trong Hình 1.13 dãy các cung

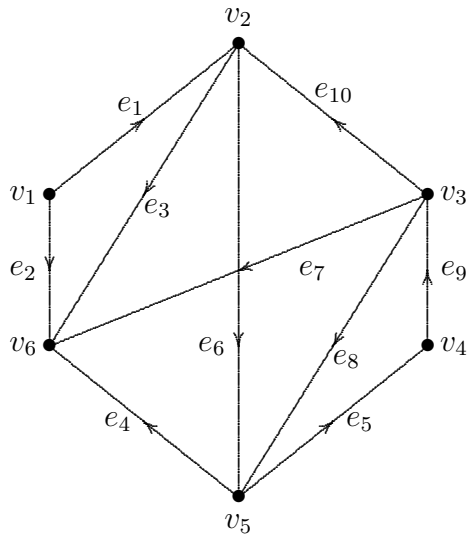
$$\begin{aligned} \mu_1 &:= \{e_6, e_5, e_9, e_8, e_4\} \\ \mu_2 &:= \{e_1, e_6, e_5, e_9\} \\ \mu_3 &:= \{e_1, e_6, e_5, e_9, e_{10}, e_6, e_4\} \end{aligned}$$

là các đường đi.

Đường đi là *đơn giản* nếu không chứa cung nào quá một lần. Suy ra các đường đi μ_1, μ_2 là đơn giản, nhưng đường đi μ_3 không đơn giản do nó sử dụng cung e_6 hai lần.

Đường đi là *sơ cấp* nếu không đi qua đỉnh nào quá một lần. Khi đó đường đi μ_2 là sơ cấp nhưng các đường đi μ_1 và μ_3 là không sơ cấp. Hiển nhiên, đường đi sơ cấp là đơn giản nhưng ngược lại không nhất thiết đúng. Chẳng hạn, chú ý rằng đường đi μ_1 là đơn giản nhưng không sơ cấp, đường đi μ_2 vừa đơn giản và vừa sơ cấp, đường đi μ_3 không đơn giản cũng không sơ cấp.

Chú ý rằng, khái niệm dây chuyền là bản sao không có hướng của đường đi và áp dụng cho các đồ thị mà không để ý đến hướng của các cung.



Hình 1.13:

Đường đi cũng có thể được biểu diễn bởi dãy các đỉnh mà chúng đi qua trong trường hợp không có *cung song song* (tức hai cung có cùng gốc và cùng ngọn). Do đó, đường đi μ_1 có thể biểu diễn bởi dãy đỉnh $\{v_2, v_5, v_4, v_3, v_5, v_6\}$.

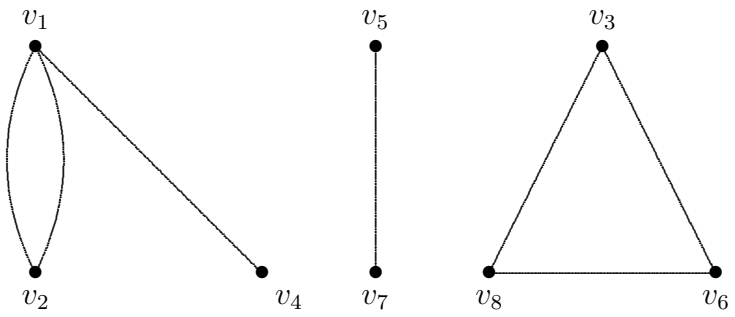
Mạch là một đường đi $\{e_1, e_2, \ldots, e_k\}$ trong đó đỉnh gốc của cung e_1 trùng với đỉnh ngọn của cung e_k. Do đó đường đi $\{e_5, e_9, e_{10}, e_6\}$ trong Hình 1.13 là mạch.

1.3.3 Đồ thị liên thông

Đồ thị vô hướng gọi là *liên thông* nếu tất cả các cặp đỉnh v_i và v_j tồn tại dây chuyền từ v_i đến v_j. Quan hệ $v_i \mathcal{R} v_j$ nếu và chỉ nếu $v_i = v_j$ hoặc tồn tại một dây chuyền nối hai đỉnh v_i và v_j là *quan hệ tương đương* (phản xạ, đối xứng và bắc cầu).

Lớp tương đương trên V xác định bởi quan hệ tương đương $\mathcal{R}$ phân hoạch tập V thành các tập con rời nhau $V_1, V_2, \ldots, V_p$. Số p gọi là *số thành phần liên thông* của đồ thị. Theo định nghĩa, đồ thị liên thông nếu và chỉ nếu số thành phần liên thông bằng một. Các đồ thị con $G_1, G_2, \ldots, G_p$ sinh bởi các tập con $V_1, V_2, \ldots, V_p$ gọi là các *thành phần liên thông* của đồ thị. Mỗi thành phần liên thông là một đồ thị liên thông.

Hình 1.14 minh họa đồ thị có ba thành phần liên thông.



Hình 1.14: Đồ thị có ba thành phần liên thông.

Nhận xét 1.3.1. Đồ thị vô hướng G có p thành phần liên thông $G_1, G_2, \ldots, G_p$ nếu và chỉ nếu ma trận kề A của G có thể viết ở dạng

$$A = \begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \cdots & A_p \end{pmatrix},$$

trong đó $A_i, i = 1, 2, \ldots, p$, là ma trận kề của thành phần liên thông G_i.

Ta cũng có nhận xét tương tự đối với ma trận liên thuộc đỉnh cạnh.

Xác định số thành phần liên thông của đồ thị là một trong những bài toán cơ bản của lý thuyết đồ thị và có nhiều ứng dụng trong thực tiễn; chẳng hạn, xác định tính liên thông của mạch điện, mạng điện thoại, v.v.

Chúng ta sẽ trình bày một số thuật toán có thời gian $O(m)$ giải bài toán này vì nó cho phép tìm lời giải của một số bài toán khác.

Bắt đầu với đỉnh nào đó của đồ thị, chúng ta liệt kê các đỉnh theo thứ tự của thuật toán *tìm kiếm theo chiều sâu*, tức là chúng ta đi, đầu tiên, xa nhất có thể được trên đồ thị mà không tạo thành chu trình, và sau đó trở về vị trí rẽ nhánh gần đây nhất mà chúng ta đã bỏ qua, và tiếp tục cho đến khi trở về đỉnh xuất phát. Do đó tập các đỉnh bắt gặp sẽ tạo thành thành phần liên thông đầu tiên.

Nếu tất cả các đỉnh của đồ thị được duyệt thì đồ thị liên thông; ngược lại, chúng ta khởi đầu lại thủ tục trên với một đỉnh mới chưa được viếng thăm; do đó ta xây dựng được thành phần liên thông thứ hai, và vân vân.

Thuật toán dưới đây trình bày giai đoạn đầu tiên, tức là tìm thành phần liên thông chứa một đỉnh đã cho-nếu thành phần này chứa tất cả các đỉnh của đồ thị thì đồ thị liên thông.

Ký hiệu num(i) là số hiệu của đỉnh v_i trong quá trình tìm kiếm. Nếu ta bắt đầu bằng đỉnh s thì đặt num(s) = 1. Ký hiệu $P(i)$ là đỉnh đứng liền trước đỉnh v_i trong *cây có gốc* (xem Chương 4) được xây dựng trong quá trình thực hiện thuật toán.

Xét đồ thị được biểu diễn bởi ánh xạ đa trị Γ. Đặt d_i^+ là số các đỉnh kề đỉnh $v_i : d_i^+ := \#\Gamma(v_i)$. Với mỗi $k = 1, 2, \ldots, n$, ký hiệu $\Gamma_k(v_i)$ là đỉnh thứ k trong tập $\Gamma(v_i)$.

Để thực hiện tìm kiếm trên đồ thị, tại mỗi bước của thuật toán chúng ta đặt $n(i)$ là chỉ số của đỉnh được viếng thăm cuối cùng từ đỉnh v_i. Khởi đầu với $n(i) = 0, i = 1, 2, \ldots, n$.

Dưới đây là thuật toán (dạng không đệ qui) của Trémaux đưa ra năm 1882 và sau đó được Tarjan cải tiến [53].

Thuật toán Trémaux-Tarjan tìm thành phần liên thông chứa đỉnh s.

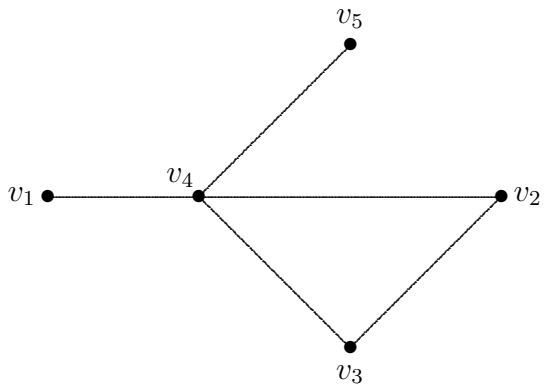
1. [Khởi tạo] Đặt $P(i) = 0, d_i^+ := \#\Gamma(v_i)$ và $n(i) = 0$ với mọi đỉnh $v_i, i = 1, 2, \ldots, n$; $k = 0$, num(s) = 1, $P(s) = s$ (tùy ý, khác không), $i = s$.

2. [Bước lặp] Trong khi $(n(i) \neq d(i))$ hoặc $(i \neq s)$ thực hiện

   - Nếu $n(i) = d(i)$ đặt $i = P(i)$ (lần ngược);
   - ngược lại, đặt $n(i) = n(i) + 1$ (viếng thăm đỉnh kế tiếp trong $\Gamma(v_i)$), và $j = \Gamma_{n(i)}(v_i)$. Nếu $P(j) = 0$ thì gán $P(j) = i, i = j, k = k + 1$, num($i$) = k.

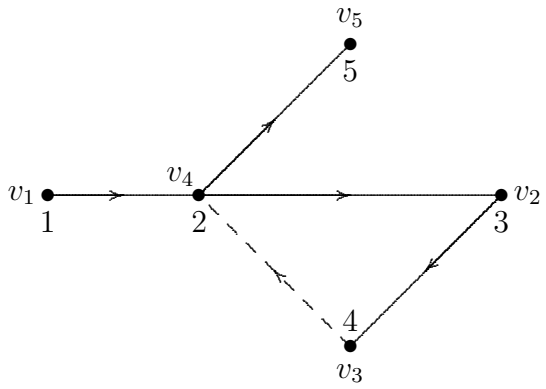
Kết thúc thuật toán, nếu $k = n$ thì đồ thị liên thông; ngược lại thành phần liên thông chứa đỉnh s gồm k đỉnh mà num(i) nhận các giá trị từ 1 đến k.

Ví dụ 1.3.2. Xét đồ thị trong Hình 1.15. Các đỉnh sẽ được viếng thăm theo thứ tự $1, 4, 2, 3$ và 5. Quá trình tìm kiếm có thể biểu diễn thành cây có gốc (đỉnh gốc là v_1)



Hình 1.15:

trong Hình 1.16.



Hình 1.16:

Thuật toán tìm kiếm theo chiều sâu

1. Thăm đỉnh xuất phát s.

2. Với mỗi đỉnh w kề với v (có hướng từ v đến w) làm các bước sau:

   Nếu w chưa được thăm, áp dụng thuật toán tìm kiếm theo chiều sâu với w như là đỉnh xuất phát.

Trong cách tìm kiếm theo chiều sâu, ta đi theo đường từ đỉnh xuất phát cho đến khi đạt đến một đỉnh có tất cả các đỉnh kề nó đã được viếng thăm. Sau đó ta quay lại đỉnh cuối cùng vừa được thăm dọc theo đường này sao cho các đỉnh kề với nó (có hướng từ nó đi ra trong trường hợp đồ thị có hướng) có thể thăm được. Để có thể quay trở lại, ta lưu trữ các đỉnh dọc theo đường này trong một ngăn xếp. Nếu thủ tục được viết dạng đệ quy thì ngăn xếp này được bảo trì một cách tự động; trong trường hợp ngược lại, cần một mảng đánh dấu các đỉnh đã được viếng thăm.

Thuật toán tìm kiếm theo chiều rộng

Trong thuật toán này, chúng ta thăm các đỉnh theo từng mức một, và khi thăm một đỉnh ở mức nào đó, ta phải lưu trữ nó để có thể trở lại khi đi hết một mức, vì vậy có thể thăm các đỉnh kề của nó. Thuật toán tìm kiếm theo chiều rộng dưới đây dùng một hàng đợi theo cách này.

1. Thăm đỉnh xuất phát.
2. Khởi động một hàng đợi chỉ chứa đỉnh xuất phát.
3. Trong khi hàng đợi không rỗng làm các bước sau:

   Lấy một đỉnh v từ hàng đợi.

   Với tất cả các đỉnh w kề với v, làm các bước sau:

   Nếu (w chưa được thăm) thì:

   Thăm w.

   Thêm w vào hàng đợi.

Các thuật toán tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu là rất cơ bản cho nhiều thuật toán khác để xử lý đồ thị. Ví dụ, để duyệt một đồ thị, ta có thể áp dụng nhiều lần một trong các cách nói trên, chọn các đỉnh xuất phát mới nếu cần thiết, cho đến khi tất cả các đỉnh được thăm.

1.3.4 Cầu, $k-$liên thông

Điểm khớp của đồ thị là một đỉnh mà xóa nó sẽ tăng số thành phần liên thông; *cầu* là cạnh mà xóa nó cũng có ảnh hưởng tương tự. Đồ thị trong Hình 1.15 có một điểm khớp là đỉnh v_4 và hai cầu là các cạnh (v_1, v_4) và (v_4, v_5).

Ví dụ 1.3.3. Trong một đồ thị không có chu trình, các đỉnh không phải là đỉnh treo, tức đỉnh có bậc ≥ 2, là điểm khớp. Ngược lại, đồ thị có chu trình Hamilton (xem Phần 5.3) không có điểm khớp.

Ví dụ 1.3.4. [Mạng thông tin] Giả sử V là tập hợp những người thuộc một tổ chức nào đó; ta đặt $(a, b) \in E$ nếu người a và b có thể báo tin với nhau. Những người liên lạc là những điểm khớp. Những người đó là những mắt xích quan trọng, vì mất họ sẽ phá vỡ tính thống nhất và sự liên kết của tổ chức.

Định lý 1.3.5. *Giả sử $G = (V, E)$ là đồ thị liên thông. Khi đó đỉnh $v \in V$ là điểm khớp nếu và chỉ nếu tồn tại hai đỉnh a và b sao cho mọi dây chuyền nối a với b đều đi qua v.*

Chứng minh. Điều kiện cần. Nếu đồ thị con sinh bởi tập hợp $V \setminus \{v\}$ không liên thông thì nó chứa ít nhất hai thành phần C và C'; giả sử a là một đỉnh nào đó của C và b là một đỉnh nào đó của C'. Trong đồ thị liên thông ban đầu G mọi dây chuyền bất kỳ nối a với b đều phải đi qua v.

Điều kiện đủ. Nếu một dây chuyền bất kỳ nối a với b đều đi qua v thì đồ thị con sinh ra bởi $V \setminus \{v\}$ không thể liên thông; bởi vậy đỉnh v là điểm khớp. ◁

Ta có thể định nghĩa: đồ thị n đỉnh ($n \geq 3$) là *2−liên thông* hay *đồ thị không tách được* nếu và chỉ nếu nó liên thông và không có điểm khớp. Các đồ thị con 2−liên thông cực đại của G tạo thành một phân hoạch của G, và gọi là các *thành phần 2−liên thông* của G.

Để tìm các điểm khớp và các thành phần 2−liên thông ta có thể sử dụng thuật toán tìm kiếm theo chiều sâu.

Với mỗi đỉnh v_i, xét tập $D(i)$ các đỉnh đứng liền trước đỉnh v_i trong cây T xác định bởi thuật toán tìm kiếm theo chiều sâu. Khi đó, với mọi đỉnh $v_j \in D(i)$ ta có

$$l(j) = \min_{k \in \Gamma(v_j)} (\text{num}(k))$$

là số nhỏ nhất được gán cho đỉnh kề với đỉnh v_j trong G. Bây giờ ta định nghĩa một chỉ số mới

$$\inf(i) := \min_{v_j \in D(i)} l(j).$$

Do đó chỉ số này tương ứng cực tiểu của $\text{num}(i)$ và số nhỏ nhất được gán cho các đỉnh mà có thể đến được bằng đúng một cạnh từ một trong các *hậu duệ* của v_i trong cây T.

Do đó, trong Ví dụ 1.3.2 (Hình 1.16), đỉnh v_2 có đúng một đỉnh trước liền kề là đỉnh v_4, và do đó $\inf(2) = \text{num}(4) = 2$.

Chú ý rằng $\inf(i) \leq \text{num}(i)$ vì khởi đầu từ *tiền bối* của v_i nó có thể trở về v_i.

Hơn nữa, dễ dàng chỉ ra rằng, nếu $\inf(i) = \text{num}(i)$ thì đỉnh v_i là điểm khớp của đồ thị. Ngoài ra, các đỉnh bắt gặp khi duyệt trở lại đỉnh v_i tạo thành một thành phần 2−liên thông.

Thuật toán dưới đây trình bày phương pháp xác định các điểm khớp của đồ thị liên thông xuất phát từ đỉnh s.

Thuật toán Tarjan tìm điểm khớp của đồ thị liên thông

1. [Khởi tạo] Đặt $P(i) = 0, d_i^+ := \#\Gamma(v_i), n(i) = 0$ và $\inf(i) = \infty$ với mọi đỉnh $v_i, i = 1, 2, \ldots, n$; $k = 0$, $\text{num}(s) = 1, P(s) = s, i = s$.

2. [Bước lặp] Trong khi $(n(i) \neq d(i))$ hoặc $(i \neq s)$ thực hiện

   - Nếu $n(i) = d(i)$ đặt

     $$q = \inf(i), i = P(i), \inf(i) = \min(q, \inf(i)).$$

     Nếu $\inf(i) = \text{num}(i)$ thì v_i là điểm khớp (và ta có thể xác định thành phần 2−liên thông).

- Ngược lại, tức là $n(i) \neq d(i)$ (viếng thăm đỉnh kế tiếp trong $\Gamma(v_i)$)

  Nếu $j = P(i)$ thì gán $n(i) = n(i) + 1, j = \Gamma_{n(i)}(i)$.

  Nếu $P(j) = 0$ thì gán

  $$\inf(i) = \min(\inf(i), \text{num}(i)), P(j) = i, i = j, k = k + 1, \text{num}(i) = k.$$

  Ngược lại nếu $P(j) \neq 0$ gán $\inf(i) = \min(\inf(i), \text{num}(j))$.

Dễ dàng kiểm tra với ví dụ trước, đỉnh v_4 là điểm khớp. Chú ý rằng có thể xác định thành phần 2−liên thông bằng cách sửa đổi Bước 2 trong Thuật toán Tarjan.

Mệnh đề 1.3.6. *Thời gian thực hiện các thuật toán Trémaux-Tarjan và Tarjan là* $O(m)$.

Chứng minh. Bài tập. ◁

Thiết diện A của đồ thị liên thông $G = (E, V)$ là tập con $A \subset V$ sao cho đồ thị con $G_{V \setminus A}$ nhận được từ G bằng cách xóa các đỉnh trong A (và các cạnh liên thuộc nó) không liên thông.

Đồ thị gọi là *k−liên thông* nếu và chỉ nếu nó liên thông, có số đỉnh $n \geq k + 1$, và không chứa một thiết diện có lực lượng $(k - 1)$.

Các đồ thị con k−liên thông cực đại của G tạo thành một phân hoạch của G và gọi là các *thành phần k−liên thông.*

Các thành phần 3−liên thông của đồ thị có thể nhận được trong thời gian $O(m)$ bằng thuật toán tương tự của Tarjan.

Đa đồ thị liên thông G gọi là *k−cạnh liên thông* nếu nó vẫn còn liên thông khi xóa đi ít hơn k cạnh.

Do đó, đa đồ thị là 2−liên thông nếu và chỉ nếu nó liên thông và không chứa cầu. Bằng cách sửa đổi lại thuật toán Tarjan, ta có thể xác định các cầu trong thời gian $O(m)$. Xét tính 2−cạnh liên thông có nhiều ứng dụng trong thực tế: mạng điện, điện thoại, v.v., phải 2−cạnh liên thông!

1.3.5 Đồ thị liên thông mạnh

Đồ thị có hướng gọi là *liên thông mạnh* nếu tất cả các cặp đỉnh v_i và v_j tồn tại đường đi từ v_i đến v_j.

Xét quan hệ $v_i \mathcal{R} v_j$ nếu và chỉ nếu hoặc $v_i = v_j$ hoặc tồn tại đường đi từ đỉnh v_i đến đỉnh v_j và đường đi từ đỉnh v_j đến đỉnh v_i. Dễ thấy đây là *quan hệ tương đương* (phản xạ, đối xứng và bắc cầu).

Lớp tương đương trên V xác định bởi quan hệ tương đương $\mathcal{R}$ phân hoạch tập V thành các tập con rời nhau $V_1, V_2, \ldots, V_p$. Số p gọi là *số thành phần liên thông mạnh* của đồ thị. Các đồ thị con $G_1, G_2, \ldots, G_p$ sinh bởi các tập con $V_1, V_2, \ldots, V_p$ gọi là các *thành phần liên thông mạnh* của G. Theo định nghĩa, đồ thị liên thông mạnh nếu và chỉ nếu số thành phần liên thông mạnh bằng một.

Nhận xét rằng, thành phần liên thông mạnh $\mathcal{C}_l$ chứa đỉnh v_l được cho bởi

$$\mathcal{C}_l = \hat{\Gamma}_{v_l} \cap \hat{\Gamma}^{-1}_{v_l},$$

và từ đó suy ra một thuật toán rất đơn giản thời gian đa thức $O(m)$ dựa trên thuật toán tìm kiếm theo chiều sâu mà có thể sử dụng nó để tìm $\mathcal{C}_l$.

Do đó tính liên thông mạnh dễ dàng kiểm tra. Chỉ cần xét khi nào $\mathcal{C}_l \equiv V$. (Hãy giải bài toán này bằng ma trận).

Nếu mặt khác, chúng ta muốn tìm tất cả các thành phần liên thông mạnh, ta sẽ sử dụng Thuật toán Tarjan.

Thật vậy ta sẽ chứng minh rằng, *thuật toán Tarjan* áp dụng với các đồ thị có hướng, cho phép tìm các thành phần liên thông mạnh.

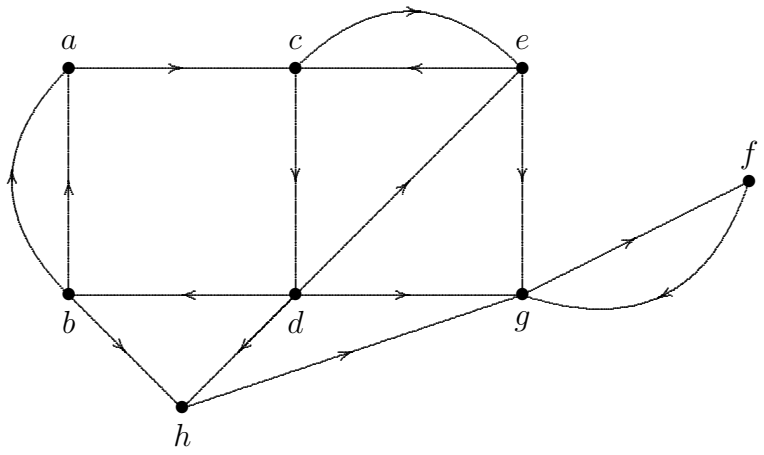
Chúng ta khởi đầu với thuật toán tìm kiếm theo chiều sâu, như trong các thuật toán tìm kiếm theo chiều sâu và Tarjan. Với mỗi đỉnh v_i xét một chỉ số mới là số nhỏ nhất của chỉ số đỉnh mà có thể đến nó bằng chỉ một cung từ một *hậu duệ* của v_i trong *cây gia phả*. Chỉ số mới này được ký hiệu là $\inf(i)$.

Nhận xét rằng chúng ta luôn luôn có $\inf(i) \leq \text{num}(i)$. Dễ dàng chỉ ra rằng khi chúng ta trở lại trong cây gia phả, thì một đỉnh mà xảy ra đẳng thức $\inf(i) = \text{num}(i)$ sẽ

phân hoạch đồ thị thành ít nhất hai thành phần liên thông mạnh, và một trong chúng tương ứng tập các đỉnh mà đã được viếng thăm trước khi tới đỉnh v_i.

Đồ thị trong Hình 1.17 có ba thành phần liên thông mạnh:

$$\mathcal{C}_1 = \{a, b, c, d, e\}, \quad \mathcal{C}_6 = \{g, f\}, \quad \mathcal{C}_8 = \{h\}.$$

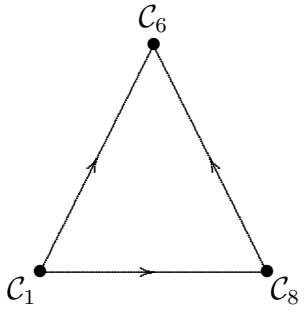


Hình 1.17:

Chúng ta sử dụng thuật ngữ *đồ thị thu gọn* để biểu diễn đồ thị G qua quan hệ liên thông mạnh $G_r := G/\mathcal{R}$. Do đó các đỉnh của G_r là các thành phần liên thông mạnh của G và tồn tại cung giữa hai đỉnh $\mathcal{C}_i$ và $\mathcal{C}_j$ nếu và chỉ nếu tồn tại ít nhất một cung giữa một đỉnh của $\mathcal{C}_i$ và $\mathcal{C}_j$ trong G. Hiển nhiên đồ thị G_r không có mạch. Hình 1.18 là đồ thị thu gọn của đồ thị có hướng trong Hình 1.17. Nghiên cứu các thành phần liên thông mạnh và tìm đồ thị thu gọn là những bài toán thực tiễn quan trọng; chẳng hạn trong mối liên hệ với xích Markov, trong phân tích cấu trúc của một hệ thống (xem [30]). Phần tiếp theo chúng ta sẽ đề cập thêm về vấn đề này.

1.4 Phạm vi và liên thông mạnh

Ta biết rằng hệ thống truyền thông của một tổ chức có thể xem như một đồ thị trong đó mỗi người tương ứng một đỉnh và các kênh truyền thông tương ứng các cung. Câu hỏi đặt ra là trong hệ thống này, thông tin từ v_i có thể đến được v_j không? Tức là có



Hình 1.18:

tồn tại đường đi từ v_i đến v_j? Nếu đường đi đó tồn tại ta nói v_j thuộc *phạm vi* của v_i. Chúng ta cũng quan tâm đến có đường đi từ v_i đến v_j với số cung hạn chế không? Mục đích của phần này là thảo luận một vài khái niệm cơ bản liên quan đến các tính chất phạm vi và liên thông mạnh của các đồ thị và giới thiệu một số thuật toán cơ sở.

Theo thuật ngữ của đồ thị biểu diễn cho một tổ chức, phần này khảo sát một số câu hỏi:

1. Số người ít nhất mà mỗi người trong tổ chức có thể liên lạc được bằng bao nhiêu?
2. Số người nhiều nhất có thể liên lạc được với nhau bằng bao nhiêu?
3. Hai bài toán trên có quan hệ gì với nhau?

1.4.1 Ma trận phạm vi

Ma trận phạm vi $R = (r_{ij})$ định nghĩa như sau:

$$r_{ij} := \begin{cases} 1 & \text{nếu tồn tại đường đi từ đỉnh } v_i \text{ đến đỉnh } v_j, \\ 0 & \text{nếu ngược lại.} \end{cases}$$

Tập $R(v_i)$ các đỉnh của đồ thị G có thể đến được từ đỉnh v_i gồm các đỉnh v_j sao cho phần tử r_{ij} bằng 1. Theo định nghĩa, tất cả các phần tử trên đường chéo của ma trận phạm vi bằng 1.

Do $\Gamma(v_i)$ là tập các đỉnh có thể đến được từ v_i theo một đường đi có độ dài 1 nên tập $\Gamma(\Gamma(v_i)) = \Gamma^2(v_i)$ gồm những đỉnh có thể đến được từ v_i theo một đường đi có độ

dài 2. Tương tự, $\Gamma^p(v_i)$ là tập những đỉnh có thể đến được từ v_i theo một đường đi có độ dài p. Do vậy, tập các đỉnh có thể đến được từ v_i là

$$R(v_i) = \{v_i\} \cup \Gamma^1(v_i) \cup \Gamma^2(v_i) \cup \cdots .$$

Nhưng do đồ thị được xét là hữu hạn và mọi đường đi đều chứa một đường đi con sơ cấp trong đó, nên tồn tại p sao cho

$$R(v_i) = \{v_i\} \cup \Gamma^1(v_i) \cup \Gamma^2(v_i) \cup \cdots \cup \Gamma^p(v_i). \tag{1.1}$$

Suy ra tập phạm vi $R(v_i)$ có thể nhận được từ Phương trình (1.1) bằng cách thực hiện các phép toán hợp từ trái sang phải cho đến khi tập hiện hành không tăng kích thước trong lần lặp kế tiếp. Số các phép toán hợp phụ thuộc vào đồ thị, mặc dù hiển nhiên rằng $p \leq n-1$, trong đó n là số đỉnh của đồ thị.

Vậy ma trận phạm vi có thể được xây dựng như sau. Tìm tập $R(v_i)$ với mỗi đỉnh v_i theo trên. Đặt $r_{ij} = 1$ nếu $v_j \in R(v_i)$, và $r_{ij} = 0$ nếu ngược lại.

Ma trận đạt được $Q = (q_{ij})$ định nghĩa như sau:

$$q_{ij} := \begin{cases} 1 & \text{nếu tồn tại đường đi từ đỉnh } v_i \text{ đến đỉnh } v_j, \\ 0 & \text{nếu ngược lại.} \end{cases}$$

Từ định nghĩa ta có $Q = R^t$.

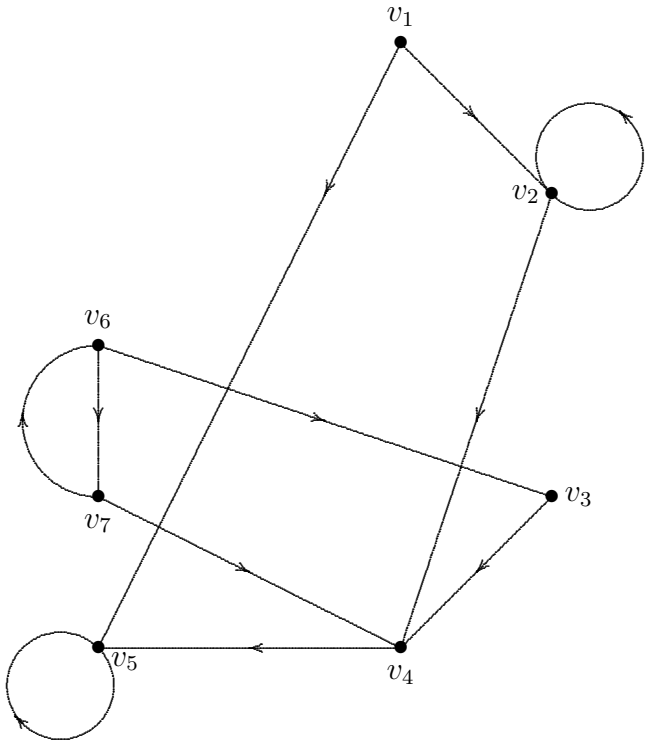
Ký hiệu $Q(v_i)$ là tập các đỉnh có thể đến được đỉnh v_i. Tương tự như trên ta cũng có

$$Q(v_i) = \{v_i\} \cup \Gamma^{-1}(v_i) \cup \Gamma^{-2}(v_i) \cup \cdots \cup \Gamma^{-p}(v_i), \tag{1.2}$$

trong đó $\Gamma^{-2}(v_i) = \Gamma^{-1}(\Gamma^{-1}(v_i)), \ldots$.

Ví dụ 1.4.1. Xét đồ thị G trong Hình 1.19. Ma trận kề của G là

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$



Hình 1.19:

Các tập phạm vi có thể xác định từ Phương trình (1.1) như sau

$$\begin{aligned}
R(v_1) &= \{v_1\} \cup \{v_2, v_5\} \cup \{v_2, v_4, v_5\} \cup \{v_2, v_4, v_5\} \\
&= \{v_1, v_2, v_4, v_5\}, \\
R(v_2) &= \{v_2\} \cup \{v_2, v_4\} \cup \{v_2, v_4, v_5\} \cup \{v_2, v_4, v_5\} \\
&= \{v_2, v_4, v_5\}, \\
R(v_3) &= \{v_3\} \cup \{v_4\} \cup \{v_5\} \cup \{v_5\} \\
&= \{v_3, v_4, v_5\}, \\
R(v_4) &= \{v_4\} \cup \{v_5\} \cup \{v_5\} \\
&= \{v_4, v_5\}, \\
R(v_5) &= \{v_5\} \cup \{v_5\} \\
&= \{v_5\}, \\
R(v_6) &= \{v_6\} \cup \{v_3, v_7\} \cup \{v_4, v_6\} \cup \{v_3, v_5, v_7\} \cup \{v_4, v_5, v_6\} \\
&= \{v_3, v_4, v_5, v_6, v_7\}, \\
R(v_7) &= \{v_7\} \cup \{v_4, v_6\} \cup \{v_3, v_5, v_7\} \cup \{v_4, v_5, v_6\} \\
&= \{v_3, v_4, v_5, v_6, v_7\}.
\end{aligned}$$

Do đó ma trận phạm vi có dạng

$$R = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}.$$

Cần chú ý rằng, do R và Q là các ma trận Bool, mỗi hàng của chúng có thể lưu dạng một hoặc hơn một từ (word). Việc tìm các ma trận này là một tính toán đơn giản do chỉ dựa vào các phép toán logic.

Từ định nghĩa, $R(v_i) \cap Q(v_j)$ là tập các đỉnh v_k mà có ít nhất một đường đi từ v_i đến v_j qua đỉnh v_k. Các đỉnh này gọi là *cốt yếu*. Tất cả các đỉnh $v_k \notin R(v_i) \cap Q(v_j)$ gọi là *không cốt yếu* hay *dư thừa* do bỏ chúng đi không ảnh hưởng đến các đường đi từ v_i đến v_j.

Các ma trận R và Q định nghĩa trên là tuyệt đối theo nghĩa số các cung trên đường đi từ v_i đến v_j không bị hạn chế. Mặt khác ta cũng có thể định nghĩa tương tự các ma trận này với số cung trên đường đi không vượt quá một số cho trước. Với mở rộng này ta cũng có thể tính được các ma trận hạn chế theo lược đồ trên.

Đồ thị được gọi là *bắc cầu* nếu tồn tại các cung (v_i, v_j) và (v_j, v_k) thì cũng tồn tại cung (v_i, v_k). *Bao đóng bắc cầu* của đồ thị $G = (V, E)$ là đồ thị $G_{tc} = (V, E \cup E')$ trong đó E' là các cung được thêm vào ít nhất để đồ thị G_{tc} có tính bắc cầu. Dễ dàng chứng minh rằng ma trận kề của đồ thị G_{tc} chính là ma trận phạm vi R và bằng

$$A + A^2 + \cdots + A^p, \quad (p \leq n-1),$$

trong đó A là ma trận kề của G. (Tại sao?)

1.4.2 Tìm các thành phần liên thông mạnh

Thành phần liên thông mạnh định nghĩa trong phần trước là đồ thị con liên thông mạnh lớn nhất trong G. Vì với đồ thị liên thông mạnh, với mọi cặp đỉnh v_i, v_j, luôn luôn tồn tại một đường đi từ đỉnh v_i đến đỉnh v_j và ngược lại, nên tồn tại duy nhất một thành phần liên thông mạnh chứa một đỉnh cho trước và v_i sẽ xuất hiện trong tập các đỉnh của một và chỉ một thành phần liên thông mạnh. Khẳng định này là hiển nhiên (tại sao?).

Nếu đỉnh v_i được coi vừa là xuất phát vừa là kết thúc thì tập các đỉnh cốt yếu tương ứng với hai đỉnh trùng nhau này (tức là tập các đỉnh thuộc mạch nào đó chứa v_i) xác định bởi $R(v_i) \cap Q(v_i)$. Vì tất cả các đỉnh cốt yếu này có thể đến từ v_i và ngược lại nên chúng cũng có thể đến được các đỉnh khác trong tập này và ngược lại. Hơn nữa, dễ thấy rằng tập $R(v_i) \cap Q(v_i)$ xác định các đỉnh của thành phần liên thông mạnh duy nhất tương ứng với đỉnh v_i.

Nếu ta loại các đỉnh này khỏi đồ thị $G = (V, \Gamma)$ thì có thể áp dụng tìm thành phần liên thông mạnh trên đồ thị con nhận được G' với tập đỉnh $V \setminus (R(v_i) \cap Q(v_i))$. Quá trình này lặp lại cho đến khi tất cả các thành phần liên thông mạnh được tìm. Kết thúc ta có đồ thị G được *phân hoạch* thành các thành phần liên thông mạnh.

Ví dụ 1.4.2. Xét đồ thị trong Hình 1.20. Chúng ta hãy tìm thành phần liên thông mạnh chứa đỉnh v_1.

Từ các phương trình (1.1) và (1.2) ta có

$$R(v_1) = \{v_1, v_2, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$$

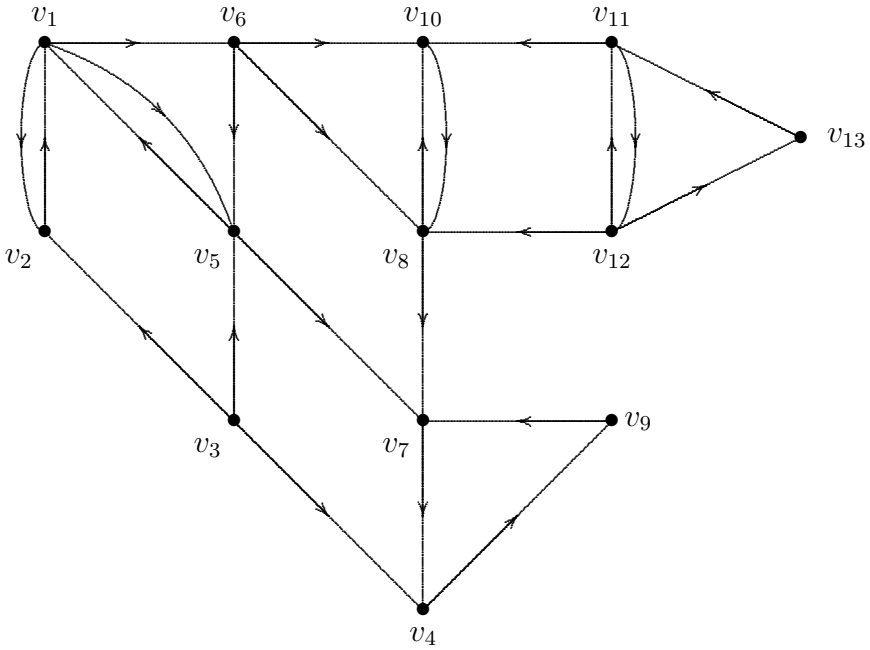
và

$$Q(v_1) = \{v_1, v_2, v_3, v_5, v_6\}.$$

Do đó thành phần liên thông mạnh chứa đỉnh v_1 là đồ thị con

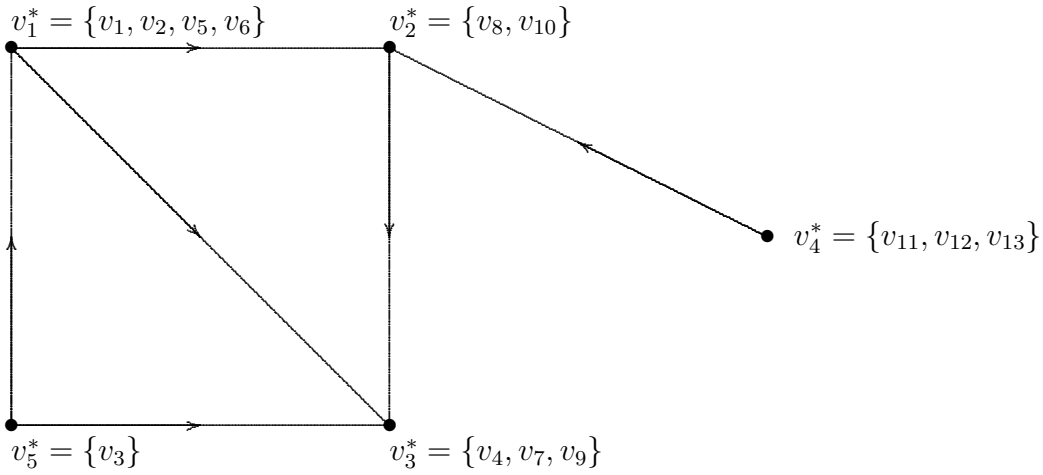
$$\langle R(v_1) \cap Q(v_1)\rangle = \langle\{v_1, v_2, v_5, v_6\}\rangle.$$

Tương tự, thành phần liên thông mạnh chứa đỉnh v_8 là đồ thị con $\langle\{v_8, v_{10}\}\rangle$, chứa đỉnh v_7 là đồ thị con $\langle\{v_4, v_7, v_9\}\rangle$, chứa đỉnh v_{11} là đồ thị con $\langle\{v_{11}, v_{12}, v_{13}\}\rangle$, và chứa đỉnh v_3 là đồ thị con $\langle\{v_3\}\rangle$. Đồ thị thu gọn G_r được cho trong Hình 1.21.



Hình 1.20: Đồ thị G.

Các phép toán được miêu tả trên để tìm các thành phần liên thông mạnh của một đồ thị cũng có thể sử dụng trực tiếp đối với các ma trận R và Q định nghĩa phần trước. Do đó nếu ta ký hiệu $R \otimes Q$ có nghĩa là ma trận nhận được từ R và Q bằng cách nhân các phần tử tương ứng thì phần tử $(R \otimes Q)_{ij} = r_{ij}q_{ij}$ bằng 1 nếu tồn tại đường đi từ v_i đến v_j và bằng 0 trong trường hợp ngược lại. Do đó hai đỉnh thuộc cùng một thành phần liên thông mạnh nếu và chỉ nếu các hàng (hoặc cột) tương ứng bằng nhau. Các đỉnh mà hàng tương ứng chứa một phần tử 1 ở cột v_j tạo thành tập đỉnh của thành phần liên thông mạnh chứa v_i. Hiển nhiên rằng có thể biến đổi ma trận $R \otimes Q$ bằng phép hoán vị các hàng và các cột sao cho ma trận thu được có dạng khối chéo; mỗi ma trận con chéo tương ứng với một thành phần liên thông mạnh của G và có các phần tử bằng 1, các phần tử khác bằng 0. Với ví dụ trên, ma trận $R \otimes Q$ được



Hình 1.21: Đồ thị thu gọn G_r.

sắp xếp lại có dạng

$$R \otimes Q = \begin{array}{c} \\ v_1 \\ v_2 \\ v_5 \\ v_6 \\ v_8 \\ v_{10} \\ v_4 \\ v_7 \\ v_9 \\ v_{11} \\ v_{12} \\ v_{13} \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} v_1 & v_2 & v_5 & v_6 & v_8 & v_{10} & v_4 & v_7 & v_9 & v_{11} & v_{12} & v_{13} & v_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & & & & & & & & & \\ 1 & 1 & 1 & 1 & & & & & & & & & \\ 1 & 1 & 1 & 1 & & & & & & & & & \\ 1 & 1 & 1 & 1 & & & & & & & & & \\ & & & & 1 & 1 & & & & & & & \\ & & & & 1 & 1 & & & & & & & \\ & & & & & & 1 & 1 & 1 & & & & \\ & & & & & & 1 & 1 & 1 & & & & \\ & & & & & & 1 & 1 & 1 & & & & \\ & & & & & & & & & 1 & 1 & 1 & \\ & & & & & & & & & 1 & 1 & 1 & \\ & & & & & & & & & 1 & 1 & 1 & \\ & & & & & & & & & & & & 1 \end{pmatrix} \end{array}.$$

1.4.3 Cơ sở

Tập B các đỉnh có thể đến được mọi đỉnh của đồ thị và nhỏ nhất theo nghĩa không có tập con thực sự nào của nó có tính chất này gọi là *cơ sở*. Do đó nếu ta viết $R(B)$ là

tập phạm vi của B :

$$R(B) = \cup_{v_i \in B} R(v_i),$$

thì B là cơ sở nếu và chỉ nếu

1. $B(V) = V$; và
2. với mọi tập con $S \subset B$ thì $R(S) \neq V$.

Điều kiện thứ hai tương đương với điều kiện $v_j \notin R(v_i)$ với hai đỉnh phân biệt $v_i, v_j \in B$; tức là một đỉnh thuộc B không thể đến được từ một đỉnh khác trong B. Điều này có thể chứng minh như sau:

Do với hai tập con H và $H' \subset H$ ta có $R(H') \subset R(H)$ nên điều kiện $R(S) \neq V$ với mọi $S \subset B$ tương đương với $R(B \setminus \{v_j\}) \neq V$ với mọi $v_j \in B$. Nói cách khác, $R(v_j) \not\subset R(B \setminus \{v_j\})$. Nhưng điều kiện này thỏa mãn nếu và chỉ nếu $v_j \notin R(B \setminus \{v_j\})$, tức là $v_j \notin R(v_i)$ với mọi $v_i, v_j \in B$.

Vậy, cơ sở là một tập B các đỉnh thỏa mãn hai điều kiện sau:

1. tất cả các đỉnh của G có thể đến được từ đỉnh nào đó của B; và
2. không đỉnh nào trong B có thể đến được từ một đỉnh thuộc B.

Từ hai điều kiện này ta suy ra

Mệnh đề 1.4.3. *(a) Không tồn tại hai đỉnh trong cơ sở B sao cho chúng thuộc cùng một thành phần liên thông mạnh của G.*

(b) Đồ thị không mạch có duy nhất một cơ sở gồm tập các đỉnh có bậc trong bằng không.

Chứng minh. Suy trực tiếp từ định nghĩa. ◁

Theo mệnh đề trên và do đồ thị thu gọn G_r của đồ thị G không có mạch nên tập cơ sở B_r của G_r là tập các đỉnh có bậc trong bằng không. Các tập cơ sở của G có thể

xác định dựa vào B_r. Cụ thể là nếu $B_r = \{S_1, S_2, \ldots, S_k\}$, trong đó k là số các tập đỉnh S_j trong cơ sở B_r của G_r, thì B là tập dạng $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ với $v_{i_j} \in S_j, j = 1, 2, \ldots, k$.

Ví dụ 1.4.4. Với đồ thị trong Hình 1.20, đồ thị thu gọn trong Hình 1.21. Cơ sở của đồ thị này là $\{v_4^*, v_5^*\}$ do v_4^* và v_5^* là hai đỉnh duy nhất của G có bậc trong bằng không. Suy ra các tập cơ sở của G là $\{v_3, v_{11}\}$, $\{v_3, v_{12}\}$ và $\{v_3, v_{13}.\}$

Đối ngẫu với khái niệm cơ sở có thể định nghĩa theo thuật ngữ của các tập hợp $Q(v_i)$ như sau.

Đối cơ sở là tập $\bar{B}$ các đỉnh của đồ thị $G = (V, \Gamma)$ thỏa mãn

$$Q(\bar{B}) = \bigcup_{v_i \in B} Q(v_i) = V,$$
$$\forall S \subset \bar{B}, Q(S) \neq V.$$

Tức là $\bar{B}$ là tập nhỏ nhất các đỉnh có thể đến được từ các đỉnh khác. Các tính chất của $\bar{B}$ tương tự với của B trong đó các khái niệm về hướng được thay bằng bản sao ngược lại.

Do vậy, đối cơ sở của đồ thị thu gọn G_r là tập các đỉnh của G_r có bậc ngoài bằng không, và từ đó ta có thể nhận được đối cơ sở của G tương tự như đã thực hiện ở trên.

Trong ví dụ của đồ thị G trong Hình 1.20, đồ thị thu gọn G_r chỉ chứa một đỉnh v_3^* có bậc ngoài bằng không. Do đó đối cơ sở của G_r là $\{v_3^*\}$ và bởi vậy đối cơ sở của G là $\{v_4\}, \{v_7\}$ và $\{v_9\}$.

Áp dụng trong cấu trúc tổ chức

Nếu đồ thị biểu diễn cấu trúc ảnh hưởng của một tổ chức thì các phần tử của một thành phần liên thông mạnh của G có ảnh hưởng và chức năng ngang nhau. Một cơ sở của G có thể hiểu là một "liên minh" có số người ít nhất nhưng lãnh đạo mọi cá nhân trong tổ chức.

Trên tập các đỉnh biểu diễn các thành viên của cùng tổ chức, ta xét đồ thị G' được xây dựng để biểu diễn các kênh truyền thông sao cho tồn tại cung (v_i, v_j) nếu v_i

có thể giao tiếp với v_j. Đồ thị G' dĩ nhiên có liên quan với G. Số người ít nhất mà biết hoặc có thể nhận tất cả các sự kiện về tổ chức tạo thành một đối cơ sở của G'. Ta có thể nói rằng một liên minh hiệu quả để lãnh đạo tổ chức là tập H những người thỏa mãn:

$$H = B(G) \cup \bar{B}(G'),$$

trong đó $B(G)$ và $\bar{B}(G')$ là một trong các cơ sở và các đối cơ sở của G và G' tương ứng được chọn sao cho $\#H$ nhỏ nhất.

Kế tiếp xét *cơ sở luỹ thừa* là tập các đỉnh $B_p \subset V$ sao cho

$$\begin{aligned} R(B_p) &= V, \qquad Q(B_p) = B_p, \\ R(S) &\neq V, \qquad \forall S \subset B_p. \end{aligned}$$

Điều kiện thứ hai nghĩa là chỉ những người bên trong B_p mới có thể có ảnh hưởng đến người khác cũng trong B_p và có thể thay bằng điều kiện tương đương: $R(V \setminus B_p) \cap B_p = \emptyset$. Điều kiện này chỉ ra rằng nếu một đỉnh trong thành phần liên thông mạnh của G thuộc B_p thì mọi đỉnh khác trong cùng thành phần liên thông mạnh này cũng thuộc B_p. Do tập cơ sở của G_r là tập các đỉnh có bậc trong bằng không nên tập cơ sở lũy thừa của G chính là

$$B_p = \bigcup_{S_i \in B^*} S_i.$$

Chẳng hạn đồ thị trong Hình 1.20 có cơ sở lũy thừa là $\{v_3, v_{11}, v_{12}, v_{13}\}$. Cần chú ý rằng, nếu đồ thị này tương ứng một tổ chức thì v_3 có thể xem là người lãnh đạo cao nhất của các nhóm v_1^*, v_2^* và v_3^*.

1.5 Đẳng cấu của các đồ thị

Ta đã biết rằng cùng một đồ thị có thể được biểu diễn bằng nhiều hình vẽ khác nhau. Việc nhận biết được trong trường hợp nào hai hình vẽ biểu diễn cùng một đồ thị, trong trường hợp nào biểu diễn hai đồ thị khác nhau, là một vấn đề không đơn giản.

Rõ ràng là hai hình cho trước biểu diễn cùng một đồ thị chỉ khi các điều kiện sau đây được thỏa mãn:

1. Số đỉnh bằng nhau;

2. Số cạnh bằng nhau;

3. Số đỉnh cùng bậc bằng nhau.

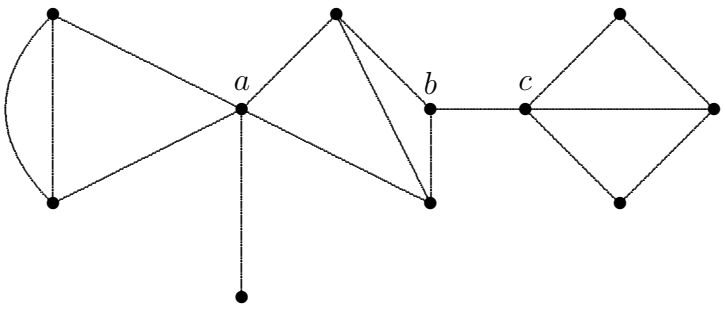
Đó là những điều kiện cần: hai hình không thỏa mãn một trong các điều kiện đó thì không biểu diễn cùng một đồ thị. Nhưng chúng cũng không phải là điều kiện đủ (hãy cho ví dụ). Hiển nhiên rằng

Định lý 1.5.1. *Điều kiện cần và đủ để hai hình biểu diễn cùng một đồ thị là tồn tại một tương ứng một-một lên giữa các đỉnh của hai hình sao cho nếu hai đỉnh của hình này được nối bởi một cạnh thì hai đỉnh tương ứng của hình kia cũng được nối với nhau bởi một cạnh và ngược lại.*

Việc tìm một tiêu chuẩn đơn giản và hiệu quả để phát hiện tính đẳng cấu của các đồ thị vẫn là bài toán mở của lý thuyết đồ thị và đang còn được tiếp tục nghiên cứu.

1.5.1 1−đẳng cấu

Đồ thị có các điểm khớp gồm ít nhất hai đồ thị con không có điểm khớp. Mỗi đồ thị con liên thông lớn nhất không có điểm khớp gọi là *khối.* Đồ thị trong Hình 1.22 có năm khối (và ba điểm khớp a, b và c). Chú ý rằng đồ thị liên thông không có điểm khớp chỉ có một khối.



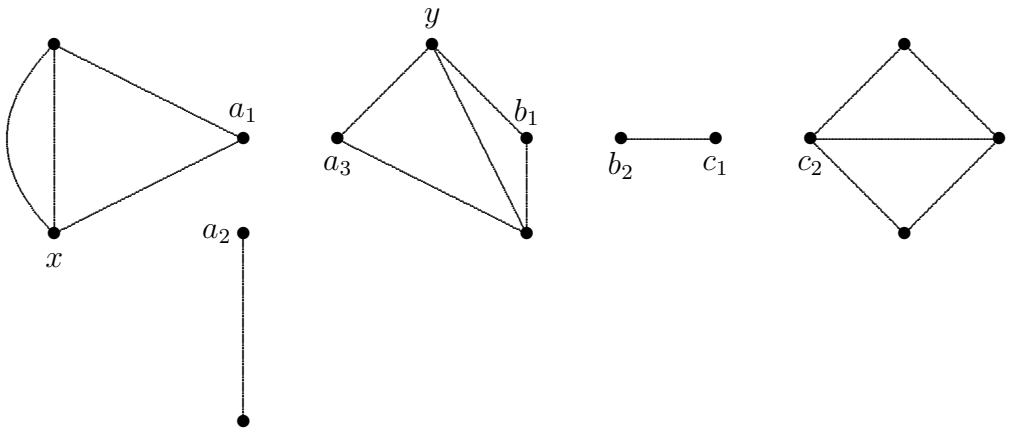
Hình 1.22:

So sánh đồ thị không liên thông trong Hình 1.23 và đồ thị trong Hình 1.22. Hiển nhiên hai đồ thị này không đẳng cấu (chúng có số đỉnh khác nhau); nhưng chúng có

mối liên hệ là các khối trong Hình 1.22 đẳng cấu với các thành phần của đồ thị trong Hình 1.23. Các đồ thị này gọi là *1-đẳng cấu*. Chính xác hơn:

Định nghĩa 1.5.2. Hai đồ thị G_1 và G_2 gọi là $1-$*đẳng cấu* nếu chúng trở thành đẳng cấu với nhau sau khi áp dụng một số lần phép toán sau:

Phép toán 1. "Tách" một điểm khớp thành hai đỉnh để tạo thành hai đồ thị con không có điểm khớp rời nhau.

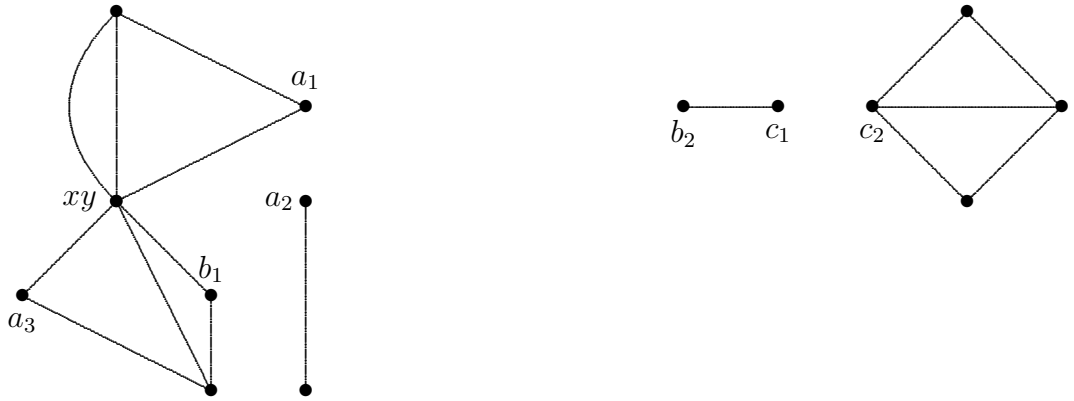


Hình 1.23:

Từ định nghĩa suy ra rằng hai đồ thị không có điểm khớp là 1-đẳng cấu nếu và chỉ nếu chúng đẳng cấu.

Điều gì sẽ xảy ra khi ta nối hai thành phần của Hình 1.23 bằng cách "dán" hai đỉnh (chẳng hạn x và y)? Chúng ta nhận được đồ thị trong Hình 1.24.

Hiển nhiên các đồ thị trong Hình 1.24 là $1-$đẳng cấu với đồ thị trong Hình 1.23. Vì các khối của đồ thị trong Hình 1.24 đẳng cấu với các khối của đồ thị trong Hình 1.22, nên hai đồ thị này là $1-$đẳng cấu. Do đó ba đồ thị trong các Hình 1.22, 1.23 và 1.24 là đôi một $1-$đẳng cấu.



Hình 1.24:

1.5.2 2−đẳng cấu

Trong phần trên chúng ta đã tổng quát hóa khái niệm đẳng cấu bằng khái niệm 1−đẳng cấu. Các đồ thị đẳng cấu thì 1−đẳng cấu nhưng ngược lại không đúng. Sự tổng quát hóa rất hữu ích trong việc nghiên cứu các đồ thị có điểm khớp.

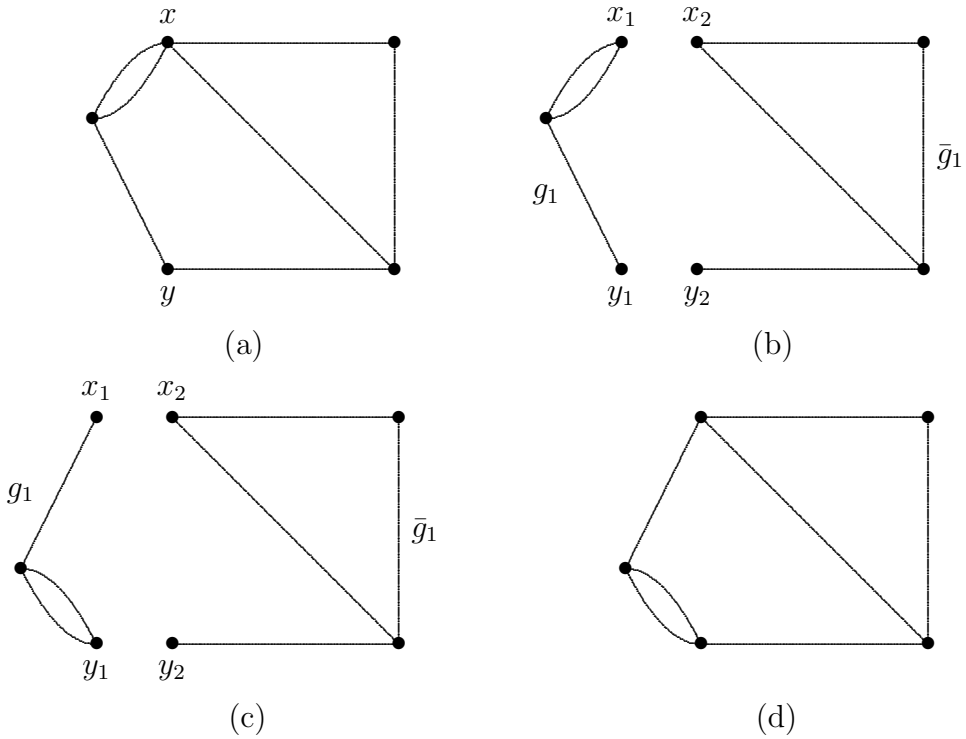
Chúng ta còn có thể mở rộng khái niệm này cho các đồ thị 2−liên thông như sau.

Trong đồ thị 2−liên thông G xét hai đỉnh x và y mà xóa chúng (và các cạnh liên thuộc chúng) thì đồ thị mất tính liên thông. Nói cách khác, G gồm một đồ thị con g_1 và phần bù của nó $\bar{g}_1$ sao cho g_1 và $\bar{g}_1$ có đúng hai đỉnh chung: x và y. Giả sử rằng chúng ta thực hiện phép toán sau trên G:

Phép toán 2. "Tách" đỉnh x thành x_1 và x_2 và đỉnh y thành y_1 và y_2 sao cho ta thu được từ G hai thành phần g_1 và $\bar{g}_1$. Giả sử các đỉnh x_1 và y_1 thuộc thành phần g_1 và x_2 và y_2 thuộc $\bar{g}_1$. Bây giờ nối hai đồ thị g_1 và $\bar{g}_1$ bằng cách hợp nhất đỉnh x_1 với đỉnh y_2 và đỉnh x_2 với đỉnh y_1. (Hiển nhiên các cạnh mà liên thuộc với x hoặc y trong G đi cùng với g_1 hoặc $\bar{g}_1$, không ảnh hưởng đến đồ thị thu được ở bước cuối).

Hai đồ thị gọi là 2−*đẳng cấu* nếu chúng trở thành đẳng cấu sau Phép toán 1 hoặc Phép toán 2, hoặc cả hai phép toán sau một số lần thực hiện. Hình 1.25 chỉ ra hai đồ thị trong Hình 1.25(a) và (d) là 2−đẳng cấu. Chú ý rằng trong Hình 1.25(a), bậc của đỉnh x bằng bốn, còn trong Hình 1.25(d) không có đỉnh nào bậc bốn.

Từ định nghĩa, suy ra 1−đẳng cấu là 2−đẳng cấu, nhưng ngược lại chưa chắc



Hình 1.25:

đúng. Tuy nhiên, với các đồ thị $k-$liên thông với $k \geq 3$ thì ba khái niệm đẳng cấu, $1-$đẳng cấu và $2-$đẳng cấu là như nhau (tại sao?).

Tương ứng chu trình

Hai đồ thị G_1 và G_2 gọi là cho một *tương ứng chu trình* nếu chúng thỏa điều kiện sau: Tồn tại tương ứng một-một giữa các cạnh của G_1 và G_2 và một tương ứng giữa các chu trình của G_1 và G_2 sao cho một chu trình trong G_1 được tạo bởi các cạnh của G_1 có một chu trình tương ứng trong G_2 được tạo bởi các cạnh tương ứng trong G_2, và ngược lại. Từ định nghĩa suy ra các đồ thị đẳng cấu có tương ứng chu trình.

Vì trong đồ thị có điểm khớp G, mỗi chu trình thuộc một khối nào đó, nên mỗi chu trình trong G vẫn tương ứng các cạnh của nó khi thực hiện Phép toán 1 trên G. Do đó, các đồ thị $1-$đẳng cấu có tính chất tương ứng chu trình.

Tương tự, xét chu trình μ trong đồ thị G sau khi thực hiện Phép toán 2 trên G.

Với chu trình μ, ta có ba trường hợp xảy ra:

1. Các cạnh thuộc μ nằm hoàn toàn trong g_1; hoặc
2. Các cạnh thuộc μ nằm hoàn toàn trong $\bar{g}_1$; hoặc
3. Các cạnh thuộc μ nằm trong cả hai đồ thị con g_1 và $\bar{g}_1$; và trong trường hợp này μ phải chứa cả hai đỉnh x và y.

Trong các Trường hợp 1 và 2, chu trình μ không ảnh hưởng qua Phép toán 2. Trong Trường hợp 3, μ vẫn gồm các cạnh cũ, ngoại trừ dây chuyền giữa các đỉnh x và y trong g_1 (thuộc chu trình μ) bị "đảo ngược lại". Do đó, mỗi chu trình sau Phép toán 2 vẫn gồm chính những cạnh cũ. Suy ra các đồ thị 2−đẳng cấu cũng có tính chất tương ứng chu trình.

Định lý 1.5.3. *Hai đồ thị là 2−đẳng cấu nếu và chỉ nếu chúng có tương ứng chu trình.*

Chứng minh. Điều kiện đủ suy từ các lý luận trên. Chứng minh điều kiện cần khó hơn và có thể xem [55]. ◁

Như sẽ thấy sau, các khái niệm 2−đẳng cấu và tương ứng chu trình đóng vai trò quan trọng khi nghiên cứu đối ngẫu của các đồ thị phẳng.

1.6 Các đồ thị đặc biệt

Phần này giới thiệu một số đồ thị đặc biệt thường gặp trong các mô hình thực tế. Các đồ thị này được quan tâm nhiều vì:

- Từ phát biểu một số bài toán;
- Từ các tính chất đặc biệt của chúng;
- Phục vụ cho một số thuật toán.

1.6.1 Đồ thị không có mạch

Đây là đồ thị thường gặp nhất khi biểu diễn các quan hệ thứ tự bộ phận trên các phần tử: với một quan hệ thứ tự $\leq$ trên tập V, xét đồ thị $G = (V, E)$ trong đó

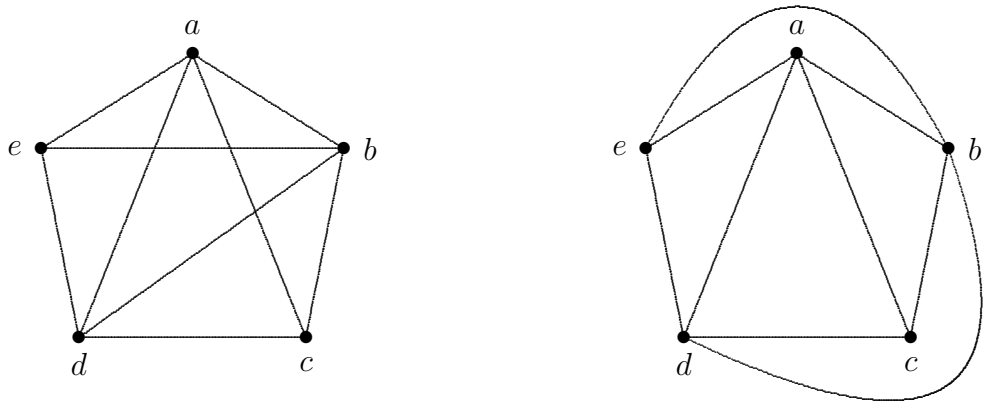
$$(v_i, v_j) \in E \quad \Leftrightarrow \quad i \leq j.$$

Các bài toán luồng trên mạng vận tải xét các đồ thị này (cũng xem các bài toán lập lịch trong [30]).

1.6.2 Đồ thị phẳng

Đồ thị vô hướng G gọi là *phẳng* nếu có thể biểu diễn trên một mặt phẳng $\mathbf{R}^2$ với các đỉnh tương ứng các điểm phân biệt trên $\mathbf{R}^2$ và các đường cong không tự cắt tương ứng các cạnh sao cho hai đường cong bất kỳ không cắt nhau ngoại trừ tại các đỉnh chung. Các đồ thị phẳng sẽ được nghiên cứu trong Chương 6.

Ví dụ 1.6.1. Đồ thị trong Hình 1.26(a) là phẳng vì chúng ta có thể vẽ lại như trong Hình 1.26(b).



(a) (b)

Hình 1.26: Đồ thị (a) được biểu diễn lại trong hình (b) là phẳng.

Chương 2

Các số cơ bản của đồ thị

2.1 Chu số

Khái niệm mà chúng ta sẽ đề cập ở đây *không phụ thuộc vào sự định hướng:* ta sẽ nói về *cạnh* chứ không phải *cung*. Để tổng quát xét đa đồ thị vô hướng $G := (V, E)$ có n đỉnh, m cạnh và p thành phần liên thông. Đặt

$$\begin{aligned} \rho(G) &:= n - p, \\ \nu(G) &:= m - \rho(G) = m - n + p. \end{aligned}$$

Ta gọi $\nu(G)$ là *chu số* của đồ thị G.

Ví dụ 2.1.1. Đồ thị vô hướng G trong hình 2.1 có $n = 5$ đỉnh, $m = 7$ cạnh và $p = 1$ thành phần liên thông nên có chu số bằng

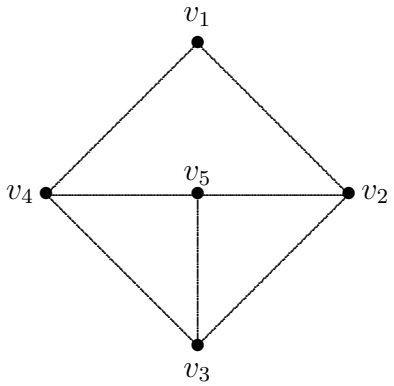
$$\nu(G) = m - n + p = 7 - 5 + 1 = 3.$$

Định lý 2.1.2. *Cho đa đồ thị vô hướng $G = (V, E)$. Giả sử G' là đồ thị nhận được từ G bằng cách nối hai đỉnh a và b của G bởi một cạnh mới; nếu a và b trùng nhau hoặc có thể nối với nhau bởi một dây chuyền của G thì*

$$\rho(G') = \rho(G), \quad \nu(G') = \nu(G) + 1;$$

trong trường hợp ngược lại

$$\rho(G') = \rho(G) + 1, \quad \nu(G') = \nu(G).$$



Hình 2.1:

Chứng minh. Theo cách xây dựng, đa đồ thị G' có $n' = n$ đỉnh, $m' = m + 1$ cạnh và giả sử G' có p' thành phần liên thông.

Nếu $a \equiv b$ hoặc có một dây chuyền nối a với b. Khi đó phép biến đổi G thành G' không thay đổi số thành phần liên thông, tức là $p = p'$. Do đó

$$\begin{aligned} \rho(G') &= n' - p' = n - p = \rho(G), \\ \nu(G') &= m' - \rho(G') = \nu(G) + 1. \end{aligned}$$

Ngược lại, nếu $a \neq b$ và không tồn tại dây chuyền nối a và b, thì do cách xác định G' ta có $p' = p - 1$. Suy ra

$$\begin{aligned} \rho(G') &= n' - p' = n - (p-1) = n - p + 1 = \rho(G) + 1, \\ \nu(G') &= m' - \rho(G') = (m+1) - (\rho(G) + 1) = m - \rho(G) = \nu(G). \end{aligned}$$

◁

Hệ quả 2.1.3. *$\rho(G) \geq 0$ và $\nu(G) \geq 0$.*

Chứng minh. Thật vậy, xuất phát từ đồ thị thành lập bằng các đỉnh của đa đồ thị vô hướng G, đỉnh nọ cô lập với đỉnh kia, ta xây dựng G' dần dần từng cạnh một; khởi đầu ta có $\rho = 0, \nu = 0$; mỗi khi thêm một cạnh, thì hoặc ρ tăng và lúc đó ν không đổi, hoặc ν tăng và lúc đó ρ không đổi. Như vậy, trong quá trình xây dựng đồ thị G', các số ρ và ν chỉ có thể tăng. ◁

Để có thể vận dụng những kết quả phong phú của đại số vector trong việc nghiên

cứu, người ta thường đặt tương ứng mỗi chu trình trong G với một vector theo cách sau đây.

Mỗi cạnh của đa đồ thị G đều được định hướng một cách tùy ý; nếu chu trình μ đi qua cạnh e_k, r_k lần thuận hướng và s_k lần ngược hướng thì ta đặt $c_k := r_k - s_k$ (nếu e_k là một khuyên thì ta luôn qui ước $s_k = 0$). Vector m chiều

$$(c_1, c_2, \ldots, c_m)$$

gọi là *vector chu trình* tương ứng với μ và ký hiệu là $\vec{\mu}$ (hay là μ nếu không thể gây ra nhầm lẫn).

Các chu trình $\mu, \mu', \mu'', \ldots$ gọi là *độc lập* nếu các vector chu trình tương ứng độc lập tuyến tính. Chú ý rằng, định nghĩa này không phụ thuộc vào hướng gán cho các cạnh.

Định lý 2.1.4. *Chu số $\nu(G)$ của $G = (V, E)$ bằng số cực đại các chu trình độc lập.*

Chứng minh. Tiến hành như trong Hệ quả 2.1.3: đầu tiên ta lấy đồ thị vô hướng không có cạnh với tập các đỉnh là V. Sau đó ta xây dựng đa đồ thị G' bằng cách thêm từng cạnh một vào. Theo Định lý 2.1.2, chu số sẽ tăng một đơn vị nếu cạnh thêm vào lập ra các chu trình mới, chu số không thay đổi trong trường hợp ngược lại.

Giả sử, trước khi thêm cạnh e_k ta đã có một cơ sở gồm các chu trình độc lập: $\mu_1, \mu_2, \mu_3, \ldots$; và sau khi thêm cạnh e_k xuất hiện thêm các chu trình sơ cấp mới $\gamma_1, \gamma_2, \ldots$, nào đó. Hiển nhiên γ_1 không thể biểu diễn tuyến tính qua hệ các chu trình μ_j (vì các vector tương ứng các chu trình μ_j có thành phần thứ k bằng không, trong khi vector tương ứng chu trình γ_1 có thành phần thứ k khác không). Mặt khác các vector $\gamma_2, \gamma_3, \ldots$ có thể biểu diễn tuyến tính qua $\gamma_1, \mu_1, \mu_2, \mu_3, \ldots$. Tóm lại mỗi khi chu số tăng một đơn vị thì số cực đại các chu trình độc lập tuyến tính cũng tăng lên một đơn vị. Định lý được chứng minh. ◁

Từ kết quả này, dễ dàng suy ra:

Hệ quả 2.1.5. (a) *Đa đồ thị vô hướng G không có chu trình nếu và chỉ nếu $\nu(G) = 0$.*

(b) *Đa đồ thị vô hướng G có đúng một chu trình nếu và chỉ nếu $\nu(G) = 1$.*

Định lý 2.1.6. *Trong đồ thị có hướng liên thông mạnh, chu số bằng số cực đại các mạch độc lập tuyến tính.*

Chứng minh. Thật vậy, xét đồ thị vô hướng lập bởi các cung khác nhau của G (mỗi cung tương ứng một cặp cạnh) và một chu trình sơ cấp μ; ta phân hoạch tập các đỉnh trên chu trình này thành: tập S các đỉnh có một cung tới nó và một cung ra khỏi nó, tập S' các đỉnh có hai cung của μ ra khỏi nó và tập S'' các đỉnh có hai cung của μ đi tới nó. Vì số các cung đi ra bằng số các cung đi tới nên $\#S' = \#S''$; giả sử $v'_1, v'_2, \ldots, v'_k$ là các phần tử của S' và $v''_1, v''_2, \ldots, v''_k$ là các phần tử của S''.

Trên chu trình μ, các phần tử của S' và của S'' xen kẽ nhau và ta giả sử rằng sau đỉnh v'_i thì đỉnh đầu tiên bắt gặp (không thuộc S) là v''_i; cuối cùng, nếu μ_0 là một đường đi gặp đỉnh x trước đỉnh y thì ta ký hiệu $\mu_0[x, y]$ là đường đi bộ phận của μ_0 từ x đến y. Vì đồ thị liên thông mạnh nên tồn tại mạch μ_1 đi qua v'_{i+1} và v''_i và dùng các cung của μ để đi từ v'_{i+1} đến v''_i. Chu trình μ là một tổ hợp tuyến tính của các mạch vì ta có thể viết

$$\begin{aligned}\mu &= \mu[v'_1, v''_1] - \mu_1[v'_2, v''_1] + \mu[v'_2, v''_2] + \cdots \\ &= \mu[v'_1, v''_1] + \mu_1[v''_1, v'_2] + \mu[v'_2, v''_2] + \mu_2[v''_2, v'_3] + \cdots - (\mu_1 + \mu_2 + \cdots).\end{aligned}$$

Vậy mọi chu trình sơ cấp đều là tổ hợp tuyến tính của các mạch, đối với các chu trình bất kỳ điều đó cũng đúng (vì nó là tổ hợp tuyến tính của các chu trình sơ cấp).

Trong $\mathbf{R}^m$, các mạch lập thành một cơ sở của không gian vector con sinh bởi các chu trình, và theo Định lý 2.1.4 thì cơ sở này có số chiều là $\nu(G)$. Vậy số cực đại các mạch độc lập tuyến tính bằng $\nu(G)$. ◁

2.2 Sắc số

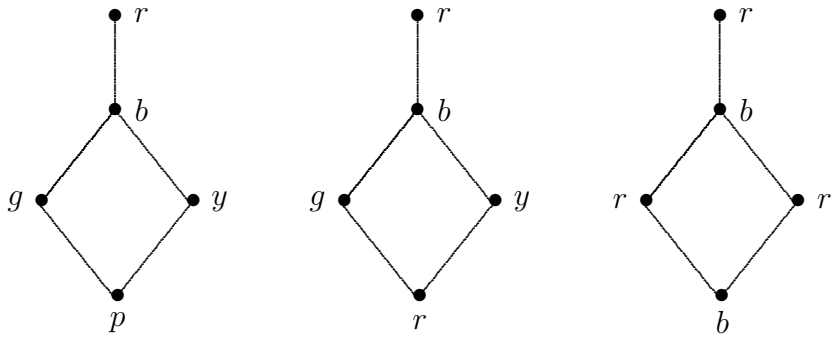
Giả sử chúng ta có một đồ thị vô hướng G với n đỉnh và cần tô màu các đỉnh sao cho hai đỉnh kề nhau có màu khác nhau. Hiển nhiên là có thể dùng n màu để tô các đỉnh đó, nhưng như thế vấn đề đặt ra lại không mang tính thực tiễn. Thế thì số màu tối thiểu đòi hỏi là bao nhiêu? Đây chính là bài toán tô màu. Khi các đỉnh được tô, chúng ta có thể nhóm chúng vào các tập khác nhau-một tập gồm các đỉnh được tô màu đỏ,

một tập các đỉnh được tô màu xanh, vân vân. Đây chính là bài toán phân hoạch. Bài toán tô màu và phân hoạch dĩ nhiên có thể xét trên các cạnh của đồ thị. Trong trường hợp đồ thị phẳng thậm chí có thể quan tâm đến việc tô màu các diện.

Trong phần này ta chỉ xét các đồ thị vô hướng liên thông.

Định nghĩa 2.2.1. Cho trước một số nguyên p, ta nói rằng đồ thị G là $p-$*sắc* nếu bằng p màu khác nhau ta có thể tô màu các đỉnh, sao cho hai đỉnh kề nhau không cùng một màu. Số p nhỏ nhất, mà đối với số đó G là $p-$sắc gọi là *sắc số* của đồ thị G và ký hiệu là $\gamma(G)$.

Ví dụ 2.2.2. Hình 2.2 minh họa ba cách tô màu khác nhau của đồ thị. Dễ dàng kiểm tra rằng đồ thị này là $2-$sắc.



Hình 2.2:

Ví dụ 2.2.3. *Bản đồ địa lý.* Ta vẽ trên mặt phẳng một bản đồ. Gọi V là tập hợp các nước, đặt $(i, j) \in E$ nếu các nước i và j có biên giới chung. Đồ thị $G = (V, E)$ đối xứng và có tính chất rất đặc biệt là: có thể vẽ nó lên mặt phẳng mà không có hai cạnh nào cắt nhau (trừ tại các đỉnh chung); nói cách khác, G là *đồ thị phẳng*. Người ta đã biết rằng sắc số của mọi đồ thị phẳng đều nhỏ hơn hoặc bằng bốn (Định lý 6.4.7). Như vậy bằng bốn màu cũng đủ để tô màu bản đồ phẳng sao cho hai nước kề nhau không cùng một màu.

Từ định nghĩa dễ dàng suy ra

1. Một đồ thị chỉ có các đỉnh cô lập là $1-$sắc.

2. Một đồ thị có một hoặc hai cạnh (không phải là một khuyên) có sắc số ít nhất bằng hai.

3. Đồ thị đầy đủ n đỉnh K_n là $n-$sắc.

4. Đồ thị là một chu trình đơn giản với n đỉnh, $n > 3$, là $2-$sắc nếu n chẵn và $3-$sắc nếu n lẻ.

5. Hiển nhiên, mọi đồ thị $2-$sắc là hai phần do chúng ta có thể phân hoạch tập các đỉnh V thành hai tập con theo màu được tô trên các đỉnh. Tương tự, đồ thị hai phần là $2-$sắc, với một trường hợp ngoại lệ tầm thường: đồ thị có ít nhất hai đỉnh cô lập và không có cạnh là hai phần nhưng là $1-$sắc.

Định nghĩa 2.2.4. Ta gọi *sắc lớp* của đồ thị G là số nguyên q có các tính chất sau:

1. Có thể dùng q màu khác nhau để tô màu các cạnh của G sao cho hai cạnh kề nhau có màu khác nhau;

2. Điều này không thể làm được với $(q-1)$ màu.

Ví dụ 2.2.5. Đồ thị G trong hình 2.2 có sắc lớp bằng 4.

Nhận xét rằng sắc lớp của đồ thị $G = (V, E)$ chính là sắc số của đồ thị $G' = (V', E')$ được xác định như sau: mỗi đỉnh của G' tương ứng một cạnh của G; cạnh $e' = (v'_1, v'_2) \in E'$ nếu các cạnh e_1 và e_2 (tương ứng với hai đỉnh v'_1, v'_2) kề nhau.

Như vậy bài toán sắc lớp đưa về bài toán sắc số. Dưới đây là một vài kết quả cơ bản về sắc số.

Định lý 2.2.6. [König] *Đồ thị vô hướng G là $2-$sắc nếu và chỉ nếu nó không có chu trình có độ dài lẻ.*

Chứng minh. Điều kiện cần. Nếu đồ thị G là $2-$sắc, thì tất nhiên G không chứa chu trình có độ dài lẻ, vì các đỉnh của một chu trình loại như vậy không thể tô bằng hai màu theo như quy tắc đã chỉ ra ở trên.

Điều kiện đủ. Giả sử đồ thị G không có chu trình có độ dài lẻ, ta chứng minh nó là $2-$sắc. Không giảm tổng quát coi G là liên thông. Ta sẽ tô màu dần các đỉnh của G theo quy tắc sau:

- tô màu xanh cho một đỉnh a nào đó;

- nếu một đỉnh x nào đó đã được tô xanh, thì ta tô đỏ tất cả các đỉnh kề với nó; nếu đỉnh y đã được tô đỏ, thì ta tô xanh tất cả các đỉnh kề với y.

Vì đồ thị G liên thông, nên sớm hay muộn thì mọi đỉnh của nó đều được tô màu hết, và một đỉnh x không thể cùng một lúc được tô xanh và tô đỏ, vì như vậy thì x và a sẽ cùng nằm trên một chu trình có độ dài lẻ. Vậy đồ thị G là $2-$sắc. ◁

Chú ý rằng tính chất đồ thị G không có chu trình với độ dài lẻ tương đương với tính chất G không có chu trình sơ cấp với độ dài lẻ. Thật vậy giả sử có một chu trình

$$\mu = \{v_0, v_1, \ldots, v_p = v_0\}$$

có độ dài p lẻ. Mỗi khi gặp hai đỉnh v_j và v_k với $j < k < p$ và $v_j = v_k$, ta phân chia μ thành hai chu trình bộ phận $\mu_1 = \{v_j, \ldots, v_k\}$ và $\mu_2 = \{v_0, \ldots, v_j, v_k, \ldots, v_0\}$; hơn nữa một trong hai chu trình có độ dài lẻ (vì nếu không như thế thì μ sẽ có độ dài chẵn). Ta thấy rằng nếu tiếp tục phân chia chu trình μ theo cách đó cho đến khi còn có thể làm được, thì mỗi lần vẫn còn được một chu trình có độ dài lẻ; vì cuối cùng mọi chu trình đều là sơ cấp, nên xảy ra mâu thuẫn; và ta có điều cần chứng minh.

Định lý sau đây cho ta biết cận trên của sắc số.

Định lý 2.2.7. *Ký hiệu $d_{\max}$ là bậc cực đại của các đỉnh trong G. Khi đó*

$$\gamma(G) \leq 1 + d_{\max}.$$

Chứng minh. Bài tập. ◁

Brooks [9] đã chứng minh rằng nếu G là đồ thị không đầy đủ, có $d_{\max}$ đỉnh thì

$$\gamma(G) \leq d_{\max}.$$

2.2.1 Cách tìm sắc số

Xét đồ thị $G = (V, \Gamma)$ có n đỉnh và m cạnh; muốn tìm sắc số của nó ta có thể dùng một phương pháp thực nghiệm rất đơn giản, áp dụng trực tiếp được, nhưng không phải lúc nào cũng có hiệu quả hoặc có thể dùng phương pháp giải tích, nó cho ta một lời giải hệ thống, nhưng nói chung cần máy tính điện tử.

Phương pháp thực nghiệm

Bằng cách tô màu tùy ý dùng các màu $1, 2, \ldots, p$ và tìm cách loại dần một màu nào đó (gọi là "màu tới hạn") trong các màu ấy. Muốn vậy, ta xét đỉnh v có màu tới hạn đó và các thành phần liên thông $C_1^{jk}, C_2^{jk}, \ldots$ của đồ thị con sinh ra bởi hai màu không tới hạn j và k. Ta có thể lập tức thay màu của đỉnh v nếu các tập hợp $C_1^{jk} \cap \Gamma(v), C_2^{jk} \cap \Gamma(v), \ldots$ không phải là hai màu: lấy riêng rẽ mỗi thành phần C^{jk} mà với thành phần đó thì các đỉnh của $C^{jk} \cap \Gamma(v)$ có màu j, hoán vị trong các thành phần đó các màu j và k (không thay đổi màu của các đỉnh khác), cuối cùng tô đỉnh v màu j (lúc này đỉnh v không kề với đỉnh nào có màu j).

Phương pháp giải tích

Kiểm tra bằng cách giải tích xem đồ thị G có thể được tô bằng p màu được không. Phương pháp đó như sau: Với mỗi cách tô bằng p màu, ta cho tương ứng với một hệ thống các số $x_{ij}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, p$, được xác định như sau:

$$x_{ij} = \begin{cases} 1 & \text{nếu đỉnh } i \text{ có màu } j, \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

Đặt

$$r_{ij} = \begin{cases} 1 & \text{nếu cạnh } e_j \text{ liên thuộc đỉnh } v_i, \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

Lúc đó bài toán trở thành tìm các số nguyên x_{ij} sao cho:

$$\begin{cases} x_{ij} & \geq & 0, & \\ \sum_{q=1}^{p} x_{iq} & \leq & 1, & i = 1, 2, \ldots, n, \\ \sum_{k=1}^{n} r_{jk} x_{kq} & \leq & 1, & j = 1, 2, \ldots, m; q = 1, 2, \ldots, p. \end{cases}$$

Ta có hệ bất đẳng thức tuyến tính. Dễ ràng thấy rằng hệ đó thích hợp với các phương pháp thông thường của quy hoạch nguyên và do đó có thể dùng phương pháp của Gomory (dựa trên phương pháp đơn hình của Dantzig) để giải.

2.3 Số ổn định trong

Với đồ thị $G = (V, \Gamma)$ cho trước ta thường quan tâm đến tập con của V có những tính chất nào đó. Chẳng hạn, tìm một tập con $S \subset V$ có số phần tử lớn nhất sao cho đồ thị con sinh bởi S là đầy đủ? Hoặc tìm tập con các đỉnh của đồ thị G có số phần tử nhiều nhất sao cho hai đỉnh trong đó không kề nhau. Một bài toán khác là tìm tập con S của V có số phần tử ít nhất sao cho mọi đỉnh thuộc $V \setminus S$ kề với một đỉnh trong S.

Các số và các tập con tương ứng lời giải các bài toán trên cho những tính chất quan trọng của đồ thị và có nhiều ứng dụng trực tiếp trong bài toán lập lịch, phân tích cluster, phân loại số, xử lý song song trên máy tính, vị trí thuận lợi và thay thế các thành phần điện tử, v.v.

Định nghĩa 2.3.1. Xét đồ thị vô hướng $G := (V, \Gamma)$; tập hợp $S \subset V$ được gọi là *tập hợp ổn định trong* nếu hai đỉnh bất kỳ của S đều không kề nhau; nói cách khác, với mọi cặp đỉnh $a, b \in S$ thì $b \notin \Gamma(a)$.

Ký hiệu $\mathcal{S}$ là họ các tập ổn định trong của đồ thị G. Khi đó

1. Tập trống $\emptyset$ thuộc $\mathcal{S}$.
2. Nếu $S \in \mathcal{S}$ và $A \subset S$ thì $A \in \mathcal{S}$. Nói cách khác, tập con của một tập ổn định trong cũng là một tập ổn định trong.

Tập ổn định trong là *cực đại* nếu thêm một đỉnh bất kỳ vào nó thì sẽ không còn ổn định trong nữa. Đại lượng

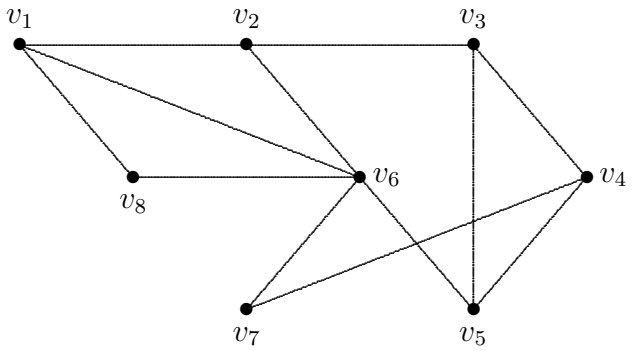
$$\alpha(G) := \max\{\#S \mid S \in \mathcal{S}\}$$

được gọi là *số ổn định trong* của G.

Ví dụ 2.3.2. Xét đồ thị G trong Hình 2.3. Tập các đỉnh $\{v_7, v_8, v_2\}$ là ổn định trong nhưng không cực đại; tập $\{v_7, v_8, v_2, v_5\}$ là ổn định trong cực đại. Các tập $\{v_1, v_3, v_7\}$ và $\{v_4, v_6\}$ cũng là tập ổn định trong cực đại và do đó, nói chung có thể có nhiều tập ổn định trong cực đại. Họ các tập ổn định trong cực đại của đồ thị này là

$$\{v_7, v_8, v_2, v_5\}, \{v_1, v_3, v_7\}, \{v_4, v_6\}, \{v_3, v_6\}, \{v_1, v_5, v_7\}, \{v_1, v_4\}, \{v_3, v_7, v_8\}.$$

Suy ra $\alpha(G) = 4$.



Hình 2.3:

Ví dụ 2.3.3. [Gauss] *Bài toán tám con hậu.* Trên bàn cờ có thể bố trí tám con hậu, sao cho không có con nào chém được con nào không? Bài toán nổi tiếng này đưa về tìm một tập hợp ổn định trong cực đại của đồ thị vô hướng có 64 đỉnh (là các ô trên bàn cờ), trong đó $y \in \Gamma(x)$ nếu các ô x và y nằm trên cùng một hàng, một cột hay một đường chéo. Thực tế, khó khăn lại lớn hơn là khi người ta mới thoạt nhìn: lúc đầu Gauss tưởng có 76 lời giải, còn tờ báo về cờ ở Berlin"Schachzeitung" năm 1854 chỉ mới đưa ra 40 lời giải thế cờ do các nhà ham cờ tìm ra. Sự thật thì có 92 lời giải như 12 sơ đồ dưới đây:

(72631485)	(61528374)	(58417263)
(35841726)	(46152837)	(57263148)
(16837425)	(57263184)	(48157263)
(51468273)	(42751863)	(35281746)

Mỗi sơ đồ trên tương ứng với một hoán vị, và từ một sơ đồ ta suy ra tám lời giải khác nhau: ba lời giải bằng cách quay $90^0, 180^0$ và 270^0; các lời giải khác suy ra bằng cách đối xứng mỗi sơ đồ nhận được qua đường chéo chính; hoán vị cuối cùng (35281746) chỉ có bốn lời giải vì sơ đồ tương ứng sẽ trùng với chính nó sau khi quay 180^0.

Ví dụ 2.3.4. *Bè* của một đồ thị G là tập hợp $C \subset V$ sao cho

$$a \in C, b \in C \quad \text{suy ra} \quad b \in \Gamma(a).$$

Nếu V là một tập hợp người, và $b \in \Gamma(a)$ chỉ sự đồng tình giữa a và b, thì thường yêu cầu tìm bè cực đại, nghĩa là hội có số người tham gia nhiều nhất. Xét đồ thị $G' := (V, \Gamma')$ xác định như sau:

$$b \in \Gamma(a') \quad \text{nếu và chỉ nếu} \quad b \notin \Gamma(a).$$

Bài toán quy về tìm một tập hợp ổn định trong cực đại của đồ thị (V, Γ').

Ví dụ 2.3.5. *Bài toán về các cô gái* là đối tượng của nhiều công trình toán học, có thể phát biểu như sau: một ký túc xá nuôi dưỡng 15 cô gái, ký hiệu là

$$a, b, c, d, e, f, g, h, i, j, k, l, m, n, o.$$

Hàng ngày các cô đi dạo chơi thành từng bộ ba, có thể đưa các cô đi chơi trong bảy ngày liền sao cho không có hai cô nào cùng đi trong một bộ ba quá một lần được không?

Với những nhận xét về đối xứng, Cayley đã tìm ra lời giải như sau:

Chủ Nhật	Thứ Hai	Thứ Ba	Thứ Tư	Thứ Năm	Thứ Sáu	Thứ Bảy
afk	abe	alm	ado	agn	ahj	aci
bgl	cno	bcf	bik	bdj	bmn	bho
chm	dfl	deh	cjl	cek	cdg	dkm
din	ghk	gio	egm	fmo	fei	eln
ejo	ijm	jkn	fhn	hil	klo	fgj

Bài toán về các cô gái cùng loại với một bài toán thứ hai cũng nổi tiếng gọi là *bài toán bổ trợ:* với 15 cô gái có thể lần lượt lập 35 bộ ba khác nhau sao cho không có

hai cô nào cùng trong một bộ ba quá một lần không? Để giải bài toán bổ trợ, chỉ cần lập đồ thị G có các đỉnh là 455 bộ ba có thể có, hai bộ ba được xem là kề nhau nếu chúng có chung hai cô gái: khi đó cần tìm một tập hợp ổn định trong cực đại. Ta có $\alpha(G) \leq 35$ vì một cô gái chỉ có mặt nhiều nhất là trong 7 bộ ba khác nhau, nên tất cả có $15 \times 7 \times \frac{1}{3} = 35$ bộ là nhiều nhất; vì vậy mọi tập hợp ổn định trong có 35 bộ ba đều là cực đại.

Để xét xem một lời giải nào đó của bài toán bổ trợ có cho lời giải của bài toán về các cô gái không, ta lập đồ thị G' có các đỉnh là 35 bộ ba cho nghiệm của bài toán bổ trợ, hai bộ ba được xem là kề nhau nếu có chung một cô gái; nếu sắc số $\gamma(G') > 7$ thì phải chọn tập hợp khác gồm các bộ ba cho nghiệm của bài toán bổ trợ. Dễ dàng kiểm tra rằng tồn tại những lời giải của bài toán bổ trợ không dẫn đến lời giải của bài toán về các cô gái.

Nhận xét 2.3.6. (a) Sắc số $\gamma(G)$ và số ổn định trong $\alpha(G)$ liên hệ với nhau bằng bất đẳng thức

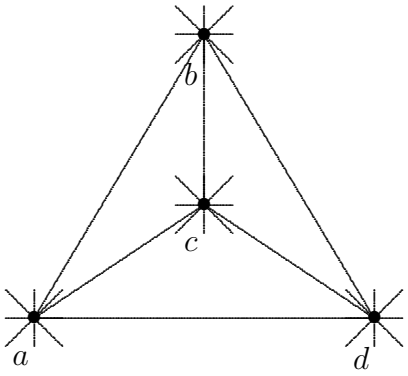
$$\gamma(G) \times \alpha(G) \geq n.$$

Thật vậy, có thể phân hoạch V thành $\gamma := \gamma(G)$ tập hợp ổn định trong, lập bởi các đỉnh cùng màu và tương ứng chứa $n_1, n_2, \ldots, n_\gamma$ đỉnh. Vậy ta có

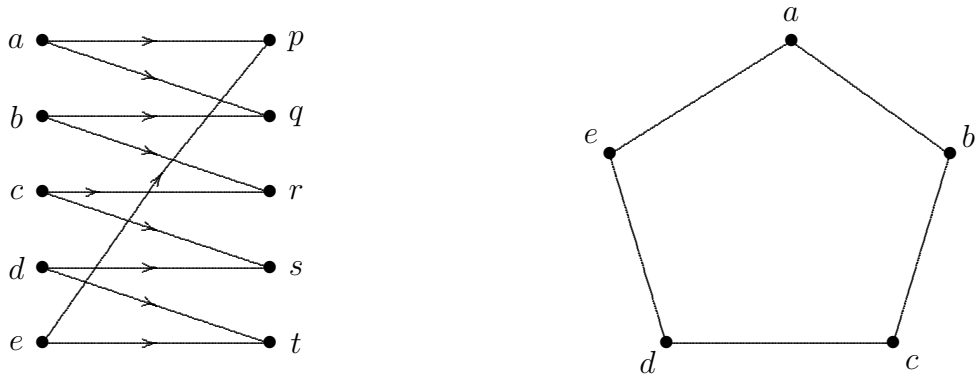
$$n = n_1 + n_2 + \cdots + n_\gamma \leq \alpha(G) + \alpha(G) + \cdots + \alpha(G) = \gamma(G).\alpha(G).$$

(b) Ta có thể hỏi: phải chăng mối liên hệ giữa hai khái niệm là chưa chặt chẽ. Phải chăng có thể tìm sắc số bằng cách: trước hết dùng màu (1) để tô tập ổn định trong cực đại S_1. Rồi dùng màu (2) để tô tập ổn định trong cực đại S_2 của đồ thị con sinh ra bởi các đỉnh của $V \setminus S_1$, sau đó dùng màu (3) để tô tập hợp ổn định trong cực đại của đồ thị con còn lại, v.v. Thực ra không phải như vậy. Đồ thị trên Hình 2.4 (có sắc số bằng 4) chứng tỏ điều đó: các đỉnh kề với các đỉnh a, b, c và d lập thành tập ổn định trong cực đại duy nhất, nếu ta tô chúng cùng một màu thì ta phải dùng ba màu chỉ để tô các đỉnh a, b, c, d, nhưng rõ ràng điều đó không thể làm được.

Ví dụ 2.3.7. [Shannon] *Bài toán về dung lượng thông tin của một tập hợp tín hiệu.* Xét trường hợp rất đơn giản là máy phát có thể truyền đi năm tín hiệu: a, b, c, d, e; ở máy thu, mỗi tín hiệu có thể cho hai cách hiểu khác nhau: tín hiệu a có thể hiểu là p hay q, tín hiệu b có thể hiểu là q hay $r, \ldots$ (Hình 2.5 (a)).



Hình 2.4:



Hình 2.5:

Số cực đại các tín hiệu mà ta có thể sử dụng là bao nhiêu để cho ở máy thu không xảy ra nhầm lẫn giữa các tín hiệu được sử dụng. Bài toán quy về tìm một tập ổn định trong cực đại S của đồ thị G (Hình 2.5(b)), trong đó hai đỉnh là kề nhau nếu chúng biểu thị hai tín hiệu có thể lẫn lộn với nhau ở máy thu, ta sẽ lấy $S = \{a, c\}$, và rõ ràng $\alpha(G) = 2$. Thay cho các tín hiệu một chữ cái, ta có thể dùng các "từ" gồm hai chữ cái với điều kiện là các từ này không gây ra lầm lẫn ở máy thu. Với các chữ cái a và c (các chữ cái này không thể lẫn lộn nhau) ta lập mã: aa, ac, ca, cc; như vậy ta được $[\alpha(G)]^2 = 4$ từ. Nhưng ta còn có thể lập mã phong phú hơn: aa, bc, ce, db, ed. (Dễ dàng kiểm tra rằng hai từ bất kỳ trong các từ này không thể lẫn lộn với nhau ở máy thu).

Tìm tất cả các tập ổn định trong cực đại

Phương pháp suy luận sau (không hiệu quả đối với các đồ thị lớn) tìm tất cả các tập ổn định trong cực đại dựa trên đại số Boole. Ta xem mỗi đỉnh của đồ thị tương ứng với một biến Boole. Ký hiệu $x + y, xy$ và x' là các phép toán tuyển, hội và phủ định của các biến Boole x, y.

Với đồ thị G cho trước, ta cần tìm một tập con cực đại các đỉnh không kề nhau trong G. Biểu diễn mỗi cạnh (x, y) bằng một tích Boole xy và sau đó lấy tổng của tất cả các tích này ta được biểu thức Boole

$$\varphi = \sum_{(x,y)\in E} xy.$$

Kế tiếp biểu diễn phủ định φ' dạng tổng của các tích Boole

$$\varphi' = f_1 + f_2 + \cdots + f_k.$$

Một tập các đỉnh là ổn định trong cực đại nếu và chỉ nếu $\varphi = 0$, nghĩa là $\varphi' = 1$; hay tương đương có ít nhất một $f_i = 1$; tức là nếu mỗi đỉnh xuất hiện trong f_i (ở dạng phủ định) được loại ra khỏi tập đỉnh của G. Do đó mỗi f_i sẽ cho tương ứng một tập ổn định trong cực đại và do đó phương pháp này cho ta tất cả các tập ổn định trong cực đại. Chẳng hạn xét đồ thị trong Hình 2.6 ta có

$$\varphi = ab + bc + bd + be + ce + de + ef + eg + fg.$$

$$\varphi' = (a' + b')(b' + c')(b' + d')(b' + e')(c' + e')(d' + e')(e' + f')(e' + g')(f' + g').$$

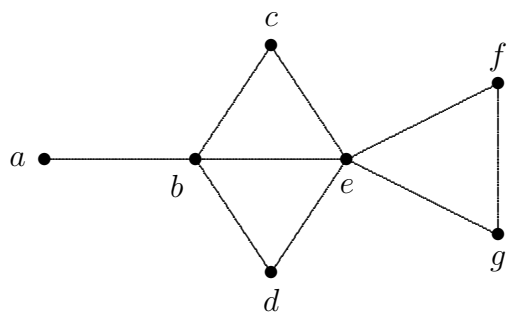
Rút gọn và đưa về dạng tổng của các tích ta được

$$\varphi' = b'e'f' + b'e'g' + a'c'd'e'f' + a'c'd'e'g' + b'c'd'f'g'.$$

Bây giờ nếu loại bỏ khỏi tập đỉnh của G các đỉnh xuất hiện trong bất kỳ một trong năm số hạng của tổng, ta sẽ thu được một tập ổn định trong cực đại. Các tập ổn định trong cực đại tương ứng là

$$\{a, c, d, g\}, \quad \{a, c, d, f\}, \quad \{b, g\}, \quad \{b, f\}, \quad \text{và} \quad \{a, e\}.$$

Và do đó số ổn định trong của đồ thị này bằng 4.



Hình 2.6:

2.4 Số ổn định ngoài

Định nghĩa 2.4.1. Cho đồ thị $G := (V, \Gamma)$. Tập hợp con T các đỉnh của V là tập *ổn định ngoài* nếu mọi đỉnh không thuộc T kề với ít nhất một đỉnh trong T; tức là $T \cap \Gamma(v) \neq \emptyset$ với mọi đỉnh $v \notin T$.

Ký hiệu $\mathcal{T}$ là họ các tập hợp ổn định ngoài của đồ thị G thì:

1. $V \in \mathcal{T}$.
2. $T \in \mathcal{T}$ và $T \subset A$ thì $A \in \mathcal{T}$.

Ta định nghĩa *số ổn định ngoài* của đồ thị G là số

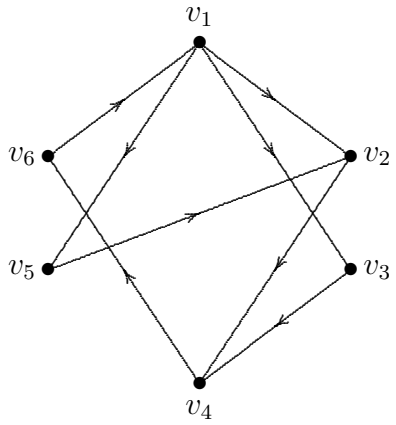
$$\beta(G) := \min\{\#T \mid T \in \mathcal{T}\}.$$

Tập ổn định ngoài cực tiểu là tập ổn định ngoài sao cho bỏ đi một đỉnh thì không còn ổn định ngoài nữa.

Ví dụ 2.4.2. Xét đồ thị G trong Hình 2.6. Các tập $\{b, g\}$ và $\{a, b, c, d, f\}$ là ổn định ngoài. Các tập $\{b, e\}$ và $\{a, c, d, f\}$ là ổn định ngoài cực tiểu. Dễ thấy $\beta(G) = 2$.

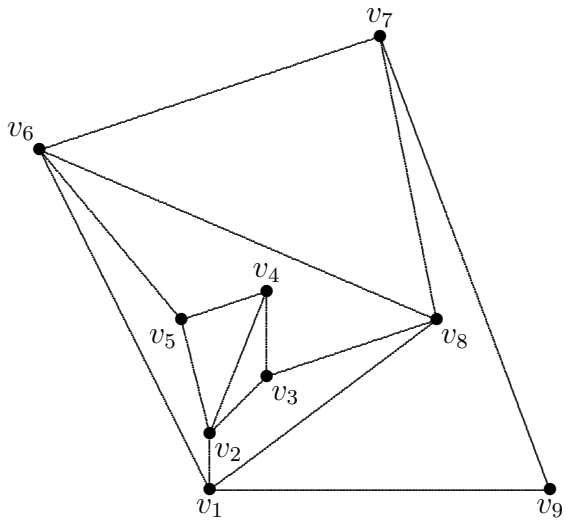
Ví dụ 2.4.3. Với đồ thị trong Hình 2.7, tập ổn định ngoài cực tiểu chứa số nhỏ nhất các phần tử là $\{v_1, v_4\}$ và do đó $\beta(G) = 2$.

Bài toán ta xét ở đây là xây dựng một tập hợp ổn định ngoài với số phần tử ít nhất.



Hình 2.7:

Ví dụ 2.4.4. *Bài toán về những người đứng gác.* Trong một trại giam ở thành phố N, mỗi nhà giam có một trạm gác độc lập, nhưng người đứng gác, chẳng hạn ở nhà giam v_1, cũng có thể nhìn thấy những gì xảy ra ở các nhà giam v_2, v_6, v_8, v_9, những nhà giam này thông với nhà giam v_1 bởi một hành lang thẳng như ta thấy trên Hình 2.8. Hỏi số người đứng gác cần thiết ít nhất là bao nhiêu để quan sát được mọi nhà giam? Ta phải tìm số ổn định ngoài của đồ thị vô hướng trong Hình 2.8, đối với sơ đồ rất đơn giản này thì số ổn định ngoài là 2.



Hình 2.8:

Ví dụ 2.4.5. *Vị trí của các "tâm" để phủ một vùng.* Có nhiều lĩnh vực liên quan đến bài toán này:

1. Vị trí của các máy phát T.V. hay radio truyền đến một số vị trí cho trước.

2. Vị trí của các trạm gác để giám sát một vùng.

3. Trạm làm việc của người bán hàng để phục vụ một vùng.

Giả sử rằng một vùng-cho bởi hình vuông lớn trong Hình 2.9-được phân hoạch thành 16 vùng nhỏ hơn. Giả sử một trạm gác được đặt trong một vùng nhỏ có thể giám sát không chỉ hình vuông nó đặt mà còn những hình vuông có chung biên với nó. Câu hỏi đặt ra là số tối thiểu các trạm gác và vị trí của chúng sao cho có thể điều khiển toàn bộ vùng. Nếu ta biểu diễn mỗi vùng nhỏ bởi một đỉnh và một cạnh liên thuộc hai đỉnh

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Hình 2.9:

nếu hai vùng tương ứng có chung một biên (hãy vẽ hình!). Bài toán đưa về tìm tập ổn định ngoài có số phần tử nhỏ nhất-là $\beta(G)$. Trong ví dụ này, $\beta(G) = 4$ và các trạm gác cần đặt tại các vị trí $\{3, 5, 12, 14\}$ hoặc tại $\{2, 9, 15, 8\}$.

Ví dụ 2.4.6. *Bài toán về 5 con hậu.* Trên bàn cờ quốc tế cần bố trí bao nhiêu con hậu để cho mọi ô trên bàn cờ đều bị ít nhất một con hậu khống chế. Bài toán này đưa về tìm một tập hợp ổn định ngoài cực tiểu cho đồ thị có 64 đỉnh (là các ô của bàn cờ), trong đó hai đỉnh kề nhau nếu và chỉ nếu các ô tương ứng nằm trên cùng một hàng, một cột hoặc cùng một đường chéo.

Số ổn định là +5 đối với các con hậu, chẳng hạn với hai cách sắp xếp:

$$(3, 3), (4, 6), (5, 4), (6, 2), (7, 5)$$

và

$$(5, 1), (8, 3), (4, 4), (3, 6), (7, 8),$$

trong đó (i, j) là vị trí của con hậu ở hàng i và cột j. Cũng dễ thấy số ổn định là: $+8$ đối với các con tháp; $+12$ đối với các con mã; $+8$ đối với các con điên.

Ví dụ 2.4.7. *Bài toán về sáu cái đĩa đỏ.* Trò chơi sau đây rất quen thuộc đối với những người ưa thích hội chợ Pháp: Trên bàn đặt một cái đĩa lớn màu trắng (bán kính 1), hãy tìm cách phủ hoàn toàn đĩa đó bởi sáu đĩa con đỏ (bán kính $r < 1$) lần lượt đặt trên bàn, và khi đã đặt rồi thì không được xê dịch nữa. Hỏi bán kính r nhỏ nhất bằng bao nhiêu để có thể giải được?

Bài toán đưa về tìm một tập ổn định ngoài cực tiểu T cho "đồ thị vô hạn" (V, E), trong đó V là tập hợp các điểm của đĩa trắng, và hai đỉnh kề nhau nếu khoảng cách giữa hai điểm tương ứng nhỏ hơn hoặc bằng r.

Từ định nghĩa dễ dàng suy ra

1. Tập gồm đúng một đỉnh trong đồ thị đầy đủ K_n là tập ổn định ngoài cực tiểu.

2. Mọi tập ổn định ngoài chứa ít nhất một tập ổn định ngoài cực tiểu.

3. Một đồ thị có thể có nhiều tập ổn định ngoài cực tiểu với kích thước khác nhau.

4. Tập ổn định ngoài cực tiểu có thể hoặc không là ổn định trong.

5. Mọi tập ổn định trong cực đại là tập ổn định ngoài. Thật vậy, nếu trái lại thì có ít nhất một đỉnh không nằm trong tập này và cũng không kề với một đỉnh bất kỳ trong đó. Một đỉnh như thế có thể thêm vào mà không phá vỡ tính ổn định trong và do đó mâu thuẫn với tính cực đại của nó.

6. Một tập ổn định trong có tính ổn định ngoài chỉ nếu nó là tập ổn định trong cực đại.

7. Với mọi đồ thị G bất đẳng thức sau luôn thỏa

$$\beta(G) \leq \alpha(G).$$

Tìm các tập ổn định ngoài cực tiểu

Tương tự cách xác định tập ổn định trong cực đại, ta có thể dùng đại số Boole để xác định các tập ổn định ngoài cực tiểu của đồ thị G.

Để kiểm tra đỉnh v_i ta cần lấy hoặc đỉnh v_i hoặc một đỉnh kề với nó. Tập nhỏ nhất các đỉnh thỏa mãn điều kiện này là tập cần tìm. Do đó, với mỗi đỉnh v_i trong G chúng ta xét tích Boole của các tổng $(v_{i_1} + v_{i_2} + \cdots + v_{i_d})$ trong đó $v_{i_1}, v_{i_2}, \ldots, v_{i_d}$ là các đỉnh kề với đỉnh v_i và d là bậc của nó:

$$\theta = \prod_{v_i \in V} (v_{i_1} + v_{i_2} + \cdots + v_{i_d}).$$

Khi θ viết dưới dạng tổng của các tích, mỗi số hạng trong đó sẽ tương ứng với một tập ổn định ngoài cực tiểu. Chúng ta minh họa thuật toán này sử dụng đồ thị trong Hình 2.6. Ta có

$$\theta = (a+b)(b+c+d+e+a)(c+b+e)(d+b+e)(e+b+c+d+f+g)(f+e+g)(g+e+f).$$

Theo luật hấp thụ $(x + y)x = x$ ta suy ra

$$\begin{aligned} \theta &= (a+b)(c+b+e)(d+b+e)(f+e+g) \\ &= ae + be + bf + bg + acdf + acdg. \end{aligned}$$

Mỗi số hạng trong sáu biểu thức này tương ứng một tập ổn định ngoài tối thiểu. Hiển nhiên $\beta(G) = 2$.

Để xác định một tập hợp ổn định trong cực đại, hay một tập hợp ổn định ngoài cực tiểu, ta có thể xét lần lượt mỗi tập hợp con T của V và kiểm tra xem nó có thỏa mãn các điều kiện đã đề ra hay không. Tất nhiên phương pháp loại dần này nói chung không dùng được. Người ta cần phải đưa ra các thuật toán. Chẳng hạn, khi giải quyết bài toán về tám con hậu, nếu dùng phương pháp loại dần cần khoảng 10 giờ để tính toán, trong khi đó nếu có một thuật toán tốt, chúng ta chỉ cần ba phút là thừa đủ; ngoài ra cần chú ý rằng: nếu ta đã đạt được việc xếp 8 con hậu trên bàn cờ, thì một lập luận rất đơn giản chỉ cho ta rõ rằng không thể làm tăng số con hậu lên được nữa trong trường hợp tổng quát hơn, cũng cần có một tiêu chuẩn đơn giản để nhận biết một tập hợp ổn định trong cực đại hay không, và chỉ có thuật toán đã được suy nghĩ kỹ mới có thể giúp ta biết điều đó (xem [4]).

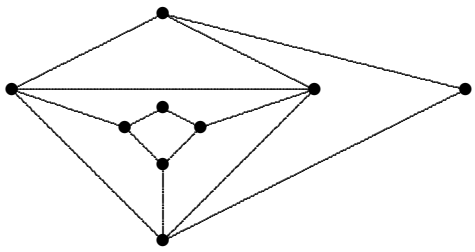
2.5 Phủ

Tập g các cạnh được gọi là *phủ* của đồ thị $G = (V, \Gamma)$ nếu mỗi đỉnh bất kỳ thuộc V liên thuộc với ít nhất một cạnh trong g.

Ví dụ 2.5.1. (a) Cây bao trùm trong đồ thị vô hướng liên thông là một phủ.

(b) Chu trình Hamilton (nếu nó tồn tại) trong đồ thị là một phủ.

Ví dụ 2.5.2. Trong trại giam của thành phố N, sơ đồ vẽ kèm theo đây, mỗi người gác ở hành lang (a, b) phải giám sát hai gian a và b là hai đầu của hành lang; hỏi số người gác ít nhất là bao nhiêu để mỗi gian ít nhất có một người giám sát? Bài toán này đưa về tìm phủ cực tiểu của một đồ thị. Trên đồ thị Hình 2.12, phủ nhỏ nhất có năm đỉnh, vậy năm người gác là đủ.

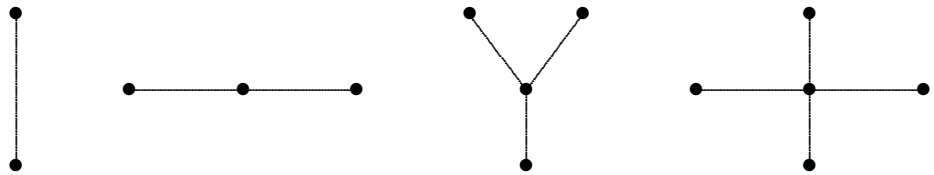


Hình 2.10:

Trong một đồ thị, có thể có nhiều phủ, vấn đề ở đây là cần nghiên cứu phủ mà có một vài tính chất đặc biệt nào đó như cây bao trùm hay chu trình Hamnilton. Ta sẽ nghiên cứu *phủ tối thiểu:* là phủ mà nếu bỏ đi một cạnh thì sẽ phá hủy thuộc tính phủ của nó.

Từ định nghĩa dễ dàng suy ra

1. Tồn tại phủ trong một đồ thị nếu và chỉ nếu đồ thị không có đỉnh cô lập.
2. Một phủ của đồ thị n đỉnh có ít nhất $[n/2]$ cạnh (ký hiệu $[x]$ là số nguyên lớn nhất không vượt quá x).
3. Mọi cạnh treo trong đồ thị nằm trong phủ của đồ thị.



(a) (b) (c) (d)

Hình 2.11: Các đồ thị hình sao.

4. Mọi phủ đều chứa phủ tối thiểu.

5. Tập các cạnh g là một phủ nếu và chỉ nếu, với mọi đỉnh $v \in V$ ta có $d_{G\setminus g}(v) \leq d_G(v) - 1$, trong đó $G \setminus g$ là đồ thị G xóa đi các cạnh thuộc g.

6. Phủ tối thiểu không chứa chu trình. Do vậy mọi phủ tối thiểu của đồ thị n đỉnh không chứa hơn $(n-1)$ cạnh.

7. Một đồ thị, nói chung, có nhiều phủ tối thiểu, và chúng có thể có kích thước khác nhau (gồm số các cạnh khác nhau). Số các cạnh trong một phủ tối thiểu có cỡ nhỏ nhất được gọi là *số phủ* của đồ thị.

Định lý 2.5.3. *Phủ g của một đồ thị là tối thiểu nếu và chỉ nếu g không chứa các dây chuyền có độ dài lớn hơn hoặc bằng ba.*

Chứng minh. Điều kiện cần. Giả sử g là phủ tối thiểu của G mà chứa một dây chuyền có độ dài ba là

$$\mu := \{v_1, e_1, v_2, e_2, v_3, e_3, v_4\}.$$

Trong g bỏ cạnh e_2 thì ta vẫn thu được một phủ của G, điều này mâu thuẫn với tính tối thiểu của g.

Điều kiện đủ. Ngược lại, nếu g không chứa một dây chuyền nào có độ dài lớn hơn hoặc bằng ba thì các thành phần của nó phải có một trong các dạng "hình sao" như trong Hình 2.11. Hiển nhiên nếu xóa bất kỳ một cạnh của đồ thị hình sao thì sẽ có ít nhất một đỉnh không liên thuộc với cạnh còn lại của g. Vậy g là phủ tối thiểu. ◁

Ví dụ 2.5.4. Giả sử đồ thị trong Hình 2.12 biểu diễn bản đồ giao thông của một thành phố. Mỗi đỉnh là một nút giao thông thường xảy ra tắc nghẽn và mỗi cạnh tương ứng

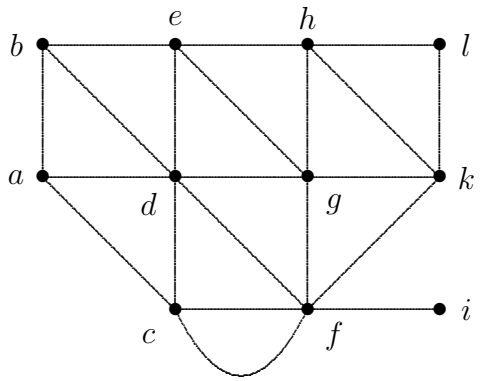
một đại lộ nối hai nút giao thông. Cần đặt các đội tuần tra trên các đại lộ sao cho có thể giám sát tất cả các nút giao thông. Vấn đề đặt ra là số tối thiểu các đội tuần tra là bao nhiêu? Câu trả lời chính là tìm phủ tối thiểu với số phần tử ít nhất của đồ thị. Dễ kiểm tra hai tập sau là phủ tối thiểu:

$$\{(a,d),(b,e),(c,f),(f,i),(g,h),(k,l)\}$$

và

$$\{(a,b),(b,d),(b,e),(c,f),(f,g),(f,i),(h,l)\}.$$

Do có 11 đỉnh và mỗi cạnh phủ nhiều nhất hai đỉnh nên không thể phủ đồ thị ít hơn sáu cạnh.



Hình 2.12:

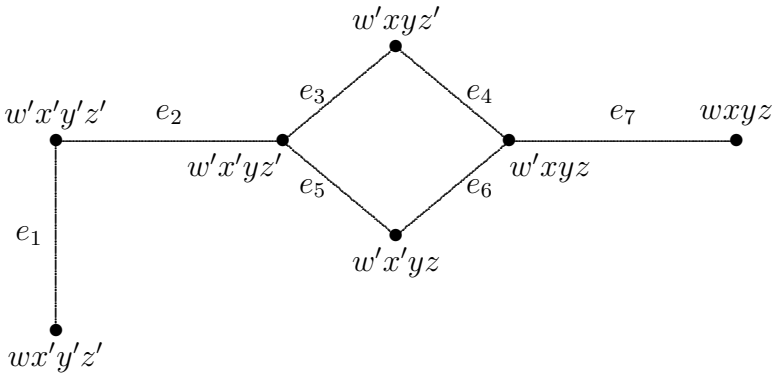
Cực tiểu hóa hàm Boole

Một phần quan trọng trong việc thiết kế các mạch số là cực tiểu hóa hàm Boole trước khi thiết kế nó. Giả sử ta muốn xây dựng mạch logic cho bởi hàm Boole bốn biến

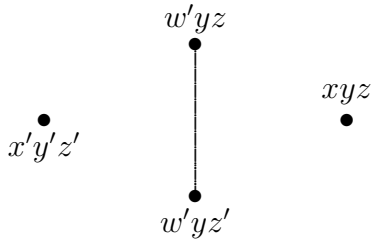
$$f(x,y,z,w) = w'x'y'z' + w'x'yz' + w'x'yz + w'xyz' + w'xyz + wxyz + wx'y'z'.$$

Chúng ta hãy biểu diễn mỗi một trong bảy thành phần của f bởi một đỉnh và một cạnh liên thuộc hai đỉnh nếu và chỉ nếu hai thành phần chỉ khác nhau đúng một biến. Đồ thị tương ứng hàm f cho trong Hình 2.13.

Một cạnh liên thuộc hai đỉnh tương ứng một thành phần với ba biến.



Hình 2.13:



Hình 2.14:

Một phủ tối thiểu của đồ thị cho ta một đơn giản hóa hàm Boole f, với cùng chức năng như f nhưng thiết kế sử dụng ít cổng logic hơn.

Các cạnh treo e_1 và e_7 cần thuộc mọi phủ của đồ thị. Bởi vậy các thành phần $x'y'z'$ và xyz là không thể khử được. Hai cạnh e_3 và e_6 (hoặc e_4 và e_5, hoặc e_3 và e_5) sẽ phủ các đỉnh còn lại. Do đó ta có thể viết lại

$$f = x'y'z' + xyz + w'yz' + w'yz.$$

Biểu thức này lại có thể biểu diễn bởi đồ thị trong Hình 2.14.

Các thành phần $x'y'z'$ và xyz không thể phủ bởi một cạnh do đó không thể cực tiểu thêm với chúng. Ngoài ra có một cạnh phủ hai đỉnh còn lại $w'yz$ và $w'yz'$. Do đó biểu thức Boole tối thiểu là

$$f = x'y'z' + xyz + w'y.$$

2.6 Nhân của đồ thị

2.6.1 Các định lý về tồn tại và duy nhất

Xét đồ thị $G := (V, \Gamma)$. Ta nói tập hợp $S \subset V$ là *nhân* của đồ thị G nếu S là tập ổn định trong và ngoài của G; tức là nếu

1. Nếu $v \in S$ thì $\Gamma(v) \cap S = \emptyset$; và
2. Nếu $v \notin S$ thì $\Gamma(v) \cap S \neq \emptyset$.

Điều kiện (1) suy ra rằng nhân S không có khuyên; điều kiện (2) suy ra nhân S chứa mọi đỉnh $v \in V$ có $\Gamma(v) = \emptyset$; ngoài ra, rõ ràng tập trống $\emptyset$ không phải là nhân.

Ví dụ 2.6.1. [Von Neumann-Morgenstern] Khái niệm nhân đầu tiên được đưa vào lý thuyết trò chơi với tên gọi là "lời giải".

Giả sử có n đấu thủ, ký hiệu là $(1), (2), \ldots, (n)$, họ có thể thỏa thuận với nhau để chọn tình thế v trong tập hợp V; nếu đấu thủ (i) ưng tình thế a hơn tình thế b thì ta viết $a \succeq b$. Hiển nhiên $\succeq$ là quan hệ tiền thứ tự toàn phần[1]. Vì những sự thích ứng của các cá nhân $\succeq$ có thể không phù hợp nhau nên ta phải đưa thêm khái niệm *thích ứng hiệu quả* $\succ$. Nếu tình huống a được thích ứng hiệu quả hơn tình huống b (ký hiệu $a \succ b$) thì nghĩa là: có một tập đấu thủ, do cho rằng a hơn b nên họ có thể cho quan điểm của mình thắng thế; nếu ngoài ra lại có $b \succ c$ thì có một tập các đấu thủ có khả năng làm cho tình huống b thắng thế; nhưng vì tập hợp thứ hai không bắt buộc phải trùng với tập hợp thứ nhất nên có thể không bắt buộc $a \succ c$: quan hệ $\succ$ không có tính bắc cầu[2]!

Bây giờ xét đồ thị $G = (V, \Gamma)$ trong đó $\Gamma(x)$ là tập các tình huống được thích ứng hiệu quả hơn x. Gọi S là nhân (nếu có) của đồ thị. Von Neumann và Morgenstern đề nghị chỉ hạn chế trò chơi với các phần tử của S thôi. Sự ổn định trong của S biểu thị

[1]Quan hệ $\succeq$ trên tập V là *tiền thứ tự toàn phần* nếu nó thỏa mãn: tính phản xạ, tính bắc cầu và hai phần tử bất kỳ $a, b \in V$ thì hoặc $a \succeq b$ hoặc $b \succeq a$.

[2]Thuật ngữ *thích ứng hiệu quả* do G. T. Guilbaud đưa ra trong [40]; Von Neumann và Morgenstern lại dùng danh từ *áp đảo*. Về các cách phát biểu toán học khác của sự thích ứng hiệu quả, xem [5].

rằng: không có một tình huống nào của S được thích ứng hiệu quả hơn một tình huống khác x; sự ổn định ngoài của S biểu thị rằng: mọi tình huống x không thuộc S không được thích ứng hiệu quả hơn một tình huống nào đó của S, do đó tình huống x lập tức được loại trừ.

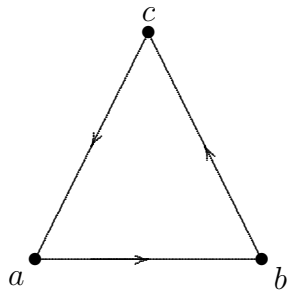
Ví dụ 2.6.2. Không phải mọi đồ thị đều có nhân. Nếu đồ thị có nhân S_0 thì các số ổn định của nó thỏa mãn điều kiện:

$$\alpha(G) = \max_{S \in \mathcal{S}} \#S \geq \#S_0 \geq \min_{T \in \mathcal{T}} = \beta(G).$$

Đồ thị trong Hình 2.15 không có nhân vì

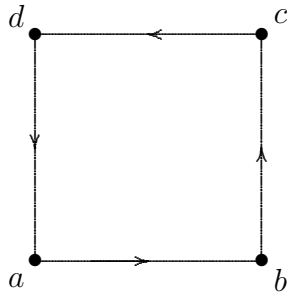
$$\alpha(G) = 1 < 2 = \beta(G).$$

Trái lại, đồ thị trong Hình 2.16 có hai nhân là $S' = \{b, d\}$ và $S'' = \{a, c\}$.



Hình 2.15:

Bây giờ ta đưa ra tiêu chuẩn để nhận biết một đồ thị có nhân hay không, và nhân có duy nhất không?



Hình 2.16:

Định lý 2.6.3. *Nếu S là nhân của đồ thị (V, Γ) thì S là tập hợp cực đại trong họ $\mathcal{S}$ các tập ổn định trong; tức là*

$$A \in \mathcal{S}, S \subset A \quad \Rightarrow \quad A = S.$$

Chứng minh. Thật vậy, giả sử ngược lại tồn tại tập ổn định trong A chứa nhân S và $A \neq S$. Khi đó tồn tại $a \in A$ sao cho $a \neq S$; từ đó suy ra $\Gamma(a) \cap S \neq \emptyset$. Vậy $\Gamma(a) \cap A \neq \emptyset$, mâu thuẫn với $A \in \mathcal{S}$. ◁

Định lý 2.6.4. *Trong đồ thị đối xứng không khuyên, mọi tập hợp cực đại của họ $\mathcal{S}$ các tập hợp ổn định trong đều là nhân*

Chứng minh. Giả sử S là tập hợp cực đại trong $\mathcal{S}$, ta phải chứng minh rằng mọi đỉnh $a \notin S$ đều thỏa mãn $\Gamma(a) \cap S \neq \emptyset$. Thật vậy, nếu $\Gamma(a) \cap S \neq \emptyset$ với đỉnh $a \in S$ thì tập $A = S \cup \{a\}$ ổn định trong (vì $a \notin \Gamma(a)$, đồng thời $S \subset A, A \neq S$, mâu thuẫn với giả thiết S là tập ổn định trong cực đại. ◁

Hệ quả 2.6.5. *Mọi đồ thị đối xứng không khuyên đều có nhân.*

Chứng minh. Thật vậy, lập đồ thị phụ $(\mathcal{S}, \bar{\Gamma})$ có các đỉnh là các tập ổn định trong của đồ thị đối xứng đã cho, và $S \in \bar{\Gamma}_{S'}$ nếu và chỉ nếu $S' \subset S$. Đồ thị phụ là cảm ứng, vậy theo Bổ đề Zorn, tồn tại đỉnh $S \in \mathcal{S}$ không có dòng dõi nên S là tập ổn định trong cực đại và theo Định lý 2.6.4 S là nhân của đồ thị đã cho. ◁

Trong đồ thị đối xứng không khuyên, quá trình tìm nhân là như sau:

Lấy một đỉnh bất kỳ v_0 và đặt $S_0 = \{v_0\}$; sau đó lấy một đỉnh $v_1 \notin \Gamma(S_0)$ và đặt $S_1 = S_0 \cup \{v_1\}$; tiếp theo lấy $v_2 \notin \Gamma(S_1)$ và đặt $S_2 = S_1 \cup \{v_2\}$; và vân vân. Vì đồ thị hữu hạn nên sớm hay muộn ta sẽ được $\Gamma(S_k) = V$, và vì S_k là tập hợp cực đại của $\mathcal{S}$ nên nó là nhân.

Định lý 2.6.6. *Điều kiện cần và đủ để tập S là nhân là hàm đặc trưng của nó*

$$\varphi_S(x) := \begin{cases} 1 & \text{nếu } x \in S, \\ 0 & \text{nếu ngược lại,} \end{cases}$$

thỏa mãn hệ thức

$$\varphi_S(x) = 1 - \max_{y\in\Gamma(x)} \varphi_S(y).$$

Nếu $\Gamma(x) = \emptyset$ *thì ta quy ước*

$$\max_{y\in\Gamma(x)} \varphi_S(y) = 0.$$

Chứng minh. Điều kiện cần. Giả sử S là nhân. Do tính ổn định trong nên $\varphi_S(x) = 1$ suy ra $x \in S$ và do đó

$$\max_{y\in\Gamma(x)} \varphi_S(y) = 0.$$

Mặt khác, do tính ổn định ngoài nên $\varphi_S(x) = 0$ suy ra $x \notin S$ và do đó

$$\max_{y\in\Gamma(x)} \varphi_S(y) = 1.$$

Từ đó suy ra hệ thức đã phát biểu.

Điều kiện đủ. Giả sử $\varphi_S(x)$ là hàm đặc trưng của một tập hợp S nào đó. Nếu hệ thức của định lý thỏa mãn thì ta có

1. $x \in S$ suy ra $\varphi_S(x) = 1$ nên $\max_{y\in\Gamma(x)} \varphi_S(y) = 0$. Vậy $\Gamma(x) \cap S = \emptyset$.
2. $x \notin S$ suy ra $\varphi_S(x) = 0$ nên $\max_{y\in\Gamma(x)} \varphi_S(y) = 1$. Vậy $\Gamma(x) \cap S \neq \emptyset$.

Do đó S là nhân. ◁

Định lý 2.6.7. [Richardson] *Nếu đồ thị hữu hạn không có mạch với độ dài lẻ, thì nó có nhân (nhưng không nhất thiết duy nhất).*

Chứng minh. Xem, chẳng hạn [4]. ◁

2.6.2 Trò chơi Nim

Giữa hai đấu thủ mà chúng ta ký hiệu là (A) và (B) có một đồ thị (V, Γ) cho phép xác định một trò chơi nào đó, trong trò chơi ấy mỗi thế là một đỉnh của đồ thị; đỉnh khởi

đầu v_0 được chọn bằng cách rút thăm, và các đấu thủ lần lượt đi: đầu tiên đấu thủ (A) chọn đỉnh v_1 trong tập hợp $\Gamma(v_0)$; sau đó (B) chọn đỉnh v_2 trong tập $\Gamma(v_1)$; tiếp theo (A) lại chọn đỉnh v_3 trong tập $\Gamma(v_2), \ldots$. Nếu một đấu thủ chọn được đỉnh v_k mà $\Gamma(v_k) = \emptyset$ thì ván đó kết thúc; đấu thủ nào chọn được đỉnh cuối cùng thì thắng cuộc và đấu thủ kia thua. Hiển nhiên do ta chỉ xét đồ thị hữu hạn nên cuộc chơi sẽ kết thúc sau một số hữu hạn bước.

Để kỷ niệm một trò chơi tiêu khiển quen thuộc mà Nim đã tổng quát hoá, ta gọi trò chơi vừa mô tả là *trò chơi Nim* và ký hiệu là (V, Γ) cũng như đồ thị xác định nó. Bài toán đặt ra là phát hiện các thế thắng; nghĩa là các đỉnh của đồ thị mà ta phải chọn để bảo đảm chắc thắng mặc dù đối phương chống trả ra sao. Kết quả chủ yếu như sau:

Định lý 2.6.8. *Nếu đồ thị có nhân S và nếu một đối thủ đã chọn một đỉnh trong nhân S, thì việc chọn này bảo đảm cho anh ta thắng hoặc hòa.*

Chứng minh. Thật vậy, nếu đấu thủ (A) chọn đỉnh $v_1 \in S$ thì hoặc $\Gamma(v_1) = \emptyset$ lúc đó anh ta thắng; hoặc đối phương buộc phải chọn một đỉnh v_2 trong $V \setminus S$, do đó đến lượt mình, đấu thủ (A) lại có thể chọn một đỉnh $v_3 \in S$ và cứ như thế mãi. Nếu đến một lúc nào đó, một trong các đấu thủ thắng bằng cách chọn một đỉnh v_k mà $\Gamma(v_k) = \emptyset$ thì ta có $v_k \in S$. Vậy đấu thủ thắng nhất định phải là (A). Ta có điều phải chứng minh. $\triangleleft$

Một phương pháp cơ bản để chơi tốt là tìm *hàm Grundy* (nếu nó tồn tại) (xem [4]) và từ đó suy ra nhân của đồ thị đã cho. Bạn đọc quan tâm về các trò chơi trên đồ thị có thể xem [4] (Chương 6).

Chương 3

Các bài toán về đường đi

Trong các ứng dụng thực tế, ta cần tìm đường đi (nếu có) giữa hai đỉnh của đồ thị. Đặc biệt, bài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị có ý nghĩa to lớn. Có thể dẫn về bài toán như vậy từ nhiều bài toán thực tế. Ví dụ, bài toán tìm hành trình tiết kiệm nhất (theo tiêu chuẩn khoảng cách, thời gian hoặc chi phí) trên một bản đồ giao thông; bài toán chọn phương pháp tiết kiệm nhất để đưa một hệ động lực từ trạng thái này sang trạng thái khác v.v... Hiện nay có rất nhiều phương pháp dựa trên lý thuyết đồ thị tỏ ra là các phương pháp có hiệu quả nhất.

Chương này trình bày các thuật toán tìm đường đi ngắn nhất trên đồ thị có trọng số.

3.1 Đường đi giữa hai đỉnh

3.1.1 Đường đi giữa hai đỉnh

Trong nhiều trường hợp, chúng ta cần trả lời câu hỏi: *Tồn tại đường đi μ từ đỉnh s đến đỉnh t của đồ thị có hướng $G := (V, E)$? Nếu có, hãy chỉ ra cách đi của đường đi μ.*

Lời giải của bài toán này khá đơn giản: chúng ta chỉ cần áp dụng thuật toán tìm kiếm theo chiều rộng (hoặc chiều sâu) trên đồ thị có hướng G như sau. Gán mỗi đỉnh của G một chỉ số. Bằng phương pháp lặp, dần dần ta sẽ cho mỗi đỉnh v một chỉ số nào đó bằng độ dài đường đi ngắn nhất (số cung ít nhất) từ s tới v. Đánh dấu đỉnh s bằng chỉ số 0. Nếu các đỉnh được đánh dấu bằng chỉ số m lập thành một tập hợp $P(m)$ đã biết, thì ta đánh dấu chỉ số $(m+1)$ cho mọi đỉnh của tập hợp:

$$P(m+1) := \{v_j \text{ chưa được đánh dấu} \mid \text{tồn tại } v_i \in P(m) \text{ với } (v_i, v_j) \in E\}.$$

Thuật toán dừng khi không thể đánh dấu được nữa. Có hai trường hợp xảy ra:

1. Đỉnh t được đánh dấu, chẳng hạn $t \in P(m)$ với m nào đó, thì ta xét các đỉnh $v_1, v_2, ...,$ sao cho

   $$v_1 \in P(m-1), v_2 \in P(m-2), \ldots, v_m \in P(0).$$

   Khi đó $\mu := \{s = v_m, v_{m-1}, ..., v_1, t\}$ là đường đi phải tìm.

2. Đỉnh t không được đánh dấu. Trong trường hợp này, ta kết luận không tồn tại đường đi từ s đến t.

Theo cách xây dựng của thuật toán, dễ dàng chứng minh rằng

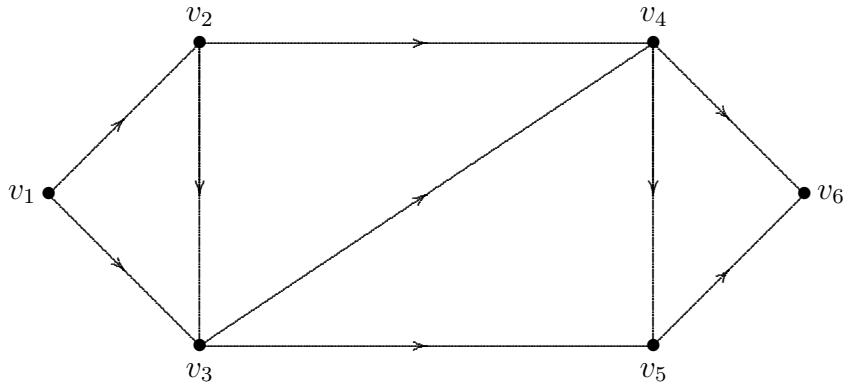
Mệnh đề 3.1.1. *Nếu đồ thị được xác định bởi dãy liên tiếp các đỉnh, thì thuật toán có thời gian $O(m)$.*

Ví dụ 3.1.2. Xét đồ thị G trong Hình 3.1. Giả sử $s = v_1$ và $t = v_6$. Áp dụng thuật toán trên ta có các đỉnh $v_1, v_2, \ldots, v_6$ được đánh dấu theo thứ tự tương ứng là $0, 1, 1, 2, 2, 3$. Suy ra tồn tại đường đi từ đỉnh v_1 đến đỉnh v_6, chẳng hạn

$$v_1, v_2, v_4, v_6.$$

3.1.2 Đồ thị liên thông mạnh

Nhắc lại là đồ thị có hướng G gọi là *liên thông mạnh* nếu hai đỉnh s và t tùy ý của G luôn luôn tồn tại một đường đi từ s đến t. Hiển nhiên rằng



Hình 3.1:

Định lý 3.1.3. *Cho $G = (V, E)$ là đồ thị có hướng, và $v \in V$. Khi đó G liên thông mạnh nếu và chỉ nếu mọi cặp đỉnh $a, b \in V$, tồn tại một đường đi từ a đến v và một đường đi từ v đến b.*

Dựa trên thuật toán tìm kiếm theo chiều sâu, ta có thể mô tả cách xác định một đồ thị có hướng có liên thông mạnh hay không thông qua định lý sau:

Định lý 3.1.4. *Cho $G = (V, E)$ là đồ thị có hướng, và $v \in V$. Ký hiệu $G' := (V, E')$ là đồ thị có hướng nhận được từ G bằng cách đảo hướng mỗi cung trong E. Khi đó G là liên thông mạnh nếu và chỉ nếu thuật toán tìm kiếm theo chiều sâu trên đồ thị có hướng G, khởi đầu từ v, đạt được mọi đỉnh của G và thuật toán tìm kiếm theo chiều sâu trên đồ thị có hướng G', khởi đầu từ v, đạt được mọi đỉnh của G'.*

Định nghĩa 3.1.5. Đồ thị vô hướng G gọi là *được định hướng mạnh* nếu có thể định hướng trên các cạnh của G sao cho đồ thị có hướng tương ứng nhận được là liên thông mạnh.

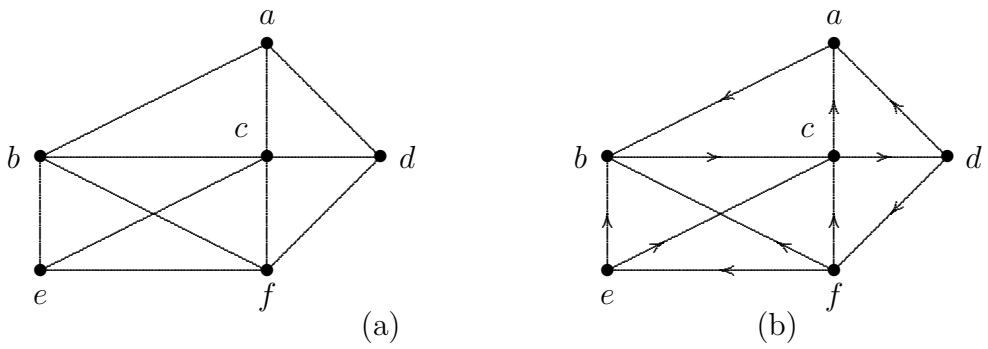
Định lý sau cho chúng ta một đặc trưng của đồ thị vô hướng được định hướng mạnh. Ta nói *cầu* trong đồ thị vô hướng liên thông là một cạnh mà bỏ đi thì đồ thị sẽ mất tính liên thông.

Định lý 3.1.6. *Đồ thị vô hướng G được định hướng mạnh nếu và chỉ nếu nó liên thông và không có cầu.*

Chứng minh. Bài tập. ◁

Dựa trên thuật toán tìm kiếm theo chiều sâu, ta có thể định hướng các cạnh của đồ thị vô hướng được định hướng mạnh như sau. Lấy một đỉnh bất kỳ trong đồ thị vô hướng G làm đỉnh khởi đầu và thực hiện thuật toán tìm kiếm theo chiều sâu. Vì đồ thị vô hướng là liên thông, kết quả của việc tìm kiếm này cho ta một *cây bao trùm*[1] $T = (V_n, E_n)$, trong đó $V_n = \{v_1, v_2, \ldots, v_n\}$ là các đỉnh của G. Nếu $e = (v_i, v_j)$ là một cạnh trong cây bao trùm T, ta định hướng nó từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn. Tức là nếu $i < j$, định hướng cạnh e từ v_i đến v_j, và nếu $j < i$ thì định hướng cạnh e từ v_j đến v_i. Nếu cạnh $e = (v_i, v_j)$ của G không thuộc cây T, thì ta định hướng cạnh này từ đỉnh có chỉ số lớn hơn đến đỉnh có chỉ số nhỏ hơn. Tức là nếu $i > j$, định hướng cạnh e từ v_i đến v_j, và nếu $j > i$ thì định hướng cạnh e từ v_j đến v_i.

Hình 3.2 minh họa đồ thị vô hướng và cách định hướng nó.



Hình 3.2: (a) Đồ thị vô hướng G. (b) Đồ thị G được định hướng.

3.2 Đường đi ngắn nhất giữa hai đỉnh

Cho đồ thị có hướng $G = (V, E)$ có trọng số: mỗi cung $e \in E$ tương ứng một số thực $w(e)$. Bài toán đặt ra là tìm đường đi μ từ một đỉnh xuất phát $s \in V$ đến một đỉnh cuối là $t \in V$ sao cho tổng các trọng lượng trên đường đi μ :

$$\sum_{e_k \in \mu} w(e_k)$$

là nhỏ nhất.

[1]Khái niệm này sẽ được trình bày trong Phần 4.5.

Giả sử $V := \{v_1, v_2, \ldots, v_n\}$. Ta chỉ cần xét G là đơn đồ thị. Xét *ma trận trọng lượng* $W := (w_{ij})_{i,j=1,2,\ldots,n}$ định nghĩa như sau: với mỗi cặp đỉnh v_i và $v_j, i \neq j$, nếu tồn tại cung e có gốc là v_i và ngọn là v_j thì đặt $w_{ij} := w(e)$; ngược lại, đặt $w_{ij} := +\infty$; cuối cùng, đặt $w_{ii} = 0$.

Các phần tử $w_{ij}, i, j = 1, ..., n$, của ma trận trọng lượng có thể dương, âm hoặc bằng không. Một điều kiện duy nhất đặt ra là đồ thị có hướng G không chứa các mạch μ với tổng trọng lượng trên mạch μ âm. Vì rằng, nếu có một mạch μ như vậy và v là một đỉnh nào đó của nó, thì xuất phát từ nút s ta đi đến đỉnh v, và sau đó đi vòng quanh mạch μ một số đủ lớn lần rồi mới đến t ta sẽ thu được một đường đi có trọng lượng đủ nhỏ. Vì vậy, trong trường hợp này, đường đi ngắn nhất là không tồn tại.

Nhận xét rằng, nếu ma trận trọng lượng W thỏa mãn

$$w_{ij} := \begin{cases} 1 & \text{nếu} \quad (v_i, v_j) \in E, \\ 0 & \text{nếu ngược lại,} \end{cases}$$

thì đường đi ngắn nhất từ s đến t được xác định bằng thuật toán tìm kiếm theo chiều rộng như đã trình bày trong phần trước.

Trước tiên chúng ta xét thuật toán Dijkstra, đơn giản và rất hiệu quả, để giải bài toán đặt ra trong trường hợp ma trận trọng lượng W có các phần tử không âm. Sau đó phát triển nó để giải quyết bài toán trong trường hợp tổng quát.

3.2.1 Trường hợp ma trận trọng lượng không âm

Thuật toán Dijkstra [20] tìm đường đi ngắn nhất từ đỉnh s đến đỉnh t trong đồ thị có hướng có trọng lượng không âm. Phương pháp này dựa trên việc gán các *nhãn tạm thời* cho các đỉnh: *Nhãn* của đỉnh v_i, ký hiệu $L(v_i)$, là một cận trên của độ dài đường đi ngắn nhất từ s đến v_i. Các nhãn này sau đó tiếp tục được giảm bớt bởi một thủ tục lặp và tại mỗi bước lặp có đúng một nhãn tạm thời trở thành *nhãn cố định.* Khi đỉnh v_i được gán nhãn cố định thì không thể giảm $L(v_i)$; số này chính là độ dài đường đi ngắn nhất từ s đến v_i.

Thuật toán Dijkstra ($w_{ij} \geq 0$)

1. [Khởi tạo] Đặt
$$L(v_i) := \begin{cases} 0 & \text{nếu } v_i = s, \\ +\infty & \text{nếu ngược lại.} \end{cases}$$
với mọi $v_i \in V$. Đánh dấu đỉnh s có nhãn cố định, các đỉnh khác có nhãn tạm thời. Đặt $c = s$.

2. [Cập nhật] Với mọi $v_i \in \Gamma(c)$ sao cho v_i có nhãn tạm thời đặt
$$L(v_i) := \min\{L(v_i), L(c) + w(c, v_i)\};$$

3. Tìm đỉnh v_{i^*} có nhãn tạm thời sao cho
$$L(v_{i^*}) := \min\{L(v_i) < +\infty \mid v_i \text{ có nhãn tạm thời}\}.$$

4. Đánh dấu đỉnh v_{i^*} có nhãn cố định và đặt $c := v_{i^*}$.

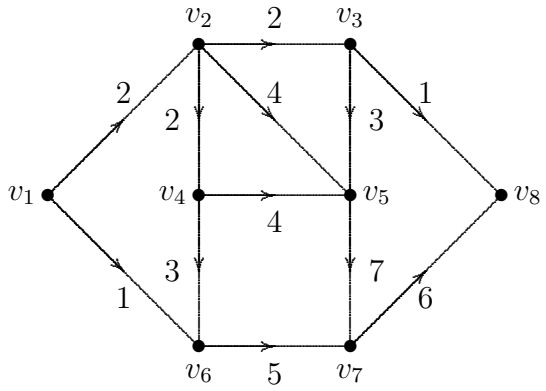
5. (a) *(Nếu tìm đường đi từ s đến t)* Nếu $v_i = t$ thì thuật toán dừng, $L(t)$ là đường đi ngắn nhất từ s đến t; ngược lại, chuyển sang Bước 2.

   (b) *(Nếu tìm tất cả các đường đi xuất phát từ s)* Nếu không thể gán nhãn cố định cho đỉnh v_{i^*} trong Bước 4 thì thuật toán dừng; giá trị $L(v_i)$ của đỉnh v_i có nhãn cố định cho ta độ dài đường đi ngắn nhất từ s đến v_i. Ngược lại chuyển sang Bước 2.

Ví dụ 3.2.1. Xét đồ thị có hướng có trọng số $G = (V, E)$ cho trong Hình 3.3. Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh $s := v_1$ đến đỉnh $t := v_8$ ta có đường đi ngắn nhất từ v_1 đến v_8 là
$$(v_1, v_2, v_3, v_8)$$
với độ dài là 5.

Định lý 3.2.2. *Thuật toán Dijkstra cho ta đường đi ngắn nhất từ s đến t (nếu có).*

Chứng minh. Trước hết chú ý rằng các đỉnh không được gán nhãn cố định sẽ không tồn tại đường đi từ s đến nó (những đỉnh này có nhãn L bằng ∞).



Hình 3.3:

Giả sử $L(v_i)$ của các đỉnh có nhãn cố định ở bước nào đó là độ dài đường đi ngắn nhất từ s đến v_i. Ký hiệu S_1 là tập tất cả các đỉnh này và S_2 là tập các đỉnh có nhãn tạm thời. Cuối Bước 2 của mỗi lần lặp, nhãn tạm thời $L(v_i)$ là độ dài đường đi ngắn nhất μ_i từ s đến v_i qua các đỉnh trong tập S_1. (Vì trong mỗi lần lặp chỉ có một đỉnh được đưa vào S_1 nên cập nhật lại $L(v_i)$ chỉ xảy ra trong Bước 2).

Xét đường đi ngắn nhất từ s đến v_{i^*} không hoàn toàn đi qua các đỉnh của S_1 mà chứa ít nhất một đỉnh thuộc S_2 và giả sử $v_j \in S_2$ là đỉnh đầu tiên như vậy trên đường đi này. Do w_{ij} không âm, đoạn đường đi từ v_j đến v_{i^*} phải có độ dài Δ không âm sao cho $L(v_j) < L(v_{i^*}) - \Delta < L(v_{i^*})$. Tuy nhiên điều này mâu thuẫn với khẳng định rằng $L(v_{i^*})$ có nhãn tạm thời nhỏ nhất, và do đó các đỉnh trên đường đi ngắn nhất đến v_{i^*} thuộc S_1 và do đó $L(v_{i^*})$ là độ dài của đường đi này.

Vì S_1 khởi tạo là $\{s\}$ và trong mỗi bước lặp, đỉnh v_{i^*} được thêm vào S_1, nên bằng qui nạp, $L(v_i)$ là độ dài đường đi ngắn nhất từ s đến v_i với mọi đỉnh $v_i \in S_1$. Do đó thuật toán cho lời giải tối ưu. ◁

Mệnh đề 3.2.3. *Thuật toán Dijkstra đòi hỏi thời gian $O(n^2)$. Nếu đồ thị thưa và được xác định bởi dãy liên tiếp các đỉnh, thì thời gian lớn nhất để thực hiện thuật toán là $O(m \log n)$.*

Chứng minh. Trong trường hợp đồ thị liên thông mạnh đầy đủ n đỉnh và cần tìm đường đi ngắn nhất từ s đến mọi đỉnh khác, thuật toán cần $n(n-1)/2$ phép cộng và so sánh trong Bước 2 và $n(n-1)/2$ phép so sánh khác trong Bước 3. Ngoài ra, các Bước 2 và 3 cần xác định các đỉnh được gán nhãn tạm thời nên cần thêm $n(n-1)/2$

phép so sánh. Các số này cũng là cận trên cho số các phép toán cần thiết để tìm đường đi ngắn nhất từ s đến t, và thật vậy, các giá trị này đạt được khi t là đỉnh cuối cùng được gán nhãn cố định. ◁

Khi Thuật toán Dijkstra kết thúc, các đường đi ngắn nhất được xác định bằng đệ qui theo Phương trình (3.1) dưới đây. Do đó nếu v_i' là đỉnh trước đỉnh v_i trong đường đi ngắn nhất từ s đến v_i thì với mọi đỉnh v_i cho trước, đỉnh v_i' có thể lấy là đỉnh mà

$$L(v_i) = L(v_i') + w(v_i', v_i). \tag{3.1}$$

Nếu tồn tại duy nhất đường đi ngắn nhất từ s đến v_i thì các cung (v_i', v_i) trên đường đi ngắn nhất tạo thành một cây có hướng (xem Chương 4) với gốc là đỉnh s. Nếu có nhiều hơn một đường đi ngắn nhất từ s đến bất kỳ một đỉnh khác, thì Phương trình 3.1 sẽ được thỏa mãn bởi hơn một đỉnh khác v_i' với v_i nào đó.

Thủ tục sau minh họa thuật toán Dijkstra. Trong thủ tục này, mảng $Mark[]$ được sử dụng để đánh dấu các đỉnh có nhãn tạm thời hay cố định: $Mark[i] =$ FALSE nếu đỉnh v_i có nhãn tạm thời; ngược lại bằng TRUE. Giá trị $Label[i]$ tương ứng nhãn $L(v_i)$ và $Pred[i]$, sau khi kết thúc thuật toán, là đỉnh liền trước đỉnh v_i trong đường đi ngắn nhất từ s đến v_i. Nếu tồn tại đường đi ngắn nhất từ s đến t, thì thủ tục PathTwoVertex() sẽ in ra thông tin của đường đi này.

```
void Dijkstra(byte Start, byte Terminal)
{
  byte i, Current;
  AdjPointer Tempt;
  Path Pred;
  int Label[MAXVERTICES], NewLabel, Min;
  Boolean Mark[MAXVERTICES];

  for(i = 1; i <= NumVertices; i++)
  {
    Mark[i] = FALSE;
```

```
  Pred[i] = 0;
  Label[i] = +Infty;
}

Mark[Start] = TRUE;
Pred[Start] = 0;
Label[Start] = 0;
Current = Start;

while (Mark[Terminal] == FALSE)
{
  Tempt = V_out[Current]->Next;
  while (Tempt != NULL)
  {
    if (Mark[Tempt->Vertex] == FALSE)
    {
      NewLabel = Label[Current] + Tempt->Length;
      if (NewLabel < Label[Tempt->Vertex])
      {
        Label[Tempt->Vertex] = NewLabel;
        Pred[Tempt->Vertex] = Current;
      }
    }
    Tempt = Tempt->Next;
  }

  Min = +Infty;
  for (i = 1; i <= NumVertices; i++)
  if ((Mark[i] == FALSE) && (Label[i] < Min))
  {
    Min = Label[i];
    Current = i;
  }
```

```
    if (Min == +Infty)
    {
      printf("Khong ton tai duong di tu %d", Start);
      printf(" den %d", Terminal);
      return;
    }
    Mark[Current] = TRUE;
  }

  printf("Ton tai duong di tu %d", Start);
  printf(" den %d", Terminal);
  printf("\nDuong di qua cac dinh:");
  printf("\n % d " , Start);
  PathTwoVertex(Pred, Start, Terminal);
  printf("\nDo dai la: %d ", Label[Terminal]);
}
```

3.2.2 Trường hợp ma trận trọng lượng tùy ý

Thuật toán Dijkstra chỉ áp dụng trong trường hợp ma trận trọng lượng W không âm. Tuy nhiên, nhiều bài toán thực tế, W là ma trận chi phí, cho nên những cung mang lại lợi nhuận phải có chi phí âm. Trong trường hợp này, phương pháp cho dưới đây tìm đường đi ngắn nhất từ s đến tất cả các đỉnh khác. Đây cũng là phương pháp lặp và dựa trên cách đánh nhãn, trong đó cuối bước lặp thứ k các nhãn biểu diễn giá trị độ dài của các đường đi ngắn nhất (từ s đến tất cả các đỉnh khác) có số cung không vượt quá $(k+1)$. Thuật toán này đưa ra lần đầu tiên bởi Ford [26] vào giữa năm 1950. Sau đó được Moore [45] và Bellman [3] cải tiến như sau.

Thuật toán Ford, Moore, Bellman

Ký hiệu $L^k(v_i)$ là nhãn của đỉnh v_i ở cuối lần lặp thứ $(k+1)$.

1. [Khởi tạo] Đặt $S = \Gamma(s), k = 1$; và

$$L^1(v_i) := \begin{cases} 0 & \text{nếu} \quad v_i = s, \\ w(s, v_i) & \text{nếu} \quad v_i \neq s, v_i \in \Gamma(s), \\ +\infty & \text{nếu ngược lại.} \end{cases}$$

2. [Cập nhật nhãn] Với mỗi $v_i \in \Gamma(S), v_i \neq s$, thay đổi nhãn của nó theo quy tắc:

$$L^{k+1}(v_i) := \min[L^k(v_i), \min_{v_j \in T_i}\{L^k(v_j) + w(v_j, v_i)\}], \tag{3.2}$$

   trong đó $T_i := \Gamma^{-1}(v_i) \cap S$. Tập S chứa tất cả các đỉnh v_i sao cho đường đi ngắn nhất (hiện hành) từ s đến v_i có số cung là k.

   Tập T_i chứa tất cả các đỉnh v_j sao cho đường đi ngắn nhất từ s đến v_i có số cung là k (tức là $v_j \in S$) và kết thúc bằng cung (v_j, v_i). Chú ý rằng, nếu $v_i \notin \Gamma(S)$ thì đường đi ngắn nhất từ s đến v_i không thể có số cung là $(k+1)$ và ta không thay đổi nhãn của đỉnh này:

$$L^{k+1}(v_i) := L^k(v_i), \quad \text{với mọi } v_i \notin \Gamma(S).$$

3. [Kiểm tra kết thúc] (a) Nếu $k \leq n-1$ và $L^{k+1}(v_i) = L^k(v_i)$, với mọi $v_i \in V$, thì thuật toán dừng; nhãn của đỉnh v_i cho ta độ dài đường đi ngắn nhất từ s đến v_i.

   (b) Nếu $k < n-1$ và $L^{k+1}(v_i) \neq L^k(v_i)$, với v_i nào đó, thì chuyển sang Bước 4.

   (c) Nếu $k = n-1$ và $L^{k+1}(v_i) \neq L^k(v_i)$, với v_i nào đó, thì đồ thị G có mạch với độ dài âm. Thuật toán kết thúc.

4. [Cập nhật S] Đặt

$$S := \{v_i \mid L^{k+1}(v_i) \neq L^k(v_i)\}.$$

   (Tập S bây giờ chứa tất cả các đỉnh mà đường đi ngắn nhất từ s đến nó có số cung là $(k+1)$).

5. Thay k bởi $(k+1)$ và lặp lại Bước 2.

Khi thuật toán kết thúc và đồ thị không có mạch độ dài âm, chúng ta có thể tìm tất cả các đường đi (nếu tồn tại) ngắn nhất từ s đến tất cả các đỉnh khác. Hơn nữa các đường đi có thể nhận được trực tiếp nếu, trong phép cộng vào các nhãn $L^k(v_i)$ ở

Phương trình 3.2, ta thêm một nhãn $P^k(v_i)$ lưu trữ thông tin mỗi đỉnh trong suốt quá trình tính toán, trong đó $P^k(v_i)$ là đỉnh kề trước đỉnh v_i trên đường đi ngắn nhất từ s đến v_i ở bước lặp thứ k. Ta có thể khởi tạo

$$P^1(v_i) := \begin{cases} s & \text{nếu} \quad v_i \in \Gamma(s), \\ 0 & \text{nếu ngược lại.} \end{cases}$$

Nhãn $P^k(v_i)$ được cập nhật theo Phương trình 3.2 trong Bước 2 sao cho $P^{k+1}(v_i) = P^k(v_i)$ nếu giá trị nhỏ nhất đạt được ở phần tử đầu tiên trong dấu ngoặc của Phương trình 3.2; hoặc $P^{k+1}(v_i) = v_j$ nếu ngược lại. Nếu ký hiệu $\mathbf{P}(v_i)$ là vector được xây dựng từ các nhãn P khi kết thúc thuật toán, thì đường đi ngắn nhất từ s đến v_i nhận được theo thứ tự ngược:

$$s, \ldots, P^3(v_i), P^2(v_i), P(v_i), v_i,$$

trong đó $P^2(v_i)$ có nghĩa là $P(P(v_i)), \ldots$.

Đoạn chương trình sau minh họa thuật toán đã trình bày.

```
void Ford_Moore_Bellman(byte Start)
{
  byte i, k, Terminal;
  AdjPointer Tempt1, Tempt2;
  Path OldPred, NewPred;
  int OldLabel[MAXVERTICES], NewLabel[MAXVERTICES], Min;
  Boolean Done, Mark[MAXVERTICES];

  for (i = 1; i <= NumVertices; i++)
  {
    Mark[i] = FALSE;
    OldPred[i] = 0;
    OldLabel[i] = +Infty;
  }

  OldLabel[Start] = 0;
  Tempt1 = V_out[Start]->Next;
  while (Tempt1 != NULL)
```

```
{
  Mark[Tempt1->Vertex] = TRUE;
  if (Tempt1->Vertex != Start)
  {
    OldPred[Tempt1->Vertex] = Start;
    OldLabel[Tempt1->Vertex] = Tempt1->Length;
  }
  Tempt1 = Tempt1->Next;
}

for (i = 1; i <= NumVertices; i++)
{
  NewPred[i] = OldPred[i];
  NewLabel[i] = OldLabel[i];
}

for (k = 1; k < NumVertices; k++)
{
  printf("\n ");
  for (i = 1; i <= NumVertices; i++)
  {
    if (Mark[i] == TRUE)
    {
      Tempt1 = V_out[i]->Next;
      while (Tempt1 != NULL)
      {
        NewPred[Tempt1->Vertex] = Tempt1->Vertex;
        Min = OldLabel[Tempt1->Vertex];
        Tempt2 = V_in[Tempt1->Vertex]->Next;
        while (Tempt2 != NULL)
        {
          if (Mark[Tempt2->Vertex] == TRUE)
          {
            if ((OldLabel[Tempt2->Vertex] + Tempt2->Length) < Min)
```

```
          {
            NewPred[Tempt1->Vertex] = Tempt2->Vertex;
            Min = OldLabel[Tempt2->Vertex] + Tempt2->Length;
          }
        }
        Tempt2 = Tempt2->Next;
      }
      NewLabel[Tempt1->Vertex] = Min;

      Tempt1 = Tempt1->Next;
    }
  }
}

Done = TRUE;
for (i = 1; i <= NumVertices; i++)
{
  if (OldLabel[i] != NewLabel[i])
  {
    Done = FALSE;
    break;
  }
}

if (Done == TRUE)
{
  for (Terminal = 1; Terminal <= NumVertices; Terminal++)
  if (OldLabel[Terminal] < +Infty)
  {
    printf("\n ");
    printf("Ton tai duong di tu %d", Start);
    printf(" den %d", Terminal);
    printf("\n Duong di qua cac dinh:");
    printf(" % d " , Start);
```

```
        PathTwoVertex(OldPred, Start, Terminal);
        i = Terminal;
        printf(" Do dai la: %d ", OldLabel[Terminal]);
        printf("\n ");
        getch();
      }
      else
      {
        printf("\n ");
        printf("Khong ton tai duong di tu %d", Start);
        printf(" den %d", Terminal);
      }
      return;
    }

    for (i = 1; i <= NumVertices; i++)
    if (OldLabel[i] != NewLabel[i])
    {
      Mark[i] = TRUE;
      OldPred[i] = NewPred[i];
      OldLabel[i] = NewLabel[i];
    }
    else Mark[i] = FALSE;
  }
  printf("\n Ton tai mach co do dai am");
}
```

Để chứng minh thuật toán tìm được cho lời giải tối ưu của bài toán cần sử dụng nguyên lý tối ưu của quy hoạch động và từ nhận xét là nếu không có đường đi tối ưu qua k cung thì sẽ không có đường đi tối ưu qua $(k+1)$ cung. Chúng ta không trình bày chứng minh đó; bạn đọc quan tâm có thể xem [6].

Đồ thị có hướng có trọng số âm nhưng không có mạch độ dài âm

Trong trường hợp đồ thị có hướng có trọng số tùy ý nhưng không có mạch có độ dài âm thì *thuật toán Moore-Bellman-d'Usopo* tìm đường đi ngắn nhất từ s đến t trình bày dưới đây sẽ hiệu quả hơn.

Nhãn của mọi đỉnh $v_i \in V$ là cặp $(P(v_i), L(v_i))$. Kết thúc thuật toán, $L(t)$ là độ dài của đường đi ngắn nhất từ s đến t và vector $\mathbf{P}$ cho ta thông tin của đường đi này.

1. [Khởi tạo] Đặt
$$L(v_i) := \begin{cases} +\infty & \text{nếu} \quad v_i \neq s, \\ 0 & \text{nếu} \quad v_i = s, \end{cases}$$
và
$$P(v_i) := \begin{cases} -1 & \text{nếu} \quad v_i \neq s, \\ 0 & \text{nếu} \quad v_i = s. \end{cases}$$
Khởi tạo Stack chỉ có một phần tử là s.

2. [Bước lặp] Trong khi Stack khác NULL

{

lấy ra phần tử v_i *ở đầu* Stack;

với mọi đỉnh $v_j \in V$ sao cho $v_j \in \Gamma(v_i)$,

{

nếu $[L(v_i) + w(v_i, v_j) < L(v_j)]$ thì gán

{

$L(v_j) = L(v_i) + w(v_i, v_j)$;

$P(v_j) = v_i$;

nếu đỉnh v_j chưa bao giờ ở trong Stack thì chèn v_j *vào cuối* Stack;

ngược lại, nếu v_j đã từng ở trong Stack, nhưng hiện tại không có trong đó thì chèn v_j *vào đầu* của Stack.

}

}

}

3.3 Đường đi ngắn nhất giữa tất cả các cặp đỉnh

Giả sử cần phải tìm đường đi ngắn nhất giữa hai đỉnh tùy ý. Rõ ràng ta có thể giải bài toán này bằng cách sử dụng n lần thuật toán mô tả ở phần trước, mà trong mỗi lần thực hiện, ta sẽ chọn s lần lượt là các đỉnh của đồ thị. Trong trường hợp đồ thị đầy đủ có ma trận trọng lượng không âm, số phép tính cần thiết tỉ lệ với n^3; trái lại với ma trận trọng lượng tổng quát, số phép tính tỉ lệ với n^4.

Trong phần này chúng ta sẽ trình bày hai thuật toán để giải bài toán tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh. Hai phương pháp này có thể áp dụng cho các ma trận trọng lượng tổng quát và chỉ đòi hỏi số phép tính tỉ lệ với n^3. Với trường hợp ma trận trọng lượng không âm, nói chung, các phương pháp này sẽ nhanh hơn 50% cách áp dụng thuật toán Dijkstra n lần.

3.3.1 Thuật toán Hedetniemi (trường hợp ma trận trọng lượng không âm)

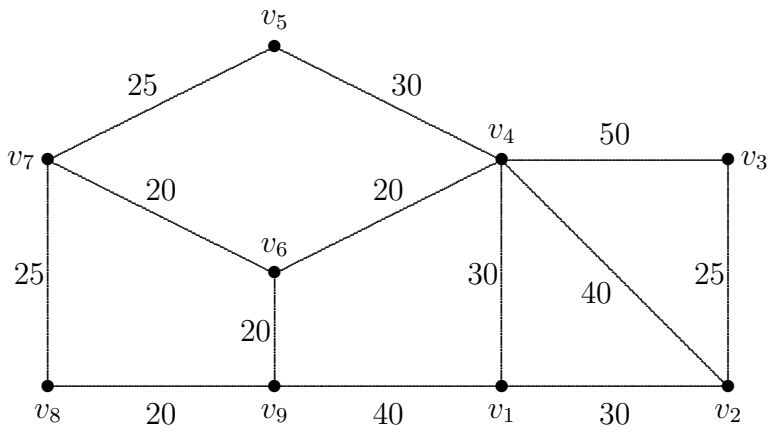
Một trong những mục đích của các thuật toán tìm đường đi ngắn nhất là xác định tất cả các độ dài của các đường đi ngắn nhất có thể có trong một đồ thị có trọng lượng tại cùng một thời điểm.

Thuật toán được xây dựng trên cơ sở một cách tính mới lũy thừa của các ma trận, mà chúng ta sẽ gọi là *tổng ma trận Hedetniemi* (xem [2]).

Xét đồ thị có hướng có trọng lượng $G = (V, \Gamma)$ với tập các đỉnh $V = \{v_1, v_2, \ldots, v_n\}$ và ma trận trọng lượng không âm $W := [w_{ij}]$ cấp $n \times n$.

Chẳng hạn, đồ thị trong Hình 3.4 có ma trận trọng lượng

$$W = \begin{pmatrix} 0 & 30 & \infty & 30 & \infty & \infty & \infty & \infty & 40 \\ 30 & 0 & 25 & 40 & \infty & \infty & \infty & \infty & \infty \\ \infty & 25 & 0 & 50 & \infty & \infty & \infty & \infty & \infty \\ 30 & 40 & 50 & 0 & 30 & 20 & \infty & \infty & \infty \\ \infty & \infty & \infty & 30 & 0 & \infty & 25 & \infty & \infty \\ \infty & \infty & \infty & 20 & \infty & 0 & 20 & \infty & 20 \\ \infty & \infty & \infty & \infty & 25 & 20 & 0 & 25 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 25 & 0 & 20 \\ 40 & \infty & \infty & \infty & \infty & 20 & \infty & 20 & 0 \end{pmatrix}.$$



Hình 3.4:

Chúng ta giới thiệu một phép toán mới trên các ma trận gọi là tổng ma trận Hedetniemi.

Định nghĩa 3.3.1. Giả sử A là ma trận cấp $m \times n$ và B là ma trận cấp $n \times p$. *Tổng ma trận Hedetniemi* của hai ma trận A và B là ma trận C cấp $m \times p$, ký hiệu $A \oplus B$, xác định bởi:

$$c_{ij} := \min\{a_{i1} + b_{1j}, a_{i2} + b_{2j}, \ldots, a_{in} + b_{nj}\}.$$

Ví dụ 3.3.2. Tìm tổng ma trận $A \oplus B$ nếu $A = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 5 & 6 & 0 \end{pmatrix}$ và $B = \begin{pmatrix} 0 & 3 & 4 \\ 5 & 0 & 4 \\ 3 & 1 & 0 \end{pmatrix}$.

Ta có

$$A \oplus B = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 5 & 6 & 0 \end{pmatrix} \oplus \begin{pmatrix} 0 & 3 & 4 \\ 5 & 0 & 4 \\ 3 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 3 & 1 & 0 \end{pmatrix}.$$

Chẳng hạn, phần tử c_{23} xác định bởi

$$c_{23} = \min\{2+4, 0+4, 3+0\} = 3.$$

Ví dụ 3.3.3. Tìm tổng ma trận $A \oplus B$ nếu $A = \begin{pmatrix} 0 & 1 & \infty \\ 1 & 0 & 4 \\ \infty & 4 & 0 \end{pmatrix}$ và $B = \begin{pmatrix} 1 & 0 & \infty \\ 1 & 0 & 4 \\ \infty & 4 & 0 \end{pmatrix}$.

Ta có

$$A \oplus B = \begin{pmatrix} 0 & 1 & \infty \\ 1 & 0 & 4 \\ \infty & 4 & 0 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & \infty \\ 1 & 0 & 4 \\ \infty & 4 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 5 \\ 1 & 0 & 4 \\ 5 & 4 & 0 \end{pmatrix}.$$

Chẳng hạn, phần tử c_{13} xác định bởi

$$c_{23} = \min\{0+\infty, 1+4, \infty+0\} = 5.$$

Bây giờ áp dụng tổng ma trận Hedetniemi vào việc tìm đường đi ngắn nhất. Xét ví dụ trong Hình 3.4. Đặt

$$\begin{aligned} W^1 &:= W, \\ W^k &:= W^{k-1} \oplus W \quad \text{nếu} \quad k \geq 2. \end{aligned}$$

Khi đó

$$W^2 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & \infty & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & \infty & \infty & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & \infty & \infty & \infty \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & \infty & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & \infty \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ \infty & \infty & \infty & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & \infty & \infty & \infty & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & \infty & 40 & \infty & 20 & 40 & 20 & 0 \end{pmatrix}.$$

Chúng ta hãy xét cách xác định một phần tử của ma trận $W^2 = [w_{ij}^{(2)}]$:

$$\begin{aligned} w_{13}^{(2)} &= \min\{0+\infty, 30+25, \infty+0, 30+50, \infty+\infty, \infty+\infty, \infty+\infty, \infty+\infty, 40+\infty\} \\ &= 55. \end{aligned}$$

Chú ý rằng giá trị 55 là tổng của 30, độ dài của đường đi ngắn nhất với số cung một từ đỉnh v_1 đến đỉnh v_2, và của 25, độ dài của cung nối đỉnh v_2 và đỉnh v_3. Do đó $w_{13}^{(2)}$ là độ dài của đường đi ngắn nhất từ v_1 đến v_2 với số cung nhiều nhất hai. Suy ra W^2 cho ta thông tin của tất cả các độ dài đường đi ngắn nhất giữa hai đỉnh có số cung nhiều nhất hai.

Tương tự, W^3 cho ta thông tin của tất cả các độ dài đường đi ngắn nhất giữa hai đỉnh có số cung nhiều nhất ba, và vân vân. Do đồ thị có n đỉnh nên có nhiều nhất $(n-1)$ cung trên đường đi ngắn nhất giữa hai đỉnh. Vậy

Định lý 3.3.4. *Trong đồ thị có trọng số không âm n đỉnh, phần tử hàng i cột j của ma trận Hedetniemi W^{n-1} là độ dài của đường đi ngắn nhất giữa đỉnh v_i và v_j.*

Với đồ thị trong Hình 3.4 có chín đỉnh, ta có

$$W^8 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & 70 & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & 80 & 90 & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & 90 & 110 & 90 \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & 60 & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & 65 \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ 70 & 80 & 90 & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & 90 & 110 & 60 & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & 90 & 40 & 65 & 20 & 40 & 20 & 0 \end{pmatrix}.$$

Do đó, đường đi ngắn nhất từ v_1 đến v_7 có độ dài 70.

Điều lý thú nhất trong ví dụ này là $W^4 = W^8$. Thật vậy, đẳng thức suy trực tiếp từ đồ thị trong Hình 3.4: mọi đường đi ngắn nhất đi qua nhiều nhất bốn cung. Bởi vậy độ dài của đường đi ngắn nhất được xác định bởi ma trận W^4 thay vì phải tính đến W^8. Tổng quát ta có

Định lý 3.3.5. *Trong đồ thị có trọng số không âm n đỉnh, nếu ma trận Hedetniemi* $W^k \neq W^{k-1}$, *còn* $W^k = W^{k+1}$, *thì* W^k *biểu thị tập các độ dài của các đường đi ngắn nhất, và số cung trên mỗi đường đi ngắn nhất không vượt quá k.*

Do đó, thuật toán này có thể dừng ở bước lặp thứ $k < (n-1)$. Dưới đây là đoạn chương trình minh họa tính ma trận luỹ thừa của ma trận trọng lượng W.

```
void Hetdetniemi()
{
  int i, j, k, t;
  int Old[MAXVERTICES][MAXVERTICES], New[MAXVERTICES][MAXVERTICES];
  Boolean Flag;

  for (i = 1; i <= NumVertices; i++)
    for  (j = 1; j <= NumVertices; j++)  Old[i][j] = w[i][j];

  for (k = 1; k < NumVertices; k++)
  {
    for (i = 1; i <= NumVertices; i++)
      for (j = 1; j <= NumVertices; j++)
      {
        New[i][j] = Old[i][1] + w[1][j];
        for (t = 2; t <= NumVertices; t++)
          New[i][j] = min(New[i][j], Old[i][t] + w[t][j]);
      }

    Flag = TRUE;
    for (i = 1; i <= NumVertices; i++)
      for (j = 1; j <= NumVertices; j++)
      {
        if (New[i][j] != Old[i][j])
        {
          Flag = FALSE;
          Old[i][j] = New[i][j];
```

```
        }
      }

    if (Flag == TRUE) break;
  }
}
```

Xác định đường đi ngắn nhất

Để xác định đường đi ngắn nhất giữa tất cả các cặp đỉnh ta cần thông tin không những của ma trận cuối cùng mà còn của các ma trận trước đó. Chẳng hạn, với đồ thị trong Hình 3.4, ta có

$$W^3 = \begin{pmatrix} 0 & 30 & 55 & 30 & 60 & 50 & 70 & 60 & 40 \\ 30 & 0 & 25 & 40 & 70 & 60 & 80 & 90 & 70 \\ 55 & 25 & 0 & 50 & 80 & 70 & 90 & \infty & 90 \\ 30 & 40 & 50 & 0 & 30 & 20 & 40 & 60 & 40 \\ 60 & 70 & 80 & 30 & 0 & 45 & 25 & 50 & 65 \\ 50 & 60 & 70 & 20 & 45 & 0 & 20 & 40 & 20 \\ 70 & 80 & 90 & 40 & 25 & 20 & 0 & 25 & 40 \\ 60 & 90 & \infty & 60 & 50 & 40 & 25 & 0 & 20 \\ 40 & 70 & 90 & 40 & 65 & 20 & 40 & 20 & 0 \end{pmatrix}.$$

Bây giờ ta tìm đường đi ngắn nhất từ v_1 đến v_7 (độ dài 70). Ta có

$$w_{14}^{(4)} = w_{1k}^{(3)} \oplus w_{k7}$$

với k nào đó. Nhưng các phần tử $w_{1k}^{(3)}$ tạo thành vector hàng

$$(0, 30, 55, 30, 60, 50, 70, 60, 40)$$

và các phần tử w_{k7} tạo thành vector cột

$$(\infty, \infty, \infty, \infty, 25, 20, 0, 25, \infty).$$

Vì giá trị nhỏ nhất đạt được tại $k = 6$ ứng với $70 = 50 + 20$ (và $k = 7$) nên đường đi ngắn nhất từ v_1 đến v_7 sẽ đi qua nhiều nhất ba cung từ v_1 đến v_6 và kết thúc với cung độ dài 20 từ v_6 đến v_7. (Thật ra, do giá trị nhỏ nhất cũng đạt được tại $k = 7$ (ứng với

$70 + 0$) nên tồn tại đường đi ngắn nhất từ v_1 đến v_7 với tổng số cung đi qua không vượt quá ba).

Tiếp đến ta tìm cung trước khi đến đỉnh v_6. Chú ý rằng $w_{16}^{(4)} = 70 - 20 = 50$. Các phần tử w_{k6} tạo thành vector cột

$$(\infty, \infty, \infty, 20, \infty, 0, 20, \infty, 20).$$

Lần này giá trị nhỏ nhất đạt tại $k = 4$ (ứng với $50 = 30 + 20$) nên đường đi ngắn nhất độ dài 50 từ v_1 đến v_6 kết thúc với cung (v_4, v_6) độ dài 20. Cuối cùng, các phần tử w_{k4} tạo thành vector cột

$$(30, 40, 50, 0, 30, 20, \infty, \infty, \infty),$$

và giá trị nhỏ nhất đạt tại $k = 1$ hoặc $k = 4$ (ứng với $30 + 0$ hoặc $0 + 30$) nên tồn tại cung độ dài 30 từ v_1 đến v_4. Vậy đường đi ngắn nhất từ v_1 đến v_7 là v_1, v_4, v_6, v_7 (các cung có độ dài 30, 20, 20).

Do đó thuật toán Hedetniemi cho một minh họa hình học cách xác định đường đi ngắn nhất tại mỗi bước lặp và sử dụng các ma trận W^k có thể phục hồi được đường đi ngắn nhất. Như vậy, chúng ta cần thêm một ma trận P lưu trữ thông tin của các đường đi ngắn nhất. Ma trận này sẽ được cập nhật trong mỗi bước lặp khi tính W^k từ W^{k-1}. Chi tiết của việc cập nhật ma trận P dành cho người đọc (cũng xem thuật toán Floyd dưới đây).

3.3.2 Thuật toán Floyd (trường hợp ma trận trọng lượng tùy ý)

Thuật toán dưới đây được đưa ra lần đầu tiên bởi Floyd [25] và sau đó được Murchland [46] cải tiến.

Thuật toán Floyd

1. [Khởi tạo] Đặt $k := 0$.

2. $k := k + 1$.

3. [Bước lặp] Với mọi $i \neq k$ sao cho $w_{ik} \neq \infty$ và với mọi $j \neq k$ sao cho $w_{kj} \neq \infty$, thực hiện phép gán

$$w_{ij} := \min\{w_{ij}, (w_{ik} + w_{kj})\}. \tag{3.3}$$

4. [Điều kiện kết thúc] (a) Nếu tồn tại chỉ số i sao cho $w_{ii} < 0$ thì tồn tại mạch với độ dài âm chứa đỉnh v_i. Bài toán vô nghiệm; thuật toán dừng.

   (b) Nếu $w_{ii} \geq 0$ với mọi $i = 1, 2, \ldots, n$, và $k = n$, bài toán có lời giải: ma trận w_{ij} cho độ dài đường đi ngắn nhất từ v_i đến v_j. Thuật toán dừng.

   (c) Nếu $w_{ii} \geq 0$ với mọi $i = 1, 2, \ldots, n$, nhưng $k < n$, chuyển sang Bước 2.

Chứng minh tính đúng đắn của thuật toán Floyd là hoàn toàn đơn giản [35], [25] và dành cho người đọc. Phép toán cơ bản của Phương trình 3.3 trong thuật toán này gọi là *phép toán bộ ba* và có nhiều ứng dụng trong những bài toán tương tự bài toán tìm đường đi ngắn nhất (xem [14], [30]).

Thông tin về các đường đi ngắn nhất có thể nhận được đồng thời cùng với các độ dài đường đi ngắn nhất bằng cách sử dụng quan hệ đệ qui tương tự Phương trình 3.2: sử dụng cơ chế của Hu [35] để lưu giữ thông tin các đường đi ngắn nhất cùng với độ dài của nó. Cụ thể là đưa vào ma trận vuông cấp $n : P := [\theta_{ij}]$; và biến đổi nó đồng thời với việc biến đổi ma trận W. Phần tử θ_{ij} của ma trận P sẽ chỉ ra đỉnh đi trước v_j trong đường đi ngắn nhất từ v_i đến v_j; ở bước đầu tiên ta gán cho ma trận P các giá trị đầu là $\theta_{ij} := v_i$ với mọi đỉnh v_i và v_j.

Cùng với việc biến đổi ma trận W theo Phương trình 3.3 trong Bước 3, ta biến đổi P theo quy tắc

$$\theta_{ij} := \begin{cases} \theta_{kj} & \text{nếu} \quad (w_{ik} + w_{kj}) < w_{ij}, \\ \theta_{ij} & \text{nếu ngược lại.} \end{cases}$$

Kết thúc thuật toán, ma trận P thu được sẽ giúp cho ta việc tìm các đường đi ngắn nhất. Chẳng hạn đường đi ngắn nhất giữa hai đỉnh v_i và v_j là

$$v_i, v_\nu, \ldots, v_\gamma, v_\beta, v_\alpha, v_j$$

trong đó $v_\alpha = \theta_{ij}, v_\beta = \theta_{i\alpha}, v_\gamma = \theta_{i\beta}, ...$, cho đến khi $v_i = \theta_{i\nu}$.

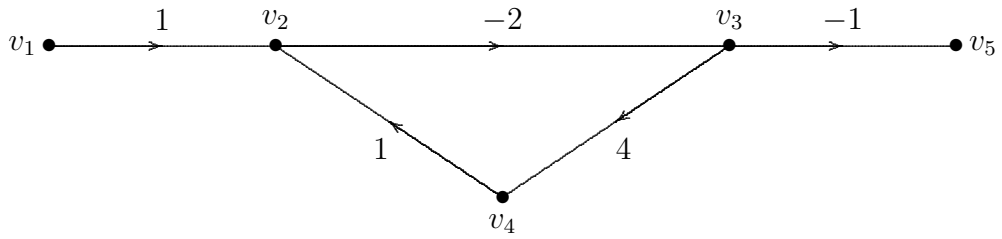
Ví dụ 3.3.6. Xét đồ thị có hướng có trọng số $G = (V, E)$ cho trong Hình 3.3. Áp dụng thuật toán Floyd tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh ta có ma trận đường đi ngắn nhất là

$$W = \begin{pmatrix} 0 & 2 & 4 & 4 & 6 & 1 & 6 & 5 \\ +\infty & 0 & 2 & 2 & 4 & 5 & 10 & 3 \\ +\infty & +\infty & 0 & +\infty & 3 & +\infty & 10 & 1 \\ +\infty & +\infty & +\infty & 0 & 4 & 3 & 8 & 14 \\ +\infty & +\infty & +\infty & +\infty & 0 & +\infty & 7 & 13 \\ +\infty & +\infty & +\infty & +\infty & +\infty & 0 & 5 & 11 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty & 0 & 6 \\ +\infty & +\infty & +\infty & +\infty & +\infty & +\infty & +\infty & 0 \end{pmatrix}$$

và ma trận cho thông tin của đường đi:

$$P = \begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 6 & 3 \\ 2 & 2 & 2 & 2 & 2 & 4 & 6 & 3 \\ 3 & 3 & 3 & 3 & 3 & 3 & 5 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 6 & 7 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 7 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 & 7 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{pmatrix}$$

Ví dụ 3.3.7. Xét đồ thị có hướng có trọng số $G = (V, E)$ cho trong Hình 3.5. Ta có



Hình 3.5:

ma trận trọng lượng

$$W = \begin{pmatrix} 0 & 1 & +\infty & +\infty & +\infty \\ +\infty & 0 & -2 & +\infty & +\infty \\ +\infty & +\infty & 0 & 4 & -1 \\ +\infty & 1 & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & +\infty & 0 \end{pmatrix}.$$

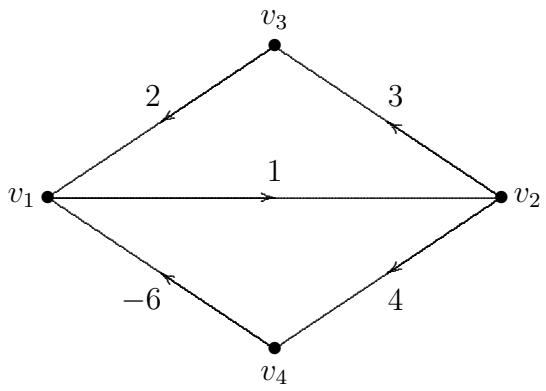
Áp dụng thuật toán Floyd tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh ta có ma trận đường đi

$$W^4 = \begin{pmatrix} 0 & 1 & -1 & 3 & 2 \\ +\infty & 0 & -2 & 2 & -3 \\ +\infty & 5 & 0 & 4 & -1 \\ +\infty & 1 & -1 & 0 & -2 \\ +\infty & +\infty & +\infty & +\infty & 0 \end{pmatrix}$$

và ma trận cho thông tin của đường đi:

$$P = \begin{pmatrix} 1 & 1 & 2 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 \\ 3 & 4 & 3 & 3 & 3 \\ 4 & 4 & 2 & 4 & 3 \\ 5 & 5 & 5 & 5 & 5 \end{pmatrix}.$$

Ví dụ 3.3.8. Xét đồ thị có hướng có trọng số $G = (V, E)$ cho trong Hình 3.6. Ta có



Hình 3.6:

ma trận trọng lượng

$$W = \begin{pmatrix} 0 & 1 & +\infty & +\infty \\ +\infty & 0 & 3 & 4 \\ 2 & +\infty & 0 & +\infty \\ -6 & +\infty & +\infty & 0 \end{pmatrix}.$$

Áp dụng thuật toán Floyd tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh ta có ma trận đường đi sau lần lặp thứ 2 là

$$W^2 = \begin{pmatrix} 0 & 1 & 4 & 5 \\ +\infty & 0 & 3 & 4 \\ 2 & 3 & 0 & 7 \\ -6 & -5 & -2 & -1 \end{pmatrix}.$$

Vì $W^2_{44} = -1 < 0$ nên tồn tại mạch có độ dài âm qua đỉnh v_4.

Nhận xét 3.3.9. Nếu tất cả các giá trị w_{ii} được khởi tạo bằng ∞ (thay cho bằng 0) lúc xuất phát thuật toán, thì các giá trị cuối cùng của w_{ii} là độ dài của mạch ngắn nhất xuất phát từ đỉnh v_i. Ngoài ra cũng có thể dễ dàng xác định mạch có độ dài âm khi $w_{ii} < 0$ dựa vào ma trận đường đi P.

Thủ tục Floyd() sau minh họa thuật toán đã trình bày.

```
void Floyd()
{
  byte i, j, k;
  AdjPointer Tempt;
  byte Pred[MAXVERTICES][MAXVERTICES];
  int Weight[MAXVERTICES][MAXVERTICES], NewLabel;
  byte Start, Terminal;

  for (i = 1; i <= NumVertices; i++)
  {
    for (j = 1; j <= NumVertices; j++)
    {
```

```
    Weight[i][j] = +INFTY;
    Pred[i][j] = i;
  }
  Weight[i][i] = 0;
}

for (i = 1; i <= NumVertices; i++)
{
  Tempt = V_out[i]->Next;
  while (Tempt != NULL)
  {
    Weight[i][Tempt->Vertex] = (long)Tempt->Length;
    Tempt = Tempt->Next;
  }
}

for (k = 1; k <= NumVertices; k++)
{
  for (i = 1; i <= NumVertices; i++)
  {
    if ((i != k) && (Weight[i][k] < +INFTY))
      for (j = 1; j <= NumVertices; j++)
        if ((j != k) && (Weight[k][j] < +INFTY))
        {
          NewLabel = Weight[i][k] + Weight[k][j];
          if (Weight[i][j] > NewLabel)
          {
            Weight[i][j] = NewLabel;
            Pred[i][j] = Pred[k][j];
          }
        }
        if (Weight[i][i] < 0)
        {
          printf("Ton tai mach do dai am qua dinh %d", i);
```

```
            printf(" %6d ", Weight[i][i]);
            return;
          }
        }
      }

      // Vi du minh hoa
      Start = 1;
      Terminal = 3;

      if (Weight[Start][Terminal] < +INFTY)
      {
        printf("Ton tai duong di tu %d", Start);
        printf(" den %d", Terminal);
        printf("\nDuong di qua cac dinh:");

        i = Terminal;
        while (i != Start)
        {
          printf("%3d ", i);
          i = Pred[Start][i];
        }
        printf("%3d ", Start);
        printf("\nTrong luong la %3d ", Weight[Start][Terminal]);
      }
      else
        printf("\n Khong ton tai duong di tu %d den %d", Start, Terminal);
}
```

3.4 Phát hiện mạch có độ dài âm

Vấn đề phát hiện các mạch có độ dài âm trong một đồ thị tổng quát là quan trọng không những trong chính bài toán tìm đường đi ngắn nhất mà còn là một bước cơ bản trong những thuật toán khác (xem Phần 3.4.1 và [14]).

Ta biết rằng Thuật toán Floyd tìm đường đi ngắn nhất giữa các cặp đỉnh có thể phát hiện các mạch độ dài âm trong đồ thị. Hơn nữa, nếu đồ thị chứa một đỉnh s mà có thể đến tất cả các đỉnh khác từ đó, thì có thể áp dụng Thuật toán Ford-Moore-Bellman (tìm tất cả các đường đi ngắn nhất xuất phát từ đỉnh s) để phát hiện các mạch có độ dài âm như đã thực hiện trong Bước 3 của phần này. Nếu tồn tại đỉnh không thể đến được từ s (chẳng hạn, khi G là đồ thị vô hướng không liên thông) thì Thuật toán Ford-Moore-Bellman sẽ kết thúc và chỉ các đỉnh có thể đến được từ s có nhãn hữu hạn, các đỉnh khác không đến được từ s có nhãn bằng ∞. Trong trường hợp này có thể tồn tại các mạch có độ dài âm trong thành phần liên thông không chứa s và không được phát hiện. Tuy nhiên, nhiều ứng dụng cần kiểm tra có mạch độ dài âm hay không, đồ thị được xét có đỉnh s đến được tất cả các đỉnh khác, và do đó với những trường hợp như vậy, để xác định sự tồn tại của mạch độ dài âm, Thuật toán Ford-Moore-Bellman sẽ cho phép tính toán hiệu quả hơn Thuật toán Floyd.

Các nguyên tắc kết thúc của Thuật toán Ford-Moore-Bellman được trình bày để tính toán ít nhất các đường đi ngắn nhất từ s khi không có mạch độ dài âm. Các nguyên tắc này có thể cải biên để phát hiện mạch độ dài âm sớm hơn như sau.

Sau khi gán lại nhãn của đỉnh v_i từ một đỉnh v_{j^*} theo Phương trình 3.2 ta kiểm tra đỉnh v_i có thuộc đường đi ngắn nhất hiện hành (mà có thể suy từ các nhãn hiện hành θ) từ s đến v_{j^*} hay không. Nếu đúng, thì đỉnh v_{j^*} đã được gán nhãn thông qua v_i và điều này dẫn đến $L^k(v_{j^*}) + w(v_{j^*}, v_i) < L^k(v_i)$; do đó một phần của đường đi ngắn nhất hiện hành từ v_i đến v_{j^*} cộng thêm cung (v_{j^*}, v_i) sẽ tạo thành mạch có độ dài âm và thuật toán kết thúc. Nếu mặt khác, nhãn của đỉnh v_i hoặc không thay đổi bởi Phương trình 3.2, hoặc nhận nhãn mới từ một đỉnh v_{j^*} nhưng v_i không thuộc đường đi ngắn nhất từ s đến v_{j^*}, thì thuật toán tiếp tục Bước 3 như trước. Cần chú ý rằng, cải tiến trên chính là giảm những dư thừa không cần thiết trong Thuật toán Ford-Moore-Bellman, do một mạch có độ dài âm được xác định ngay khi nó được tạo

ra và không cần đợi kết thúc thủ tục.

3.4.1 Mạch tối ưu trong đồ thị có hai trọng lượng

Một vấn đề nảy sinh trong nhiều lĩnh vực liên quan đến một đồ thị có hướng mà mỗi cung (v_i, v_j) được gán hai trọng lượng: w_{ij} và b_{ij}. Bài toán là tìm một mạch Φ mà cực tiểu (hoặc cực đại) hàm mục tiêu

$$z(\Phi) := \frac{\sum_{(v_i,v_j)\in\Phi} w_{ij}}{\sum_{(v_i,v_j)\in\Phi} b_{ij}}.$$

Chẳng hạn, xét hành trình của một con tàu hay một máy bay trên mạng giao thông và giả sử rằng w_{ij} là "lợi nhuận" và b_{ij} là "thời gian" cần thiết để di chuyển trên cung (v_i, v_j). Bài toán đặt ra là tìm hành trình để tối đa lợi nhuận với thời gian nhỏ nhất.

Các vấn đề khác có thể phát biểu dạng bài toán tìm các mạch tối ưu trong đồ thị có hai trọng lượng: Lập lịch để tính toán song song, tối ưu đa mục tiêu trong xử lý công nghiệp.

Bài toán tìm một mạch Φ trong đồ thị có hai trọng lượng để cực tiểu hàm mục tiêu $z(\Phi)$ có thể giải quyết nhờ thuật toán phát hiện các mạch có độ dài âm như sau. Giả sử rằng các trọng lượng w_{ij} và b_{ij} là các số thực tùy ý (dương, âm hoặc bằng không) nhưng thỏa mãn ràng buộc $\sum_{(v_i,v_j)\in\Phi} b_{ij} > 0$, với mọi mạch Φ trong G. (Trong hầu hết các tình huống, chẳng hạn các vấn đề nêu trên, $w_{ij} \in \mathbf{R}$ nhưng $b_{ij} \geq 0$ với mọi i và j).

Chúng ta chọn một giá trị thử z^k đối với hàm mục tiêu $z(\Phi)$ và xét đồ thị có trọng lượng

$$w_{ij}^k = w_{ij} - z^k b_{ij}.$$

Với ma trận trọng lượng w_{ij}^k mới, xét bài toán đường đi ngắn nhất trên G. Có ba trường hợp xảy ra:

Trường hợp A. Tồn tại mạch có độ dài âm Φ^- sao cho

$$\sum_{(v_i,v_j)\in\Phi^-} w_{ij}^k < 0.$$

Trường hợp B. Không tồn tại mạch có độ dài âm và

$$\sum_{(v_i,v_j)\in\Phi} w_{ij}^k > 0$$

với mọi mạch Φ.

Trường hợp C. Tồn tại mạch có độ dài bằng không (nhưng không có mạch độ dài âm); tức là

$$\sum_{(v_i,v_j)\in\Phi^0} w_{ij}^k = 0$$

với mạch Φ^0 nào đó.

Trong Trường hợp A chúng ta có thể nói rằng z^* (giá trị cực tiểu của z) nhỏ hơn z^k vì

$$\sum_{(v_i,v_j)\in\Phi^-} w_{ij}^k \equiv \sum_{(v_i,v_j)\in\Phi^-} w_{ij} - z^k \sum_{(v_i,v_j)\in\Phi^-} b_{ij} < 0$$

chỉ có thể đúng nếu

$$\frac{\sum_{(v_i,v_j)\in\Phi^{-1}} w_{ij}}{\sum_{(v_i,v_j)\in\Phi^{-1}} b_{ij}} < z^k$$

mà hiển nhiên chỉ ra rằng $z^* < z^k$.

Tương tự, trong Trường hợp B ta có thể nói rằng $z^* > z^k$; và trong Trường hợp C thì $z^* = z^k$.

Do đó thuật toán tìm kiếm nhị phân để giải quyết bài toán như sau: Khởi đầu với giá trị thử z^1; nếu nó quá lớn (tức là Trường hợp A) thử với $z^2 < z^1$; nếu nó quá nhỏ (tức là Trường hợp B) thử với $z^2 > z^1$. Khi đã có các cận trên và dưới (z_u và z_l tương ứng) ta có thể xác định z^* bằng cách thử $z^k = (z_u + z_l)/2$ và thay z_u bằng z^k nếu xảy ra Trường hợp A, hoặc thay z_l bằng z^k nếu xảy ra Trường hợp B. Do số các phép thử tỉ lệ với $1/\eta$, trong đó $0 < \eta \ll 1$ là sai số cho trước, và do mỗi lần thử (xác định mạch độ dài âm hoặc tính toán ma trận trọng lượng) đòi hỏi n^3 phép toán, nên tìm nghiệm của bài toán trên đòi hỏi $O[n^3 \log 1/\eta]$ phép toán.

Chương 4

Cây

4.1 Mở đầu

Cây là một đồ thị có những tính chất đặc biệt và có nhiều ứng dụng.

Trong chương này, trước hết sẽ nghiên cứu *cây Huffman* và những ứng dụng của nó trong việc nén dữ liệu. Kế tiếp chúng ta xét trình bày các thuật toán tìm cây bao trùm, cây bao trùm có trọng lượng nhỏ nhất khi các cạnh của đồ thị được gắn với các chi phí (trọng lượng). Cây bao trùm nhỏ nhất của đồ thị có nhiều ứng dụng trong những trường hợp các đường dẫn (ống dẫn ga, dây dẫn trong mạng điện, v.v) được sử dụng để nối n điểm với nhau theo cách tốt nhất: tổng khoảng cách của các đường dẫn là nhỏ nhất. Nếu n điểm được nối với nhau trên một mặt phẳng, ta có thể biểu diễn bởi một đồ thị đầy đủ trong đó các chi phí cạnh là khoảng cách giữa hai điểm tương ứng. Khi đó cây bao trùm với trọng lượng nhỏ nhất sẽ cho mạng giao thông với chi phí ít nhất. Nếu có thể nối thêm ngoài n điểm cho phép, ta có thể thậm chí xây dựng được mạng với chí phí rẻ hơn và xác định nó chính là giải quyết bài toán Steiner. Bài toán sau này sẽ được đề cập ở phần cuối chương.

Định nghĩa 4.1.1. Các định nghĩa sau của *cây* (vô hướng) là tương đương:

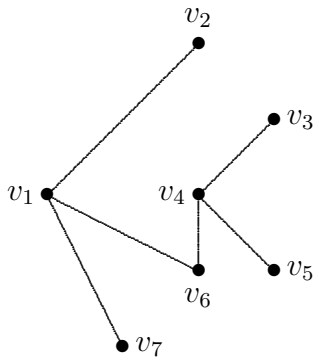
1. Đồ thị liên thông có n đỉnh và $(n-1)$ cạnh.

2. Đồ thị liên thông không có chu trình.

3. Đồ thị mà mọi cặp đỉnh được nối với nhau bởi một và chỉ một dây chuyền sơ cấp.

4. Đồ thị liên thông và khi bớt một cạnh bất kỳ thì mất tính liên thông.

Hình 4.1 minh họa cây có bảy đỉnh và sáu cạnh.



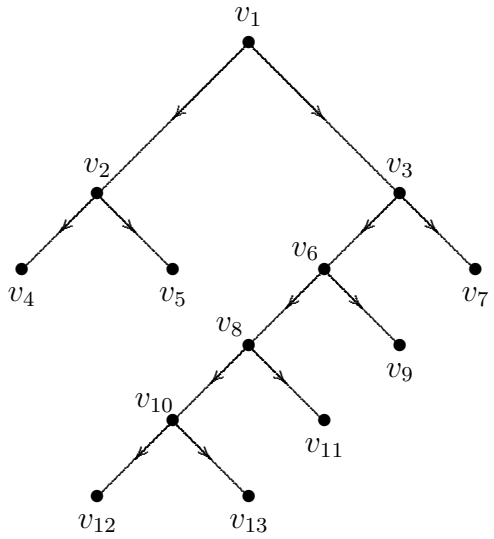
Hình 4.1: Một ví dụ về cây.

Khái niệm về cây như một thực thể của toán học được đưa ra lần đầu tiên bởi Kirchhoff [37] khi liên hệ với định nghĩa các mạch cơ bản được sử dụng trong phân tích các mạng điện. Khoảng 10 năm sau đó, một cách độc lập, Cayley [11] đã phát hiện lại các cây và những tính chất của nó khi nghiên cứu các tính chất hóa học của các chất đồng phân của hydrocarbon.

Cây có gốc (còn gọi là *cây gia phả*) được định nghĩa tương tự như sau:

Định nghĩa 4.1.2. *Cây có gốc* T là đồ thị có hướng không mạch mà mọi đỉnh, ngoại trừ một đỉnh (chẳng hạn v_1), có bậc trong bằng một: bậc trong của đỉnh v_1 (gọi là *gốc* của cây) bằng không; nói cách khác, mọi đỉnh $v \in T$ tồn tại duy nhất một đường đi từ gốc đến v.

Hình 4.2 minh họa một đồ thị là cây có gốc với đỉnh v_1 là gốc. Từ định nghĩa suy ra rằng cây có gốc n đỉnh có $(n-1)$ cung và là đồ thị liên thông (khi bỏ qua hướng trên các cung).



Hình 4.2: Một ví dụ về cây có gốc.

Cần chú ý rằng, có thể định hướng trên một cây (vô hướng) sao cho đồ thị thu được là cây có gốc: Ta chỉ cần chọn một đỉnh tùy ý, chẳng hạn v_1, làm gốc và định hướng các cung theo dây chuyền từ v_1 đến các đỉnh treo. Ngược lại, nếu bỏ qua các hướng trên cây có gốc ta thu được một cây.

Cây gia phả mà trong đó mỗi người đàn ông biểu thị một đỉnh và các cung được vẽ từ các cha đến các con của họ là một ví dụ quen thuộc của cây có gốc, gốc của cây là người đầu tiên trong dòng họ mà có thể xác định được.

4.2 Cây Huffman

Tiến trình gán dãy các bit cho các ký hiệu gọi là *mã hóa.* Trong phần này chúng ta một tả một thuật toán mã hóa rất quen thuộc-*thuật toán mã hóa Huffman.*

4.2.1 Các bộ mã "tốt"

Khi ta nói về *mã hoá* có nghĩa là gán dãy các bit cho các phần tử của một bảng chữ cái. Tập các chuỗi nhị phân gọi là *bộ mã* và các phần tử của chúng gọi là *từ mã.* Một

bảng chữ cái là một tập hợp các ký hiệu, gọi là các *ký tự*. Chẳng hạn, bảng chữ cái sử dụng trong hầu hết các sách (tiếng Anh) gồm 26 ký tự thường, 26 ký tự hoa, và các dấu ngắt câu (như dấu phẩy). Mã ASCII (viết tắt của các chữ cái đầu tiên của chuỗi American Standard Code for Information Interchange của ký tự A là 01000001, của ký tự a là 01100001 và ký tự , là 0011010. Chú ý rằng trong mã ASCII số các bit sử dụng để biểu diễn các ký tự là bằng nhau. Mã như vậy gọi là *mã có độ dài cố định*. Nếu ta muốn giảm số các bit đòi hỏi để biểu diễn các thông báo khác nhau, ta cần các chuỗi bit biểu diễn ký tự có độ dài (nói chung) không bằng nhau. Nếu biểu diễn các bit dài hơn cho các ký tự thường xuyên xuất hiện và ngược lại, chúng ta có thể, về trung bình, giảm số bit biểu diễn các ký hiệu. Chẳng hạn, mã Morse [31] sử dụng các từ mã ngắn hơn cho các ký tự xuất hiện thường xuyên: mã của của a là $\cdot-$, của e là $\cdot$, trong khi của z là $--\cdots$, q là $--\cdot-$, j là $\cdot---$.

Tuy nhiên độ dài trung bình của mã không phải là tiêu chuẩn quan trọng khi thiết kế một bộ mã "tốt". Xét ví dụ sau. Giả sử bảng chữ cái gồm bốn ký tự a_1, a_2, a_3, a_4 với các xác suất xuất hiện tương ứng là $P(a_1) = \frac{1}{2}, P(a_2) = \frac{1}{4}, P(a_3) = \frac{1}{8}, P(a_4) = \frac{1}{8}$. Entropy của mã nguồn này là 1.75 bit/ký hiệu (xem [31] để có khái niệm về entropy). Xét các bộ mã trong nguồn này cho bởi bảng sau

Các ký tự	Mã C_1	Mã C_2	Mã C_3	Mã C_4
a_1	1	1	1	1
a_2	1	0	01	10
a_3	0	11	001	100
a_4	01	00	000	1000
Độ dài trung bình	1.125	1.125	1.75	1.875

Độ dài trung bình l của mỗi mã xác định bởi

$$l = \sum_{i=1}^{4} P(a_i)n(a_i) \quad \text{(bit/ký hiệu)},$$

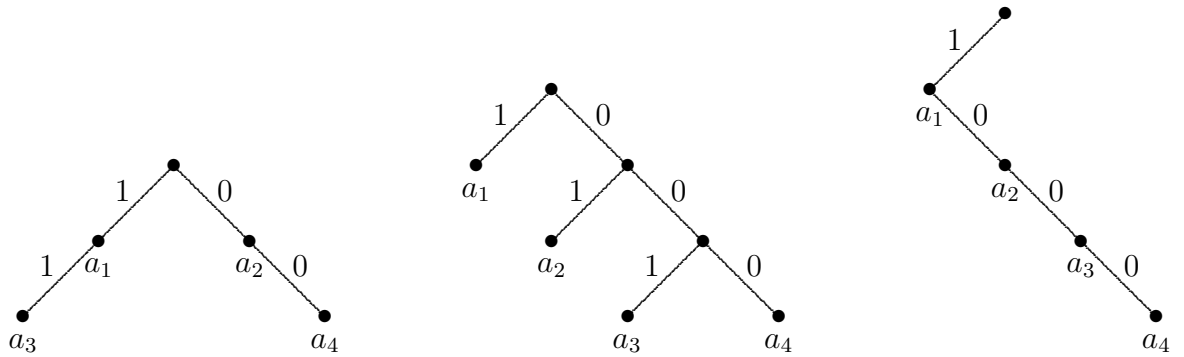
trong đó $n(a_i)$ là số các bit của từ mã a_i.

Dựa trên độ dài trung bình, mã C_1 là tốt nhất. Tuy nhiên, các mã cần phải có khả năng truyền thông tin sao cho người nhận có thể hiểu được rõ ràng. Hiển nhiên mã

C_1 không có tính chất này do cả hai ký hiệu a_1 và a_2 đều được gán là 1 nên khi nhận được thông tin là 1, người nhận không thể phân biệt đó là ký hiệu a_1 hay a_2. Chúng ta muốn mỗi ký hiệu được gán duy nhất một từ mã.

Mã C_2 có các ký tự được gán các chuỗi bit khác nhau cho các ký tự. Tuy nhiên, giả sử mã hóa dãy $a_2a_1a_1$ được 011 và gửi đi chuỗi bit này. Người nhận chuỗi 011 có một số cách giải mã: $a_2a_1a_1$ hoặc a_2a_3. Điều này có nghĩa là một dãy được mã hóa với bộ mã C_2 thì dãy này có thể được giải mã không trùng với thông báo ban đầu. Nói chung, đây không phải là điều chúng ta mong muốn khi thiết kế các bộ mã. Chúng ta muốn có tính chất *giải mã duy nhất*; tức là mọi dãy các từ mã chỉ có duy nhất một cách giải mã. Có thể chứng minh các mã C_3 và C_4 thỏa mãn tính chất này.

Mặc dù việc kiểm tra tính duy nhất khi giải mã là khó, ta có thể chứng minh tính chất này cho bộ mã C_3 dựa trên thuộc tính của bộ mã: không có từ mã nào trong mã C_3 là "tiền tố" của từ mã khác. Bộ mã như vậy gọi là *mã tiền tố.* Một cách đơn giản để kiểm tra một bộ mã có phải là tiền tố hay không ta vẽ *cây nhị phân* (mỗi đỉnh có bậc ≤ 2) tương ứng với bộ mã. Cây được vẽ xuất phát từ một nút đơn (*nút gốc*) và mỗi nút trong có bậc ngoài nhỏ hơn hoặc bằng hai. Một trong hai nhánh con tương ứng bit 1 và nhánh còn lại tương ứng bit 0. Để thuật tiện, ta quy ước nhánh con bên phải tương ứng 0 và nhánh con bên trái tương ứng 1. Theo cách này, các cây nhị phân tương ứng các mã C_2, C_3 và C_4 cho trong Hình 4.3. (Để đơn giản, ta sẽ không vẽ các mũi tên trong các cây nhị phân).



Hình 4.3:

Chú ý rằng ngoài nút gốc, cây nhị phân có hai loại nút: các *nút lá* có bậc ngoài

bằng không; và các *nút trong* có bậc ngoài khác không. Trong bộ mã tiền tố, các từ mã chỉ tương ứng các nút lá. Mã C_4 không phải là mã tiền tố do tồn tại từ mã tương ứng nút trong. Duyệt cây từ gốc đến các nút lá cho ta biểu diễn chuỗi bit tương ứng ký hiệu. Mỗi nhánh đóng góp một bit vào từ mã của nó: bit 1 cho nhánh trái và bit 0 cho nhánh phải.

Mã tiền tố luôn luôn được giải mã duy nhất nhưng ngược lại không đúng (chẳng hạn mã C_4). Tuy nhiên có thể chứng minh rằng bộ mã có thể giải mã duy nhất tương đương với mã tiền tố theo nghĩa: số trung bình các bit biểu diễn các ký hiệu bằng nhau.

4.2.2 Mã Huffman

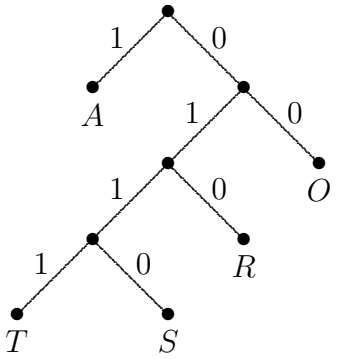
Mã Huffman là mã tiền tố và tối ưu với các xác xuất cho trước. Phương pháp xây dựng mã Huffman dựa trên hai quan sát sau:

1. Trong một bộ mã tối ưu, các ký hiệu xuất hiện thường xuyên (có xác suất hay tần số xuất hiện lớn) sẽ có các từ mã ngắn hơn các ký hiệu ít xuất hiện.

2. Trong một bộ mã tốt ưu, hai ký hiệu xuất hiện ít nhất sẽ có các từ mã cùng độ dài.

Để xây dựng mã Huffman, chúng ta có thể biểu diễn qua cây nhị phân mà các nút lá tương ứng các ký hiệu. Duyệt cây nhị phân sẽ cho ta các từ mã của bộ mã: xuất phát từ nút gốc và đi đến các nút lá, thêm bit 1 vào từ mã mỗi lần qua nhánh trái và bit 0 mỗi lần qua nhánh phải. Với cây trong Hình 4.4, ta có biểu diễn các ký tự qua các từ mã như sau:

Ký tự	Mã hóa
A	1
O	00
R	010
S	0110
T	0111



Hình 4.4:

Để giải mã một chuỗi bit, chúng ta bắt đầu từ gốc và di chuyển dọc theo cây cho đến khi gặp ký tự: đi theo nhánh trái nếu đó là bit 1, ngược lại đi theo nhánh phải. Chẳng hạn, chuỗi bit

01010111

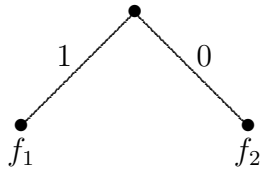
tương ứng từ RAT. Với một cây xác định mã Huffman như Hình 4.4, chuỗi bit bất kỳ được giải mã duy nhất mặc dù các ký tự tương ứng với những chuỗi bit có độ dài thay đổi.

Huffman đã chỉ ra thuật toán xây dựng mã Huffman từ bảng các tần số xuất hiện của các ký tự như sau:

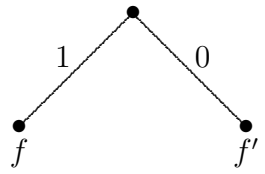
Thuật toán xây dựng mã Huffman

Xét chuỗi cần mã hóa s từ n ký tự với $n \geq 2$.

1. Xây dựng dãy tần số $f_i, i = 1, 2, \ldots, n$, xuất hiện của các ký tự trong chuỗi s.
2. Nếu $n = 2$ (giả sử $f_1 \leq f_2$), xuất cây như trong Hình 4.5 và dừng.
3. Giả sử f và f' là hai tần số nhỏ nhất và $f \leq f'$. Tạo một danh sách tần số mới bằng cách thay f và f' bởi $f + f'$. Gọi thuật toán này sử dụng danh sách tần số mới để tạo cây T'. Thay đỉnh được gán nhãn $f + f'$ để nhận được cây T trong Hình 4.6. Xuất T.



Hình 4.5:

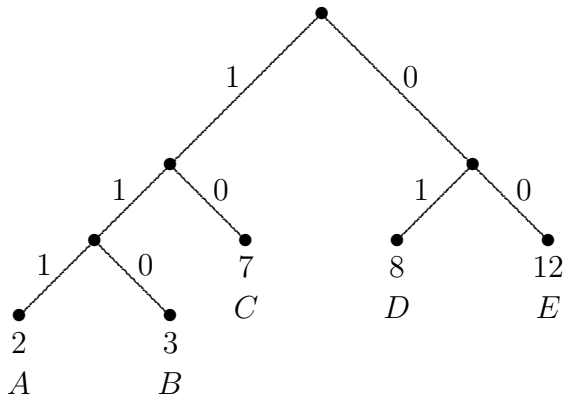


Hình 4.6:

Ví dụ 4.2.1. Cho bảng tần số

Ký tự	tần số
A	2
B	3
C	7
D	8
E	12

Khi đó cây Huffman tương ứng cho trong Hình 4.7.



Hình 4.7:

4.3 Liệt kê cây

Năm 1857, nhà toán học người Anh, A. Caylay (độc lập với G. Kirchoff) đã khám phá các cây khi cố gắng liệt kê tất cả các chất đồng phân của hydrocarbon. Các phân tử hydrocarbon được cấu tạo từ các nguyên tử hydrogen và carbon, trong đó mỗi nguyên tử carbon có thể liên kết hóa học với bốn nguyên tử khác và mỗi nguyên tử hydrogen có thể liên kết hóa học với một nguyên tử khác. Một hydrocarbon bảo hòa là hydrocarbon chứa số cực đại các nguyên tử hydrogen (với số các nguyên tử carbon cho trước). A. Caylay đã chứng minh rằng, nếu hydrocarbon bảo hòa có k nguyên tử carbon thì nó cần có $2k+2$ nguyên tử hydrogen và do đó có công thức hóa học C_kH_{2k+2}. Ông đã dùng đồ thị liên thông để biểu diễn cấu trúc của một phân tử hydrocarbon C_kH_{2k+2} : các đỉnh là các nguyên tử carbon và hydrogen; các cạnh tương ứng các liên kết hóa học giữa các nguyên tử. Trong trường hợp này, một nguyên tử carbon tương ứng với một đỉnh bậc bốn và một nguyên tử hydrogen tương ứng với một đỉnh bậc một (đỉnh treo). Tổng số các đỉnh trong đồ thị tương ứng như vậy là: $n = k + (2k+2) = 3k+2$; và tổng số các cạnh là:

$$\frac{1}{2}\ (\text{tổng các bậc})\ = (4k+2k+2)/2 = 3k+1.$$

Đồ thị này liên thông và có số cạnh ít hơn số đỉnh là 1 nên nó là một cây. Như vậy vấn đề đếm các cấu trúc đồng phân của một hydrocarbon đưa về bài toán đếm các cây (dĩ nhiên có cùng các tính chất xác định).

Câu hỏi đầu tiên của Cayley đặt ra như sau: *Số các cây khác nhau có thể xây dựng từ n đỉnh (hay nhãn) khác nhau là bao nhiêu?*

Nếu $n = 4$, thì chúng ta có 16 cây (tại sao?).

Để trả lời câu hỏi trên, ta xét đồ thị mà mỗi đỉnh có một tên hay nhãn duy nhất (tức là không có hai đỉnh mang cùng một nhãn) được gọi là *đồ thị được gán nhãn* (labeled graph).

Kết quả sau lần đầu tiên được đưa ra và chứng minh bởi Cayley. Sau đó nhiều chứng minh khác cũng được công bố. Chứng minh sau đây của H. Prüfer năm 1918.

Định lý 4.3.1. (A. Cayley) *Số các cây được gán nhãn n đỉnh ($n \geq 2$) là n^{n-2}; tức là*

số các cây bao trùm của đồ thị đầy đủ K_n là n^{n-2}.

Chứng minh. Giả sử $V = \{1, 2, \ldots, n\}$. Với mỗi cây bao trùm T của đồ thị K_n ta thiết lập tương ứng một-một với vector $a = (a_1, a_2, \ldots, a_{n-2})$, trong đó các số nguyên a_i thỏa mãn $1 \leq a_i \leq n$, như sau:

+ Ký hiệu b_1 là đỉnh treo đầu tiên (có chỉ số nhỏ nhất) trong tập được sắp thứ tự V sao cho $e_1 = (a_1, b_1)$ là cạnh treo của T tương ứng (tồn tại do mọi cây có ít nhất một đỉnh treo). Loại cạnh e_1 và đỉnh b_1 ra khỏi cây T ta được cây T_1 mới.

+ Ký hiệu b_2 là đỉnh treo đầu tiên (có chỉ số nhỏ nhất) trong tập được sắp thứ tự V sao cho $e_2 = (a_2, b_2)$ là cạnh treo tương ứng trong cây T_1 (tồn tại do mọi cây có ít nhất một đỉnh treo). Loại cạnh e_2 và đỉnh b_2 ra khỏi cây T_1 ta được cây T_2 mới.

+ Lặp lại theo quy nạp cho đến khi loại cạnh $e_{n-2} = (a_{n-2}, b_{n-2})$ ta được cây gồm đúng một cạnh $e_{n-1} = (a_{n-1}, b_{n-1})$ nối hai đỉnh còn lại.

Khi đó vector $a = (a_1, a_1, \ldots, a_{n-2}) \in V^{n-2}$ được xác định duy nhất bởi cây T và với hai cây khác nhau T và T', ta có tương ứng hai vector khác nhau. Mỗi đỉnh a_i xuất hiện $d(a_i) + 1$ lần trong vector a.

Ngược lại, với mỗi vector $a \in V^{n-2}$, ta có thể xây dựng một cây T như sau:

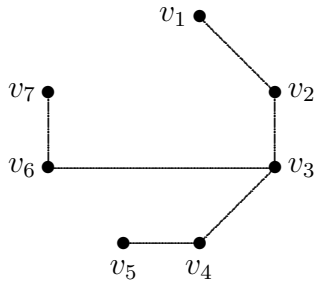
+ Lấy đỉnh đầu tiên (có chỉ số nhỏ nhất) $b_1 \in V$ mà không xuất hiện trong thành phần của vector a. Lấy phần tử đầu tiên a_1 trong vector a; đặt cạnh $e_1 = (a_1, b_1)$. Loại a_1 khỏi vector a và đỉnh b_1 ra khỏi tập V.

+ Tiếp tục lặp lại theo thủ tục trên với các số còn lại, cuối cùng ta sẽ thu được cây T.

Nhận xét rằng $\#V = n^{n-2}$. Định lý được chứng minh. ◁

Ví dụ 4.3.2. Xác định vector độ dài năm tương ứng cây bao trùm trong Hình 4.8.

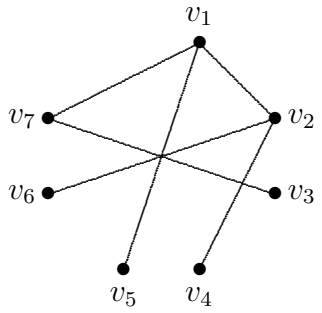
Chú ý rằng v_1 là đỉnh treo với chỉ số nhỏ nhất và v_2 liên thuộc với v_1, do đó $a_1 = 2$. Xóa cạnh (v_1, v_2). Đồ thị còn lại có v_2 là đỉnh treo với chỉ số nhỏ nhất; do đó $a_2 = 3$. Xóa cạnh (v_2, v_3) và lặp lại tiến trình trên ta có vector tương ứng cây là $(2, 3, 4, 3, 6)$.



Hình 4.8:

Ví dụ 4.3.3. Tìm cây bao trùm của K_7 tương ứng vector $(7, 2, 1, 2, 1)$.

Lời giải cho trong Hình 4.9. Thật vậy, bắt đầu với danh sách $\{1, 2, 3, 4, 5, 6, 7\}$. Số 3 là số nhỏ nhất trong danh sách nhưng không thuộc vector $a := (7, 2, 1, 2, 1)$ và 7 là số đầu tiên trong vector a. Bằng cách nối, ta có cạnh (v_3, v_7). Loại bỏ 3 ra khỏi danh sách và 7 ra khỏi vector a ta có danh sách và vector a mới tương ứng là $\{1, 2, 4, 5, 6, 7\}$ và $a = (2, 1, 2, 1)$. Số 4 là số nhỏ nhất trong danh sách nhưng không thuộc vector a và 2 là số đầu tiên trong vector a. Bằng cách nối, ta có cạnh (v_4, v_2). Lặp lại thủ tục trên cho đến khi ta có cạnh cuối cùng (v_1, v_7).



Hình 4.9:

Ví dụ 4.3.4. Có bao nhiêu cách để xây dựng mạng điện với 12 nút nối tất cả các nút sử dụng số dây dẫn ít nhất có thể.

Chúng ta xây dựng đồ thị con của K_{12} như sau: mỗi nút tương ứng với một đỉnh và mỗi dây dẫn tương ứng các cạnh. Đồ thị T biểu diễn mạng điện với 12 nút nối tất cả các nút sử dụng số dây dẫn ít nhất phải là đồ thị liên thông không chu trình. Vì vậy T là cây bao trùm của K_{12}. Theo Định lý 4.3.1, có 12^{10} cây bao trùm của K_{12}. Do đó có 12^{10} cách để xây dựng mạng điện.

Nhận xét 4.3.5. Định lý 4.3.1 không cho ta chính xác số các chất đồng phân của C_kH_{2k+1}. Để hạn chế về bậc của các đỉnh, ta nhận xét rằng:

(a) Vì các đỉnh biểu diễn hydrogen là các đỉnh treo, chúng sẽ kết hợp với các nguyên tử carbon theo cùng một cách và do đó không đóng góp vào hiện tượng đồng phân. Vì vậy ta không cần quan tâm đến các đỉnh hydrogen.

(b) Suy ra cây biểu diễn C_kH_{2k+1} đưa về cây có k đỉnh, mỗi đỉnh biểu diễn một nguyên tử carbon. Với cây này ta không phân biệt các đỉnh, và do đó nó là *cây không được gán nhãn.*

Vậy với butane C_4H_{10} chỉ có hai cây khác nhau (hãy vẽ chúng). (Trong hóa học, ta biết rằng có chính xác hai loại buttane khác nhau là: n-buttan và isobuttane).

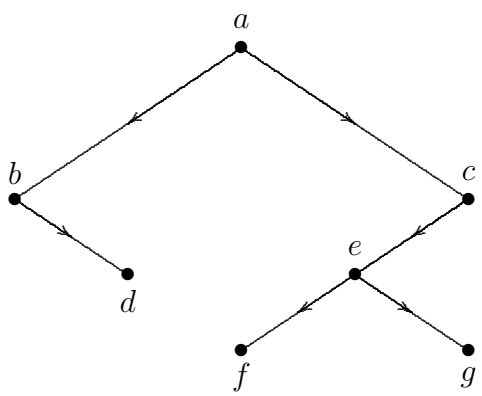
Việc phân biệt giữa đồ thị được gán nhãn và đồ thị không được gán nhãn là rất quan trọng trong bài toán đếm số các đồ thị khác nhau. Bài toán liệt kê các cây không được gán nhãn liên quan đến một số khái niệm khác và vượt ra phạm vi của giáo trình, bạn đọc quan tâm có thể xem tài liệu dẫn [19].

4.4 Cây nhị phân

"Cây nhị phân" (binary tree) là một trong những lớp quan trọng nhất của cây có gốc. Mỗi đỉnh trong cây nhị phân có nhiều nhất hai con (xem Hình 4.10). Hơn nữa, mỗi đỉnh con được ký hiệu hoặc là "con trái" hoặc là "con phải". Khi vẽ cây nhị phân, đỉnh con trái được vẽ bên trái và đỉnh con phải được vẽ bên phải.

Định nghĩa 4.4.1. *Cây nhị phân* là một cây có gốc trong đó mỗi đỉnh hoặc không có con, hoặc có một con, hoặc có hai con. Nếu đỉnh có một con, thì con này được xem là con trái hoặc con phải; nếu một đỉnh có hai con, thì một con bên trái và một con bên phải.

Ví dụ 4.4.2. Trong cây nhị phân Hình 4.10, đỉnh b là con trái của a và đỉnh c là con phải của a. Đỉnh d là con phải của b; đỉnh b không có con trái. Đỉnh e là con trái của c; đỉnh c không có con phải.



Hình 4.10:

Ví dụ 4.4.3. Một cây xác định bởi mã Huffman là cây nhị phân. Chẳng hạn, với cây Huffman trong Hình 4.4, di chuyển từ một đỉnh đến đỉnh con bên trái tương ứng sử dụng bit 1 và di chuyển từ một đỉnh đến đỉnh con bên phải tương ứng sử dụng bit 0.

Cây nhị phân đầy đủ là cây nhị phân mà mỗi đỉnh hoặc có hai con hoặc không có con.

Định lý 4.4.4. *Nếu T là cây nhị phân đầy đủ với i đỉnh trong thì T có $i+1$ đỉnh treo và có tất cả $2i+1$ đỉnh.*

Chứng minh. Tập các đỉnh của T gồm những đỉnh là các đỉnh con (một vài đỉnh là cha) và những đỉnh không phải là đỉnh con. Tồn tại duy nhất một đỉnh không phải là con-đỉnh gốc. Do có i đỉnh trong và mỗi đỉnh có hai con nên tồn tại $2i$ đỉnh con. Vậy số các đỉnh của T là $2i+1$ và số các đỉnh treo bằng

$$(2i+1) - i = i + 1.$$

◁

Định lý 4.4.5. *Nếu T là cây nhị phân có độ cao h và t đỉnh treo thì*

$$\log t \leq h. \tag{4.1}$$

Chứng minh. Ta sẽ chứng minh quy nạp theo h bất đẳng thức tương đương:

$$t \leq 2^h. \tag{4.2}$$

Nếu $h = 0$ thì cây T gồm một đỉnh; suy ra $t = 1$. Do đó (4.2) đúng.

Giả sử khẳng định đúng với mọi cây nhị phân có độ cao nhỏ hơn h. Giả sử T là cây nhị phân có độ cao $h > 0$ với t đỉnh treo. Xét trường hợp đỉnh gốc của T chỉ có một con. Nếu ta khử gốc và cạnh liên thuộc với gốc thì ta được cây nhị phân có độ cao $h - 1$ và cùng số đỉnh treo như T. Theo quy nạp, $t \leq 2^{h-1} < 2^h$ và do vậy (4.2) đúng.

Bây giờ giả sử gốc của T có hai con là v_1 và v_2. Ký hiệu $T_i, i = 1, 2$, là cây con với gốc tại v_i và giả sử T_i có độ cao h_i và t_i đỉnh treo. Theo giả thiết quy nạp

$$t_i \leq 2^{h_i}, \qquad i = 1, 2. \tag{4.3}$$

Nhận xét rằng các đỉnh treo của T gồm các nút lá của T_1 và T_2. Do đó

$$t = t_1 + t_2. \tag{4.4}$$

Từ (4.3) và (4.4) ta có

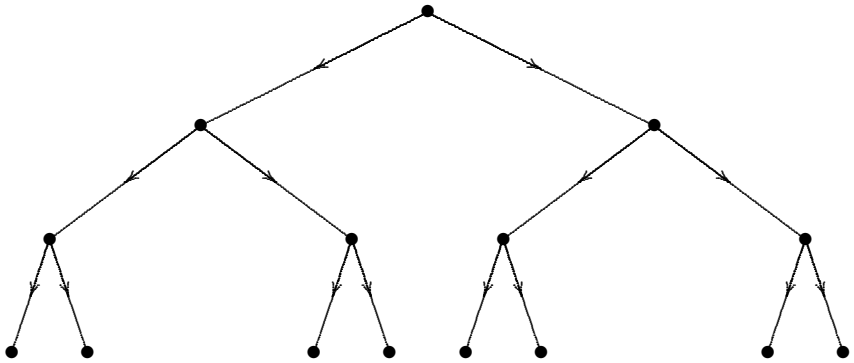
$$t = t_1 + t_2 \leq 2^{h_1} + 2^{h_2} \leq 2^{h-1} + 2^{h-1} = 2^h.$$

◁

Ví dụ 4.4.6. Cây nhị phân trong Hình 4.11 có độ cao $h = 3$ và số các nút lá là $t = 8$. Trong trường hợp này, (4.1) trở thành đẳng thức.

Giả sử ta có một tập S mà các phần tử của nó được sắp thứ tự. Chẳng hạn, nếu $S \subset \mathbb{R}$ với thứ tự thông thường; nếu S là các chuỗi ký tự, ta có thể sử dụng thứ tự từ điển. Cây nhị phân được sử dụng rất nhiều trong tin học nhằm lưu trữ các phần tử của một tập được sắp thứ tự. Giả sử tại mỗi đỉnh v ta lưu trữ dữ liệu $d(v)$. Khi đó nếu v là con trái (hoặc con phải) của w thì sẽ có một mối quan hệ thứ tự giữa $d(v)$ và $d(w)$.

Định nghĩa 4.4.7. *Cây tìm kiếm nhị phân* (binary seach tree) là một cây nhị phân trong đó dữ liệu liên kết với mỗi đỉnh. Dữ liệu được sắp xếp sao cho với mỗi đỉnh v dữ liệu trong cây con bên trái của v nhỏ hơn dữ liệu trong v; và mỗi dữ liệu trong cây con bên phải của v lớn hơn dữ liệu trong v.

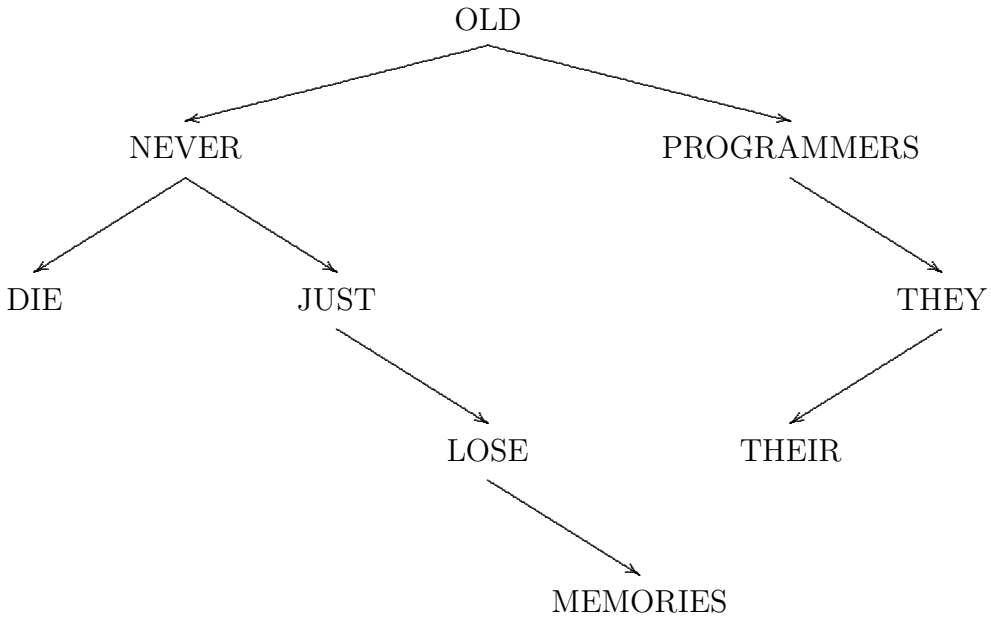


Hình 4.11:

Ví dụ 4.4.8. Chuỗi S

OLD PROGRAMMERS NEVER DIE
THEY JUST LOSE THEIR MEMORIES

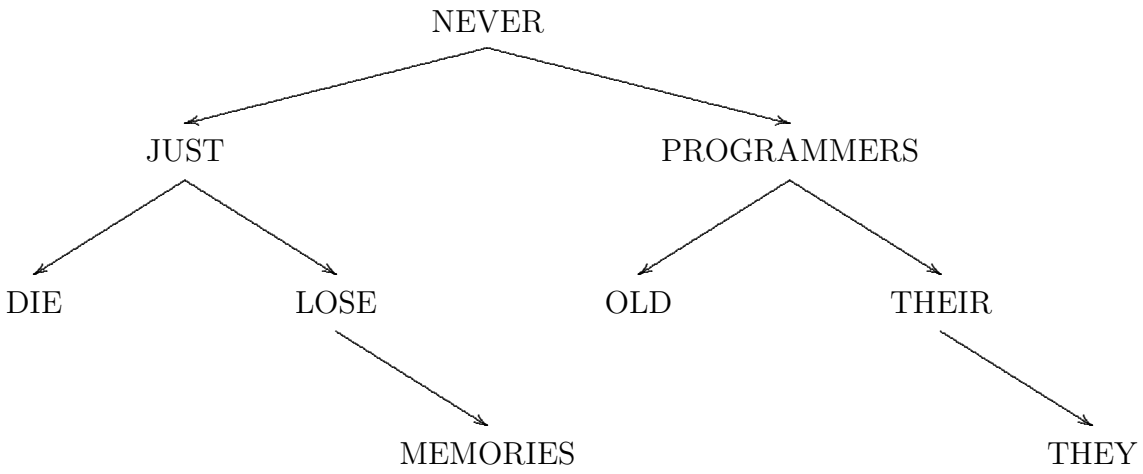
có thể đặt trong một cây tìm kiếm nhị phân như Hình 4.12.



Hình 4.12:

Nói chung, có nhiều cách đặt dữ liệu vào cây tìm kiếm nhị phân. Hình 4.13 minh

họa cây nhị phân khác lưu trữ các từ trong chuỗi S.



Hình 4.13:

Dưới đây là thuật toán xây dựng cây tìm kiếm nhị phân.

4.4.1 Thuật toán xây dựng cây tìm kiếm nhị phân

Nhập: Dãy các từ phân biệt: S.

Xuất: Cây tìm kiếm nhị phân T.

1. [Xây dựng nút gốc] Giả sử w là từ đầu tiên trong dãy S. Nếu $S = \emptyset$, đặt T là cây không đỉnh và cạnh và dừng; ngược lại, thiết lập T là cây có đúng một đỉnh (là gốc) và lưu trữ w trong gốc.

2. [Lấy ký tự tiếp] Giả sử w là ký tự kế tiếp trong S. Nếu không tồn tại, dừng.

3. [Bắt đầu tìm kiếm để lưu trữ vị trí] Giả sử v là gốc của T.

4. [Kết thúc?] Nếu w nhỏ hơn từ trong v và v không có cây con bên trái thì thêm đỉnh con bên trái vào v và lưu w vào cây con trái sau đó chuyển sang Bước 2.

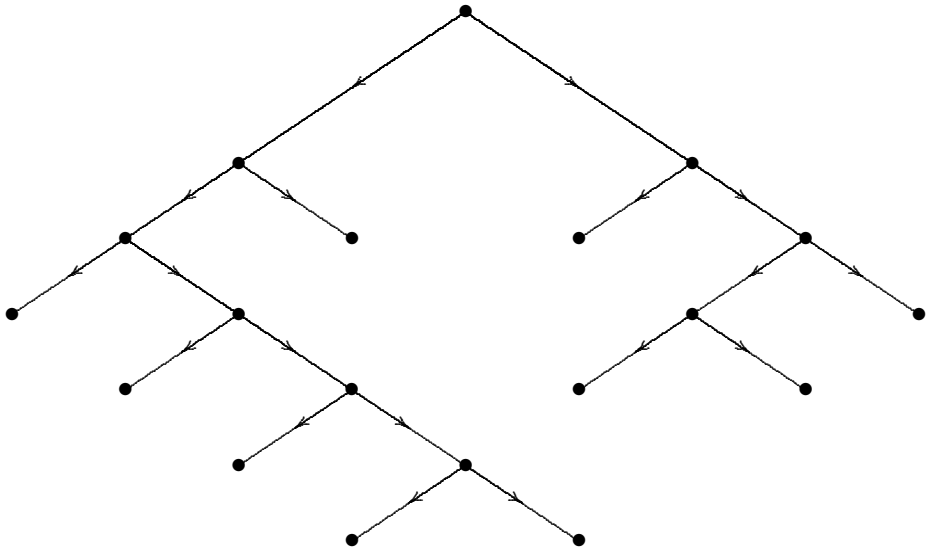
Nếu w lớn hơn từ trong v và v không có cây con bên phải, thêm đỉnh con bên phải vào v và lưu w vào sau đó chuyển sang Bước 2.

5. [Tiếp tục tìm] Nếu w nhỏ hơn từ trong v đặt v là con bên trái của v và chuyển sang Bước 4. Nếu w lớn hơn từ trong v đặt v là con bên phải của v và chuyển sang Bước 4.

Cây tìm kiếm nhị phân rất tiện lợi trong việc tìm kiếm dữ liệu. Tức là nếu cho dữ liệu D ta có thể xác định vị trí của nó D trong cây tìm kiếm nhị phân (nếu có). Để xác định dữ liệu D trong cây tìm kiếm nhị phân, chúng ta bắt đầu từ gốc. Sau đó ta lặp lại tiến trình so sánh D với dữ liệu tại nút hiện hành. Nếu D bằng dữ liệu tại nút hiện hành, tức đã tìm thấy D và thuật toán dừng. Nếu D nhỏ hơn (tương ứng, lớn hơn) dữ liệu tại nút hiện hành v ta di chuyển xuống nút con bên trái (tương ứng, bên phải) của v và lặp lại quá trình này. Tại thời điểm nào đó, ta không thể di chuyển được nữa thì kết luận D không có trong cây.

Thời gian tìm kiếm dữ liệu trong cây tìm kiếm nhị phân là tối đa khi dữ liệu không nằm trong cây và theo đó ta có dây chuyền dài nhất từ nút gốc. Do đó thời gian tối đa để tìm kiếm tỉ lệ với chiều cao của cây. Hệ quả là độ cao của cây tìm kiếm nhị phân càng nhỏ thì thời gian tìm kiếm càng ít. Có nhiều cách để cực tiểu hóa độ cao của cây (xem [9]).

Để phân tích trường hợp xấu nhất trong cây tìm kiếm nhị phân T (có n đỉnh, t đỉnh treo và độ cao h) ta gọi T^* là cây nhị phân đầy đủ nhận được từ T bằng cách thêm các nút con bên trái và bên phải (nếu cần). Chẳng hạn, Hình 4.14 là cây nhị phân đầy đủ từ cây T trong Hình 4.12. Các đỉnh thêm vào được đánh dấu $*$. Việc tìm kiếm không thành công trong T tương ứng đến các đỉnh đánh dấu $*$ trong T^*. Theo Định lý 4.4.5, $\log t \leq h$. Nhưng theo cách xây dựng, cây nhị phân đầy đủ T^* có n đỉnh trong và t đỉnh treo, nên theo Định lý 4.4.4, $t = n + 1$. Do đó trong trường hợp xấu nhất thời gian tìm kiếm ít nhất là $\log t = \log(n + 1)$. Dễ dàng thấy rằng nếu độ cao của T tối thiểu thì trường hợp xấu nhất thời gian tìm kiếm bằng $[\log(n + 1)]$.



Hình 4.14:

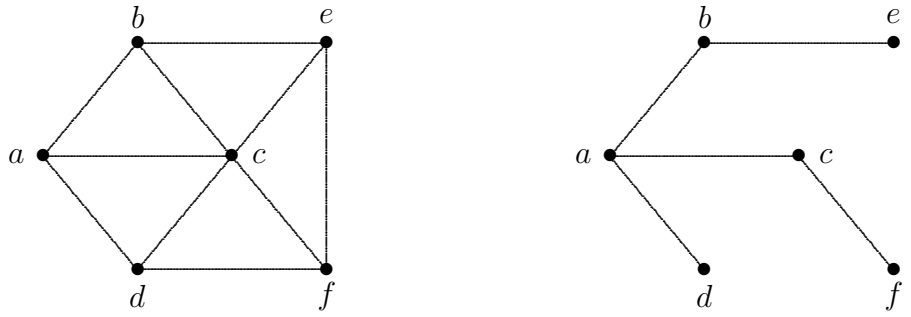
4.5 Cây bao trùm

Chúng ta đã nghiên cứu riêng biệt các tính chất của một cây, trong mục này chúng ta sẽ nghiên cứu cây khi gắn nó như một đồ thị con của một đồ thị khác. Chúng ta biết rằng cho đồ thị có m cạnh, có thể xây dựng được 2^m đồ thị con khác nhau; rõ ràng là trong số đó có một vài đồ thị con là một cây. Chúng ta quan tâm đến một loại cây đặc biệt: "cây bao trùm". Khái niệm cây bao trùm lần đầu tiên được sử dụng và phát triển lý thuyết về cây bởi nhà vật lý người Đức Kirchoff năm 1847. Kirchoff đã sử dụng cây bao trùm nhằm giải hệ các phương trình tuyến tính để xác định cường độ dòng điện trong mỗi nhánh và xung quanh mạch của một mạng điện.

Ví dụ 4.5.1. Đồ thị trong Hình 4.15(a) có cây bao trùm trong Hình 4.15(b).

Định nghĩa 4.5.2. Cây T được gọi là *cây bao trùm* của đồ thị liên thông G nếu T là đồ thị con của G và T chứa tất cả các đỉnh của G.

Định lý 4.5.3. *Đồ thị $G = (V, E)$ có đồ thị bộ phận là một cây nếu và chỉ nếu G liên thông. Nói cách khác, cho trước một đồ thị liên thông và có n đỉnh, bao giờ ta cũng có thể bỏ đi một số cạnh của G để được một cây chứa tất cả các đỉnh của G (cây có n đỉnh).*



(a) (b)

Hình 4.15:

Chứng minh. Điều kiện cần. Nếu G liên thông thì ta thử tìm xem có cạnh nào mà khi xóa đi không làm cho đồ thị mất tính liên thông không. Nếu không có một cạnh nào như vậy thì G là một cây; nếu có một cạnh như vậy thì xóa nó đi, và ta lại đi tìm một cạnh mới để xóa... Cho tới khi không thể xóa một cạnh nào được nữa thì ta có một cây mà tập hợp các đỉnh của nó đúng bằng V.

Điều kiện đủ. Giả sử a, b là hai đỉnh trong G và do đó thuộc cây bao trùm T của G. Khi đó tồn tại dây chuyền μ trong T từ a đến b. Suy ra μ cũng thuộc G. Vậy G liên thông. ◁

Chúng ta sẽ sử dụng *thuật toán tìm kiếm theo chiều rộng* để xây dựng cây bao trùm của đồ thị liên thông.

4.5.1 Thuật toán tìm kiếm theo chiều rộng xác định cây bao trùm

Trong thuật toán này, S ký hiệu là một dãy.

Nhập: Đồ thị liên thông $G := (V, E)$ với các đỉnh được đánh số thứ tự

$$v_1, v_2, \ldots, v_n.$$

Xuất: Cây bao trùm T.

1. [Khởi tạo] Đặt $S := [v_1]$ và T là đồ thị gồm đỉnh v_1 và không có cạnh. Ký hiệu v_1 là đỉnh gốc.

2. [Thêm cạnh] Với mỗi $x \in S$, theo thứ tự, thêm cạnh $(x, y) \in E$ và đỉnh y (theo thứ tự) vào T nếu $T \cup (x, y)$ không tạo thành chu trình. Nếu không có cạnh như vậy, dừng. T là cây bao trùm.

3. [Cập nhật S] Thay S bởi con (trong T) của S theo thứ tự. Chuyển sang Bước 2.

Để tìm cây bao trùm của đồ thị liên thông ta còn có thể dùng thuật toán *tìm kiếm theo chiều sâu* (còn gọi là *quay lui*) như sau:

4.5.2 Thuật toán tìm kiếm theo chiều sâu xác định cây bao trùm

Nhập: Đồ thị liên thông $G := (V, E)$ với các đỉnh được đánh số thứ tự

$$v_1, v_2, \ldots, v_n.$$

Xuất: Cây bao trùm T.

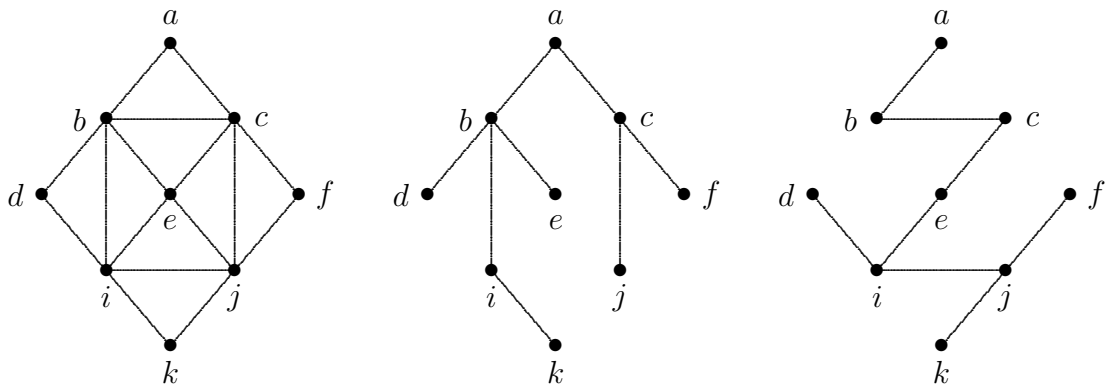
1. [Khởi tạo] Đặt $w := v_1$ và T là đồ thị gồm đỉnh v_1 và không có cạnh. Ký hiệu v_1 là đỉnh gốc.

2. [Thêm cạnh] Chọn cạnh (w, v_k) với chỉ số k nhỏ nhất sao cho việc thêm cạnh này vào T không tạo ra chu trình. Nếu không tồn tại, chuyển sang Bước 3. Ngược lại, thêm cạnh (w, v_k) và đỉnh v_k vào T; đặt $w := v_k$ và chuyển sang Bước 2.

3. [Kết thúc?] Nếu $w = v_1$, thuật toán dừng, T là cây bao trùm.

4. [Quay lui] Đặt x là cha của w (trong T); gán $w := x$ và chuyển sang Bước 2.

Ví dụ 4.5.4. Đồ thị trong Hình 4.16(a) có các cây bao trùm, Hình 4.16(b) và 4.16(c), được xây dựng theo các thuật toán tìm kiếm theo chiều rộng và chiều sâu tương ứng.



Hình 4.16: (a) Đồ thị G. (b) Cây bao trùm sinh bởi thuật toán tìm kiếm theo chiều rộng. (c) Cây bao trùm sinh bởi thuật toán tìm kiếm theo chiều sâu.

4.5.3 Tìm cây bao trùm dựa trên hai mảng tuyến tính

Để cài đặt các thuật toán tìm kiếm theo chiều rộng và chiều sâu trên đồ thị liên thông G tìm cây bao trùm T ta có thể dùng cấu trúc dữ liệu ma trận kề hay cải biên, mảng các danh sách kề $V\_out[]$. Tuy nhiên, trong trường hợp đồ thị được biểu diễn bởi hai mảng tuyến tính α và β thì cách tiếp cận sau sẽ hiệu quả hơn. Ngoài ra, phương pháp sau này cũng cho ta một "rừng" (tập các cây bao trùm) chứa $(n-p)$ cạnh trong trường hợp đồ thị có $p > 1$ thành phần liên thông. Hiển nhiên, với những thuật toán xây dựng cây bao trùm ta có thể kiểm tra đồ thị có liên thông hay không, và nếu nó không liên thông thì có thể xác định các thành phần liên thông. Nếu mặt khác, đồ thị có trọng số thì chúng ta có thể tìm cây bao trùm có tổng trọng lượng nhỏ nhất (xem Phần 4.6). Hơn nữa chúng ta cũng có thể xây dựng hệ các chu trình độc lập dựa trên cây bao trùm của đồ thị như trong Phần 4.5.4.

Xét đồ thị vô hướng G không có khuyên n đỉnh và m cạnh. Các đỉnh được gán nhãn $v_1, v_2, \ldots, v_n$, và đồ thị xác định bởi hai mảng tuyến tính α, β, trong đó α_i và $\beta_i, i = 1, 2, \ldots, m$, là các đỉnh được liên thuộc bởi cạnh e_i.

Mỗi bước lặp của thuật toán, một cạnh mới được đưa vào kiểm tra để xác định các đỉnh của cạnh đó có xuất hiện trong cây nào đó (đã được thiết lập ở bước trước; bước đầu tiên chúng ta chưa có cây bao trùm nào). Ở bước thứ $i, 1 \leq i \leq m$, khi kiểm

tra cạnh (α_i, β_i) có năm trường hợp xảy ra:

1. Nếu cả hai đỉnh không nằm trong bất cứ một cây nào đã được xây dựng ở $(i-1)$ bước trước, khi đó các đỉnh α_i, β_i được gán số thành phần liên thông là số c, sau đó tăng c lên một đơn vị.

2. Nếu α_i thuộc cây T_j còn β_i thuộc cây T_k $(j, k = 1, \ldots, c$ và $j < k)$, cạnh thứ i được sử dụng để nối hai cây này; do đó, mọi đỉnh trong cây T_k được gán là số thành phần liên thông của T_j. Giá trị c giảm một đơn vị.

3. Nếu cả hai đỉnh cùng nằm trong một cây, thì cạnh (α_i, β_i) cùng với một số cạnh khác của cây sẽ tạo thành một chu trình và do đó không xử lý trường hợp này.

4. Nếu đỉnh α_i thuộc cây T_j còn β_i không thuộc cây nào, khi đó cạnh (α_i, β_i) được thêm vào cây T_j bằng cách gán số thành phần liên thông của T_j cho đỉnh β_i.

5. Nếu đỉnh β_i thuộc cây T_k còn α_i không thuộc cây nào, khi đó cạnh (α_i, β_i) được thêm vào cây T_k bằng cách gán số thành phần liên thông của T_k cho đỉnh α_i.

Để thực hiện việc kiểm tra các đỉnh cuối của một cạnh được khảo sát có xuất hiện trong cây nào không, chúng ta xây dựng một mảng tuyến tính n phần tử Vertex[]. Khi một cạnh (v_i, v_j) nằm trong cây thứ c thì các phần tử thứ i và j của mảng này được đặt là c.

Trong các quá trình xử lý kế sau, khi một cạnh khác (α_i, β_i) được đưa vào kiểm tra, chúng ta chỉ cần kiểm tra các phần tử thứ α_i và β_i trong mảng Vertex[] có khác 0 không. Phần tử thứ q trong mảng Vertex[] bằng 0 chỉ ra rằng đỉnh thứ q này không nằm trong bất cứ cây nào. Kết thúc chương trình, mảng Vertex[] cho chúng ta biết các thành phần liên thông của đồ thị G.

Nhận xét rằng, cây không chỉ được mô tả bởi tập các đỉnh. Bởi vậy, chúng ta cần có một mảng các cạnh để xuất dữ liệu. Đặt mảng này là Edge[]. Nếu cạnh thứ i nằm trong cây thứ c, ta có Edge$[k] = c$; ngược lại, nó được đặt bằng 0. Tất cả các phần tử 0 trong mảng này tương ứng với các khuyên cô lập (tức là các cạnh không nằm trong bất cứ cây bao trùm hay rừng nào). Mảng này cùng với các mảng α và β, xác định duy nhất cây bao trùm (hoặc rừng) được sinh bởi thuật toán này.

Trong thuật toán này, vòng lặp chính thực hiện m lần. Thời gian đòi hỏi để kiểm tra các đỉnh có xuất hiện trong cây hay không là hằng số-không phụ thuộc vào n và m. Do đó thời gian thực hiện thuật toán tỉ lệ với m[1]. Trong trường hợp $m \gg n$, ta có thể giảm thời gian thực hiện bằng cách lưu trữ một biến đếm số các cạnh được đặt vào cây. Khi biến này đạt giá trị $(n-1)$ chương trình sẽ kết thúc (nếu đồ thị liên thông; trái lại ta cần kiểm tra mọi cạnh).

Thủ tục sau minh họa thuật toán tìm cây bao trùm dựa trên hai mảng tuyến tính $\alpha[]$ và $\beta[]$:

```
void SpanningTree()
{
  byte i, j, Tempt, Count = 0;
  byte Vertex[MAXVERTICES], Edge[MAXEDGES];

  for (j = 1; j <= NumVertices; j++) Vertex[j] = 0;
  for (i = 1; i <= NumEdges; i++) Edge[i] = 0;

  for (i = 1; i <= NumEdges; i++)
  {
    if (Vertex[alpha[i]] == 0)
    {
      if (Vertex[beta[i]] == 0)
      {
        Count++;
        Edge[i]          = Count;
        Vertex[alpha[i]] = Count;
        Vertex[beta[i]]  = Count;
      }
      else
      {
        Edge[i]          = Vertex[beta[i]];
```

[1]Thời gian đòi hỏi để trộn hai cây T_i và T_j được thực hiện trong ngôn ngữ C không phụ thuộc vào n. Tuy nhiên, có những thuật toán trộn rất hiệu quả cho phép thực hiện tiến trình này.

```
    Vertex[alpha[i]] = Vertex[beta[i]];
  }
}
else
{
  if (Vertex[beta[i]] == 0)
  {
    Edge[i]         = Vertex[alpha[i]];
    Vertex[beta[i]] = Vertex[alpha[i]];
  }
  else
  {
    if (Vertex[alpha[i]] < Vertex[beta[i]])
    {
      Edge[i] = Vertex[alpha[i]];
      Tempt   = Vertex[beta[i]];

      for (j = 1; j <= NumEdges; j++)
      {
        if (Edge[j] == Tempt)
        {
          Edge[j]          = Vertex[alpha[i]];
          Vertex[alpha[j]] = Vertex[alpha[i]];
          Vertex[beta[j]]  = Vertex[alpha[i]];
        }
      }
    }
    else if (Vertex[alpha[i]] > Vertex[beta[i]])
    {
      Edge[i] = Vertex[beta[i]];
      Tempt   = Vertex[alpha[i]];

      for (j = 1; j <= NumEdges; j++)
      {
```

```
            if (Edge[j] == Tempt)
            {
              Edge[j]          = Vertex[beta[i]];
              Vertex[alpha[j]] = Vertex[beta[i]];
              Vertex[beta[j]]  = Vertex[beta[i]];
            }
          }
        }
        else Tempt = 0;

        if (Tempt == Count)
        {
          Count--;
        }
        else if (Tempt > 0)
        {
          for (j = 1; j <= NumEdges; j++)
          {
            if (Edge[j] == Count)
            {
              Edge[j]          = Tempt;
              Vertex[alpha[j]] = Tempt;
              Vertex[beta[j]]  = Tempt;
            }
          }
          Count--;
        }
      }
    }
  }

printf("So thanh phan lien thong la %d", Count);
for (j = 1; j <= Count; j++)
{
```

```
    printf("\n Cay bao trum thu %d gom cac canh:", j);
    for (i = 1; i <= NumEdges; i++)
    if (Edge[i] == j) printf("\n %d %d ", alpha[i], beta[i]);
  }
}
```

4.5.4 Thuật toán tìm tất cả các cây bao trùm

Việc phân tích các mạch điện về cơ bản có thể đưa về bài toán tìm tất cả các cây bao trùm của đồ thị (xem [19]). Do tầm quan trọng của nó, có nhiều thuật toán khác nhau giải quyết bài toán này. Một trong những phương pháp là hoán đổi các chu trình như sau: Xuất phát từ một cây bao trùm T nào đó. Với mỗi cạnh không nằm trên cây T, khi thêm vào cây này sẽ tạo ra duy nhất một chu trình. Xóa bỏ một cạnh bất kỳ trong chu trình này sẽ cho ta một cây bao trùm mới.

Do số các cây bao trùm là rất lớn thậm chí cả với những đồ thị nhỏ, tính hiệu quả của những thuật toán giải quyết bài toán rất quan trọng (xem [14]). Một tổng quan của các phương pháp này là một luận án tiến sĩ của Chase [12]. Chase đã chỉ ra rằng thuật toán đưa ra bởi Minty có hiệu quả nhất: liên tiếp thu gọn đồ thị bằng các phép toán xóa một cạnh và hợp nhất các đỉnh đầu cuối của nó. Từ các cây bao trùm của các đồ thị thu gọn (mà nhỏ hơn rất nhiều) ta có thể dựng được tất cả các cây bao trùm của đồ thị ban đầu. Để bảo đảm thuật toán kết thúc, các đồ thị có kích thước dưới một ngưỡng cho trước sẽ không được thu gọn thêm; thay vào đó là các cây bao trùm của chúng được suy ra.

4.5.5 Hệ cơ sở của các chu trình độc lập

Xét đồ thị vô hướng G có n đỉnh, m cạnh, p thành phần liên thông. Ta biết rằng chu số $\nu(G) = m - n + p$ là số cực đại các chu trình độc lập. Phần dưới đây ta xây dựng thuật toán tìm hệ cơ sở của các chu trình độc lập dựa trên cây bao trùm của đồ thị. Không giảm tổng quát, có thể giả thiết đồ thị G liên thông, vì trong trường hợp trái lại, thì ta xét từng thành phần liên thông. Ý tưởng ở đây là

1. Xây dựng cây bao trùm $T := (V, E_T)$.

2. Giả sử $e_1, e_2, \ldots, e_{m-n+1}$ là các cạnh của E nhưng không thuộc cây T. Khi thêm một cạnh bất kỳ trong các cạnh này, chẳng hạn cạnh e_k, vào cây T ta được một và chỉ một chu trình μ_k. Các chu trình $\mu_1, \mu_2, \ldots, \mu_{m-n+1}$ là độc lập vì mỗi chu trình chứa một cạnh không thuộc các chu trình kia; mặt khác ta có $(m-n+1) = \nu(G)$ chu trình như vậy, nên ta đã xác định được hệ cơ sở của các chu trình độc lập.

Như vậy bài toán đưa về tìm các chu trình μ_k khi thêm cạnh e_k vào cây T. Trong khi kiểm tra cạnh $e_k = (\alpha_k, \beta_k)$ trong Thuật toán 4.5.3, nếu điều kiện 3 đúng (tức là cả hai đỉnh α_k và β_k xuất hiện trong cùng cây T_i) thì thay cho việc loại bỏ cạnh này chúng ta cần tìm các cạnh trong T_i mà tạo thành dây chuyền giữa hai đỉnh α_k và β_k. Dây chuyền này cùng với cạnh (α_k, β_k) tạo thành một chu trình cơ bản. Vấn đề chính ở đây là tìm dây chuyền này. Có nhiều phương pháp hiệu quả giải quyết bài toán, trong số đó thuật toán của Paton [50] là hiệu quả nhất.

Thuật toán Paton tìm hệ cơ sở của các chu trình độc lập

Chúng ta cũng sẽ kiểm tra mỗi cạnh có tạo thành chu trình với những cạnh trong cây được xây dựng trong quá trình thực hiện thuật toán, nhưng thay việc lấy các cạnh theo thứ tự tùy ý (như trong Thuật toán 4.5.3), ta chọn một đỉnh z và kiểm tra các cạnh liên thuộc với nó. (Đỉnh z là đỉnh vừa được thêm vào cây). Để đơn giản, ta sẽ sử dụng ma trận kề A biểu diễn đồ thị. Ký hiệu T là tập hiện hành các đỉnh trong cây được xây dựng ở bước nào đó và W là tập các đỉnh chưa được kiểm tra (tức là những đỉnh thuộc T hoặc không nhưng có ít nhất một cạnh liên thuộc với nó chưa được kiểm tra).

Chúng ta khởi đầu thuật toán bằng cách đặt $T = \{v_1\}$ và $W = V$. Đỉnh v_1 sẽ là gốc của cây. Sau quá trình khởi tạo, thực hiện các bước sau:

1. Nếu $T \cap W = \emptyset$ thuật toán dừng.

2. Nếu $T \cap W \neq \emptyset$ chọn đỉnh $z \in T \cap W$.

3. Kiểm tra z bằng cách xét mỗi cạnh liên thuộc với nó. Nếu không có cạnh nào, khử z từ W và chuyển sang Bước 1. Ngược lại, giả sử tồn tại đỉnh $p \in V$ sao cho

$(z, p) \in E$.

4. Nếu $p \in T$ tìm chu trình sơ cấp gồm cạnh (z, p) và một dây chuyền (duy nhất) từ z đến p trong cây đang được xây dựng. Xóa cạnh (z, p) và chuyển sang Bước 3.

5. Nếu $p \notin T$ thêm cạnh (z, p) vào cây và đỉnh p vào T. Xóa cạnh (z, p) và chuyển sang Bước 3.

Như đã đề cập, vấn đề cơ bản là Bước 5. Chúng ta phải tìm dây chuyền từ z đến p như thế nào? Phân tích dưới đây trả lời câu hỏi này.

Chúng ta sử dụng một ngăn xếp (stack) $TW = T \cap W$ để lưu trữ các đỉnh trong cây nhưng chưa được kiểm tra. Đỉnh được thêm gần đây nhất vào cây sẽ được chèn vào đầu ngăn xếp. Đỉnh được kiểm tra được lấy ra khỏi từ ngăn xếp. Ta sử dụng thêm hai mảng tuyến tính độ dài n: mảng $l[j]$ là khoảng cách từ gốc v_1 đến đỉnh v_j trong cây bao trùm; và Pred$[j]$ là đỉnh v_i sao cho cạnh (v_i, v_j) là một cạnh trong cây với v_i gần gốc hơn v_j. Nói cách khác, Pred$[j]$ là đỉnh liền trước đỉnh v_j trong dây chuyền từ v_1 đến v_j. Nhãn $l[j] = -1$ nếu và chỉ nếu đỉnh v_j không thuộc tập T. Khởi tạo $l[1] = 0$ và $l[j] = -1$ với mọi $j = 2, 3, \ldots, n$.

Trong Bước 5, khi xét đỉnh z và cạnh (z, p) được tìm thấy với $p \in T$. Để xác định chu trình cơ bản sinh bởi cạnh (z, p) chúng ta lần theo dây chuyền từ z đến p trong cây bằng cách tìm liên tiếp các "tiền bối" Pred$[z]$, Pred[Pred$[z]$], ... cho đến khi chúng ta bắt gặp Pred$[p]$-tiền bối của p. Nói cách khác, như chỉ ra trong Hình 4.17, chu trình cơ bản được tạo ra là

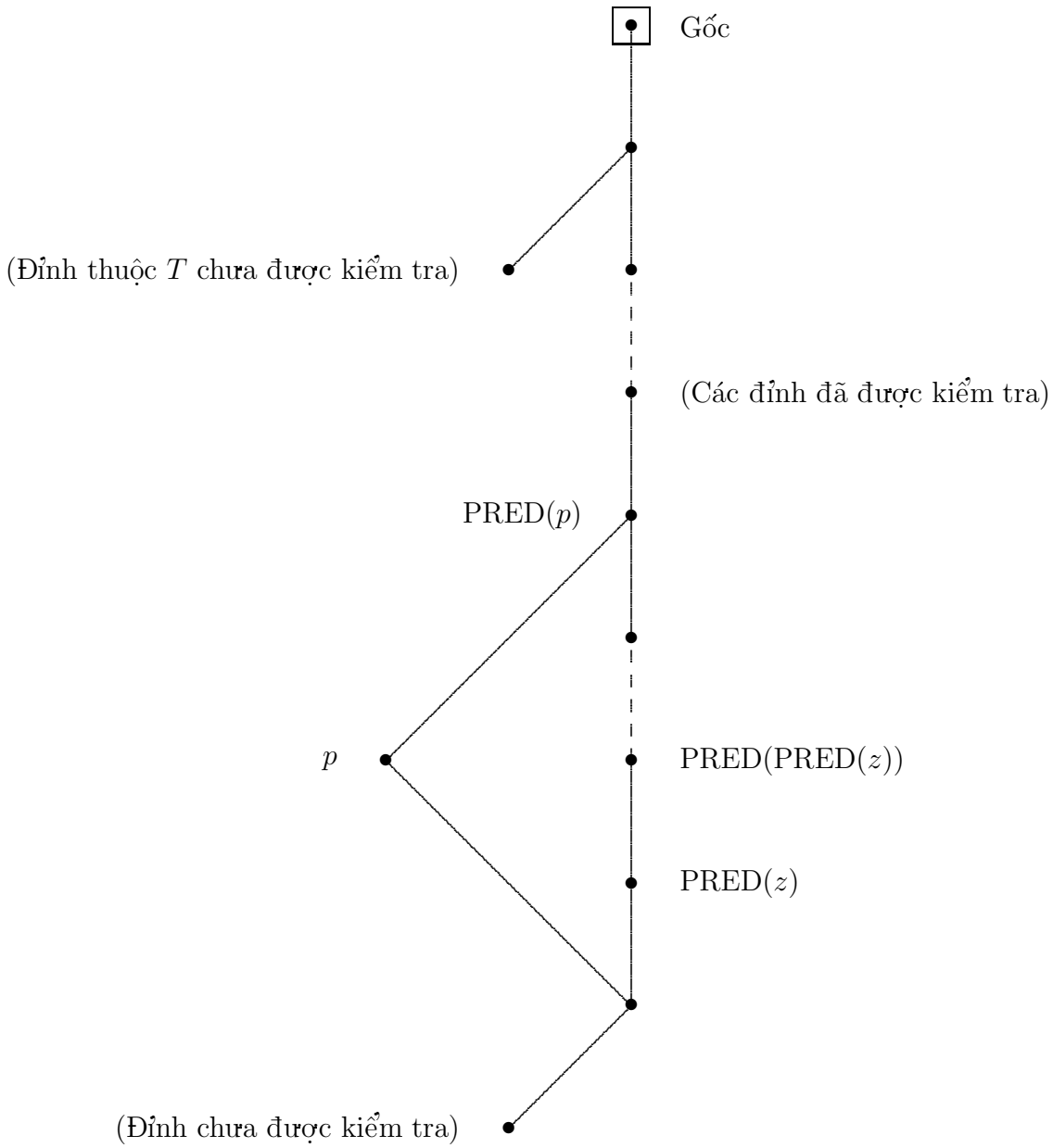
$$z, \text{Pred}[z], \text{Pred}[\text{Pred}[z]], \ldots, \text{Pred}[p], p, z.$$

Chú ý rằng tiền bối Pred$[k]$ của mọi đỉnh $k \in T$ là một đỉnh mà hoặc đã được kiểm tra hoặc đang được kiểm tra. Tức là, nếu $k \in T \cap W$ thì

$$\text{Pred}[k] \notin W \quad \text{nhưng} \quad \text{Pred}[k] \in T.$$

Thủ tục FundamentalCircuits() dưới đây minh họa các bước đã trình bày ở trên.

```
void FundamentalCircuits(byte Start, byte **A, byte NumVertices)
```



Hình 4.17: Xây dựng chu trình cơ bản.

```
{
  Path Pred;
  byte i, z, p, nu_G = 0;
  Boolean Sum;
  AdjPointer TW;
  int T[MAXVERTICES];

  for (i = 1; i <= NumVertices; i++) T[i] = -1;
  T[Start] = 0;

  Create(&TW);
  Push(&TW, Start);

  while (!Empty(TW))
  {
    Pop(&TW, &z);

    printf("\n z = %d", z);
    getch();

    for (p = 1; p <= NumVertices; )
    if (A[z][p] > 0)
    {
      printf (", p = %d", p);
      if (T[p] == -1)
      {
        printf (": not in T (-> T)");
        Push(&TW, p);
        Pred[p] = z;
        T[p]      = 0;
      }
      else
      {
        printf (": in T(%d: ", ++nu_G);
```

```
        i = z;
        while (i != Pred[p])
        {
          printf("%d->", i);
          i = Pred[i];
        }
        printf("%d->%d->%d)", Pred[p], p, z);
      }
      getch();
      A[z][p]--;
      A[p][z]--;
    }
    else p++;
  }
}
```

Thời gian thực hiện của thuật toán bị chặn trên bởi n^ν trong đó giá trị thực $\nu \in [2, 3]$ phụ thuộc vào cấu trúc của đồ thị cũng như cách đánh nhãn các đỉnh [50].

Mặc dù để đơn giản chúng ta đã giả sử G liên thông, tuy nhiên thuật toán có thể dễ dàng cải biên cho trường hợp tổng quát. Đầu tiên, thuật toán sẽ tạo ra tất cả các chu trình cơ bản trong thành phần liên thông chứa đỉnh xuất phát v_1. Sau đó ta chọn đỉnh y mà $l[y] = -1$ và bắt đầu với y là gốc của cây bao trùm thứ hai. Thuật toán lặp lại cho đến khi tất cả các đỉnh được gán nhãn l khác -1.

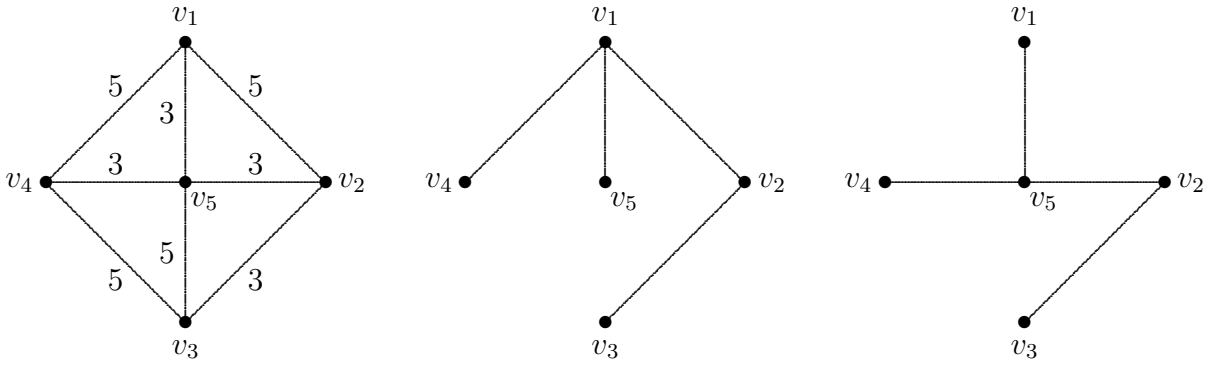
4.6 Cây bao trùm tối thiểu

Định nghĩa 4.6.1. Giả sử G là đồ thị có trọng số. *Cây bao trùm tối thiểu* của G là một cây bao trùm của G với trọng lượng nhỏ nhất.

Bài toán này rất hay gặp trong thông tin liên lạc và trong các ngành khác. Chẳng hạn ta đặt câu hỏi như sau: độ dài dây điện ngắn nhất cần thiết để nối n thành phố đã

định là bao nhiêu? Khi đó coi các thành phố là các đỉnh của đồ thị và $w(a, b)$ là khoảng cách tính bằng km giữa các thành phố a và b. Mạng dây điện cần phải liên thông, và vì độ dài dây điện cần ngắn nhất nên nó không có chu trình: vậy mạng đó là một cây. Ở đây ta cần tìm một cây "tối thiểu" có thể được và là một đồ thị bộ phận của đồ thị đầy đủ có n đỉnh. Một ứng dụng gián tiếp: bài toán tìm cây bao trùm tối thiểu là bước trung gian trong việc tìm lời giải của bài toán người du lịch-một bài toán thường xuất hiện trong thực tế.

Cần chú ý rằng, cây bao trùm tối thiểu của đồ thị không liên quan đến cây sinh bởi tất cả các dây chuyền ngắn nhất từ một đỉnh cho trước. Do đó, đồ thị trong Hình 4.18(a), với các số trên các cạnh tương ứng các trọng lượng cạnh, cây sinh ra bởi đường đi ngắn nhất từ đỉnh cho trước, chẳng hạn v_1, trong Hình 4.18(b); ngược lại, cây bao trùm tối thiểu cho trong Hình 4.18(c).



(a) (b) (c)

Hình 4.18: (a) Đồ thị G. (b) Cây sinh bởi đường đi ngắn nhất xuất phát từ v_1. (c) Cây bao trùm nhỏ nhất.

Tìm cây bao trùm tối thiểu là một trong những bài toán của lý thuyết đồ thị có thể giải quyết triệt để. Do đó, đặt T_i và T_j là hai cây con được tạo ra trong quá trình xây dựng cây bao trùm tối thiểu. Nếu T_i được sử dụng để biểu diễn tập các đỉnh của cây con thì Δ_{ij} có thể định nghĩa là khoảng cách ngắn nhất từ một đỉnh trong T_i đến

một đỉnh trong T_j; tức là

$$\Delta_{ij} := \min_{v_i \in T_i}[\min_{v_j \in T_j}\{w(v_i, v_j)\}], \qquad i \neq j. \tag{4.5}$$

Khi đó, dễ dàng chứng minh rằng lặp lại phép toán sau sẽ cho ta cây bao trùm tối thiểu:

Phép toán I. Với cây con T_s nào đó, tìm cây con T_{j^*} sao cho $\Delta_{sj^*} = \min_{T_j}[\Delta_{sj}]$, và đặt (v_s, v_{j^*}) là cạnh tương ứng giá trị Δ_{sj^*} trong (4.5) đạt được. Khi đó (v_s, v_{j^*}) thuộc cây bao trùm tối thiểu và có thể được thêm vào với các cạnh khác trong quá trình lặp để tạo ra cây bao trùm tối thiểu.

Thật vậy, giả sử các cạnh trong các cây con ở bước k nào đó thuộc cây bao trùm tối thiểu T_m ở bước cuối cùng. Giả sử cạnh (v_s, v_{j^*}) được chọn như trên không thuộc cây bao trùm tối thiểu T_m. Theo định nghĩa, cây con T_s ở bước cuối cùng được nối với cây con khác nào đó bằng cạnh (v_i, v_j) trong đó $v_i \in T_s$ và $v_j \notin T_s$ và ngoài ra T_s phải chứa trong cây bao trùm tối thiểu T_m. Xóa cạnh (v_i, v_j) khỏi cây T_m sẽ cho ta hai thành phần liên thông và thay nó bởi cạnh (v_s, v_{j^*}) sẽ tạo một cây mới có trọng lượng nhỏ hơn trọng lượng cây T_m, mâu thuẫn. Vậy, với những giả thiết trên, ta có thể thêm cạnh (v_s, v_{j^*}) để tạo thành cây (chứa trong cây bao trùm tối thiểu) ở bước k và tiếp tục với bước $(k+1)$. Cũng cần chú ý rằng, phương pháp trên không phụ thuộc vào cây T_s được chọn. Hơn nữa, do bước khởi tạo (tức là trước khi bất cứ cạnh nào được chọn) giả thiết là không tồn tại (và do đó đúng) nên lặp lại thuật toán trên cuối cùng sẽ tạo ra cây bao trùm tối thiểu.

Hầu hết các phương pháp tìm cây bao trùm tối thiểu đều dựa trên những trường hợp đặc biệt của thủ tục trên. Đầu tiên, trong số đó là phương pháp của Kruskal [39] như sau.

4.6.1 Thuật toán Kruskal

Ý tưởng của thuật toán Kruskal tìm cây bao trùm trong đồ thị liên thông có trọng số G như sau: Khởi tạo với đồ thị T gồm tất cả các đỉnh của G và không có cạnh. Tại

mỗi bước lặp chúng ta thêm một cạnh có trọng lượng nhỏ nhất lên cây T mà không tạo thành chu trình trong T. Thuật toán dừng khi T có $(n-1)$ cạnh.

1. [Khởi tạo] Giả sử T là đồ thị gồm n đỉnh và không có cạnh.

2. [Sắp xếp] Sắp xếp thứ tự các cạnh của đồ thị G theo thứ tự trọng lượng tăng dần.

3. [Thêm cạnh] Thêm cạnh (bắt đầu từ cạnh đầu tiên) trong danh sách các cạnh được sắp xếp vào cây T sao cho không tạo thành chu trình trong T khi thêm. (Cạnh được thêm vào gọi là *chấp nhận được).*

4. [Kết thúc] Nếu T có $(n-1)$ cạnh thì thuật toán dừng; T là cây bao trùm tối thiểu.

Thuật toán này thêm vào cây T những cạnh có trọng lượng nhỏ nhất chấp nhận được hơn là thêm cạnh giữa một cây con của T, chẳng hạn T_s, và một cây con nào khác (như chỉ ra trong Phép toán I). Hiển nhiên cạnh được chọn có trọng lượng nhỏ nhất giữa một cây con *nào đó* và bất kỳ cây con khác, nên nguyên tắc chọn của thuật toán Kruskal là một trường hợp đặc biệt của Phép toán I. Tuy nhiên có thể xảy ra trường hợp cạnh e được kiểm tra để thêm vào liên thuộc giữa hai đỉnh của cùng một cây con và do đó nó sẽ tạo ra một chu trình nếu thêm vào; tức là cạnh e là không chấp nhận được. Vì vậy, trong Bước 3, các cạnh cần được kiểm tra tính chấp nhận của nó trước khi thêm vào T. Tiến trình kiểm tra này có thể thực hiện hiệu quả bằng cách sử dụng thuật toán xây dựng cây bao trùm dựa trên hai mảng tuyến tính như đã trình bày trong Phần 4.5.3.

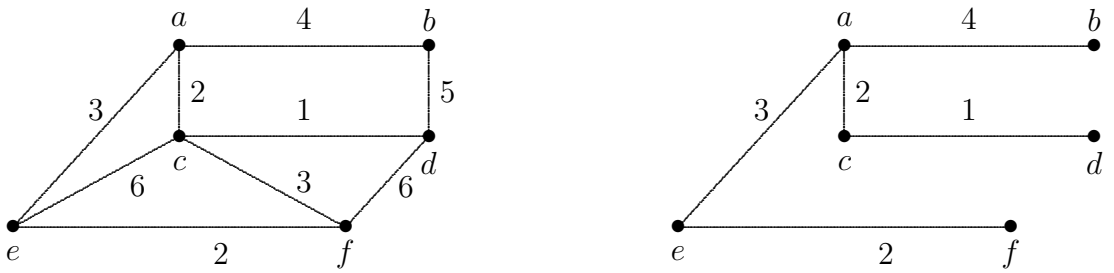
Ví dụ 4.6.2. Xét đồ thị trong Hình 4.19(a). Sắp xếp các cạnh theo trọng lượng tăng dần (sử dụng hai mảng tuyến tính α và β) ta được

k	1	2	3	4	5	6	7	8	9
α	c	a	e	a	c	a	b	c	d
β	d	c	f	e	f	b	d	e	f
Trọng lượng	1	2	2	3	3	4	5	6	6

Các cạnh (không tạo thành chu trình) được thêm vào cây T theo thứ tự là

$$(c,d),(a,c),(e,f),(a,e),(a,b).$$

T là cây bao trùm nhỏ nhất có trọng lượng 12 (Hình 4.19(b)).



Hình 4.19:

Bổ đề 4.6.3. *Nếu $K_n = (V, E)$ là đồ thị đầy đủ, và nếu tất cả các các trọng lượng của các cạnh khác nhau thì tồn tại duy nhất một cây bao trùm tối thiểu $T = (V, E_T)$.*

Chứng minh. Ký hiệu $E_k := \{e_1, e_2, \ldots, e_k\}$ là tập các cạnh được thêm vào cây T trong Thuật toán 4.6.1 ở bước lặp thứ $k, 1 \leq k \leq n-1$. Hiển nhiên theo cách xây dựng, T là đồ thị có $(n-1)$ cạnh và không có chu trình nên T là cây bao trùm của K_n.

Giả sử $T' = (V, E_{T'})$ là cây bao trùm tối thiểu, ta chứng minh $E_{T'} = E_{n-1}$. Thật vậy, giả sử ngược lại tồn tại chỉ số k nhỏ nhất sao cho cạnh e_k không thuộc $E_{T'}$. Khi đó theo tính chất của cây, tồn tại một và chỉ một chu trình μ trong $T' \cup \{e_k\}$. Trên chu trình này có một cạnh e_0 mà $e_0 \notin E_{n-1}$, vì nếu ngược lại tồn tại một chu trình, là μ, trong cây $T-$mâu thuẫn. Nếu đặt $E_{T''} := (E_{T'} \cup \{e_k\}) \setminus \{e_0\})$ thì đồ thị $T'' := (V, E_{T''})$ không có chu trình và có $(n-1)$ cạnh nên nó là một cây.

Mặt khác $E_{k-1} \cup \{e_0\} \subset E_{T''}$ nên $E_{k-1} \cup \{e_0\}$ không chứa chu trình. Suy ra

$$w(e_0) > w(e_k).$$

Nhưng cây T'' nhận được từ cây T' bằng cách thay cạnh e_0 thành cạnh e_k nên $W(T'') < W(T')$. Mâu thuẫn vì T' là cây bao trùm tối thiểu. ◁

Định lý 4.6.4. *Thuật toán Kruskal là đúng; tức là, kết thúc thuật toán T là cây bao trùm tối thiểu.*

Chứng minh. Thật vậy trước hết ta thu xếp để mọi cạnh đều có độ dài khác nhau; chẳng hạn nếu $w(e_1) = w(e_2) = \cdots = w(e_s)$ thì thực hiện phép biến đổi:

$$\begin{aligned} w(e_1) &= w(e_1) + \epsilon, \\ w(e_2) &= w(e_2) + \epsilon^2, \\ &\vdots \\ w(e_s) &= w(e_s) + \epsilon^s, \end{aligned}$$

trong đó ϵ là số dương đủ bé sao cho không làm đảo lộn thứ tự về quan hệ giữa trọng lượng của các cạnh.

Hơn nữa, ta cũng có thể thêm các cạnh f với trọng lượng đủ lớn $w(f) > \sum_{e\in E} w(e)$ và khác nhau sao cho đồ thị nhận được $K_n = (V, E')$ là đầy đủ.

Theo Bổ đề 4.6.3 tồn tại duy nhất một cây bao trùm tối thiểu T trong đồ thị K_n. Mặt khác, mọi cây bao trùm của đồ thị G có trọng lượng không vượt quá $\sum_{e\in E} w(e)$ và mọi cây bao trùm của G cũng là cây bao trùm của K_n. Suy ra T là cây bao trùm tối thiểu của G.

◁

Nhận xét rằng, có thể dùng phương pháp đối ngẫu: loại dần các cạnh có trọng lượng lớn nhất của đồ thị mà không làm mất tính liên thông của nó cho đến khi không thể loại cạnh được nữa.

Độ phức tạp tính toán của thuật toán Kruskal phụ thuộc vào Bước 2: đồ thị có m cạnh cần $m \log_2 m$ phép toán để thực hiện sắp xếp mảng theo trọng lượng tăng dần. Tuy nhiên, nói chung, ta không cần duyệt toàn bộ mảng vì cây bao trùm tối thiểu gồm $(n-1)$ cạnh chấp nhận được nên chỉ cần kiểm tra $r < m$ phần tử đầu tiên của mảng. Do đó ta có thể cải tiến thuật toán này để tăng tốc độ thực hiện (xem [14], [36]).

Mặc dù có những cải tiến, nhưng thuật toán Kruskal chỉ thích hợp với những đồ thị tương đối thưa. Với những đồ thị khác, chẳng hạn đồ thị đầy đủ có số cạnh

$m = n(n-1)/2$, Prim [49] và Dijkstra [20] đã đưa ra những thuật toán khác dựa trên việc đặc biệt hóa hiệu quả hơn Phép toán I.

4.6.2 Thuật toán Prim

Thuật toán này xây dựng cây T bằng cách liên tiếp thêm các cạnh có trọng lượng nhỏ nhất liên thuộc một đỉnh trong cây đang được hình thành và một đỉnh không thuộc cây này cho đến khi nhận được cây bao trùm tối thiểu. Trong mỗi bước lặp, quá trình thêm cạnh này sẽ bảo đảm không tạo thành chu trình trong T.

Chúng ta minh họa thuật toán bằng cách sử dụng "ma trận trọng lượng" $W = (w_{ij})_{n\times n}$, trong đó

$$w_{ij} = \begin{cases} 0 & \text{nếu } i = j, \\ +\infty & \text{nếu không tồn tại cạnh } (v_i, v_j), \\ w(v_i, v_j) & \text{nếu tồn tại cạnh } (v_i, v_j). \end{cases}$$

Khởi đầu từ đỉnh v_1 và nối nó đến một đỉnh kề gần nhất (tức là đỉnh mà có phần tử nhỏ nhất trong hàng thứ nhất trong ma trận W), giả sử là v_k. Bây giờ khảo sát v_1 và v_k như một đồ thị con, và nối đồ thị con này với một đỉnh kề với nó và gần nhất (tức là đỉnh khác v_1 và v_k mà có phần tử nhỏ nhất trong tất cả các phần tử của hàng thứ nhất và thứ k trong W). Giả sử đỉnh mới là v_i. Kế tiếp, xem cây với các đỉnh v_1, v_k, v_i như một đồ thị con, và tiếp tục xử lý cho đến khi tất cả n đỉnh được nối bởi $(n-1)$ cạnh.

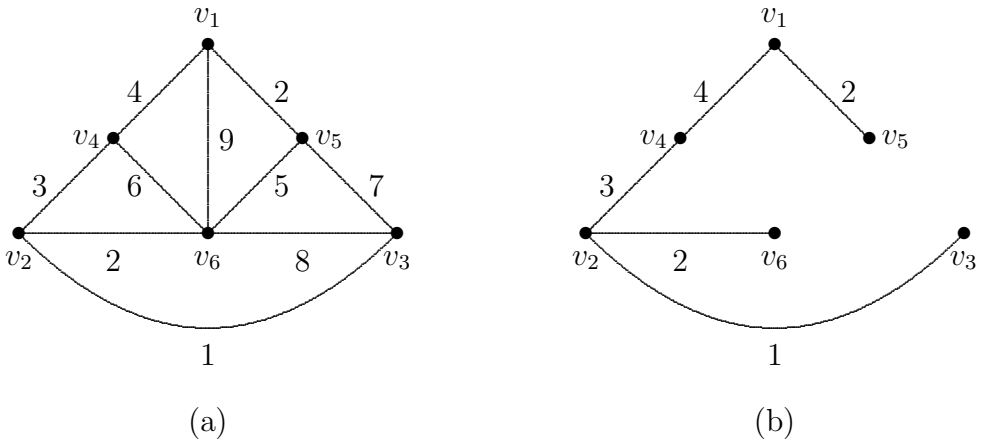
Tổng quát ta có thuật toán được trình bày như sau:

1. [Khởi tạo] Giả sử T là đồ thị chỉ có một đỉnh v_1 và không có cạnh.
2. [Kết thúc] Nếu T có $(n-1)$ cạnh thì thuật toán dừng; T là cây bao trùm tối thiểu.
3. [Thêm cạnh] Trong số tất cả các cạnh e không thuộc cây T mà liên thuộc với một đỉnh a trong T và một đỉnh b không thuộc cây T, chọn cạnh có trọng lượng nhỏ nhất và thêm nó (và đỉnh không thuộc T mà nó liên thuộc) vào cây T. Chuyển sang Bước 2.

Ví dụ 4.6.5. Áp dụng Thuật toán Prim cho đồ thị trong Hình 4.20(a) với đỉnh xuất phát là v_1 ta có các cạnh được lần lượt thêm vào cây T theo thứ tự là

$$(v_1, v_5), (v_1, v_4), (v_4, v_2), (v_2, v_3), (v_2, v_6).$$

Cây bao trùm nhỏ nhất có trọng lượng 12 (Hình 4.20(b)).

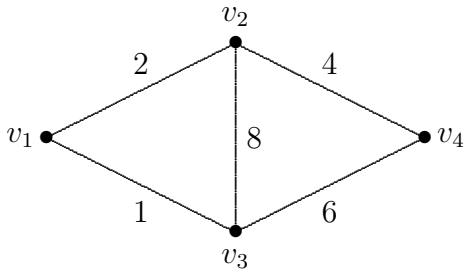


Hình 4.20: (a) Đồ thị G. (b) Cây bao trùm nhỏ nhất của G sinh bởi Thuật toán Prim.

Thuật toán Prim là một ví dụ của *thuật toán tham lam* (greedy algorithm). Thuật toán tham lam là thuật toán chọn lựa tối ưu tại mỗi bước lặp mà không quan tâm đến sự chọn lựa ở bước trước. Có thể nói nguyên lý này là "thực hiện tốt nhất về mặt địa phương". Trong thuật toán Prim, vì chúng ta muốn tìm cây bao trùm tối thiểu nên tại mỗi bước lặp chúng ta thêm một cạnh có trọng lượng nhỏ nhất (nếu việc thêm không tạo thành chu trình).

Tối ưu hóa tại mỗi bước lặp không nhất thiết cho lời giải tối ưu của bài toán gốc. Tính đúng đắn của thuật toán Prim sẽ được chứng minh trong Định lý 4.6.6. Một ví dụ của thuật toán tham lam nhưng không đưa đến lời giải tối ưu như sau: xét "thuật toán tìm đường đi ngắn nhất" trong đó, mỗi bước lặp chúng ta chọn cạnh có trọng lượng nhỏ nhất (mà không tạo thành chu trình khi thêm vào) liên thuộc với đỉnh được thêm vào gần đây nhất. Nếu ta áp dụng thuật toán này với đồ thị có hướng trong Hình 4.21 để tìm đường đi ngắn nhất từ v_1 đến v_4, chúng ta sẽ chọn cung (v_1, v_3) và sau đó cung (v_3, v_4). Tuy nhiên, đây không phải là đường đi ngắn nhất từ v_1 đến v_4.

Định lý sau chứng minh tính đúng đắn của thuật toán Prim.



Hình 4.21:

Định lý 4.6.6. *Thuật toán Prim là đúng; tức là, kết thúc thuật toán T là cây bao trùm tối thiểu.*

Chứng minh. Trong chứng minh này chúng ta sẽ đặc trưng cây bởi mảng các cạnh của nó.

Theo cách xây dựng, kết thúc Thuật toán Prim, T là đồ thị con liên thông không chu trình của đồ thị G và chứa tất cả các đỉnh của G; do đó T là cây bao trùm của G.

Để chỉ ra T là cây bao trùm tối thiểu, chúng ta chứng minh bằng quy nạp rằng ở bước thứ k của Thuật toán Prim, T được chứa trong một cây bao trùm tối thiểu. Và do đó, kết thúc thuật toán T là cây bao trùm tối thiểu. Ký hiệu T_k là cây được tạo ra ở bước lặp thứ k.

Nếu $k = 1$ thì T_1 gồm một đỉnh và do đó được chứa trong mọi cây bao trùm tối thiểu. Khẳng định đúng.

Giả sử rằng ở bước thứ $(k-1)$ của Thuật toán Prim, cây T_{k-1} chứa trong cây bao trùm tối thiểu T'. Ký hiệu V_{k-1} là tập các đỉnh của T_{k-1}. Theo Thuật toán Prim, cạnh $e = (v_i, v_j)$ có trọng lượng nhỏ nhất với $v_i \in V_{k-1}$ và $v_j \notin V_{k-1}$ được thêm vào T_{k-1} để tạo ra T_k. Nếu $e \in T'$ thì T_k chứa trong cây bao trùm tối thiểu T'. Nếu $e \notin T'$ thì $T' \cup \{e\}$ chứa một chu trình μ. Chọn cạnh $(v_x, v_y) \neq e$ trên chu trình μ sao cho $v_x \in V_{k-1}$ và $v_y \notin V_{k-1}$. Ta có

$$w(x, y) \geq w(i, j).$$

Suy ra đồ thị T" $:= [T' \cup \{e\}] - \{(v_x, v_y)\}$ có trọng lượng nhỏ hơn hoặc bằng trọng lượng của T'. Vì T" là cây bao trùm nên T" là cây bao trùm tối thiểu. Do đó ta có điều cần chứng minh. ◁

Thuật toán Prim cần kiểm tra $O(n^3)$ cạnh trong trường hợp xấu nhất (bài tập) để xác định cây bao trùm tối thiểu trong đồ thị n đỉnh. Chúng ta có thể chỉ ra Thuật toán 4.6.3 dưới đây chỉ cần kiểm tra $O(n^2)$ cạnh trong trường hợp xấu nhất. Vì K_n có n^2 cạnh nên thuật toán sau hiệu quả hơn.

Thủ tục Prim() dưới đây tìm cây bao trùm nhỏ nhất xuất phát từ đỉnh *Start* theo thuật toán Prim. Đồ thị được biểu diễn bởi mảng các danh sách kề $V\_out[]$. Cây bao trùm nhỏ nhất được lưu trữ trong hai mảng tuyến tính *alpha*[] và *beta*[].

```
void Prim(byte Start)
{
  AdjPointer Tempt;
  unsigned int Min, Sum = 0;
  byte i, k;
  byte alpha[MAXVERTICES], beta[MAXVERTICES], Weight[MAXVERTICES];
  Boolean Mark[MAXVERTICES];

  for (i = 1; i <= NumVertices; i++) Mark[i] = FALSE;
  Mark[Start] = TRUE;

  for (k = 1; k < NumVertices; k++)
  {
    Min = +INFTY;
    for (i = 1; i <= NumVertices; i++)
    {
      if (Mark[i] == TRUE)
      {
        Tempt = V_out[i]->Next;
        while (Tempt != NULL)
        {
          if (Mark[Tempt->Vertex] == FALSE)
            if (Min > Tempt->Length)
            {
              Min = Tempt->Length;
```

```
                alpha[k] = i;
                beta[k] = Tempt->Vertex;
                Weight[k] = Tempt->Length;
              }
              Tempt = Tempt->Next;
          }
        }
      }

      Mark[beta[k]] = TRUE;
      printf("\n Canh thu %d = (%d, %d) co trong luong = %d");
      printf(k, alpha[k], beta[k], Weight[k]);
      Sum += Weight[k];
    }
    printf("\n Trong luong cua cay bao trum nho nhat la %d ", Sum);
}
```

4.6.3 Thuật toán Dijkstra-Kevin-Whitney

Thuật toán dưới đây của Dijkstra [20], Kevin và Whitney [41] tìm cây bao trùm tối thiểu trong đồ thị liên thông có trọng số hiệu quả hơn phương pháp của Prim.

Ý tưởng của thuật toán là gán mỗi đỉnh $v_j \notin T_s$ một nhãn (α_j, β_j), trong đó ở bước nào đó α_j là đỉnh thuộc T_s gần với đỉnh v_j nhất và β_j là độ dài cạnh (α_j, v_j). Tại mỗi bước trong suốt quá trình thực hiện thuật toán, một đỉnh mới, chẳng hạn v_{j^*}, với giá trị β_j nhỏ nhất được thêm cùng với cạnh (α_{j^*}, v_{j^*}) vào cây T_s. Vì cây T_s được thêm đỉnh v_{j^*} nên các nhãn (α_j, β_j) của các đỉnh $v_j \notin T_s$ cần được cập nhật lại (nếu, chẳng hạn, $w(v_j, v_{j^*})$ nhỏ hơn nhãn trước đó β_j), và tiến trình được xử lý tiếp tục. Thủ tục gán nhãn này tương tự với thủ tục gán nhãn của thuật toán Dijkstra tìm đường đi ngắn nhất.

Thuật toán như sau:

1. [Khởi tạo] Đặt $T_s := \{s\}$, trong đó s là đỉnh được chọn tùy ý, và $A_s = \emptyset$. (A_s là

tập các cạnh của cây bao trùm nhỏ nhất được tạo ra trong quá trình lặp).

2. [Gán nhãn] Với mọi đỉnh $v_j \notin T_s$ tìm đỉnh $\alpha_j \in T_s$ sao cho

$$w(\alpha_j, v_j) = \min\{w(v_i, v_j) \mid v_i \in T_s\} = \beta_j$$

và gán nhãn của đỉnh v_j là (α_j, β_j).

Nếu không tồn tại đỉnh α_j, tức là $\Gamma(v_j) \cap T_s = \emptyset$ thì đặt nhãn của v_j là $(0, +\infty)$.

3. [Thêm cạnh] Chọn đỉnh v_{j^*} sao cho

$$\beta_{j^*} = \min\{\beta_j \mid v_j \notin T_s\}.$$

Cập nhật $T_s := T_s \cup \{v_{j^*}\}, A_s := A_s \cup \{(\alpha_{j^*}, v_{j^*})\}$.

Nếu $\#T_s = n$ thuật toán dừng; các cạnh trong A_s cho cây bao trùm nhỏ nhất. Ngược lại, chuyển sang Bước 4.

4. [Cập nhật nhãn] Với mọi $v_j \notin T_s$ và $v_j \in \Gamma(v_{j^*})$ cập nhật các nhãn như sau:

Nếu $\beta_j > w(v_{j^*}, v_j)$ thì đặt $\beta_j = w(v_{j^*}, v_j), \alpha_j = v_{j^*}$. Chuyển sang Bước 3.

Thủ tục Dijkstra_Kenvin_Whitney() dưới đây tìm cây bao trùm nhỏ nhất xuất phát từ đỉnh *Start*. Đồ thị được biểu diễn bởi mảng các danh sách kề $V\_out[]$. Cây bao trùm nhỏ nhất được lưu trữ trong hai mảng tuyến tính $alpha[]$ và $beta[]$.

```
void Dijkstra_Kenvin_Whitney(byte Start)
{
  byte i, Current, jstart;
  AdjPointer Tempt;
  Path Pred;
  int Label[MAXVERTICES], Min, Sum = 0;
  byte alpha[MAXEDGES], beta[MAXEDGES], Weight[MAXEDGES], NumEdges = 0;
  Boolean Mark[MAXVERTICES];

  for(i = 1; i <= NumVertices; i++) Mark[i] = FALSE;
  Mark[Start] = TRUE;
```

```
while (NumEdges < NumVertices - 1)
{
  for (i = 1; i <= NumVertices; i++)
  if (Mark[i] == FALSE)
  {
    Current = 0;
    Label[i] = +INFTY;
    Tempt = V_out[i]->Next;
    while (Tempt != NULL)
    {
      if (Mark[Tempt->Vertex] == TRUE)
        if (Label[i] > Tempt->Length)
        {
          Current = Tempt->Vertex;
          Label[i] = Tempt->Length;
        }
        Tempt = Tempt->Next;
      }
      Pred[i] = Current;
    }

    Min = +INFTY;
    jstart = 0;
    for (i = 1; i <= NumVertices; i++)
    if (Mark[i] == FALSE)
      if (Min > Label[i])
      {
        jstart = i;
        Min = Label[i];
      }
      Mark[jstart] = TRUE;
      NumEdges++;
      alpha[NumEdges] = Pred[jstart];
```

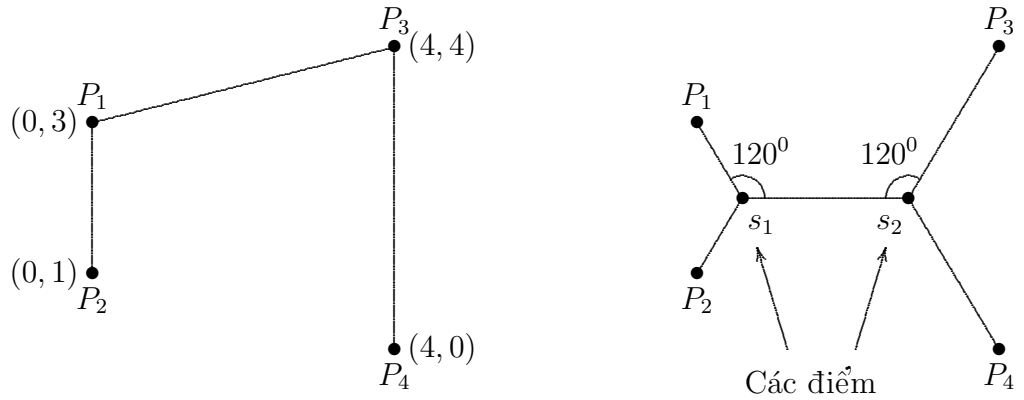
```
        beta[NumEdges] = jstart;
        Weight[NumEdges] = Label[jstart];
        Sum += Weight[NumEdges];

        Tempt = V_out[jstart]->Next;
        while(Tempt != NULL)
        {
          if (Mark[i] == FALSE);
            if (Label[Tempt->Vertex] > Tempt->Length)
            {
              Pred[Tempt->Vertex] = jstart;
              Label[Tempt->Vertex] = Tempt->Length;
            }
            Tempt = Tempt->Next;
          }
        }
        printf("\n Cay bao trum nho nhat:");

        for (i = 1; i <= NumEdges; i++)
        {
          printf("\n Canh thu %d = (%d, %d), trong luong = %d");
          printf(i, alpha[i], beta[i], Weight[i]);
        }

        printf("\n Ttrong luong cua cay bao trum nho nhat la %d ", Sum);
}
```

4.7 Bài toán Steiner

Trong phần trước chúng ta đã xét bài toán tìm cây bao trùm nhỏ nhất. Một bài toán có liên quan đến việc tìm cây bao trùm nhỏ nhất nhưng khó hơn nhiều là "*bài toán Steiner trên các đồ thị*" [21]. Trong bài toán Steiner, cây bao trùm nhỏ nhất T đòi hỏi

đi qua một tập con $P \subset V$ cho trước của đồ thị G. Các đỉnh khác (thuộc $V \setminus P$) có thể thuộc hoặc không thuộc cây với điều kiện cực tiểu trọng lượng của cây T. Do đó, bài toán Steiner trên đồ thị tương đương với tìm cây bao trùm nhỏ nhất trong đồ thị con $G' = (V', \Gamma')$ của G với $P \subseteq V' \subseteq V$.

Thật ra, *bài toán Steiner* được phát biểu ở dạng nguyên thủy là một bài toán hình học [15], trong đó tập P các điểm trên mặt phẳng được nối với nhau bằng các đoạn thẳng sao cho tổng độ dài các đoạn thẳng được vẽ là nhỏ nhất. Nếu giả thiết hai đoạn thẳng chỉ có thể cắt nhau tại các đỉnh của P thì bài toán trở thành tìm cây bao trùm nhỏ nhất của đồ thị tương đương có $\#P$ đỉnh với ma trận trọng lượng xác định bởi ma trận khoảng cách giữa các điểm thuộc tập P. Tuy nhiên, khi các "đỉnh nhân tạo" khác (gọi là các *điểm Steiner*) được tạo thêm trên mặt phẳng thì trọng lượng của cây bao trùm nhỏ nhất tương ứng tập đỉnh mới $P' \supset P$ sẽ có thể giảm đi. Chẳng hạn, xét bốn điểm trong Hình 4.22(a) với cây bao trùm nhỏ nhất T; tuy nhiên khi thêm hai điểm mới s_1 và s_2 ta được cây bao trùm nhỏ nhất mới T' qua sáu đỉnh có trọng lượng nhỏ hơn cây T (xem Hình 4.22(b)). Do đó trong bài toán Steiner gốc, nhiều điểm Steiner được thêm vào trong mặt phẳng để tạo ra cây bao trùm nhỏ nhất qua những đỉnh cho trước P. Kết quả được đồ thị gọi là *cây Steiner nhỏ nhất*.



(a) (b)

Hình 4.22: (a) Cây bao trùm nhỏ nhất có trọng lượng bằng 10.123. (b) Cây Steiner nhỏ nhất có trọng lượng 9.196.

Có rất nhiều công trình nghiên cứu xung quanh bài toán Steiner trên mặt phẳng

Euclide, và nhiều tính chất của cây Steiner nhỏ nhất được phát hiện. Trong số đó, các tính chất sau là quan trọng nhất (chứng minh các tính chất này có thể xem [28]):

1. Đối với điểm Steiner s_i có bậc $d(s_i) = 3$. Bằng hình học dễ dàng chỉ ra rằng góc liên thuộc giữa các điểm Steiner bậc ba bằng 120^0, và có đúng ba cạnh liên thuộc với điểm Steiner s_i. Do đó điểm này là "tâm" (tâm Steiner) của "tam giác ảo" mà ba đỉnh của tam giác khác nhau được nối với đỉnh s_i trong cây bao Steiner nhỏ nhất.

   Một số điểm tạo thành các đỉnh của tam giác này có thể là các điểm Steiner khác. Chẳng hạn, trong Hình 4.22, điểm Steiner s_2 là tâm Steiner của tam giác ảo với các đỉnh là P_3, P_4 và s_1.

2. Với đỉnh $P_i \in P$ nào đó mà $d(P_i) \leq 3$. Nếu $d(P_i) = 3$ thì góc giữa hai cạnh bất kỳ trong ba cạnh liên thuộc với P_i bằng 120^0, và $d(P_i) = 2$ thì góc giữa hai cạnh liên thuộc P_i lớn hơn hoặc bằng 120^0.

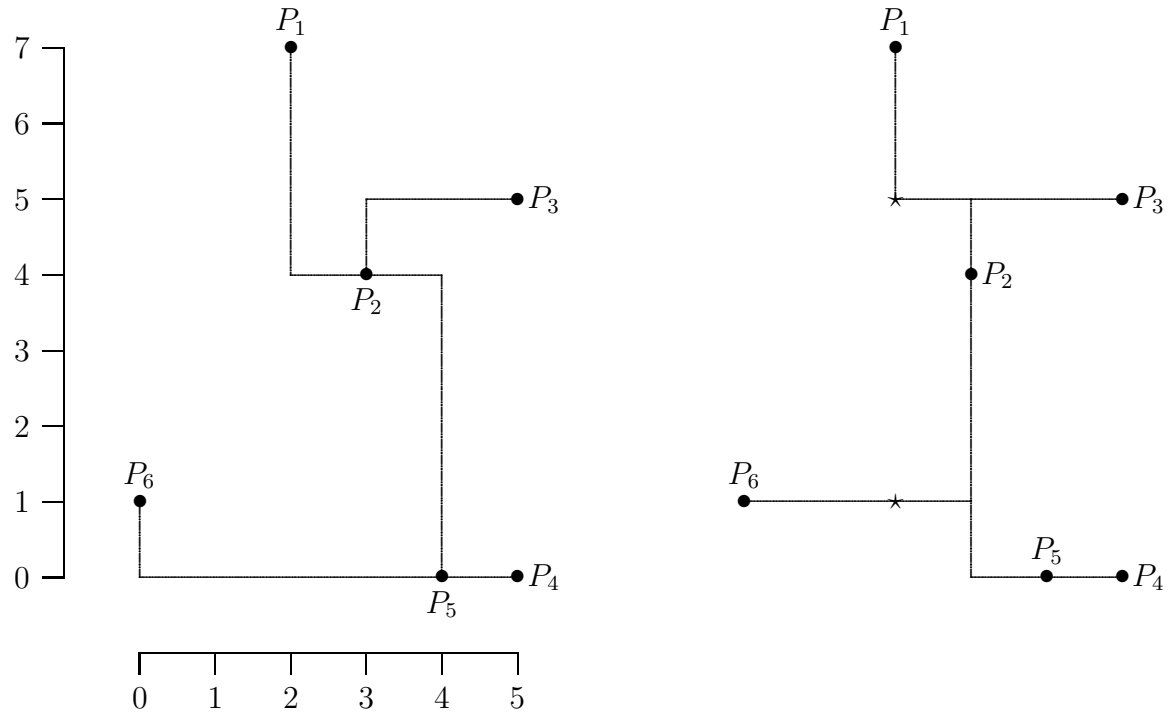
3. Số k các điểm Steiner trong cây bao trùm nhỏ nhất Steiner thỏa ràng buộc $0 \leq k \leq n-2$, trong đó $n = \#P$.

Mặc dù có nhiều cố gắng trong việc giải quyết bài toán Steiner trên mặt phẳng Euclide, nhưng chúng ta chỉ mới thu được những kết quả rất hạn chế (trong trường hợp $\#P \leq 10$). Với những bài toán có kích thước lớn ta cần dùng các phương pháp heuristics.

Một dạng khác của bài toán Steiner sử dụng khoảng cách *bàn cờ* thay cho khoảng cách Euclide. Bài toán này được xét lần đầu tiên bởi Hanan [32] và có liên quan với các đường dẫn được in trên các bảng mạch điện tử. Nhắc lại rằng, khoảng cách bàn cờ của hai điểm (x_1, y_1) và (x_2, y_2) trong mặt phẳng $\mathbf{R}^2$ xác định bởi

$$d_{1,2} := |x_1 - x_2| + |y_1 - y_2|.$$

Với các điều kiện này, có thể chứng minh rằng nếu kẻ một lưới các đường thẳng nằm ngang và đứng đi qua các điểm của tập P thì lời giải của bài toán Steiner có thể xác định bằng cách xem vị trí các điểm Steiner thuộc vào các giao điểm của lưới này.



Hình 4.23: (a) Cây bao trùm nhỏ nhất có trọng lượng bằng 18. (b) Cây Steiner nhỏ nhất có trọng lượng bằng 15 ($\star$ = các điểm Steiner).

Do đó, nếu đồ thị G được xác định bởi tập đỉnh V là các giao điểm của lưới và một cạnh liên thuộc hai đỉnh tương ứng đoạn thẳng thuộc lưới nối hai giao điểm. Khi đó bài toán Steiner trên mặt phẳng với khoảng cách bàn cờ trở thành bài toán Steiner trên đồ thị hữu hạn G. Hình 4.23(b) minh họa ví dụ của cây Steiner nhỏ nhất trong bài toán Steiner 6 điểm với khoảng cách bàn cờ và Hình 4.23(a) là cây bao trùm nhỏ nhất của đồ thị xác định bởi 6 điểm (để đơn giản lưới không được vẽ ra).

Bài toán Steiner trên đồ thị vô hướng tổng quát được nghiên cứu bởi nhiều tác giả và đã có những thuật toán (không hiệu quả về mặt tính toán) giải quyết chúng. Tuy nhiên (như trong trường hợp bài toán Steiner với khoảng cách Euclide), kích thước lớn nhất của một bài toán Steiner trên đồ thị có thể giải trên máy tính điện tử với thời gian cho phép vẫn không vượt quá 10 đỉnh (trong P). Nói cách khác, bài toán Steiner trên đồ thị cũng có thể xem là bài toán chưa có lời giải.

Chương 5

Bài toán Euler và bài toán Hamilton

Lý thuyết đồ thị phát triển bắt nguồn từ những bài toán cổ điển, trong số đó *bài toán Euler* (tìm hành trình đi qua mỗi cạnh đúng một lần) và *bài toán Hamilton* (tìm hành trình đi qua mỗi đỉnh đúng một lần) đóng vai trò quan trọng.

Hai bài toán này có liên quan đến những ứng dụng: các bài toán tìm hành trình tốt nhất (*người đưa thư Trung Hoa, người chào hàng)*, tự động hóa thiết kế bằng máy tính, lập lịch, vân vân.

Mặc dù hai bài toán này được phát biểu rất giống nhau, nhưng mức độ khó trong việc giải quyết chúng là rất khác nhau.

Chúng ta sẽ chứng minh rằng trong đồ thị vô hướng, tồn tại thuật toán đa thức tìm hành trình Euler và bài toán người đưa thư Trung Hoa có thể đưa về tìm cặp ghép hoàn hảo có trọng lượng nhỏ nhất (xem Phần 7.5). Các thuật toán này sẽ được trình bày trong các Phần 5.1 và 5.2.

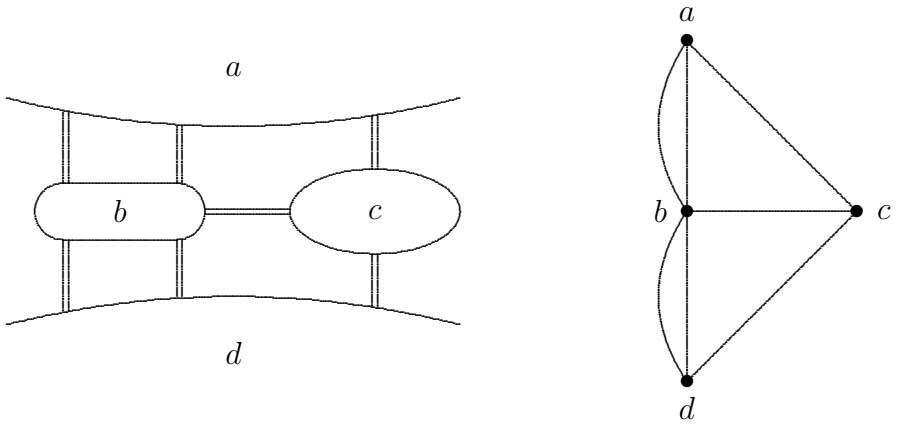
Mặt khác, vấn đề tồn tại chu trình hay mạch Hamilton là những bài toán không đa thức không được đề cập ở đây. Bạn đọc quan tâm có thể xem, chẳng hạn [30]. Chúng ta chỉ trình bày trong Phần 5.3 những kết quả chính liên quan đến sự tồn tại của các chu trình hay mạch Hamilton. Khi có điều kiện, các chứng minh có tính kiến thiết thuật toán hoặc có thể đề xuất những phương pháp heuristic.

5.1 Bài toán Euler

Định nghĩa 5.1.1. Giả sử $G = (V, E)$ là đồ thị vô hướng (đơn hoặc đa đồ thị). *Dây chuyền Euler* là dây chuyền chứa tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần. *Chu trình Euler* là dây chuyền Euler mà đỉnh đầu trùng với đỉnh cuối.

Ví dụ 5.1.2. (Bài toán Euler) Cách đây khoảng ba trăm năm, nhiều người dân thành phố Königsberg của nước Nga (sau này là thành phố Kaliningrat) đã từng thắc mắc vấn đề như sau: Thành phố có sông Pregel chảy qua, giữa sông có cù lao Kneiphof, và có 7 chiếc cầu bắc qua sông như trên Hình 5.1(a); có thể đi dạo qua khắp các cầu nhưng mỗi cầu chỉ đi một lần thôi không? Nếu ta coi mỗi khu vực a, b, c, d của thành phố như một đỉnh, mỗi cầu qua lại hai khu vực như một cạnh nối hai đỉnh, thì bản đồ thành phố Königsberg là một đồ thị (Hình 5.1(b)). Thắc mắc của người dân thành phố chính là: có thể vẽ được đồ thị bằng một nét bút liền hay không? Nói cách khác: tồn tại chu trình Euler?

Nhà toán học L. Euler (1707-1783) là người đầu tiên đã chứng minh bài toán không có lời giải (năm 1736, xem [22], [23]), và vì vậy bài toán thường được gọi là bài toán Euler về các cầu ở Königsberg.

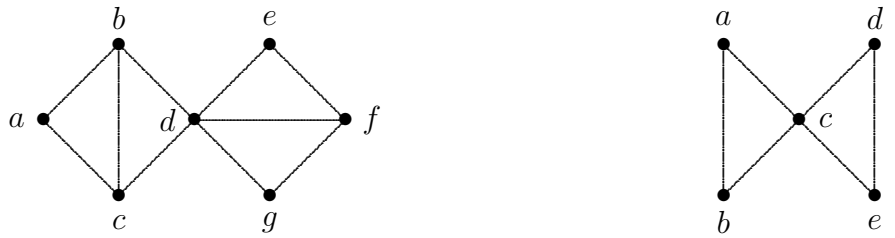


(a) (b)

Hình 5.1: (a) Bản đồ của thành phố Königsberg. (b) Đồ thị tương đương.

Ví dụ 5.1.3. Có thể vẽ đồ thị vô hướng trên Hình 5.2(a) mà không nhấc bút khỏi mặt giấy và không vẽ một cạnh nào quá hai lần không? Sau khi thử nhiều cách vẽ, dù

không có kinh nghiệm bạn đọc cũng có thể kết luận rằng bài toán đó không thể giải được. Trái lại, đồ thị vô hướng ở Hình 5.2(b) có thể vẽ bằng một nét. Tại sao?



Hình 5.2:

Định lý 5.1.4. [Euler] *Đa đồ thị vô hướng liên thông $G = (V, E)$ có dây chuyền Euler nếu và chỉ nếu số các đỉnh bậc lẻ bằng 0 hoặc 2.*

Chứng minh. Trước hết chú ý rằng đồ thị không liên thông không thể chứa dây chuyền hoặc chu trình Euler.

Bây giờ ta chứng minh điều kiện cần. Nếu μ là dây chuyền Euler, thì chỉ có hai đỉnh đầu và cuối có bậc lẻ. Nếu ngoài ra, hai đỉnh này trùng nhau (chu trình Euler) thì không có đỉnh bậc lẻ.

Kế tiếp ta chứng minh điều kiện đủ bằng quy nạp theo số cạnh m của G. Hiển nhiên định lý đúng nếu $m = 1$. Giả sử định lý đúng cho mọi đồ thị liên thông m cạnh. Nếu G có hai đỉnh bậc lẻ, giả sử các đỉnh này là a và b (nếu tất cả các đỉnh của G có bậc chẵn, chọn đỉnh a bất kỳ và lấy $b = a$). Ký hiệu μ là dây chuyền mà ta đi trên đồ thị G xuất phát từ a theo hướng tùy ý, đi qua mỗi cạnh đúng một lần. Nếu tại thời điểm nào đó ta ở đỉnh $x \neq b$ nghĩa là ta đã sử dụng một số lẻ cạnh liên thuộc với x nên có thể đi theo cạnh khác chưa được sử dụng. Nếu ta không thể đi được nữa, nghĩa là đang ở đỉnh b. Nếu tất cả các cạnh đã được sử dụng, μ là một dây chuyền Euler và định lý đúng. Trong trường hợp ngược lại, đồ thị con G' được xác định bởi các cạnh chưa được sử dụng chỉ có những đỉnh bậc chẵn. Ký hiệu $G'_1, G'_2, \ldots, G'_p$ là các thành phần liên thông của G' chứa ít nhất một cạnh. Khi đó các đồ thị G'_i có số cạnh ít hơn m và do đó theo giả thiết quy nạp, chúng chứa một chu trình Euler μ_i. Vì G liên thông, dây chuyền μ có chung với các đồ thị $G'_1, G'_2, \ldots, G'_p$ các đỉnh theo thứ tự $i_1, i_2, \ldots, i_p$. Khi đó hành trình: xuất phát từ a đi trên μ đến đỉnh i_1, đi dọc theo chu trình μ_1 từ i_1

về i_1, đi trên μ đến đỉnh i_2 đi dọc theo chu trình μ_2 từ i_2 về i_2, và vân vân, là một dây chuyền Euler xuất phát từ a và kết thúc tại b. Định lý được chứng minh. $\triangleleft$

Đồ thị thỏa các điều kiện của Định lý Euler gọi là *đồ thị Euler.*

5.1.1 Thuật toán tìm dây chuyền Euler

Cách chứng minh Định lý Euler 5.1.4 cho ta một thuật toán xây dựng dây chuyền Euler trong một đồ thị Euler.

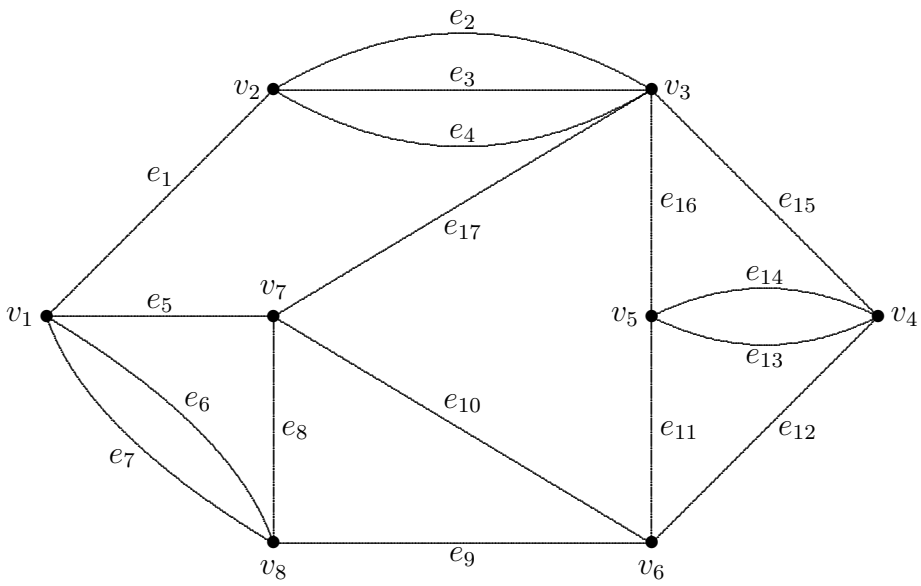
1. Xây dựng một dây chuyền đơn giản μ xuất phát từ s.
2. Nếu tất cả các cạnh của G đã được sử dụng thì dừng và ta có μ là dây chuyền Euler. Ngược lại sang Bước 3.
3. Ký hiệu G_1 là đồ thị con của G gồm các cạnh chưa được sử dụng. Chọn đỉnh c của G_1 nằm trên dây chuyền μ. Xây dựng chu trình đơn giản μ_1 trong đồ thị G_1 xuất phát từ đỉnh c.
4. Mở rộng dây chuyền μ bằng cách gắn thêm chu trình μ_1 tại đỉnh c (tức là dãy các cạnh của μ_1 được chèn vào dãy các cạnh của μ).
5. Thay G bởi G_1 và lặp lại bước 2.

Ví dụ 5.1.5. Đồ thị trong Hình 5.3 có một chu trình Euler

$$(v_1, e_1, v_2, e_2, v_3, e_3, v_2, e_4, v_3, e_{15}, v_4, e_{14}, v_5, e_{13}, v_4, e_{12}, v_6, e_{11},$$

$$v_5, e_{16}, v_3, e_{17}, v_7, e_{10}, v_6, e_9, v_8, e_8, v_7, e_5, v_1, e_7, v_8, e_6, v_1).$$

Một dây chuyền hay chu trình Euler có thể được xác định bởi một danh sách có thứ tự các đỉnh của G chứa trong nó. Chẳng hạn, ta có thể sử dụng cấu trúc danh sách liên kết kiểu *AdjPointer* để đánh dấu các đỉnh liên tiếp trên dây chuyền, trong đó $Vertex$ là số hiệu đỉnh trên dây chuyền; và con trỏ $Next$ chỉ nút kế tiếp chứa số



Hình 5.3: Một ví dụ về đồ thị Euler.

hiệu của đỉnh kề $Vertex$ trên dây chuyền. Dễ thấy rằng, danh sách này có số nút bằng $(m + 1)$.

Giả sử đồ thị G được cho bởi mảng các danh sách kề $V\_out[]$ (có trọng số), trong đó với mỗi đỉnh $v_i \in V$, $V\_out[i]$ là danh sách các đỉnh liên thuộc đỉnh v_i và trường độ dài $Length$ ở nút thứ j (tương ứng đỉnh v_j) là số cạnh liên thuộc với đỉnh v_i. Trong quá trình thực hiện thuật toán, mỗi khi đi qua cạnh (v_i, v_j) nào đó, ta giảm độ dài $Length$ một đơn vị ở nút thứ j trong danh sách $V\_out[i]$ để đánh dấu cạnh đã được sử dụng.

Vì mỗi cạnh của đồ thị được kiểm tra nhiều nhất hai lần nên độ phức tạp của thuật toán là $O(m)$.

Đoạn chương trình sau minh họa thuật toán tìm chu trình Euler.

```
#include "d:\phamts\graph\ctrinh\Graph.h"
byte A[MAXVERTICES][MAXVERTICES];
byte NumVertices;

byte Convert(ArrayOfPointer V_out)
{
```

```
  byte i, j, Count = 0;
  byte Degree[MAXVERTICES];
  AdjPointer Tempt;

  for (i = 1; i <= NumVertices; i++)
    for (j = 1; j <= NumVertices; j++) A[i][j] = 0;

  for(i = 1; i <= NumVertices; i++)
  {
    Tempt = V_out[i]->Next;
    while (Tempt != NULL)
    {
      A[i][Tempt->Vertex]++;
      A[Tempt->Vertex][i]++;
      Tempt = Tempt->Next;
    }
  }

  for (i = 1; i <= NumVertices; i++)
  {
    Degree[i] = 0;
    for (j = 1; j <= NumVertices; j++)
    {
      A[i][j] = A[i][j] / 2;
      printf("%d ", A[i][j]);
      Degree[i] += A[i][j];
    }

    Degree[i] += A[i][i];
    if ((Degree[i] % 2) == 1) Count++;
    printf("%5d  \n", Degree[i]);
  }
  return Count;
}
```

```
void FindCycle(AdjPointer *Cycle, byte Start)
{
  byte i, j;

  (*Cycle) = NULL;
  Push(Cycle, Start);

  i = Start;
  do
  {
    for (j = 1; j <= NumVertices; j++)
      if (A[i][j] > 0) break;

    A[i][j]--;
    A[j][i]--;
    Push(Cycle, j);
    i = j;
  }
  while (i != Start);
}

AdjPointer FindJointVertex(AdjPointer Cycle)
{
  AdjPointer c = Cycle;
  byte j;

  while (c->Next != NULL)
  {
    for (j = 1; j <= NumVertices; j++)
      if (A[c->Vertex][j] > 0) return(c);
    c = c->Next;
  }
```

```
  return(NULL);
}

void Connect(AdjPointer *mu, AdjPointer c)
{
  AdjPointer Last;
  byte Item;

  Pop(mu, &Item);

  Last = *mu;
  while (Last->Next != NULL) Last = Last->Next;

  Last->Next = c->Next;
  c->Next = *mu;
}

void DisplayCycle(AdjPointer Cycle)
{
  AdjPointer Tempt = Cycle;

  while (Tempt != NULL)
  {
    printf("%d -->", Tempt->Vertex);
    Tempt = Tempt->Next;
  }
  printf("NULL\n");
}

void  main()
{
  ArrayOfPointer V_out;
  AdjPointer EulerCycle, c, mu;
  byte Start;
```

```
clrscr();
MakeV_out("d:\\phamts\\graph\\ctrinh\\Euler2.txt",
                      V_out, &NumVertices, FALSE);
DisplayI(V_out, NumVertices, FALSE);
if ((Start = Convert(V_out)) != 0)
{
  printf(" \n khong ton tai chu trinh Euler ");
  getch();
}
else
{
  printf(" \n ton tai chu trinh Euler \n ");
  getch();

  Start = 1;
  FindCycle(&EulerCycle, Start);
  printf(" \n Old Cycle");
  DisplayCycle(EulerCycle);
  getch();

  while ((c = FindJointVertex(EulerCycle)) != NULL)
  {
    printf(" \n Joint Vertex %d ", c->Vertex);

    FindCycle(&mu, c->Vertex);
    printf(" \n mu Cycle ");
    DisplayCycle(mu);

    Connect(&mu, &c);
    printf(" \n New Cycle");
    DisplayCycle(EulerCycle);
  }
```

```
    getch();
  }
  PopHeadPtr(NumVertices, V_out);
}
```

5.2 Bài toán người đưa thư Trung Hoa

Xét đồ thị vô hướng liên thông $G := (V, E)$ có *trọng số* (tức là mỗi cạnh $e \in E$ ta gán một số $w(e)$ gọi là *trọng lượng* của cạnh e).

Bài toán người đưa thư Trung Hoa (không được định hướng) phát biểu rằng tìm một dây chuyền giữa hai đỉnh cho trước $a, b \in V$ sử dụng mỗi cạnh của G ít nhất một lần và có độ dài nhỏ nhất (xem [44]).

Nhiều bài toán về hành trình (người đưa thư, người giao sữa, người chào hàng, v.v) có thể phát biểu ở dạng này. Trong trường hợp đồ thị có hướng, trong đó mỗi cung của G cần được sử dụng ít nhất một lần, bài toán có thể đưa về bài toán luồng với chi phí nhỏ nhất (bài tập).

Từ đây về sau chúng ta chỉ xét đồ thị vô hướng. Không mất tính tổng quát có thể giả thiết đỉnh xuất phát a và đỉnh kết thúc b trên dây chuyền là trùng nhau. Trong trường hợp ngược lại, ta chỉ cần thêm một cạnh (a, b) với độ dài bằng không. Với mỗi chu trình có độ dài nhỏ nhất trên đồ thị mới này, tồn tại một dây chuyền trên G có cùng độ dài và do đó là nhỏ nhất.

Nếu G là đồ thị Euler thì tồn tại một chu trình Euler đi qua mỗi cạnh đúng một lần và vì vậy là một nghiệm tối ưu của bài toán.

Nói chung, G không phải là đồ thị Euler, nên tồn tại một số các đỉnh bậc lẻ. Ký hiệu V_1 là tập các đỉnh của G có bậc lẻ. Dễ thấy rằng số phần tử của tập V_1 là một số chẵn. Khi đó bài toán người đưa thư Trung Hoa đưa về việc thêm một số cạnh vào G để trở thành đồ thị Euler và cùng lúc, cực tiểu hóa tổng các trọng lượng của các cạnh được thêm vào.

Chúng ta không thêm một cạnh $e' = (v_i, v_j)$ trừ khi đã tồn tại một cạnh $e = (v_i, v_j)$ trong G và gán trọng lượng của cạnh e' là $w(e') := w(e)$. Cạnh e' gọi là *bản sao* của e.

Xét một lời giải tối ưu của bài toán và đặt E' là tập các cạnh được thêm vào G. Ký hiệu $G' = (V, E + E')$ là đồ thị Euler nhận được.

Bổ đề 5.2.1. *Giả sử v_i là một đỉnh bậc lẻ trong G. Khi đó tập E' chứa một dây chuyền sơ cấp nối đỉnh v_i với một đỉnh $v_j \neq v_i$ có bậc lẻ trong G.*

Chứng minh. Với mọi đỉnh $v_k \in V_1$ ta có $d_G(v_k) \equiv 1 \pmod 2$ và $d_{G'}(v_k) \equiv 0 \pmod 2$; ngoài ra, theo cách xây dựng $d_{G'}(v_k) \geq d_G(v_k)$. Do đó tồn tại ít nhất một cạnh $e_1 \in E'$ liên thuộc đỉnh v_i.

Ký hiệu v_{i_1} là đỉnh khác v_i mà cạnh e_1 liên thuộc. Nếu $d_G(v_{i_1}) \equiv 1 \pmod 2$ thì bổ đề được chứng minh với $v_j = v_{i_1}$. Ngược lại, nếu $d_G(v_{i_1}) \equiv 0 \pmod 2$ thì $d_{G'}(v_{i_1}) \geq d_G(v_{i_1}) + 2$ và tồn tại cạnh $e_2 \in E', e_2 \neq e_1$, liên thuộc đỉnh v_{i_1}. Ký hiệu v_{i_2} là đỉnh khác v_{i_1} mà cạnh e_2 liên thuộc. Nếu $d_G(v_{i_2}) \equiv 1 \pmod 2$ thì bổ đề được chứng minh với $v_j = v_{i_2}$. Ngược lại, tồn tại cạnh $e_3 \in E', e_3 \neq e_2$, liên thuộc đỉnh v_{i_2}, và vân vân.

Do đó ta xây dựng được một dây chuyền sơ cấp dài nhất có thể

$$(v_i, e_1, v_{i_1}, e_2, v_{i_2}, \ldots, e_p, v_{i_p}).$$

Nếu $d_G(v_{i_p}) \equiv 1 \pmod 2$ thì bổ đề được chứng minh với $v_j = v_{i_p}$. Ngược lại, tồn tại cạnh $e_{p+1} \in E', e_p \neq e_{p+1}$, liên thuộc đỉnh v_{i_p} và $v_{i_{p+1}}$. Trong trường hợp này, tồn tại chỉ số $q, 1 \leq q \leq p$, sao cho $v_{i_q} \equiv v_{i_{p+1}}$ và ta có một chu trình xuất hiện. Loại bỏ tất cả các cạnh trong chu trình này ta được một đồ thị con G'' của G' sao cho nó vẫn là đồ thị Euler và hơn nữa

$$d_{G''}(v_k) \geq d_G(v_k),$$

với mọi đỉnh $v_k \in V$.

Lặp lại cách xây dựng dây chuyền trên, xuất phát từ đỉnh v_{i_q} chỉ sử dụng các cạnh của G''.

Do số các cạnh trong E' là hữu hạn, nên sau một số hữu hạn bước ta được một đỉnh v_{i_p} sao cho $d_G(v_{i_p}) \equiv 1 \pmod 2$ và bổ đề được chứng minh với $v_j = v_{i_p}$. ◁

Bổ đề 5.2.2. *Giả sử v_i và v_j là hai đỉnh thỏa mãn các điều kiện của Bổ đề 5.2.1 và ký hiệu dây chuyền tương ứng là*

$$\mu' := \{e'_1, e'_2, \ldots, e'_p\}$$

trong đó $e'_k \in E', k = 1, 2, \ldots, p$. Các cạnh $e'_1, e'_2, \ldots, e'_p$ là các bản sao của các cạnh $e_1, e_2, \ldots, e_p$ trong G và xét dây chuyền $\mu := \{e_1, e_2, \ldots, e_p\}$ trong G. Khi đó μ là dây chuyền (trong G) có trọng lượng nhỏ nhất nối đỉnh v_i với đỉnh v_j.

Chứng minh. Nếu tồn tại một dây chuyền $\bar{\mu} = \{\bar{e}_1, \bar{e}_2, \ldots, \bar{e}_q\}$ nối đỉnh v_i với đỉnh v_j trong G có độ dài nhỏ hơn thì bằng cách loại các cạnh $e'_1, e'_2, \ldots, e'_p$ khỏi G' và thêm các bản sao $\bar{e}'_1, \bar{e}'_2, \ldots, \bar{e}'_q$ của $\bar{e}_1, \bar{e}_2, \ldots, \bar{e}_q$ ta nhận được một đồ thị Euler mới có tổng trọng lượng nhỏ hơn, mâu thuẫn. ◁

Bây giờ xét đồ thị đầy đủ $\mathcal{K}(V_1)$ trên tập các đỉnh V_1 trong đó các cạnh thêm vào (v_i, v_j) có trọng lượng w_{ij} bằng độ dài của dây chuyền nhỏ nhất trong G giữa hai đỉnh v_i và v_j. Khi đó mỗi cạnh của $\mathcal{K}(V_1)$ tương ứng với một dây chuyền trong G. Vì $w(e) \geq 0$ với mọi cạnh $e \in E$ nên w_{ij} có thể được xác định bằng thuật toán tìm đường đi ngắn nhất, chẳng hạn Floyd (xem 3.3.2) hay Dantzig [16].

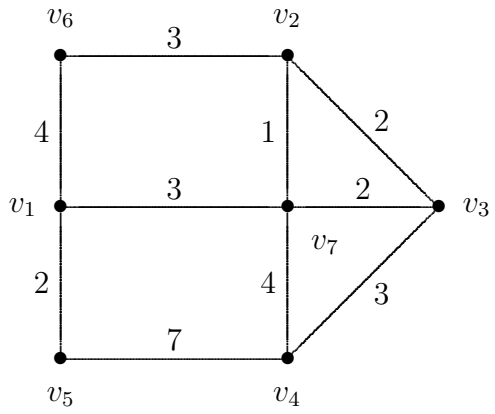
Định lý 5.2.3. *Tồn tại tương ứng một-một giữa lời giải tối ưu của bài toán người đưa thư Trung Hoa với một cặp ghép hoàn hảo có trọng lượng nhỏ nhất trong đồ thị $\mathcal{K}(V_1)$.*

Chứng minh. Xét một lời giải tối ưu của bài toán người đưa thư Trung Hoa và đặt E' là tập các cạnh thêm vào G. Theo Bổ đề 5.2.1 ta có thể thiết lập tương ứng với mỗi đỉnh $v_i \in V_1$ với một đỉnh $v_j \in V_1$ bằng một dây chuyền sơ cấp μ_{ij} mà các cạnh thuộc E'. Theo Bổ đề 5.2.2, μ_{ij} có độ dài nhỏ nhất. Trong đồ thị $\mathcal{K}(V_1)$ các dây chuyền μ_{ij} tương ứng cạnh (v_i, v_j). Do đó tất cả các đỉnh của V_1 được kết hợp, hai với hai, và các cạnh (v_i, v_j) tương ứng dây chuyền μ_{ij} của G', tạo thành một cặp ghép hoàn hảo K của đồ thị $\mathcal{K}(V_1)$. (Trong đồ thị đầy đủ với số chẵn đỉnh luôn luôn tồn tại một cặp ghép hoàn hảo; xem Phần 7.5). Vì trọng lượng của cặp ghép hoàn hảo K bằng tổng các trọng lượng của các cạnh của E' nên lời giải của bài toán người đưa thư Trung Hoa là tối ưu nếu và chỉ nếu K là một cặp ghép hoàn hảo với trọng lượng nhỏ nhất. Ta có điều phải chứng minh. ◁

Do đó nghiệm của bài toán người đưa thư Trung Hoa đưa về bài toán tìm một cặp ghép hoàn hảo có trọng lượng nhỏ nhất của đồ thị đầy đủ K_n. Việc xác định nghiệm của bài toán sau là một thuật toán khá phức tạp và do đó sẽ không được trình bày ở đây. Bạn đọc quan tâm có thể tham khảo các tài liệu [14], [30].

Nhận xét 5.2.4. Nếu tồn tại cạnh e trong G sao cho $w(e) < 0$ thì bài toán không có nghiệm tối ưu: Thật vậy, bằng cách thêm một tập E' hữu hạn các bản sao của các cạnh của G ta có thể thêm cạnh e một số chẵn lần đủ lớn, và do đó nhận được một đồ thị Euler với độ dài nhỏ tùy ý. Vậy giả thiết các cạnh có trọng lượng không âm là không mất tính tổng quát để loại trừ trường hợp tầm thường này.

Ví dụ 5.2.5. Xét đồ thị trong Hình 5.4 với các số trên các cạnh là trọng lượng cạnh. Ta cần tìm một chu trình qua mỗi cạnh ít nhất một lần và có độ dài nhỏ nhất.



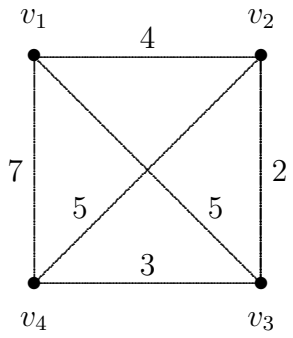
Hình 5.4: Đồ thị G.

Tổng các trọng lượng các cạnh của G bằng 31. Vì G không là đồ thị Euler nên độ dài của chu trình cần tìm sẽ lớn hơn 31.

Tập các đỉnh bậc lẻ là $V_1 = \{v_1, v_2, v_3, v_4\}$. Theo thuật toán tìm đường đi ngắn nhất (xem Chương 3), ta tìm tất cả các dây chuyền có độ dài nhỏ nhất giữa các cặp đỉnh của V_1 trong G. Ta nhận được ma trận độ dài đường đi ngắn nhất:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 4 & 5 & 7 \\ 4 & 0 & 2 & 5 \\ 5 & 2 & 0 & 3 \\ 7 & 5 & 3 & 0 \end{pmatrix} \end{array}.$$

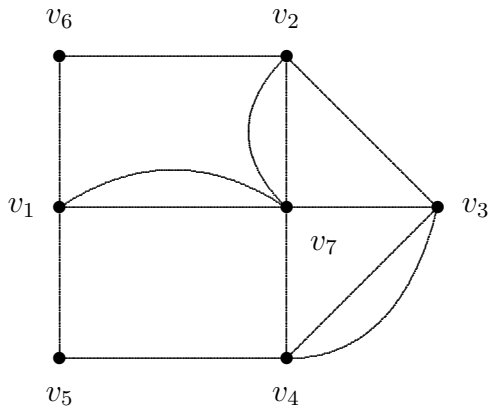
Tiếp đến ta xây dựng đồ thị đầy đủ $\mathcal{K}(V_1)$ trong đó trọng lượng cạnh (v_i, v_j) là độ dài của dây chuyền ngắn nhất giữa v_i và v_j (xem Hình 5.5).



Hình 5.5: Đồ thị đầy đủ $\mathcal{K}(V_1)$.

Cặp ghép hoàn hảo với trọng lượng nhỏ nhất trên $\mathcal{K}(V_1)$ gồm các cạnh (v_1, v_2) và (v_3, v_4) (trọng lượng bằng $4 + 3 = 7$). Các dây chuyền tương ứng là $\{v_1, v_7, v_2\}$ và $\{v_3, v_4\}$.

Nghiệm tối ưu của bài toán nhận được bằng cách thêm vào đồ thị ban đầu các cạnh $(v_1, v_7), (v_7, v_2)$ và (v_3, v_4). Đồ thị G' nhận được là đồ thị Euler (Hình 5.6).



Hình 5.6: Đồ thị Euler G' nhận được từ G bằng cách thêm các cạnh tương ứng các dây chuyền nhỏ nhất giữa 1 và 2 và giữa 3 và 4.

Cuối cùng ta chỉ cần tìm một chu trình Euler trong G', chẳng hạn

$$\{v_6, v_2, v_3, v_7, v_2, v_7, v_1, v_7, v_4, v_5, v_1, v_6\}$$

là chu trình có độ dài $31 + 7 = 38$ là nghiệm tối ưu cần tìm.

5.3 Bài toán Hamilton

Giả sử $G := (V, E)$ là đồ thị liên thông (hay liên thông mạnh trong trường hợp có hướng) có n đỉnh.

Định nghĩa 5.3.1. Dây chuyền (hay đường đi) đi qua tất cả các đỉnh của đồ thị G, mỗi đỉnh một lần, gọi là *dây chuyền Hamilton* (*hay đường đi Hamilton*).

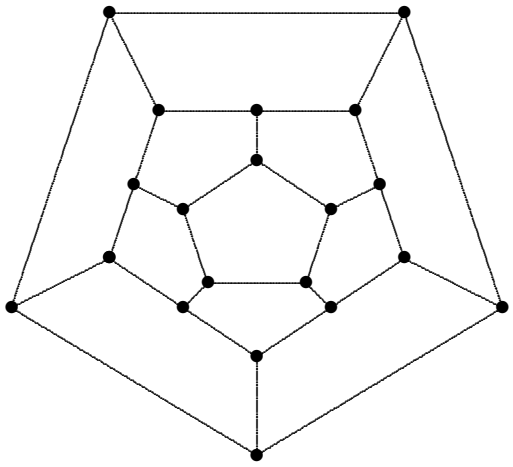
Theo định nghĩa, dây chuyền (hay đường đi Hamilton) là sơ cấp, và có độ dài $(n-1)$.

Chu trình (hay *mạch*) *Hamilton* là một chu trình (hay mạch) đi qua tất cả các đỉnh của đồ thị G, mỗi đỉnh đúng một lần. Dễ thấy rằng, chu trình Hamilton là chu trình sơ cấp có độ dài n. Ta nói rằng, G là *đồ thị Hamilton* nếu nó chứa một chu trình Hamilton (trong trường hợp vô hướng) hoặc một mạch Hamilton (trong trường hợp có hướng).

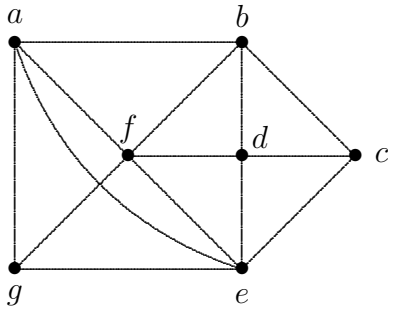
Ví dụ 5.3.2. Năm 1859, nhà toán học Hamilton (1805-1865) người Ailen đã cho bán một đồ chơi độc đáo, phần chính là một khối nhị diện đều (khối đa diện có 12 mặt ngũ giác đều và 20 đỉnh, mỗi đỉnh có 3 cạnh) làm bằng gỗ. Ở mỗi đỉnh có ghi tên một thành phố lớn: Beruych, Quảng châu, Deli, Frangfua, v.v... Cách chơi là tìm một đường đi dọc theo các cạnh của thập nhị diện đều và qua mỗi đỉnh (thành phố) vừa đúng một lần. Muốn trò chơi được hấp dẫn hơn có thể quy định trước trình tự qua một vài thành phố đầu tiên, và để giúp nhớ dễ dàng các thành phố đã đi qua, ở mỗi đỉnh của khối thập nhị diện đều có đóng một chiếc đinh mũ to, quanh đó có thể quấn sợi dây nhỏ để chỉ đoạn đường đã đi qua. Về sau để đơn giản, Hamilton đã thay khối thập nhị diện đều bằng một hình phẳng. Bài toán được phát biểu dưới dạng đồ thị như sau. Ta biết rằng hình thập nhị diện đều có 12 mặt, 30 cạnh, 20 đỉnh; mỗi mặt là một ngũ giác đều, mỗi đỉnh là đầu mút của 3 cạnh. Các đỉnh và các cạnh của hình thập nhị diện đều lập thành một đồ thị như Hình 5.7. Bài toán đặt ra là hãy tìm một chu trình Hamilton của đồ thị G.

Ví dụ 5.3.3. $\mu := (a, b, c, d, e, f, g, a)$ là một chu trình Hamilton trong đồ thị vô hướng của Hình 5.8.

Ví dụ 5.3.4. Các đồ thị vô hướng trong Hình 5.9 không chứa chu trình Hamilton.



Hình 5.7: Hành trình xung quanh thế giới (khối thập nhị diện đều) của Hamilton.

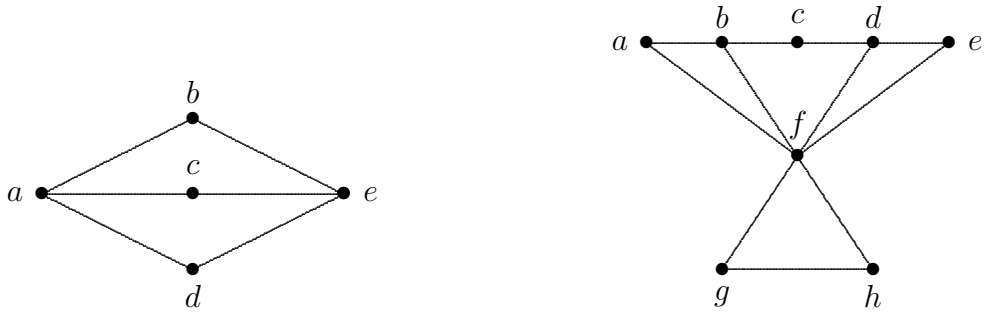


Hình 5.8:

Ví dụ 5.3.5. (*Bài toán người chào hàng*). Một người chào hàng viếng thăm n khách hàng $v_1, v_2, \ldots, v_n$, xuất phát từ thành phố v_0 và sau đó trở về vị trí xuất phát. Anh ta biết khoảng cách d_{0j} từ v_0 đến tất cả các khách hàng v_j và khoảng cách d_{ij} giữa hai khách hàng v_i và v_j (đặt $d_{ij} = d_{ji}$).

Người chào hàng cần đi đến các khách hàng của mình theo thứ tự nào để tổng quãng đường đi là nhỏ nhất? Nói cách khác cần tìm một chu trình Hamilton với độ dài nhỏ nhất trên đồ thị đầy đủ có trọng số được xây dựng từ tập các đỉnh $v_0, v_1, v_2, \ldots, v_n$, và trọng lượng cạnh (v_i, v_j) là d_{ij}. Về các thuật toán giải bài toán này có thể xem, chẳng hạn [30].

Trong trường hợp đỉnh cuối v_{n+1} khác đỉnh xuất phát v_0, bài toán đưa về tìm dây chuyền Hamilton từ v_0 đến v_{n+1} có tổng độ dài nhỏ nhất. Bằng cách biến đổi một



Hình 5.9:

cách thích hợp trên đồ thị, ta có thể đưa về bài toán tìm chu trình Hamilton có tổng độ dài nhỏ nhất.

Ví dụ 5.3.6. (*Bài toán lập lịch*). Trong một số bài toán lập lịch, ta cần tìm một thứ tự thực hiện n tiến trình cho trước (hai tiến trình không được thực hiện cùng một lúc) và thỏa mãn những ràng buộc nhất định; bằng cách xây dựng đồ thị G trong đó tập các đỉnh của G tương ứng với tập các tiến trình và một cung liên thuộc hai đỉnh v_i và v_j nếu tiến trình i được thực hiện trước tiến trình j, bài toán đưa về tìm một đường đi Hamilton trong G.

Ví dụ 5.3.7. (*Lập lịch và bài toán người chào hàng trên đồ thị có hướng*). Trong thực tế ta thường lập kế hoạch mà mỗi cung (v_i, v_j), biểu diễn một ràng buộc nào đó, gắn với một số thực t_{ij} là khoảng thời gian ít nhất có thể bắt đầu thực hiện công việc thứ j khi công việc thứ i đã tiến hành.

Thời gian nhỏ nhất để thực hiện tất cả các tiến trình được xác định bằng cách tìm một đường đi Hamilton có độ dài nhỏ nhất trên đồ thị có hướng. Đây chính là bài toán người chào hàng trên đồ thị có hướng. Về thuật toán giải bài toán này có thể xem, chẳng hạn [30].

Bài toán người chào hàng trên đồ thị vô hướng hoặc có hướng thường gặp trong cuộc sống và có nhiều ứng dụng: lập thời khóa biểu, lập lịch, lắp đặt hệ thống điện, tổng hợp các mạch logic tuần tự, v.v.

Ngoài ra, nhiều bài toán có thể đưa về bài toán người chào hàng: bài toán nhiều người chào hàng, một vài bài toán tìm đường đi ngắn nhất với điều kiện qua tất cả

các đỉnh (hay cung) của một tập cho trước chỉ một lần, các chu trình (hay mạch) Euler có chi phí nhỏ nhất.

Cuối cùng, ta có thể chỉ ra rằng, bài toán người chào hàng trên đồ thị có hướng có thể đưa về trường hợp đồ thị vô hướng.

Trái với bài toán người đưa thư Trung Hoa, ngoại trừ những trường hợp đặc biệt, người ta chưa tìm được một thuật toán đa thức để giải bài toán người chào hàng. Các thuật toán hiệu quả nhất sử dụng các phương pháp nhánh và cận [30].

Định nghĩa 5.3.8. Một chu trình (tương ứng, mạch) đi qua tất cả các đỉnh, mỗi đỉnh ít nhất một lần, gọi là *chu trình* (tương ứng, *mạch*) *tiền Hamilton.* Đồ thị G chứa một chu trình hay mạch như vậy gọi là *đồ thị tiền Hamilton.* Dễ thấy rằng, điều kiện cần và đủ để G là đồ thị tiền Hamilton là G liên thông (liên thông mạnh).

Tìm kiếm một chu trình (hay mạch) Hamilton có độ dài nhỏ nhất trên đồ thị có trọng số đưa về bài toán xác định chu trình (hay mạch) trong đồ thị đầy đủ G' nhận được tập các đỉnh của G và trọng lượng trên cạnh (cung) (v_i, v_j) của G' bằng độ dài của dây chuyền (đường đi) ngắn nhất từ v_i đến v_j trong G.

Nhiều dạng bài toán người chào hàng chính là các bài toán tiền Hamilton, và để giải chúng trước hết ta cần tìm ma trận tương ứng các dây chuyền (đường đi) ngắn nhất.

Cuối cùng nhận xét rằng, tồn tại một trường hợp mà bài toán chu trình tiền Hamilton có độ dài nhỏ nhất có thể đưa về bài toán người đưa thư Trung Hoa; điều này xảy ra khi G là *đồ thị đối ngẫu*[1] của đơn đồ thị vô hướng G^* nào đó và khi độ dài của các cạnh (v_i, v_j) có dạng $a_i + a_j$. Trong trường hợp này, bài toán tìm chu trình tiền Hamilton trong G chính là bài toán người đưa thư Trung Hoa trong G^* với độ dài cạnh e_i^* của G^* là a_i.

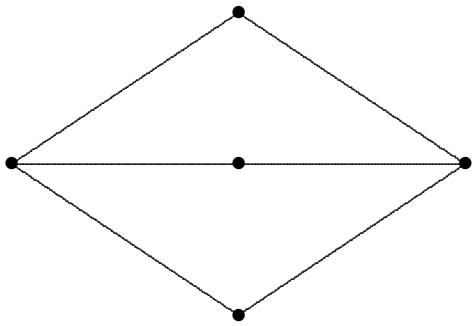
Nhận xét tương tự cho bài toán tìm mạch tiền Hamilton có độ dài nhỏ nhất trong trường hợp đồ thị có hướng G là đồ thị đối ngẫu của đa đồ thị có hướng nào đó.

[1]Đồ thị vô hướng G^* là *đồ thị đối ngẫu* của $G = (V, E)$ nếu mỗi đỉnh của G^* tương ứng với một cạnh $e \in E$ và hai đỉnh trong G^* kề nhau nếu hai cạnh tương ứng kề nhau. Đồ thị có hướng G^* là *đồ thị đối ngẫu* của $G = (V, E)$ nếu mỗi đỉnh e^* của G^* tương ứng với một cung $e \in E$ và tồn tại cung (e_1^*, e_2^*) trong G^* nếu đỉnh ngọn của cung e_1 là đỉnh gốc của cung e_2.

5.3.1 Các điều kiện cần để tồn tại chu trình Hamilton

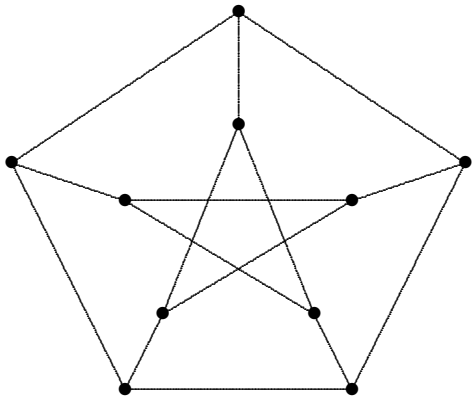
Hiển nhiên rằng, một điều kiện cần để tồn tại chu trình Hamilton là G 2-liên thông. Tuy nhiên, đây không phải điều kiện đủ.

Hình 5.10 là một ví dụ đồ thị 2-liên thông không chứa chu trình Hamilton (hơn nữa, có thể chỉ ra rằng, đó là đồ thị có số đỉnh ít nhất thỏa mãn tính chất này).



Hình 5.10: Đồ thị 2-liên thông có số đỉnh ít nhất không có chu trình Hamilton.

Đồ thị vô hướng Petersen (Hình 5.11) là ví dụ khác không có chu trình Hamilton. Đây là đồ thị chính quy 3-liên thông nhỏ nhất có tất cả các đỉnh bậc ba. Nhiều phản ví dụ về bài toán Hamilton được xây dựng từ đồ thị Petersen.



Hình 5.11: Đồ thị Petersen.

Có nhiều điều kiện cần khác (xem [30], [14]) nhưng không có một điều kiện cần và đủ về sự tồn tại chu trình (mạch) Hamiton.

Do đó chúng ta sẽ tập trung một số điều kiện đủ dẫn đến các phương pháp có tính xây dựng chu trình Hamilton.

5.3.2 Các điều kiện đủ để tồn tại chu trình Hamilton

Mệnh đề 5.3.9. *K_n là đồ thị Hamilton.*

Chứng minh. Hiển nhiên. ◁

Xét đồ thị vô hướng n đỉnh $G := (V, E)$. Giả sử s và t là hai đỉnh không kề nhau sao cho

$$d_G(s) + d_G(t) \geq n.$$

Ký hiệu $G + (s, t)$ là đồ thị nhận được từ G bằng cách thêm cạnh (s, t). Khi đó

Mệnh đề 5.3.10. *Nếu $G + (s, t)$ là đồ thị Hamilton thì G là đồ thị Hamilton. Ta nói tính chất này là ổn định Hamilton qua phép biến đổi $G \to G + (s, t)$.*

Chứng minh. Giả sử $G + (s, t)$ chứa chu trình Hamilton $\mu := (v_1, v_2, \ldots, v_n)$. Nếu μ không đi qua cạnh (s, t) thì G là Hamilton. Ngược lại, G chứa một dây chuyền Hamilton $\mu \setminus (s, t)$ nối s và t. (Không mất tính tổng quát, có thể giả thiết $s = v_1, t = v_n$).

Ta chứng minh rằng tồn tại ít nhất một chỉ số $i, 3 \leq i \leq n$, sao cho s kề với v_i và t kề với v_{i-1}.

Thật vậy, xét các tập con của tập $Y := \{v_3, v_4, \ldots, v_{n-1}\}$:

$$\begin{aligned} A &= \{v_i \mid (s, v_i) \in E \text{ và } 3 \leq i \leq n-1\}, \\ B &= \{v_i \mid (t, v_{i-1}) \in E \text{ và } 3 \leq i \leq n-1\}. \end{aligned}$$

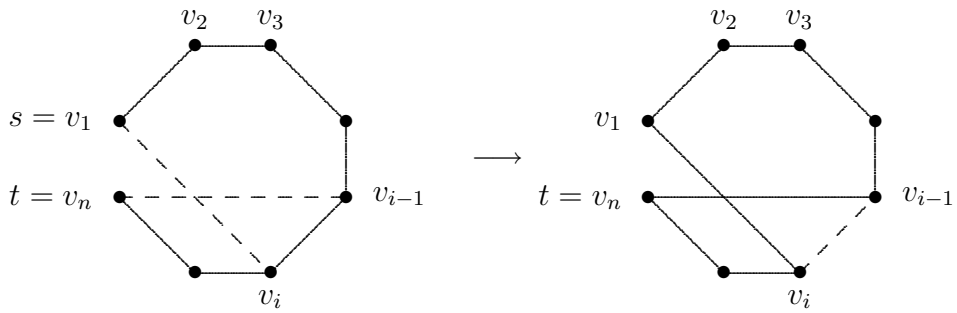
Vì s và t không kề nhau trong G nên

$$\#A + \#B = d_G(s) + d_G(t) - 2 \geq n - 2$$

và nhận xét rằng $\#Y = n - 3$ suy ra tồn tại

$$v_i \in A \cap B \quad (3 \leq i \leq n-1).$$

Do đó, thêm các cạnh (s, v_i) và (t, v_{i-1}) và xóa cạnh (v_i, v_{i-1}) ta được một chu trình Hamilton trong G (xem Hình 5.12), và mệnh đề được chứng minh. ◁



Hình 5.12:

Giả sử G là đơn đồ thị n đỉnh và k là số nguyên thỏa $1 \leq k \leq n$. Xét thủ tục đệ quy sau: Xuất phát từ G, thêm các cạnh nối các đỉnh không kề nhau mà tổng các bậc của chúng lớn hơn hoặc bằng k. (Số phép toán đòi hỏi trong thủ tục này tỉ lệ với n^4). Vì các bậc không giảm, đồ thị nhận được không phụ thuộc vào thứ tự các cạnh được thêm. Đồ thị này (chứa G) gọi là *$k-$bao đóng* của G và ký hiệu là $[G]_k$. Với $k = n$, từ Mệnh đề 5.3.10 suy ra

Định lý 5.3.11. [8] *G là đồ thị Hamilton nếu và chỉ nếu $[G]_n$ là đồ thị Hamilton.*

Trong trường hợp tổng quát, tìm chu trình Hamilton trong $[G]_n$ không phải lúc nào cũng dễ hơn trong G. Tuy nhiên, với những trường hợp đặc biệt, chẳng hạn khi $[G]_n$ là đồ thị đầy đủ K_n ta có thể dễ dàng xây dựng chu trình Hamilton trong $[G]_n$:

Định lý 5.3.12. *Điều kiện đủ để G là đồ thị Hamilton là $[G]_n = K_n$.*

Có thể chỉ ra rằng hầu hết các điều kiện đủ đã biết (được liệt kê dưới đây) liên quan đến bậc của G suy ra $[G]_n = K_n$ và là hệ quả của Định lý 5.3.12.

Giả sử G là đơn đồ thị liên thông n đỉnh.

Hệ quả 5.3.13. [Ore] *[47] Nếu $d_G(v_i) + d_G(v_j) \geq n$ với mọi $(v_i, v_j) \notin E$ thì G là đồ thị Hamilton.*

Hệ quả 5.3.14. [Dirac] *[17] Nếu $d_G(v_i) \geq \frac{n}{2}$ với mọi đỉnh $v_i \in V$ thì G là đồ thị Hamilton.*

Hệ quả 5.3.15. [Pósa] *[51] Nếu các đỉnh của G được đánh số thự tự sao cho*

$$d_G(v_1) \leq d_G(v_2) \leq \cdots \leq d_G(v_n)$$

và nếu

$$\forall k : 1 \leq k < \frac{n}{2} \Rightarrow d_G(v_k) > k,$$

thì G là đồ thị Hamilton.

Hệ quả 5.3.16. [Bondy] *[7] Giả sử các đỉnh của G được đánh số thự tự sao cho*

$$d_G(v_1) \leq d_G(v_2) \leq \cdots \leq d_G(v_n).$$

Nếu điều kiện sau đúng:

$$p < q,\ d_G(v_p) \leq p, d_G(v_q) \leq q - 1 \Rightarrow d_G(v_p) + d_G(v_q) \geq n,$$

thì G là đồ thị Hamilton.

Hệ quả 5.3.17. [Chvátal] *[13] Nếu các đỉnh của G được đánh số thự tự sao cho*

$$d_G(v_1) \leq d_G(v_2) \leq \cdots \leq d_G(v_n)$$

và nếu

$$d_G(v_k) \leq k < \frac{n}{2} \Rightarrow d_G(v_{n-k}) \geq n - k$$

thì G là đồ thị Hamilton.

Hệ quả 5.3.18. [Las Vergnas] *[42] [30] Nếu các đỉnh của G được đánh số thự tự $v_1, v_2, \ldots, v_n$ sao cho*

$$\left.\begin{array}{l} j < k, k \geq n - j \\ (v_j, v_k) \notin E \\ d_G(v_j) \leq j, d_G(v_k) \leq k - 1 \end{array}\right\} \Rightarrow d_G(v_j) + d_G(v_k) \geq n$$

thì G là đồ thị Hamilton.

Các chứng minh. Chứng minh của Hệ quả 5.3.13 suy trực tiếp từ cách xây dựng $[G]_n$: tất cả các cạnh $(v_i, v_j) \notin E$ có thể được thêm và do đó $[G]_n = K_n$. Hệ quả 5.3.14 suy trực tiếp từ Hệ quả 5.3.13.

Hơn nữa, dễ dàng thấy rằng, đồ thị thỏa mãn các điều kiện của Hệ quả 5.3.15, 5.3.16 hay 5.3.17 cũng thỏa mãn các giả thiết của Hệ quả 5.3.18.

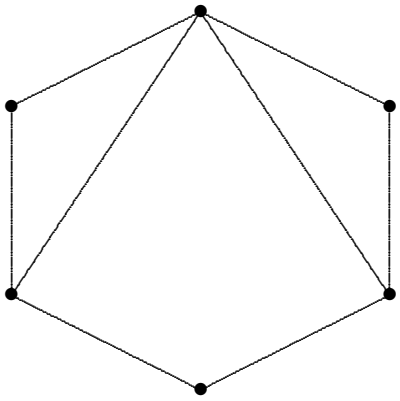
Để chứng minh Hệ quả 5.3.18, ta sẽ sử dụng kết quả sau cho điều kiện đủ để $[G]_n = K_n$.

Định lý 5.3.19. [8] *Nếu các đỉnh của G được đánh số thự tự $v_1, v_2, \ldots, v_n$ sao cho*

$$\left.\begin{array}{l} i < j \\ (v_i, v_j) \notin E \\ d_G(v_i) \leq i + k - n \\ d_G(v_j) \leq j + k - n - 1 \\ i + j \geq 2n - k \end{array}\right\} \Rightarrow d_G(v_i) + d_G(v_j) \geq k$$

thì $k-$bao đóng của G là đồ thị đầy đủ K_n.

Hệ quả 5.3.18 suy trực tiếp từ Định lý 5.3.19 với $k = n$. Hệ quả 5.3.18 là tổng quát nhất của tất cả các điều kiện đã biết có liên quan đến bậc của đồ thị. Tuy nhiên, điều kiện đủ của Định lý 5.3.12 là tổng quát hơn: đồ thị trong Hình 5.13 có $[G]_6 = K_6$ nhưng không tồn tại cách đánh số các đỉnh của G thỏa các điều kiện của Hệ quả 5.3.18.



Hình 5.13: Với đồ thị này, $[G]_6 = K_6$ nhưng không thể áp dụng Hệ quả 5.3.18.

5.3.3 Các điều kiện đủ để tồn tại mạch Hamilton

Trong trường hợp đồ thị có hướng, có một số các điều kiện đủ bảo đảm sự tồn tại của mạch Hamilton. Kết quả tổng quát nhất là:

Định lý 5.3.20. [Meyniel, 1973] *Giả sử $G = (V, E)$ là đồ thị có hướng n đỉnh liên thông mạnh không khuyên sao cho*

$$(v_i, v_j) \notin E \quad và \quad (v_j, v_i) \notin E \Rightarrow d_G(v_i) + d_G(v_j) \geq 2n - 1.$$

Khi đó G chứa một mạch Hamilton.

Chúng ta sẽ đưa ra chứng minh có tính kiến thiết định lý này (theo Minoux, 1977) và một thuật toán có độ phức tạp $O(n^4)$ để tìm mạch Hamilton. Chứng minh dựa trên phương pháp (không kiến thiết) của Bondy và Thmassen (1977). Trước hết, chúng ta cần một số khái niệm.

Giả sử $S \subset V$. Với mỗi $v_i \in V$ (có thể $v_i \in S$), ký hiệu $\delta_S(i)$ là số các cung nối (theo hướng bất kỳ) đỉnh v_i với tập con S. Vì G không có khuyên nên ta luôn luôn có

$$v_i \in S \Rightarrow \delta_S(i) \leq 2\#S - 2.$$

Với S là tập con thực sự của V ta gọi *$S-$bộ hành* là một đường đi mà các đỉnh đầu và cuối (không nhất thiết phân biệt) thuộc S và các đỉnh trung gian là phân biệt và tạo thành một tập con khác trống của tập $V \setminus S$. Một *$S-$đường đi* là một $S-$bộ hành mà các điểm đầu và cuối khác nhau.

Ta có bổ đề sau:

Bổ đề 5.3.21. *Giả sử $\mu = (v_0, v_1, \ldots, v_p)$ là đường đi sơ cấp của G, S là tập các đỉnh của nó và đặt $v \in V \setminus S$. Nếu không tồn tại hai đỉnh liên tiếp v_k và v_{k+1} $(0 \leq k \leq p-1)$ của μ sao cho $(v_k, v) \in E$ và $(v, v_{k+1}) \in E$ thì*

$$\delta_S(v) \leq \#S + 1.$$

Chứng minh. Xét các tập con của $S \setminus \{v_p\}$:

$$A := \{v_i \in S \setminus \{v_p\} \mid (v_i, v) \in E\}$$

và

$$A := \{v_i \in S \setminus \{v_p\} \mid (v, v_{i+1}) \in E\}.$$

Theo giả thiết, $A \cap B = \emptyset$ và $\#(A \cup B) \leq \#S - 1$ (do $v_p \notin A, v_p \notin B$). Vậy

$$\delta_S(v) \leq \#A + \#B + 2 = \#(A \cup B) - \#(A \cap B) + 2 \leq \#S + 1,$$

điều phải chứng minh. ◁

Bây giờ chúng ta chứng minh Định lý 5.3.20.

Chứng minh của Định lý 5.3.20. Xét đồ thị G thỏa mãn các điều kiện của định lý.

(1) Vì G liên thông mạnh nên tồn tại một mạch độ dài ít nhất hai.

Ta nói mạch μ trong G là *tựa cực đại* nếu với mọi đỉnh $v \notin \mu$ không tồn tại hai đỉnh v_k và v_{k+1} liên tiếp trên mạch μ sao cho

$$(v_k, v) \in U \quad \text{và} \quad (v_{k+1}, v) \in U.$$

Hiển nhiên rằng để kiểm tra một mạch có phải tựa cực đại hay không, hoặc để phát hiện một mạch độ dài lớn hơn 1 ta cần thực hiện $O(n^2)$ phép toán sơ cấp.

Do đó, thuật toán xây dựng một mạch tựa cực đại trong G bắt đầu từ một mạch tùy ý đòi hỏi $O(n^3)$ phép toán sơ cấp.

Giả sử μ là mạch tựa cực đại, và ký hiệu S là tập các đỉnh của nó. Nếu $S = V$ ta dừng: μ là mạch Hamilton. Ngược lại, ta sẽ chỉ ra rằng có thể mở rộng mạch μ để nhận được một mạch μ' có độ dài lớn hơn, và do đó, theo quy nạp, ta sẽ có một mạch Hamilton.

(2) Chúng ta chứng minh tồn tại một $S-$đường đi trong G.

Bằng phản chứng, giả sử không tồn tại $S-$đường đi. Vì G liên thông mạnh, tồn tại ít nhất một $S-$bộ hành P mà các đỉnh đầu cuối của nó là $v \in S$.

Đặt R là tập các đỉnh thuộc P và khác v; đặt T là tập các đỉnh của G không thuộc $S \cup R$.

Theo giả thiết không tồn tại $S-$đường đi nên không tồn tại đỉnh thuộc R và kề với một đỉnh trong $S \setminus \{v\}$.

Do đó với mỗi đỉnh $v_i \in R$ và đỉnh $v_j \in S \setminus \{v\}$ ta có

$$\begin{aligned} &\delta_R(v_i) \leq 2\#R - 2, \quad \delta_R(v_j) = 0, \\ &\delta_S(v_i) \leq 2, \qquad\qquad \delta_S(v_j) \leq 2\#S - 2. \end{aligned}$$

Hơn nữa ta cần có

$$\delta_T(v_i) + \delta_T(v_j) \leq 2\#T.$$

(Nếu không, tồn tại ít nhất một đường đi độ dài 2 hoặc giữa v_i và v_j, hoặc giữa v_j và v_i; đường đi này (có một phần nằm trên P) tạo thành một $S-$đường đi).

Có thể chứng minh rằng các đỉnh v_i và v_j là hai đỉnh không kề nhau sao cho

$$\begin{aligned} \delta(v_i) + \delta(v_j) &= \delta_S(v_i) + \delta_R(v_i) + \delta_T(v_i) + \delta_S(v_j) + \delta_R(v_j) + \delta_T(v_j) \\ &\leq 2(\#R + \#S + \#T) - 2 = 2n - 2, \end{aligned}$$

mâu thuẫn với giả thiết.

(3) Vì tồn tại ít nhất một $S-$đường đi, ta sẽ tìm một $S-$đường đi P xuất phát từ x và kết thúc tại y sao cho độ dài của đường đi con $\mu(x, y)$ từ x đến y trên μ là nhỏ nhất (theo số các cung).

Với một đỉnh $x \in S$ cho trước, chúng ta có thể sử dụng thuật toán tìm kiếm theo chiều rộng trên đồ thị G như đã trình bày trong Chương 3. Phương pháp này đòi hỏi $O(m)$ phép toán sơ cấp. Do đó thuật toán tìm một $S-$đường đi P với tính chất đòi hỏi cần nhiều nhất $O(mn)$ phép toán sơ cấp.

Ký hiệu R là tập các đỉnh trung gian của P, và S_1 là tập các đỉnh trung gian trên đường đi $\mu(x, y)$ của μ.

Nếu $S_1 = \emptyset$ thì thay cung (x, y) bởi $S-$đường đi P và chúng ta nhận được một mạch μ' có độ dài lớn hơn hoặc bằng $(\#S + 1)$.

(4) Kế tiếp ta giả sử $S_1 \neq \emptyset$, và đặt $S_2 := S \setminus S_1$ và T là tập các đỉnh của G không thuộc R và S.

Chúng ta xây dựng một đường đi sơ cấp tựa cực đại Q giữa y và x mà tập các đỉnh của nó là S' thỏa mãn $S' \subset S$ và $S_2 \subset S'$ bằng phương pháp lặp như sau:

1. Khởi tạo với $Q := \mu(x, y), S' := S_2, S'' := S_1$.

2. Tìm đỉnh $s \in S''$ sao cho $(k, s) \in U$ và $(s, l) \in U$ với hai đỉnh k và l liên tiếp trên Q. (Nhận xét rằng, điều này đòi hỏi nhiều nhất $O(n)$ phép toán sơ cấp). Nếu không tồn tại đỉnh s như vậy (hoặc là $S'' = \emptyset$) thì dừng: đường đi Q là tựa cực đại (hoặc cực đại).

3. Giả sử tồn tại đỉnh s thì chèn nó vào giữa hai đỉnh k và l để nhận được một đường đi mới Q' có độ dài lớn hơn đường đi trước đó một. Thay S' bởi $S' \cup \{s\}, S''$ bởi $S'' \setminus \{s\}, Q$ bởi Q' và lặp lại Bước 2.

Chú ý rằng, số phép toán cần thực hiện trong thuật toán này không vượt quá $O(n^3)$.

Ta sẽ chỉ ra rằng, kết thúc thuật toán thì $S'' = \emptyset$.

Thật vậy, giả sử ngược lại $S'' \neq \emptyset$.

Theo cách xây dựng của P, các đỉnh của R không kề với một đỉnh của S_1 (vì nếu ngược lại, ta có thể tìm một $S-$đường đi mà các đỉnh đầu cuối của nó gần μ hơn P). Do đó, với mọi đỉnh $v_i \in R$ và mọi đỉnh $v_j \in S_1$ ta có

$$\delta_R(v_j) = \delta_{S_1}(v_i) = 0.$$

Mặt khác, vì μ là mạch tựa cực đại, đỉnh v_i thỏa mãn các giả thiết của Bổ đề 5.3.21 với đường đi con $\mu(x, y)$ sao cho

$$\delta_{S_2}(v_i) \leq \#S_2 + 1.$$

Chọn $v_j \in S'' \subset S_1$. Theo cách xây dựng của Q, áp dụng lại Bổ đề 5.3.21 với đỉnh v_j và đường đi Q, ta được

$$\delta_{S'}(v_j) \leq \#S' + 1.$$

Mặt khác, ta cần có (như đã chứng minh trong Phần (2))

$$\delta_T(v_i) + \delta_T(v_j) \leq 2\#T$$

(ngược lại, ta có thể tìm một $S-$đường đi mà các đỉnh đầu cuối của nó gần μ hơn P).

Cuối cùng, ta luôn luôn có

$$\delta_{S''}(v_j) \leq 2\#S'' - 2, \quad \text{và} \quad \delta_R(v_i) \leq 2\#R - 2.$$

Suy ra

$$\delta(v_i) + \delta(v_j) \leq 2\#R + \#S_2 + 2\#S'' + \#S' - 2$$

và

$$\#S_2 \leq \#S' \Rightarrow \delta(v_i) + \delta(v_j) \leq 2n - 2$$

với hai đỉnh không kề nhau v_i và v_j, mâu thuẫn với giả thiết.

Vậy $S'' = \emptyset$ và từ đường đi Q với $S-$đường đi P ta có thể xây dựng một mạch μ' có độ dài $\geq \#S + 1$.

(5) Trong tất cả các trường hợp (tham khảo các Phần (3) và (4) trên) khi $\#S < n$ ta có thể tìm (nhiều nhất $O(mn + n^3)$ phép toán sơ cấp) một mạch μ' có độ dài lớn hơn mạch cũ 1. Trước khi lặp lại thuật toán, ta cần kiểm tra μ' có phải là mạch tựa cực đại. (Nếu không phải, xây dựng một mạch tựa cực đại μ'' từ μ'-đòi hỏi nhiều nhất $O(n^3)$ phép toán sơ cấp).

Sau nhiều nhất $(n-1)$ lần lặp mở rộng mạch, thuật toán cho chúng ta một mạch Hamilton của đồ thị G. Định lý 5.3.20 được chứng minh bằng thuật toán kiến thiết cách xây dựng mạch Hamilton với độ phức tạp thuật toán là $O(n^4)$. ◁

5.4 Mã Gray

Khảo sát sự hoạt động của một đồng hồ đo. Có những vị trí mà tại đó một vài chữ số cần phải thay đổi đồng thời. Do giới hạn của bộ phận cơ khí, các thay đổi này không hoàn toàn đồng thời. Do đó, số ghi tại các vị trí này có thể xảy ra lỗi. Chẳng hạn, trong quá trình thay đổi từ 36999 đến 37000, số ghi trên đồng hồ đo có thể là 36000 hoặc là 37999; và thậm chí nếu chúng ta giả sử rằng các chữ số thay đổi tuần tự từ phải sang trái, thì giá trị 36000 vẫn có thể xuất hiện trên đồng hồ đo.

Để loại trừ lỗi loại này, chúng ta cần sắp xếp các số theo thứ tự (khác với thứ tự thông thường) sao cho chỉ có một chữ số thay đổi tại mỗi bước. Nếu điều đó thực hiện

được, thì các lỗi chỉ có thể xảy ra khi các thao tác của máy bị trễ, và nhiều nhất là một lỗi. Trong trường hợp biểu diễn nhị phân, một dãy như vậy được gọi là *mã Gray*.

Xét đồ thị vô hướng $Q_n := (V, E)$, trong đó mỗi đỉnh tương ứng với một vector n chiều trong $\mathbb{B}^n$; hai đỉnh kề nhau nếu hai vector tương ứng chỉ khác nhau đúng một vị trí. (Chú ý rằng đồ thị vô hướng này có 2^n đỉnh mà chúng ta viết như biểu diễn nhị phân của các số nguyên từ 0 đến $2^n - 1$).

Định lý 5.4.1. *Đồ thị vô hướng $Q_n, n \geq 2$, chứa chu trình Hamilton.*

Chứng minh. định lý bằng quy nạp.

+ Với $n = 2$: ta có chu trình Hamilton $\{00, 01, 11, 10, 00\}$.

+ Giả sử định lý đúng đến n: Đồ thị Q_n có chu trình Hamilton $(v_0, v_1, ..., v_k)$, với $k = 2^n - 1$.

Khi đó

$$0v_0, 0v_1, \ldots, 0v_{k-2}, 0v_{k-1}, 1v_{k-1}, 1v_{k-2}, 1v_1, 1v_0, 0v_0$$

là chu trình Hamilton trong Q_{n+1}. ◁

Chu trình Hamilton trong Q_n cho ta mã Gray. Đặc biệt, chúng ta có thể "mã hóa" và "giải mã" bộ mã này. Thật vậy ta có thể tính số hạng thứ $N, 2^n \geq N$, trong bộ mã Gray, và vị trí (biểu diễn nhị phân) của N trong bộ mã.

Định lý 5.4.2. *Giả sử $x_{n-1}x_{n-2}\ldots x_0$ là biểu diễn nhị phân của N, và*

$$y_i = x_i + x_{i+1} \ (mod\ 2),\ i = 0, 1, \ldots, n-1 \qquad (x_n = 0).$$

Nếu $y_{n-1}y_{n-2}\ldots y_0$ là biểu diễn nhị phân của M thì số ở vị trí thứ N trong mã Gray là M.

Ngược lại đặt $y_{n-1}y_{n-2}\ldots y_0$ là biểu diễn nhị phân của M. Giả sử x_i là số các số một trong tập $\{y_{n-1}, y_{n-2}, \ldots, y_i\}$ (mod 2); và đặt $x_{n-1}x_{n-2}\ldots x_0$ là biểu diễn nhị phân của N. Khi đó số M xuất hiện ở vị trí thứ N trong mã Gray.

Chứng minh. Bài tập. ◁

Chương 6

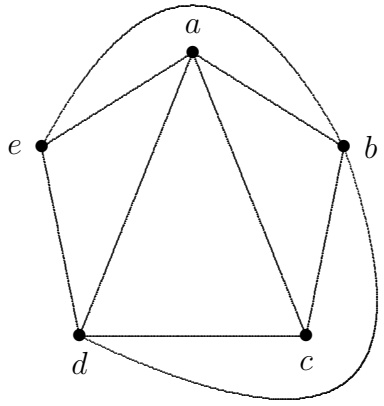
Đồ thị phẳng

Chúng ta đã nghiên cứu các tính chất của các đồ thị con, chẳng hạn, các dây chuyền, chu trình và các cây bao trùm trong một đồ thị đã cho. Trong chương này, chúng ta sẽ nghiên cứu toàn bộ đồ thị G với câu hỏi sau: *khi nào G là phẳng*?

Chương này bắt đầu với hai ví dụ về đồ thị Kuratowski đóng vai trò trung tâm trong việc kiểm tra tính phẳng của đồ thị. Kế tiếp là các tính chất của đồ thị phẳng, chẳng hạn định lý Euler và giả thuyết bốn màu nổi tiếng "mọi đồ thị phẳng là 4−sắc" (phát biểu năm 1850 và được chứng minh năm 1976). Chúng ta cũng tìm hiểu tiêu chuẩn cần và đủ để đồ thị là phẳng-Định lý Kuratowski. Cuối cùng là đối ngẫu hình học của đồ thị phẳng.

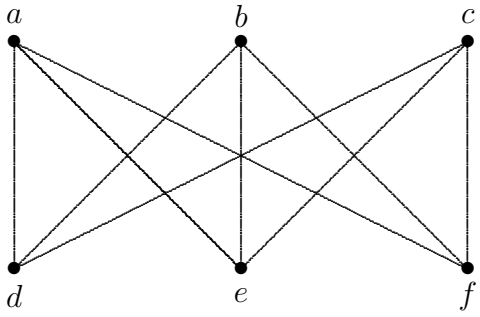
6.1 Định nghĩa và các ví dụ

Nhắc lại rằng đồ thị G được gọi là *phẳng* nếu tồn tại một phép biểu diễn G lên một mặt phẳng sao cho hai cạnh bất kỳ của đồ thị không cắt nhau ngoại trừ tại đỉnh của chúng. Các đồ thị mà có thể biến đổi cho trùng nhau bằng phép biến dạng co dãn liên tục trên mặt phẳng thì không coi là những đồ thị phẳng khác nhau. Theo định nghĩa, đồ thị trong Hình 6.1 là phẳng.



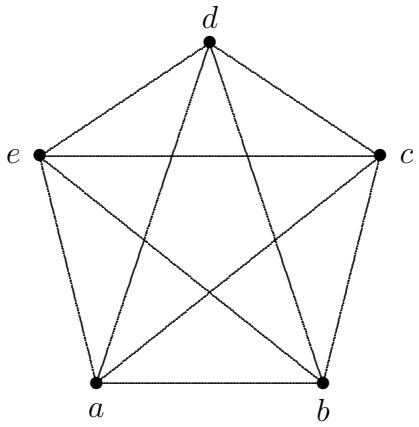
Hình 6.1: Ví dụ về đồ thị phẳng.

Ví dụ 6.1.1. *(Bài toán ba biệt thự và ba nhà máy).* Có ba biệt thự a, b, c và ba nhà máy: một nhà máy nước d, một nhà máy hơi đốt e và một nhà máy điện f. Mỗi biệt thự nối với các nhà máy bằng những ống dẫn nước, ống dẫn hơi và đường dây điện. Vậy có thể vẽ trên mặt phẳng ba biệt thự và ba nhà máy và tất cả các đường vận chuyển sao cho không có hai đường nào cắt nhau ở một điểm khác các đầu mút của chúng hay không? Ta có mô hình hóa bởi đồ thị hai phần $K_{3,3}$: Các đỉnh của đồ thị tương ứng các nhà và các nhà máy (xem Hình 6.2); một cạnh liên thuộc hai đỉnh tương ứng nét vẽ từ nhà đến nhà máy. Bài toán đưa về kiểm tra tính phẳng của *đồ thị Kuratowski* $K_{3,3}$. Bằng thực nghiệm có thể chỉ ra rằng đồ thị này không phẳng.



Hình 6.2: Đồ thị Kuratowski $K_{3,3}$.

Ví dụ 6.1.2. Cũng bằng thực nghiệm, có thể chứng tỏ đồ thị đầy đủ K_5 (xem Hình 6.3) là không phẳng.



Hình 6.3: Đồ thị Kuratowski K_5.

Về tính không phẳng của các đồ thị $K_{3,3}$ và K_5 sẽ được giải thích trong những phần sau. Chú ý rằng các đồ thị K_5 và $K_{3,3}$ có những tính chất:

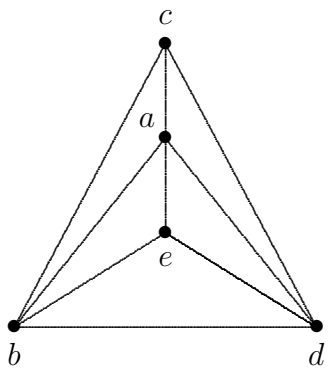
1. Cả hai là không phẳng;
2. Nếu xóa một cạnh hoặc một đỉnh của đồ thị thì sẽ nhận được một đồ thị phẳng.
3. K_5 là đồ thị không phẳng có số đỉnh ít nhất; $K_{3,3}$ là đồ thị không phẳng có số cạnh ít nhất.

6.2 Các biểu diễn khác nhau của một đồ thị phẳng

Trước hết ta có kết quả sau.

Định lý 6.2.1. [Fary] *Mọi đơn đồ thị phẳng có thể biểu diễn trên một mặt phẳng sao cho các cạnh là các đoạn thẳng và chúng chỉ cắt nhau tại các đỉnh chung.*

Chứng minh. Chứng minh của định lý này khá phức tạp và không đóng góp nhiều trong việc hiểu tính phẳng của đồ thị. Bạn đọc quan tâm có thể xem [24]. Chẳng hạn, đồ thị trong Hình 6.1 có thể vẽ lại chỉ sử dụng các đoạn thẳng như trong Hình 6.4. Trong định lý này, đồ thị cần phải đơn vì các khuyên hoặc một trong hai cạnh song song không thể vẽ bởi các đoạn thẳng. ◁



Hình 6.4:

Diện

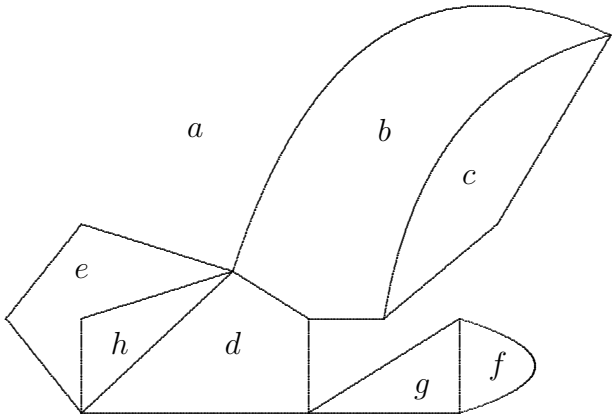
Ký hiệu $R(G)$ là đồ thị phẳng G được biểu diễn trên mặt phẳng. Các vùng liên thông (theo tô pô của mặt phẳng $\mathbf{R}^2$) của tập $\mathbf{R}^2 \setminus R(G)$ gọi là các *diện.* Một diện là một vùng của mặt phẳng được giới hạn bởi các cạnh gọi là *biên;* hai diện gọi là *kề nhau* nếu các đường biên của chúng có ít nhất một cạnh chung.

Nói chung, biên của diện gồm các chu trình đơn giản với các cạnh rời nhau, các *cạnh treo* (cạnh có một đỉnh bậc một), hoặc các cạnh nối hai chu trình. Diện phía ngoài, gọi là *diện vô hạn,* luôn luôn tồn tại một chu trình là đường biên chứa bên trong nó các cạnh khác; đó là chu tuyến của diện vô hạn.

Ví dụ 6.2.2. Bản đồ địa lý là đồ thị tô pô phẳng không có cầu; đồ thị đó có tính chất đặc biệt là: bậc của mỗi đỉnh ≥ 3. Một diện có thể kề với diện khác ở nhiều cạnh. Trên Hình 6.5 ta chú ý rằng các diện d và g không kề nhau mặc dù chúng có một đỉnh chung; a là diện vô hạn.

Biểu diễn trên mặt cầu

Để tránh phân biệt giữa các diện hữu hạn và diện vô hạn, chúng ta thường biểu diễn các đồ thị phẳng trên mặt cầu $\mathbf{S}^2 \subset \mathbf{R}^3$ dựa vào phép chiếu nổi từ $\mathbf{S}^2$ lên $\mathbf{R}^2$. Ký hiệu S là điểm tiếp xúc của mặt cầu với mặt phẳng, và N là điểm thuộc mặt cầu sao cho NS vuông góc với mặt phẳng. Với mỗi điểm $P \in \mathbf{S}^2, P \neq N$, đường thẳng NP cắt mặt phẳng $\mathbf{R}^2$ tại đúng một điểm Q. Như vậy ta có tương ứng một-một từ mặt cầu



Hình 6.5:

(bỏ đi điểm N) lên mặt phẳng.

Theo cách xây dựng này, hiển nhiên rằng đồ thị là phẳng nếu và chỉ nếu có thể biểu diễn nó trên một mặt cầu sao cho các cạnh của nó không cắt nhau ngoài các đỉnh chung. Do đó, theo Định lý 6.2.1 ta có

Định lý 6.2.3. *Một đồ thị có thể biểu diễn trên mặt cầu sao cho các cạnh không tự cắt nếu và chỉ nếu nó có thể biểu diễn phẳng trên mặt phẳng.*

Một đồ thị phẳng được biểu diễn trên mặt cầu sẽ phân chia mặt cầu thành nhiều vùng khác nhau. Mỗi vùng trên mặt cầu là miền giới nội, diện vô hạn trên mặt phẳng được ánh xạ thành vùng trên mặt cầu chứa *điểm cực bắc* N. Hiển nhiên rằng, bằng phép quay thích hợp quả cầu chúng ta có thể ánh xạ một vùng bất kỳ thành diện vô hạn trên mặt phẳng. Từ đó ta nhận được:

Định lý 6.2.4. *Một đồ thị phẳng có thể biểu diễn trên một mặt phẳng sao cho diện bất kỳ cho trước là diện vô hạn.*

Mối liên quan các vùng trên mặt cầu cho phép chúng ta thấy rằng không có sự phân biệt thực sự giữa diện vô hạn và các diện hữu hạn trên mặt phẳng. Do đó, khi nói về các vùng trong một biểu diễn của đồ thị phẳng trên mặt phẳng, chúng ta xét

cả diện vô hạn. Ngoài ra, do không có sự khác nhau cốt yếu giữa các phép biểu diễn đồ thị trên một mặt phẳng hay mặt cầu (mặt phẳng có thể xem là mặt cầu bỏ đi một điểm), thuật ngữ "biểu diễn phẳng" của một đồ thị thường được sử dụng để bao hàm các phép biểu diễn trên mặt cầu cũng như trên mặt phẳng.

Biểu diễn trên mặt phẳng và tính liên thông

Trong một đồ thị không liên thông, việc biểu diễn mỗi thành phần trên mặt phẳng có thể được khảo sát một cách độc lập. Do đó, đồ thị là phẳng nếu và chỉ nếu các thành phần liên thông của nó là phẳng. Tương tự, trong một đồ thị tách được (tức 1−liên thông) việc biểu diễn mỗi khối (tức là đồ thị con không tách được cực đại) trên mặt phẳng có thể được khảo sát độc lập. Do đó một đồ thị tách được là phẳng nếu và chỉ nếu mỗi khối là phẳng.

Vậy, kiểm tra tính phẳng của đồ thị chỉ cần xét đối với đồ thị không tách được.

Một đồ thị phẳng không tách được có thể biểu diễn duy nhất lên một mặt cầu không? Trước khi trả lời câu hỏi này, chúng ta cần giải thích ý nghĩa của từ "biểu diễn duy nhất". Hai biểu diễn của một đồ thị phẳng là *giống nhau* nếu chúng có thể làm trùng nhau bằng phép quay mặt cầu này thành mặt cầu kia và có thể biến dạng các vùng (không di chuyển đỉnh băng qua cạnh). Nếu tất cả các phép biểu diễn lên một mặt cầu là trùng nhau thì ta nói đồ thị *có duy nhất một phép biểu diễn.* Định lý dưới đây cho chúng ta chính xác câu trả lời khi nào thì một đồ thị được biểu diễn duy nhất trên một mặt cầu.

Định lý 6.2.5. [Whitney] *Phép biểu diễn lên mặt cầu của mỗi đồ thị phẳng* 3*−liên thông là duy nhất.*

Chứng minh. Xem [54]. ◁

Định lý này đóng một vai trò quan trọng trong việc xác định xem một đồ thị có phẳng hay không: Với các đồ thị 3−liên thông, nếu có thể biểu diễn trên một mặt cầu thì phép biểu diễn này là duy nhất.

6.3 Các tính chất của đồ thị phẳng

Phần này liệt kê nghiên cứu một số tính chất cơ bản của đồ thị phẳng. Trước hết ta có

Định lý 6.3.1. [Berge] *Trong một đồ thị tô pô phẳng, các chu tuyến của diện hữu hạn tạo thành một hệ cơ sở các chu trình độc lập.*

Chứng minh. Chúng ta sẽ chứng minh bằng quy nạp theo số diện f của đồ thị G.

Hiển nhiên định lý đúng khi G có một hoặc hai diện hữu hạn. Giả sử định lý đúng cho mọi đồ thị phẳng có số diện hữu hạn là $(f-1)$; ta chứng minh định lý đúng cho mọi đồ thị phẳng có f diện hữu hạn.

Thật vậy, giả sử không phải tất cả các chu tuyến đều là các chu trình không giao nhau (theo nghĩa là có cạnh chung) vì với trường hợp đó thì định lý là hiển nhiên. Khi đó ta có thể bỏ bớt một cạnh e nào đó của G và được đồ thị G' có $(f-1)$ diện hữu hạn mà các chu tuyến của nó theo giả thiết quy nạp lập thành một cơ sở gồm các chu trình độc lập. Khi trả lại cạnh e, ta có một diện hữu hạn mới mà chu tuyến là một chu trình không phụ thuộc vào các chu trình trước (vì nó chứa một cạnh không thuộc vào một chu trình nào trong số các chu trình ấy). Do bổ sung một cạnh không thể làm tăng chu số lên quá một đơn vị, nên số diện hữu hạn của đồ thị G quả là xác định một cơ sở gồm các chu trình độc lập. ◁

Hệ quả 6.3.2. [Euler, 1752] *Giả sử G là đồ thị tô pô phẳng liên thông n đỉnh, m cạnh và f diện. Khi đó*

$$f = m - n + 2.$$

Chứng minh. Thật vậy, số các diện hữu hạn bằng chu số $\nu(G)$, nên

$$f = \nu(G) + 1 = (m - n + 1) + 1 = m - n + 2.$$

Cũng có thể chứng minh cách khác như sau.

Nhận xét rằng chỉ cần chứng minh cho một đơn đồ thị, vì việc thêm một khuyên hoặc thêm một cạnh song song chỉ đơn giản là tăng số diện và số cạnh của đồ thị lên

một đơn vị. Chúng ta cũng có thể xóa tất cả các đỉnh treo và các cạnh liên thuộc nó. Nếu thêm (hoặc bớt) một cạnh và một đỉnh thì hiệu số $(m-n)$ không đổi.

Vì mỗi đơn đồ thị phẳng có thể biểu diễn trên mặt phẳng sao cho mỗi cạnh là một đoạn thẳng (Định lý 6.2.1), nên một đồ thị phẳng có thể được vẽ trên mặt phẳng sao cho các diện là các đa giác. Ký hiệu f là số diện và k_p là số các diện có p cạnh. Vì mỗi cạnh thuộc biên của đúng hai đa giác, nên

$$3k_3 + 4k_4 + 5k_5 + \cdots + rk_r = 2m,$$

trong đó k_r là số các đa giác với số cạnh cực đại.

Mặt khác

$$k_3 + k_4 + \cdots + k_r = f.$$

Tổng tất cả các góc tại mỗi đỉnh trong đa giác tương ứng đồ thị là $2n\pi$. Nhắc lại rằng tổng tất cả các góc trong một đa giác p cạnh là $(p-2)\pi$ và tổng tất cả các góc ngoài là $(p+2)\pi$. Do đó

$$\begin{aligned} 2n\pi &= (3-2)\pi k_3 + (4-2)\pi k_4 + \cdots + (r-2)\pi k_r + 4\pi \\ &= (2m-2f)\pi + 4\pi. \end{aligned}$$

Từ các đẳng thức trên, ta suy ra

$$2(m-f)\pi + 4\pi = 2n\pi.$$

Vậy $f = m - n + 2$. ◁

Ví dụ 6.3.3. [Euler] Ta xét trong không gian ba chiều một đa diện lồi có n đỉnh, m cạnh và f diện. Hiển nhiên ta có thể vẽ nó trên một mặt cầu sao cho hai cạnh bất kỳ không cắt nhau tại những điểm khác các đầu mút. Sau đó tiến hành phép chiếu nổi với tâm ở giữa một trong các diện. Ta được một biểu diễn của đa diện trên mặt phẳng. Vì đồ thị là phẳng nên ta có hệ thức cơ bản sau đây của các đa diện lồi:

$$n - m + f = 2.$$

Hệ quả 6.3.4. *Trong một đơn đồ thị phẳng, liên thông có f diện, n đỉnh, m cạnh $(m \geq 2)$, các bất đẳng thức sau luôn nghiệm đúng*

$$\begin{cases} 2m & \geq & 3f, \\ 3n-6 & \geq & m. \end{cases}$$

Chứng minh. Mỗi diện có biên ít nhất là ba cạnh, mỗi cạnh là chung của chính xác hai diện, do đó

$$2m \geq 3f.$$

Thay thế $f = m - n + 2$ trong công thức Euler, ta có bất đẳng thức thứ hai. ◁

Chúng ta lưu ý rằng bất đẳng thức thứ hai chỉ là một điều kiện cần mà không đủ về tính phẳng của một đồ thị. Nói cách khác, mọi đơn đồ thị phẳng cần phải thỏa mãn các điều kiện này, nhưng điều ngược lại là không đúng. Chẳng hạn đồ thị Kuratowski $K_{3,3}$ thỏa

$$3n - 6 = 3.6 - 6 = 12 \geq m = 9.$$

Nhưng

Ví dụ 6.3.5. Đồ thị $K_{3,3}$ không phẳng. Để chứng minh chúng ta cần chỉ ra rằng trong đồ thị này không tồn tại diện mà biên của nó có số cạnh nhỏ hơn 5. Thật vậy, nếu đồ thị là phẳng thì $2m \geq 4f$.

Theo công thức Euler

$$2m \geq 4(m - n + 2).$$

Vậy

$$2.9 \geq 4(9 - 6 + 2).$$

Hay $18 \geq 20$! Mâu thuẫn. Do đó đồ thị $K_{3,3}$ không phẳng.

Ví dụ 6.3.6. Đồ thị đầy đủ có năm đỉnh K_5 là không phẳng. Thật vậy, nếu ngược lại thì K_5 sẽ có

$$f = m - n + 2 = 10 - 5 + 2 = 7$$

diện. Chu tuyến của mỗi diện đều có ít nhất ba cạnh. Nếu lập đơn đồ thị liên thuộc diện-cạnh, thì số cạnh một mặt $\leq 2m$ và mặt khác lại $\geq 3f$. Từ đó

$$20 = 2m \geq 3f = 21.$$

Vô lý, vậy K_5 là không phẳng.

Hệ quả 6.3.7. *Mỗi bản đồ địa lý có ít nhất một diện mà số cạnh của chu tuyến nhỏ hơn hoặc bằng 5.*

Chứng minh. Thật vậy, trong bản đồ địa lý, mỗi đỉnh là đầu mút của ít nhất 3 cạnh; nếu lập đơn đồ thị liên thuộc đỉnh-cạnh thì số cạnh một mặt $\leq 2m$, mặt khác lại $\geq 3n$, vậy $n \leq 2m/3$. Nếu giả thiết rằng mỗi diện có chu tuyến gồm ít nhất 6 cạnh và nếu lập đơn đồ thị liên thuộc diện-cạnh, thì số cạnh của nó, một mặt $\leq 2m$, mặt khác lại $\geq 6f$. Vậy $f \leq 2m/6$. Ta luôn luôn có thể giả thiết là đồ thị liên thông (nếu không thì chứng minh hệ quả cho từng thành phần), và do đó có thể viết

$$2 = n - m + f \leq 2m/3 - m + m/3 = 0.$$

Điều đó không thể xảy ra. ◁

Hệ quả 6.3.8. *Giả sử G là đơn đồ thị phẳng liên thông. Khi đó tồn tại đỉnh $v \in V$ sao cho $d(v) \leq 5$.*

Chứng minh. Thật vậy, trong đồ thị G, mỗi diện được bao ít nhất bởi ba cạnh khác nhau; nếu lập đồ thị H liên thuộc diện-cạnh (tức là đồ thị gồm tập X các điểm dùng để biểu thị các diện, tập hợp Y các điểm dùng để biểu thị các cạnh và các cạnh nối điểm $x \in X$ với $y \in Y$ nếu diện x liên thuộc cạnh y) thì số cạnh của đồ thị H, một mặt $\leq 2m$, mặt khác lại $\geq 3f$, vậy $f \leq 2m/3$. Nếu mỗi đỉnh là đầu mút ít nhất của 6 cạnh, thì cũng bằng cách như vậy ta thu được $n \leq 2m/6$, nghĩa là (theo công thức Euler)

$$2 = n - m + f \leq m/3 - m + 2m/3 = 0.$$

Vô lý! ◁

6.4 Phát hiện tính phẳng

Việc xác định đồ thị G là phẳng hay không là một bài toán quan trọng, chẳng hạn trong hóa học: ở đó nhiều chất hóa học mà công thức cấu tạo của nó có thể biểu diễn như một đồ thị phẳng; và trả lời câu hỏi này bằng cách cố gắng thử vẽ nó lên một mặt phẳng rõ ràng không phải là một câu trả lời tốt. Về thuật toán thời gian $O(n)$ kiểm tra một đồ thị có phẳng hay không, và nếu phẳng thì biểu diễn nó như thế nào có thể xem [34], [18].

Cũng có một thuật toán [33] thời gian $O(n \log n)$ để kiểm tra tính đẳng cấu của hai đồ thị phẳng. Bài toán này cũng thường nảy sinh trong hóa học khi chúng ta muốn nghiên cứu các phân tử.

Các tiêu chuẩn đơn giản và hiệu quả sau cho phép phát hiện tính phẳng của đồ thị.

Phép rút gọn cơ sở

1. Do đồ thị là phẳng nếu và chỉ nếu các thành phần liên thông của nó là phẳng, chúng ta chỉ cần xét một thành phần liên thông của nó. Cũng vậy, một đồ thị có điểm khớp là phẳng nếu và chỉ nếu các khối của nó là phẳng. Vì vậy đối với một đồ thị G cho trước, xác định

   $$G = \{G_1, G_2, \ldots, G_k\},$$

   trong đó $G_i, i = 1, 2, \ldots, k$, là các đồ thị con không tách được của đồ thị G. Rồi thì chúng ta chỉ cần kiểm tra tính phẳng của mỗi G_i.

2. Do việc thêm hay xóa các khuyên không làm mất tính phẳng, ta xóa các khuyên.

3. Tất cả các cạnh song song cũng không ảnh hưởng trong việc xét tính phẳng, vì vậy xóa tất cả các cạnh song song trừ một cạnh còn lại.

4. Loại bỏ tất cả các đỉnh bậc hai và coi hai cạnh nối với đỉnh đó là một (điều này không ảnh hưởng đến tính phẳng của đồ thị). Nói cách khác thực hiện "rút gọn chuỗi".

   Lặp lại các Bước 3 và 4 cho đến khi ta được một đồ thị đơn giản nhất cho phép xác định dễ dàng tính phẳng của nó.

Đối với một đồ thị liên thông không có điểm khớp G_i, sau khi áp dụng các Bước 3 và 4 ta sẽ được một đồ thị H_i có các đặc trưng:

Định lý 6.4.1. *Đồ thị H_i có một trong ba dạng*

1. Một cạnh đơn; hoặc là

2. Một đồ thị đầy đủ có bốn đỉnh; hoặc là

3. Một đơn đồ thị không tách được có số đỉnh $n \geq 5$ và số cạnh $m \geq 7$.

Chứng minh. Định lý có thể được chứng minh bằng cách khảo sát tất cả các đồ thị liên thông không tách được với số cạnh nhỏ hơn hoặc bằng sáu. ◁

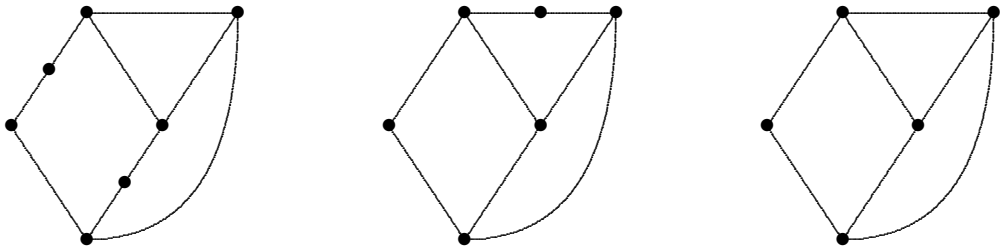
Áp dụng định lý, chúng ta thấy rằng tất cả các đồ thị H_i rơi vào loại 1 hoặc 2 là phẳng, do đó không cần kiểm tra thêm.

Từ những điều nêu trên, chúng ta cần nghiên cứu các đơn đồ thị liên thông không tách được với ít nhất năm đỉnh, và mỗi đỉnh có bậc lớn hơn hoặc bằng ba. Kế đến, chúng ta cần kiểm tra bất đẳng thức $m \leq 3n - 6$. Nếu bất đẳng thức không thỏa mãn, thì H_i không phẳng. Ngược lại, chúng ta kiểm tra thêm bằng cách áp dụng định lý Kuratowski dưới đây. Trước hết ta cần định nghĩa sau.

Định nghĩa 6.4.2. (a) Trong một đồ thị, hai cạnh được gọi là trong một *chuỗi*, nếu chúng có đúng một đỉnh chung bậc hai.

(b) Hai đồ thị được gọi là *đồng phôi* nếu đồ thị này có thể nhận được từ đồ thị kia bằng cách tạo thêm các cạnh trong chuỗi (tức là chèn thêm các đỉnh bậc hai) hay hợp nhất các cạnh trong chuỗi.

Ví dụ 6.4.3. Các đồ thị G_1 và G_2 trong Hình 6.6 là đồng phôi, do có thể đưa về cùng đồ thị G'.



G_1 G_2 G'

Hình 6.6:

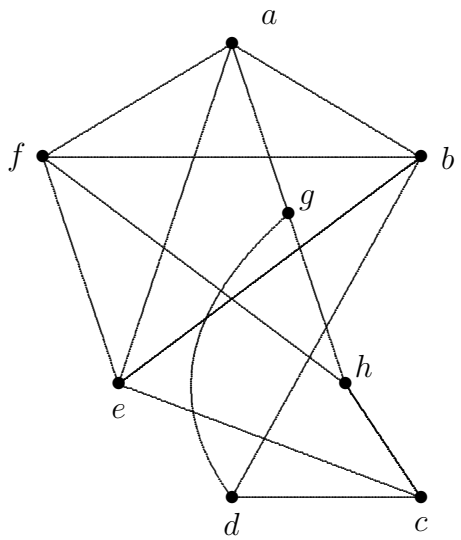
Hiển nhiên rằng đồ thị là phẳng nếu và chỉ nếu mọi đồ thị đồng phôi với nó là phẳng. Hơn nữa, quan hệ $\mathcal{R}$ trên tập các đồ thị xác định bởi $G_1 \mathcal{R} G_2$ nếu và chỉ nếu G_1 và G_2 đồng phôi là quan hệ tương đương.

Định lý 6.4.4. [Kuratowski, 1930] *Đồ thị G là phẳng nếu và chỉ nếu G không chứa đồ thị con đồng phôi với hoặc K_5 hoặc $K_{3,3}$.*

Chứng minh. Điều kiện cần. Rõ ràng, vì G không thể biểu diễn trên mặt phẳng sao cho các cạnh của chúng không cắt nhau ngoại trừ tại các đỉnh chung của chúng nếu nó chứa đồ thị con đồng phôi với K_5 hoặc $K_{3,3}$.

Điều kiện đủ. Xem [4]. ◁

Ví dụ 6.4.5. Bằng cách áp dụng Định lý Kuratowski ta có đồ thị trong Hình 6.7 không phẳng. Thật vậy, trước hết xóa các cạnh $(a, b), (e, f)$ và (g, h); sau đó "rút gọn" các chuỗi $(a, g), (g, d)$ và $(f, h), (h, c)$ để được các cạnh mới (a, d) và (f, c). Đồ thị thu được là $K_{3,3}$. Suy ra G chứa một đồ thị con đồng phôi với $K_{3,3}$ và do đó không phẳng.



Hình 6.7:

Việc kiểm tra tính phẳng dựa vào Định lý Kuratowski là khó đối với các đồ thị lớn (chẳng hạn, đơn đồ thị không tách được có 25 đỉnh và bậc của mỗi đỉnh ≥ 3). Có một số tính chất khác đặc trưng cho các đồ thị phẳng. Một trong những đặc trưng đó sẽ được xét trong phần kế tiếp.

Định lý 6.4.6. *Mọi đồ thị phẳng đều* $5-$*sắc.*

Chứng minh. Hiển nhiên định lý đúng cho các đồ thị phẳng có số đỉnh ≤ 5. Giả sử khẳng định đúng cho mọi đồ thị phẳng có $(n-1)$ đỉnh. Ta sẽ chứng minh định lý cũng đúng cho những đồ thị phẳng có n đỉnh.

Thật vậy, theo Hệ quả 6.3.8, trong G tồn tại đỉnh v có bậc ≤ 5. Đồ thị con nhận được từ G bằng cách xóa đỉnh v và các cạnh liên thuộc nó là $5-$sắc; ta tô các đỉnh của nó bằng năm màu $\alpha, \beta, \gamma, \delta, \epsilon$. Ta có thể tô màu đỉnh v, trừ khi nó kề với năm đỉnh mang năm màu khác nhau.

Trong trường hợp đó, giả sử a, b, c, d, e là năm đỉnh kề với v (theo thứ tự vòng quanh v) và giả sử $\alpha, \beta, \gamma, \delta, \epsilon$ là các màu tương ứng. Ký hiệu $G_{\alpha,\beta}$ là đồ thị con sinh bởi tập hợp các đỉnh có màu α hay β, v.v. Có hai trường hợp xảy ra:

1. Nếu a và c không thuộc những thành phần của đồ thị $G_{\alpha,\gamma}$ thì trong thành phần chứa a ta tô màu γ cho các đỉnh có màu α và ngược lại. Cách tô màu mới cho G là chấp nhận được, và đỉnh v bây giờ được bao quanh bởi các đỉnh có màu $\gamma, \beta, \gamma, \delta, \epsilon$ nên có thể tô nó bằng màu α.

2. Nếu a và c thuộc cùng một thành phần của đồ thị $G_{\alpha,\gamma}$ thì các đỉnh b và d không thể nối với nhau trong đồ thị $G_{\beta,\delta}$ (vì nếu ngược lại, G chứa đồ thị K_5). Khi hoán vị các màu β và δ trong thành phần của đồ thị $G_{\beta,\delta}$ (chứa đỉnh b) ta có thể tô màu β cho đỉnh v được.

Định lý được chứng minh. ◁

Tuy nhiên, ta còn có thể làm tốt hơn:

Định lý 6.4.7. [Giả thuyết bốn màu] *Sắc số của mọi đồ thị phẳng bằng bốn.*

Chứng minh. Xem [1]. ◁

Giả thuyết bốn màu xuất hiện lần đầu tiên trong cuộc trao đổi giữa Morgan và người học trò của ông vào năm 1850. Sau đó, Morgan đã đề cập bài toán này với

Hamilton trong một bức thư viết ngày 23 tháng 10 năm 1852. Kết quả là rất nhiều "chứng minh" cũng như các "phản ví dụ" và các giả thuyết khác có liên quan xuất hiện (để có thêm những thông tin chi tiết, xem [48]). Tuy nhiên, phải đến năm 1976, hai nhà toán học người Mỹ là Appel và Haken (xem [1]), sử dụng hàng ngàn giờ trên máy tính điện tử, đã chứng minh tính đúng đắn của giả thuyết bốn màu.

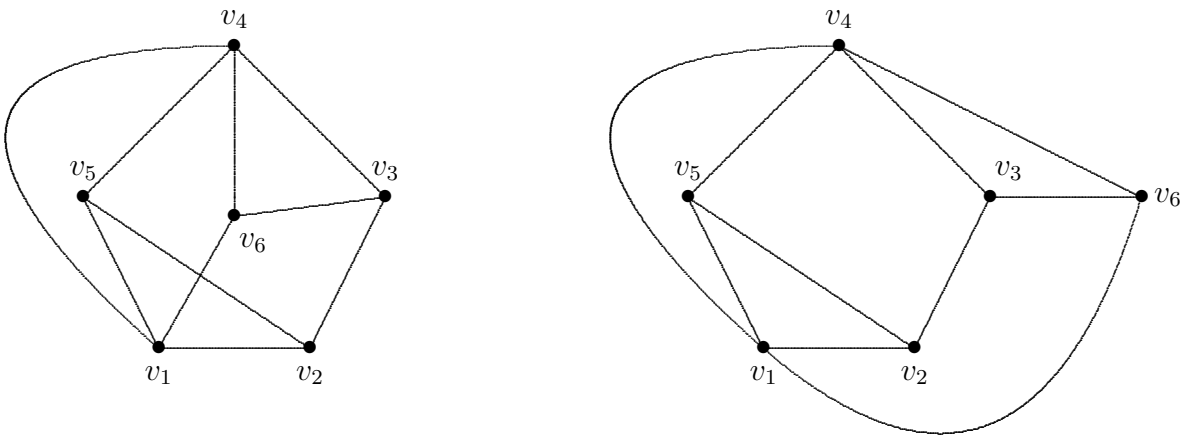
6.4.1 Kiểm tra tính phẳng

Việc xác định một đồ thị có phẳng hay không là một bài toán có ý nghĩa. Như đã chỉ ra, các đặc trưng của Kuratowski và Whitney cho phép kiểm tra tính phẳng của đồ thị. Tuy nhiên các đặc trưng này không phù hợp khi thực hiện trên máy tính do khó thể hiện bằng chương trình; ngoài ra nếu đồ thị là phẳng thì các kết quả này không chỉ ra phương pháp vẽ nó như thế nào. Người ta biết rằng [52], nếu sử dụng tiêu chuẩn của Kuratowski để kiểm tra tính phẳng của đồ thị n đỉnh ($n > 5$) thì thời gian thực hiện ít nhất là $O(n^6)$.

Có một số thuật toán kiểm tra tính phẳng của đồ thị (xem [52] để có tổng quan về các kết quả này). Hầu hết các phương pháp này áp dụng cách "xây dựng bản đồ": Đầu tiên, chọn một đồ thị con phẳng g của G và biểu diễn nó trên mặt phẳng. (Trong hầu hết các thuật toán, g là một chu trình). Sau đó thêm các cạnh vào đồ thị g sao cho nó không cắt các cạnh khác (ngoại trừ tại các đỉnh chung). Nếu chúng ta thành công khi biểu diễn lại thì đồ thị G là phẳng; ngược lại kết luận G không phẳng.

Khó khăn chính của một thuật toán như vậy ở chỗ trong những giai đoạn trước đó, việc thêm các cạnh vào g từ những khả năng có thể không để ý đến các giai đoạn sau này. Một cách đặt cạnh ở bước nào đó có thể dẫn đến không thể đặt cạnh ở bước sau, mặc dù đồ thị đã cho là phẳng. Chẳng hạn, trong Hình 6.8(a), giả sử ta bắt đầu với đồ thị con g là chu trình $\{v_1, v_2, v_3, v_4, v_5, v_1\}$. Sau đó ta thêm các cạnh $(v_1, v_4), (v_4, v_6), (v_3, v_6)$ và (v_1, v_6) sao cho đồ thị thu được vẫn phẳng. Hiển nhiên là cạnh (v_2, v_5) không thể đặt được! Từ đó ta cho rằng đồ thị không phẳng. Tuy nhiên, đồ thị này có thể biểu diễn phẳng như trong Hình 6.8(b).

Đây chính là một trở ngại của phương pháp xây dựng bản đồ; và ta cần một số thủ tục khác giải quyết.



(a) (b)

Hình 6.8: Hai cách biểu diễn của cùng một đồ thị.

Như đã biết, một đồ thị, nói chung, có thể thu gọn thành đồ thị nhỏ hơn để đơn giản hóa tiến trình kiểm tra. Các bước thu gọn không ảnh hưởng đến tính phẳng (hoặc không phẳng) của đồ thị.

1. Kiểm tra tính liên thông của đồ thị. Nếu đồ thị không liên thông, xét từng thành phần liên thông.

2. Xóa các khuyên, và thay tất cả các cạnh song song bởi đúng một cạnh.

3. Xóa các đỉnh bậc hai và hợp nhất hai cạnh liên thuộc đỉnh này. Lặp lại các Bước 2 và 3 cho đến khi không thể rút gọn thêm.

4. Áp dụng Thuật toán Tremaux-Tarjan để phân hoạch đồ thị thành các thành phần không tách được .

5. Với mỗi thành phần không tách được, thực hiện phép thu gọn sử dụng các Bước 3 và 2 cho đến khi không thể thu gọn thêm.

6. Với mỗi thành phần không tách được có n đỉnh, m cạnh, kiểm tra các bất đẳng thức

$$n \geq 5, \quad m \geq 9, \quad m \leq 3n - 6.$$

Nếu các ràng buộc này không thỏa mãn, thuật toán kết thúc; ngược lại kiểm tra thành phần không tách được khác. Mọi đồ thị có $n < 5$ hoặc $m < 9$ là phẳng, và mọi đơn đồ thị có $m > 3n - 6$ không phẳng.

Mỗi đồ thị không tách được có thể tách thành các thành phần 3−liên thông. Sử dụng kết quả của Tutte [54]: *Đồ thị là phẳng nếu và chỉ nếu tất cả các thành phần* 3−*liên thông của nó là phẳng.* Tuy nhiên, phương pháp này không hiệu quả bằng thuật toán của Hopcroft và Tarjan sau đây.

Thuật toán Hopcroft-Tarjan

Thuật toán kiểm tra tính phẳng của đồ thị rất phức tạp. Do đó chúng ta sẽ chỉ trình bày những đặc trưng cốt yếu. Để hiểu thuật toán chính, xét thủ tục phân rã sau áp dụng cho các đơn đồ thị không tách được G có n đỉnh và m cạnh.

Phân rã chu trình-dây chuyền:

1. Tìm chu trình μ trong G. Gán $g = \mu$. Gán nhãn các đỉnh và các cạnh của g là $v_1, v_2, \ldots,$ và $e_1, e_2, \ldots,$ tương ứng. Đặt $i = 1$.

2. Nếu có một cạnh nào của G chưa có nhãn thì tìm một dây chuyền μ_i bắt đầu và kết thúc tại các đỉnh được gán nhãn nhưng dây chuyền này chỉ gồm những cạnh chưa được gán nhãn. Lưu trữ μ_i. Nếu tất cả các cạnh đã được gán nhãn, chuyển sang Bước 4.

3. Đặt $g = g \cup \mu_i$; và $i = i + 1$. Gán nhãn các cạnh và các đỉnh chưa được gán nhãn trong g và chuyển sang Bước 2.

4. Xuất g và các dây chuyền $\mu_1, \mu_2, \ldots, \mu_e$. Dừng.

Có thể chứng minh rằng [52] thủ tục trên sẽ phân rã đơn đồ thị không tách được G thành một chu trình và $e = m - n$ dây chuyền. Vì chu trình có thể xem là kết nối của hai dây chuyền không có chung cạnh, nên G được phân rã thành $m - n + 2$ dây chuyền. Cần chú ý rằng, mặc dù phân rã trên là không duy nhất, nhưng số các dây

chuyền được phân rã từ G là hằng số. Chẳng hạn, Hình 6.8, xét hai phân rã sau thành một chu trình và bốn $(10 - 6 = 4)$ dây chuyền:

$$\{v_1, v_2, v_3, v_4, v_5, v_1\}, \{(v_1, v_4)\}, \{(v_2, v_5)\}, \{(v_4, v_6), (v_1, v_6)\}, \{(v_3, v_6)\},$$

và

$$\{v_1, v_2, v_3, v_4, v_5, v_1\}, \{(v_4, v_6), (v_3, v_6)\}, \{(v_1, v_6)\}, \{(v_1, v_4)\}, \{(v_2, v_5)\}.$$

Trong quá trình phân rã chu trình-dây chuyền, chúng ta có thể biểu diễn chu trình μ trên mặt phẳng, và tiếp tục thêm các dây chuyền mới $\mu_1, \mu_2, \ldots,$ khi chúng được sinh ra. Một dây chuyền μ_i mới hoặc sẽ phân chia một diện đang tồn tại thành hai diện mới hoặc sẽ khiến cho $g \cup \mu_i$ không phẳng khi thêm nó vào g. Do chúng ta thêm tùy ý, nên có thể xảy ra tình trạng như trong Hình 6.8. Để giải quyết khó khăn này, ta có thể hoặc

1. Tiếp tục thêm các dây chuyền cho đến khi không thể thực hiện được nữa. Sau đó trở lại tìm kiếm chọn lựa cách biểu diễn khác (tức là dùng phương pháp lần ngược) thay cho quá trình trước đó; hoặc là
2. Tiếp tục tìm các dây chuyền khác nhau nhưng không đặt chúng vào μ, cho đến khi tìm được diện mà một dây chuyền cần được đặt vào, hoặc là biết chắc rằng không có trở ngại với diện mà đặt dây chuyền vào.

Có một số thuật toán tiếp cận theo hướng thứ nhất, nhưng Hopcroft và Tarjan tiếp cận theo cách thứ hai và đã chứng minh rằng thuật toán của họ là hiệu quả hơn. Ý chính của thuật toán này là giải quyết sự nhập nhằng khi thêm các dây chuyền; cụ thể là:

Giả sử rằng ở bước nào đó, chúng ta có một dây chuyền μ_i (trên đỉnh của một danh sách chứa các dây chuyền cần xử lý) mà chúng ta cần giải quyết sự nhập nhằng của nó. Giả sử a và b là các đỉnh xuất phát và kết thúc của dây chuyền này. Các trường hợp khác nhau có thể xảy ra và phương pháp xử lý tương ứng sẽ được trình bày dưới đây. Các tình huống này được giải thích dựa vào Hình 6.9.

1. Nếu không tồn tại dây chuyền khác P_i và chứa hai đỉnh a và b.

(a) Nếu stack khác trống thì lưu trữ P_i và giải quyết sự nhập nhằng của dây chuyền kế tiếp trong stack.

(b) Ngược lại (stack bằng trống) thì thêm P_i vào g; giải quyết sự nhập nhằng của dây chuyền kế tiếp trong stack.

2. Ngược lại giả sử P_j là dây chuyền (khác P_i) chứa hai đỉnh a và b (xem Hình 6.9(a)). Giữ lại P_i trong danh sách. Xây dựng dây chuyền P_k xuất phát từ đỉnh x nào đó trên P_i và nằm giữa a và b (giả sử P_k kết thúc tại y).

   (a) Nếu y thuộc P_k (xem Hình 6.9(b)) thì đặt dây chuyền P_k trên P_i vào stack và giải quyết tính nhập nhằng của dây chuyền P_k.

   (b) Ngược lại và nếu y không thuộc P_j (Hình 6.9(c)) thì dây chuyền P_i đã được giải. Loại P_i khỏi stack. Giải sự nhập nhằng của dây chuyền kế tiếp trong stack.

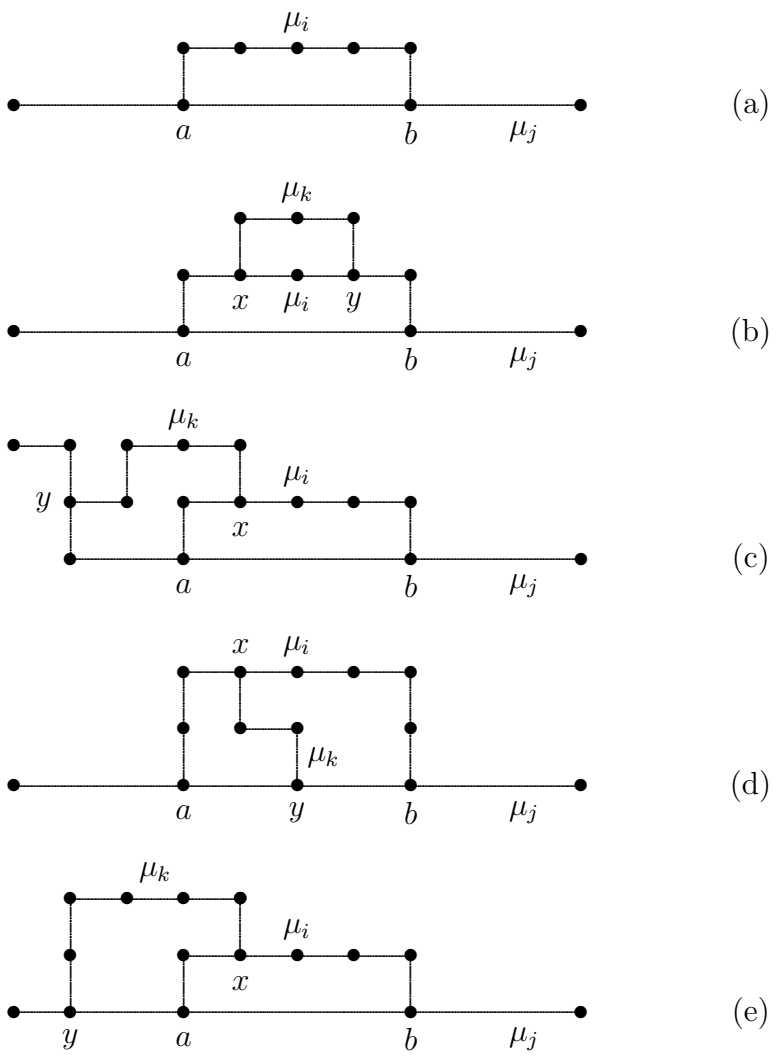
   (c) Ngược lại nếu y nằm giữa a và b (Hình 6.9(d)) thì lưu trữ P_k. Tiếp tục giải tính nhập nhằng của dây chuyền P_i bằng cách tạo dây chuyền khác giữa a và b.

   (d) Ngược lại (Hình 6.9(e)), lưu trữ P_k. Tiếp tục giải tính nhập nhằng của dây chuyền P_i bằng cách tạo dây chuyền khác giữa y và b.

Trong Hình 6.9(a), dây chuyền μ_i có thể xoay tại hai đỉnh a và b, và do đó sẽ phân đôi diện "trên" hoặc "dưới" dây chuyền μ_j. Sự nhập nhằng này cần được xử lý. Để giải quyết, ta xây dựng một dây chuyền μ_k xuất phát từ đỉnh x nào đó trên dây chuyền μ_i. (Dây chuyền μ_k chứa các cạnh chưa được gán nhãn và kết thúc ngay khi nó gặp một đỉnh được gán nhãn, chẳng hạn y).

Nếu cả hai đỉnh x và y của dây chuyền μ_k thuộc dây chuyền μ_i như trong Hình 6.9(b), thì μ_k có thể được xoay tại hai đỉnh x và y và do đó sẽ phân đôi diện phía "trên" hoặc phía "dưới" dây chuyền μ_i. Vì vậy chúng ta không những cần xử lý cho μ_i mà trước hết còn phải giải quyết sự nhập nhằng cho cả dây chuyền mới μ_k. Do đó cần cất μ_k sau khi cất μ_i trong một ngăn xếp, và ta bắt đầu xử lý tương tự μ_i đối với μ_k.

Khả năng khác là đỉnh y không nằm trên hai dây chuyền μ_i và μ_j nhưng lại thuộc một dây chuyền khác như trong Hình 6.9(c). Trong trường hợp này dây chuyền μ_i



Hình 6.9: Phân tích các khả năng xảy ra của dây chuyền μ_i.

không thể xoay sang phía bên kia nên không có sự nhập nhằng về diện mà dây chuyền này phân chia.

Như trong Hình 6.9(d), nếu đỉnh cuối của dây chuyền μ_k nằm trên dây chuyền μ_j giữa hai đỉnh a và b thì dây chuyền μ_i (cùng với μ_k) vẫn có thể xoay tại a và b. Do đó vẫn có sự nhập nhằng đối với dây chuyền μ_i. Tuy nhiên khi đó dây chuyền μ_k phân đôi diện xác định bởi chu tuyến (a, x, b, y, a) không có sự nhập nhằng. Vì vậy, chúng ta cần một dây chuyền khác xuất phát từ một đỉnh trên μ_i để giải quyết sự nhập nhằng với μ_i.

Cuối cùng, trong trường hợp dây chuyền μ_k có đỉnh y nằm trên dây chuyền μ_j nhưng nằm ngoài đoạn xác định bởi a và b như trong Hình 6.9(e). Dây chuyền μ_i vẫn có thể xoay. Nhưng khác với trường hợp của Hình 6.9(b), không có sự nhập nhằng của dây chuyền μ_k tương ứng với μ_i. Do đó, cần tìm một dây chuyền mới để giải quyết sự nhập nhằng của dây chuyền μ_i và dây chuyền mới này có thể xuất phát từ một đỉnh bất kỳ trên dây chuyền y, a, x, b; ngoài ra ta cũng mở rộng dây chuyền μ_i đến đỉnh y trên μ_j.

Tóm lại, thuật toán kiểm tra tính phẳng của đồ thị gồm các thủ tục:

1. Tìm các thành phần không tách được;
2. Phân hoạch các thành phần không tách đượcthành các dây chuyền;
3. Xác định vị trí để thêm các dây chuyền này; và
4. Xây dựng một biểu diễn phẳng.

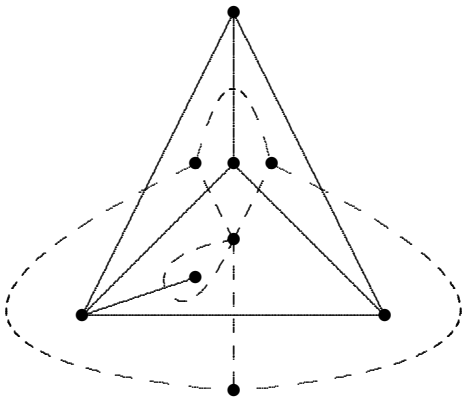
Sử dụng thuật toán tìm kiếm theo chiều sâu để giải quyết các Bước 1 và 2. Các dây chuyền được tạo bởi hai bước trên có những tính chất thích hợp cho các tiến trình về sau. Thời gian đòi hỏi trong mỗi bước (ngoại trừ Bước 4) tỉ lệ với n. Thời gian xây dựng một biểu diễn phẳng là $O(n \log n)$. Vì vậy thời gian thực hiện của thuật toán là $O(n \log n)$ (do công thức Euler, số cạnh m tỉ lệ với số đỉnh n trong một đồ thị phẳng).

Thuật toán này còn có thể cải thiện tốt hơn với thời gian thực hiện là $O(n)$ (xem [34]).

6.5 Đối ngẫu hình học

Xét đồ thị phẳng liên thông G không có đỉnh cô lập. Khi đó ta có thể thiết lập một đồ thị phẳng G^* như sau: bên trong mỗi diện s của $R(G)$ đặt một đỉnh s^* của G^*; với mỗi cạnh e của G thiết lập tương ứng một cạnh e^* của G^* bằng cách nối các đỉnh s^* và t^* tương ứng các diện s và t trên hai phía của cạnh e trong $R(G)$. Đồ thị G^* gọi là *đối ngẫu (tô pô)* của G (hay còn gọi là *đối ngẫu hình học* của G).

Ví dụ 6.5.1. Đồ thị G (đường liền nét) và đối ngẫu của nó (đường đứt nét) cho trong Hình 6.10.



Hình 6.10:

Một chu trình sơ cấp của $R(G)$ chia mặt phẳng thành hai vùng liên thông khác trống và rời nhau. Bỏ đi tất cả các cạnh đối ngẫu tương ứng các cạnh trên chu trình này sẽ phân hoạch $R(G^*)$ thành hai thành phần liên thông. Nói cách khác, mỗi chu trình sơ cấp của G, qua đối ngẫu, tương ứng một đối chu trình sơ cấp của G^*, và ngược lại. Tính chất này rất hữu ích vì ta có thể tìm một đối chu trình của G bằng cách tìm một chu trình của G^*.

Rõ ràng có một sự tương ứng một-một giữa các cạnh của đồ thị G và đồ thị đối ngẫu G^*−một cạnh của G^* giao với một cạnh của G. Hơn nữa chúng ta có một vài nhận xét sau:

1. Cạnh là khuyên trong G tương ứng cạnh treo trong G^*.
2. Cạnh là cạnh treo trong G tương ứng khuyên trong G^*.
3. Nếu G có điểm khớp thì G^* cũng có điểm khớp.
4. Các cạnh trong chuỗi của G tương ứng các cạnh song song trong G^*.
5. Các cạnh song song trong G tương ứng các cạnh trong chuỗi của G^*.
6. Số các cạnh tạo thành biên trong diện F của G, bằng bậc của đỉnh tương ứng trong G^* và ngược lại.

7. Đồ thị G^* là phẳng.

8. Đồ thị đối ngẫu của G^* là G; tức là $(G^*)^* = G$.

9. Nếu ký hiệu n, m và f là số các đỉnh, cạnh và diện tương ứng của đồ thị phẳng liên thông G, và ký hiệu n^*, m^* và f^* là số các đỉnh, cạnh và diện của đồ thị đối ngẫu G^* tương ứng, thì
$$n^* = f, m^* = m, f^* = n.$$

Ví dụ 6.5.2. Xét một mạng vận tải phẳng (vô hướng) G mà mỗi cạnh e có khả năng thông qua $c(e) \geq 0$, đỉnh vào s và đỉnh ra t được xếp trên cùng một đường nằm ngang. Ta lập đồ thị đối ngẫu G^* bằng cách cho hai đỉnh khác nhau s^* và t^* tương ứng với diện vô hạn của G: đỉnh s^* ứng với nửa mặt phẳng dưới, đỉnh t^* ứng với nửa mặt phẳng trên. Gán trọng lượng $w(e^*) = c(e)$ cho cạnh e^* (ứng với e). Bây giờ ta xét các bài toán sau:

Bài toán 1. Trong G tìm thiết diện A có khả năng thông qua nhỏ nhất; tức là làm cực tiểu
$$c(A) = \sum_{v \in A} c(v).$$

Bài toán 2. Trong G^* tìm đường đi ngắn nhất μ^* đi từ s^* đến t^*; tức là làm cực tiểu
$$w(\mu^*) = \sum_{v^* \in \mu^*} w(v^*).$$

Bài toán 1 tương đương với bài toán tìm luồng lớn nhất; Bài toán 2 tương đương với bài toán tìm đường đi ngắn nhất và rõ ràng đối với đồ thị phẳng thì cả hai bài toán đó là một.

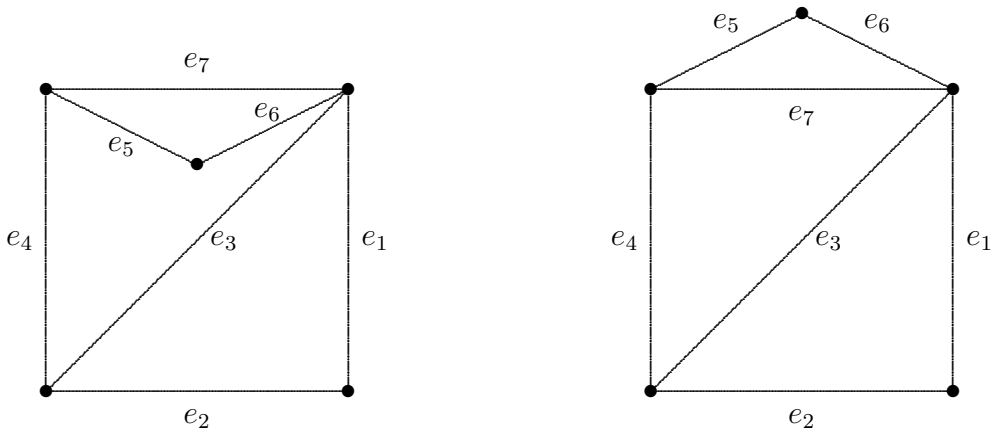
Bài toán bốn màu còn có thể phát biểu dưới dạng đối ngẫu là: *có thể tô các diện của một đồ thị phẳng G bằng bốn màu sao cho không có hai diện kề nhau có cùng một màu.*

Tính duy nhất của đồ thị đối ngẫu

Vấn đề đặt ra là đối ngẫu của một đồ thị là duy nhất? Nói cách khác, các đồ thị đối ngẫu của một đồ thị đã cho là đẳng cấu? Từ phương pháp xây dựng của đồ thị đối

ngẫu, dường như hợp lý cho rằng một đồ thị phẳng G có duy nhất đồ thị đối ngẫu nếu và chỉ nếu nó có một biểu diễn phẳng (trên mặt phẳng hoặc mặt cầu) duy nhất.

Ví dụ 6.5.3. Hai đồ thị trong Hình 6.11 là đẳng cấu với hai biểu diễn phẳng khác nhau. Ngược lại, các đồ thị đối ngẫu của hai đồ thị này không đẳng cấu như trong Hình 6.12.



Hình 6.11: Hai biểu diễn khác nhau của cùng một đồ thị.

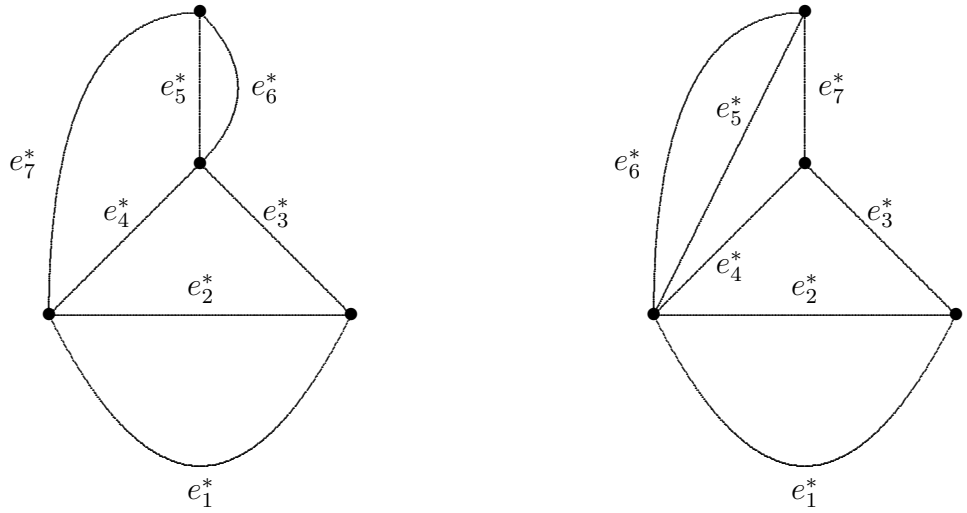
Tuy nhiên các đồ thị trong Hình 6.12 là 2−đẳng cấu. Định lý sau tổng quát ví dụ này.

Định lý 6.5.4. *Tất cả các đồ thị đối ngẫu của một đồ thị phẳng G là 2−đẳng cấu; và mọi đồ thị 2−đẳng cấu với một đối ngẫu của G cũng là một đối ngẫu của G.*

Vì một đồ thị phẳng 3−liên thông có biểu diễn duy nhất trên mặt cầu, nên đối ngẫu của nó là duy nhất. Nói cách khác, tất cả các đối ngẫu của đồ thị 3−liên thông là đẳng cấu.

6.6 Đối ngẫu tổ hợp

Trong các phần trước chúng ta đã định nghĩa và thảo luận về đối ngẫu của các đồ thị phẳng theo nghĩa thuần túy hình học. Phần này cung cấp một định nghĩa tương đương của khái niệm đối ngẫu không phụ thuộc vào các khái niệm hình học. Trước hết ta có



(a) (b)

Hình 6.12: (a) Đối ngẫu của đồ thị trong Hình 6.11(a). (b) Đối ngẫu của đồ thị trong Hình 6.11(b).

Định lý 6.6.1. *Điều kiện cần và đủ để hai đồ thị phẳng G_1 và G_2 là đối ngẫu của nhau là: Tồn tại tương ứng một-một giữa các cạnh trong G_1 và các cạnh trong G_2 sao cho một tập các cạnh trong G_1 tạo thành một chu trình nếu và chỉ nếu tập tương ứng trong G_2 tạo thành một thiết diện.*

Chứng minh. Điều kiện cần. Xét một biểu diễn phẳng của đồ thị G. Chúng ta cũng vẽ một đối ngẫu (hình học) G^* của G. Xét một chu trình μ tùy ý trong G. Hiển nhiên, chu trình μ sẽ tạo thành một đường cong đóng đơn nào đó trong mặt phẳng biểu diễn của G và phân chia mặt phẳng thành hai vùng (Định lý Đường cong Jordan). Do đó các đỉnh của G^* được phân hoạch thành hai tập con khác trống rời nhau-một bên trong μ và một bên ngoài μ. Nói cách khác, tập các cạnh μ^* trong G^* tương ứng tập μ trong G là một thiết diện trong G^*. (Không tập con thực sự nào của μ^* là thiết diện của G^*; tại sao?). Tương tự, hiển nhiên rằng với mỗi thiết diện S^* trong G^* tồn tại duy nhất một chu trình gồm tập các cạnh tương ứng S trong G sao cho S là một chu trình.

Điều kiện đủ. Giả sử G là đồ thị phẳng và G' là đồ thị mà có tương ứng một-một giữa các thiết diện trong G và các chu trình trong G' và ngược lại. Giả sử G^* là đồ thị đối ngẫu của G. Khi đó có một tương ứng một-một giữa các chu trình trong G' và các thiết

diện trong G, và giữa các thiết diện trong G và các chu trình trong G^*. Do đó tồn tại tương ứng một một giữa các chu trình của G' và G^* mà điều này chỉ ra rằng G' và G^* là 2−đẳng cấu (Định lý 1.5.3). Theo Định lý 6.5.4, đồ thị G' là một đối ngẫu của G. $\triangleleft$

Đối ngẫu của đồ thị con

Giả sử G là đồ thị phẳng và G^* là đối ngẫu của nó. Giả sử a là một cạnh của G và cạnh tương ứng nó trong G^* là a^*. Bây giờ ta xóa cạnh a khỏi đồ thị G và tìm đối ngẫu của đồ thị $G \setminus \{a\}$. Nếu a là cạnh biên của hai diện, thì việc xóa cạnh a sẽ hợp nhất hai diện này làm một. Do đó đồ thị đối ngẫu $(G \setminus \{a\})^*$ có thể nhận được từ G^* bằng cách xóa cạnh tương ứng a^* và sau đó đồng nhất hai đỉnh mà a^* liên thuộc trong $G^* \setminus \{a^*\}$. Mặt khác nếu cạnh a không nằm trên biên, thì a^* là một khuyên. Trong trường hợp này $G^* \setminus \{a^*\}$ chính là $(G \setminus \{a\})^*$. Do đó nếu đồ thị G có một đối ngẫu G^* thì đối ngẫu của đồ thị con của G có thể nhận được bằng cách áp dụng một số lần thủ tục này.

Đối ngẫu của đồ thị đồng phôi

Giả sử G là đồ thị phẳng và G^* là đối ngẫu của nó. Giả sử a là một cạnh của G và cạnh tương ứng nó trong G^* là a^*. Bây giờ ta thêm một đỉnh mới có bậc bằng hai nằm trên cạnh a vào G (tức là a trở thành hai cạnh trong chuỗi). Khi đó đối ngẫu của đồ thị mới nhận được chính là thêm cạnh song song với a^* vào đồ thị G^*. Tương tự xử lý ngược lại bằng cách rút gọn chuỗi sẽ tương ứng với xóa một cạnh song song trong G^*. Do đó nếu G có một đối ngẫu G^*, thì mọi đối ngẫu của đồ thị đồng phôi với G^* có thể nhận được từ G^* bằng phương pháp trên.

Đến nay chúng ta mới chỉ nghiên cứu đối ngẫu của các đồ thị phẳng. Đó là vì mọi định nghĩa của khái niệm đối ngẫu phụ thuộc vào cách biểu diễn phẳng của đồ thị trên mặt phẳng. Tuy nhiên, Định lý 6.6.1 cho phép chúng ta có thể định nghĩa lại khái niệm này mà không phụ thuộc vào cách biểu diễn trên mặt phẳng; cụ thể là tương ứng giữa các chu trình và các thiết diện. Khái niệm này có thể mở rộng cho các đồ thị không phẳng? Nói cách khác, với đồ thị không phẳng G, có tồn tại một đồ thị G' tương ứng một-một giữa các cạnh của chúng sao cho mọi chu trình trong G tương ứng duy nhất với một thiết diện trong G', và ngược lại? Câu trả lời là *không* do kết quả sau

Định lý 6.6.2. [Whitney] *Đồ thị G có đối ngẫu nếu và chỉ nếu G phẳng.*

Chứng minh. Chúng ta chỉ cần chứng minh điều kiện cần. Tức là ta sẽ chứng minh đồ thị không phẳng sẽ không có đối ngẫu. Giả sử G là đồ thị không phẳng. Khi đó theo Định lý Kuratowski, G chứa K_5 hoặc $K_{3,3}$ hoặc một đồ thị con đồng phôi với một trong hai đồ thị này. Ta biết rằng, G có một đối ngẫu chỉ nếu mọi đồ thị con g của G và mọi đồ thị đồng phôi với g có một đối ngẫu. Do đó nếu chúng ta chứng minh được cả hai đồ thị Kuratowski K_5 và $K_{3,3}$ đều không có đối ngẫu thì định lý được chứng minh. Ta sẽ chứng minh phản chứng như sau:

(a) Giả sử $K_{3,3}$ có một đối ngẫu D. Nhận xét rằng các thiết diện trong $K_{3,3}$ tương ứng các chu trình trong D và ngược lại (Định lý 6.6.1). Do $K_{3,3}$ không có thiết diện gồm hai cạnh nên D không có chu trình độ dài hai. Tức D không có các cạnh song song. Do mọi chu trình trong $K_{3,3}$ có độ dài bốn hoặc sáu nên D không có thiết diện với số cạnh ít hơn bốn. Vì vậy, bậc của các đỉnh trong D ít nhất bằng bốn. Nhưng D không có cạnh song song và bậc của các đỉnh ít nhất bằng bốn nên D có ít nhất năm đỉnh, mỗi đỉnh có bậc lớn hơn hoặc bằng bốn. Tức là D có ít nhất $(5 \times 4)/2 = 10$ cạnh. Điều này mâu thuẫn vì $K_{3,3}$ có chín cạnh và đối ngẫu của nó cũng phải có chín cạnh. Do đó $K_{3,3}$ không có đối ngẫu.

(b) Giả sử K_5 có một đối ngẫu là H. Chú ý rằng K_5 có

1. 10 cạnh;
2. không có cạnh song song;
3. không có thiết diện với hai cạnh; và
4. các thiết diện chỉ có bốn hoặc sáu cạnh.

Suy ra đồ thị H có

1. 10 cạnh;
2. không có đỉnh bậc nhỏ hơn ba;
3. không có cặp cạnh song song;

4. các chu trình có độ dài bốn hoặc sáu.

Khi đó H chứa một lục giác (một chu trình có độ dài sáu) và không có hơn ba cạnh mà khi thêm vào lục giác sẽ không tạo thành chu trình độ dài ba hoặc một cặp cạnh song song. Nhưng điều này không thể xảy ra trong H và H có 10 cạnh, nên H có ít nhất bảy đỉnh. Bậc của mỗi đỉnh này ít nhất bằng ba. Điều này dẫn đến H có ít nhất 11 cạnh-mâu thuẫn. ◁

Chương 7

Mạng vận tải

7.1 Mở đầu

Một trong những bài toán lý thú và quan trọng của lý thuyết đồ thị là xác định giá trị lớn nhất của luồng được truyền từ một *đỉnh nguồn* s của đồ thị đến một *đỉnh đích* t. Trong khung cảnh đó, mỗi cung (v_i, v_j) của đồ thị G được gắn với một khả năng thông qua q_{ij} là số lượng luồng lớn nhất có thể tải qua cung này. Bài toán này và những cải biên của nó có rất nhiều ứng dụng trong thực tế, chẳng hạn, xác định mật độ giao thông lớn nhất giữa hai vùng trong bản đồ giao thông được biểu diễn bởi một đồ thị. Trong ví dụ này, lời giải của bài toán luồng lớn nhất cũng chỉ ra những nơi "bảo hòa" trên mạng giao thông và tạo một "tắc nghẽn" khi luồng tập trung vào giữa hai vị trí nào đó.

Phương pháp giải bài toán *luồng lớn nhất* từ s đến t đưa ra lần đầu tiên bởi Ford và Fulkerson [27] và kỹ thuật "gán nhãn" của họ là cơ sở cho những thuật toán khác giải quyết những vấn đề liên quan. Có một số cải biên của bài toán luồng lớn nhất:

1. Giả sử rằng mỗi cung của đồ thị không chỉ được gắn với khả năng q_{ij} cho biết cận trên của luồng trên cung (v_i, v_j) mà còn "khả năng" r_{ij} cho cận dưới của luồng trên cung này. Trong trường hợp như vậy, không phải lúc nào một tập chấp nhận được các giá trị của luồng cũng thỏa mãn cùng lúc hai ràng buộc này. Tuy

nhiên-nói chung-nhiều luồng thỏa điều kiện này; và nếu ngoài các khả năng còn có các chi phí c_{ij} tương ứng một đơn vị luồng dọc theo các cung thì bài toán trở thành *tìm luồng chấp nhận được với chi phí nhỏ nhất* từ s đến t.

2. Xét trường hợp đòi hỏi *luồng lớn nhất giữa mọi cặp đỉnh.* Mặc dù bài toán này có thể giải bằng $n(n-1)/2$ lần lặp các bài toán luồng lớn nhất từ s đến t nhưng cách làm này quá thô! Tương tự với tìm tất cả các đường đi ngắn nhất, ở đây cũng cần một thuật toán chuyên dụng để giải nó-và trong trường hợp đồ thị vô hướng, phương pháp giải quyết nó không liên quan đến lời giải của bài toán luồng lớn nhất giữa hai đỉnh s và t.

3. Nếu thay cho một đỉnh nguồn và một đỉnh đích, ta khảo sát một số nguồn và một số đích, với các hàng hóa khác nhau giữa hai tổ hợp nguồn-đích khác nhau, thì bài toán cực đại tổng số tất cả các luồng từ các nguồn đến các đích là *bài toán luồng đa-hàng hóa.* Trong bài toán này, khả năng q_{ij} của cung (v_i, v_j) là cận trên của tổng các luồng với các loại hàng hóa trên cung đó.

4. Giả thiết (ẩn) trong tất cả các trường hợp trên là số lượng luồng đến bằng số lượng luồng ra. Nếu ta bỏ giả thiết này và xét trường hợp luồng ra khỏi một cung bằng luồng đến nhân với một số nguyên không âm nào đó, thì bài toán luồng lớn nhất (từ s đến t) được xem như *bài toán trong các đồ thị với các lợi nhuận.* Trong dạng này, các luồng có thể "được sinh ra" hoặc "bị hấp thụ" bởi đồ thị, và do vậy luồng vào s và luồng ra khỏi t có thể thay đổi hoàn toàn độc lập.

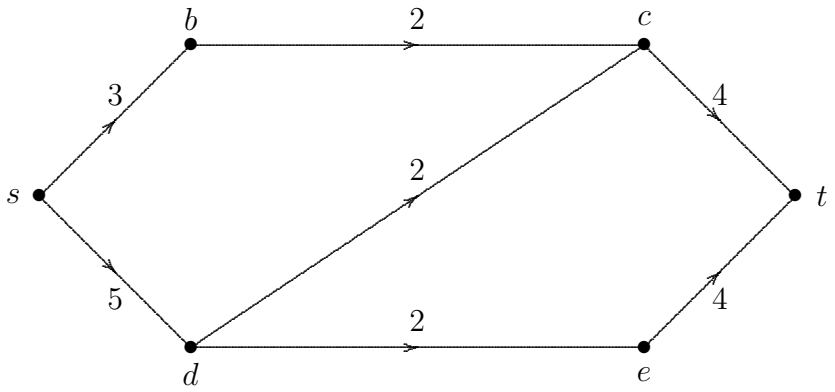
Các bài toán về luồng được nghiên cứu nhiều và có những ứng dụng thực tiễn. Mục đích của chương này trình bày ngắn gọn các bài toán thường gặp, chỉ ra mối liên hệ giữa chúng và xây dựng một số thuật toán giải quyết.

Chương này sẽ tìm hiểu bài toán luồng lớn nhất từ s đến t và các vấn đề liên quan. Do các thuật toán giải bài toán luồng đa-hàng hóa rất khác biệt về bản chất nên sẽ không được đề cập ở đây. Bạn đọc quan tâm có thể xem [30] và các tài liệu dẫn kèm theo.

7.2 Bài toán luồng lớn nhất

Luồng lớn nhất trên mạng vận tải G là một luồng với giá trị lớn nhất. Nói chung có một vài luồng với cùng giá trị lớn nhất. Phần này trình bày thuật toán tìm luồng lớn nhất. Ý tưởng của thuật toán là: khởi đầu với luồng nào đó và tăng giá trị của luồng cho đến khi không thể tăng được nữa. Kết quả ta có luồng lớn nhất.

Ví dụ 7.2.1. Đồ thị có hướng trong Hình 7.1 biểu diễn sơ đồ mạng dẫn dầu. Dầu được dẫn từ tàu s và được bơm thông qua mạng đến nhà máy lọc dầu t. Các đỉnh b, c, d và e là các trạm bơm trung gian. Các cung biểu thị các ống dẫn dầu và hướng di chuyển của hệ thống. Các nhãn trên các cung là khả năng thông qua tối đa của ống dẫn. Bài toán đặt ra là tìm cách chuyển nhiều nhất lượng dầu từ tàu đến nhà máy và tính giá trị này. Hình 7.1 là một ví dụ của một "mạng vận tải". Trước hết ta có:



Hình 7.1: Ví dụ về mạng vận tải.

Định nghĩa 7.2.2. *Mạng vận tải* là một đơn đồ thị có hướng, có trọng số G sao cho

(a) có một và chỉ một đỉnh s, gọi là *đỉnh vào (hoặc đỉnh nguồn) của mạng,* không có cung đến nó.

(b) có một và chỉ một đỉnh t, gọi là *đỉnh ra (hoặc đỉnh đích) của mạng,* không có cung đi ra khỏi nó.

(c) mỗi cung $u := (v_i, v_j) \in E$ được gán một số nguyên không âm q_{ij}, gọi là *khả năng* của cung u.

(d) đồ thị vô hướng nhận được từ G bằng cách bỏ đi các hướng là liên thông.

Ví dụ 7.2.3. Đồ thị có hướng trong Hình 7.1 là mạng vận tải. Lối vào là đỉnh s và lối ra là đỉnh t. Khả năng cung (s, b) là $q_{sb} = 3$ và cung (b, c) là $q_{bc} = 2$.

Trong chương này, nếu G là mạng vận tải thì chúng ta sẽ ký hiệu đỉnh vào là s và đỉnh ra là t.

Định nghĩa 7.2.4. Giả sử G là mạng vận tải với khả năng cung (v_i, v_j) là q_{ij}. *Luồng (chấp nhận được)* F trong G gán mỗi cung (v_i, v_j) một số không âm f_{ij} sao cho

(a) $0 \leq f_{ij} \leq q_{ij}$; và

(b) với mỗi đỉnh $v_j \neq s, t$, ta có

$$\sum_i f_{ij} = \sum_i f_{ji}. \tag{7.1}$$

(Tổng trong (7.1) lấy trên tất cả các giá trị i; nếu không có cung (v_i, v_j), ta đặt $f_{ij} = 0$).

Ta nói f_{ij} là *luồng trên cung* (v_i, v_j). Với mỗi j, ta gọi

$$\sum_i f_{ij}$$

là *luồng đến đỉnh* v_j và

$$\sum_i f_{ji}$$

là *luồng ra khỏi đỉnh* v_j.

Điều kiện (7.1) gọi là *bảo toàn luồng*. Trong ví dụ dẫn dầu của Hình 7.1, bảo toàn luồng có nghĩa dầu không được sử dụng và cũng không được cấp thêm tại các trạm bơm b, c, d và e.

Ví dụ 7.2.5. Các phép gán

$$f_{sb} = 2, f_{bc} = 2, f_{cz} = 3, f_{sd} = 3,$$

$$f_{dc} = 1, f_{de} = 2, f_{ez} = 2,$$

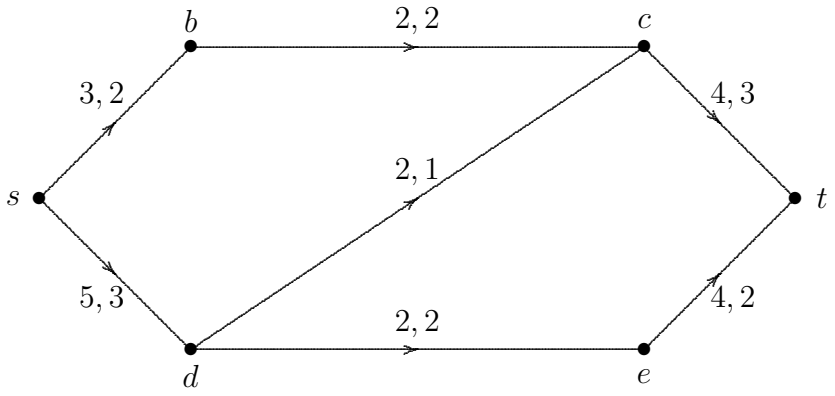
xác định luồng trên mạng vận tải của Hình 7.1. Chẳng hạn, luồng đến đỉnh d là

$$f_{sd} = 3$$

và bằng luồng ra khỏi đỉnh d :

$$f_{dc} + f_{de} = 1 + 2 = 3.$$

Luồng F được vẽ trong Hình 7.2, trong đó cung e được gán nhãn x, y nếu khả năng của e là x và luồng trên e là y.



Hình 7.2: Luồng trên mạng vận tải của Hình 7.1.

Chú ý rằng trong ví dụ này, luồng ra khỏi đỉnh s là

$$f_{sb} + f_{sd}$$

bằng luồng đến đỉnh t :

$$f_{ct} + f_{et}$$

và bằng 5. Thật vậy, ta có

Định lý 7.2.6. *Giả sử F là luồng trên mạng vận tải $G := (V, E)$. Khi đó luồng ra khỏi đỉnh s bằng luồng đến đỉnh t; tức là*

$$\sum_i f_{si} = \sum_i f_{it}.$$

Chứng minh. Ta có

$$\sum_{j \in V} \left(\sum_{i \in V} f_{ij} \right) = \sum_{j \in V} \left(\sum_{i \in V} f_{ji} \right)$$

do mỗi vế bằng $\sum_{e\in E} f_e$. Vì vậy

$$\begin{aligned} 0 &= \sum_{j\in V}\left(\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji}\right) \\ &= \left(\sum_{i\in V} f_{it} - \sum_{i\in V} f_{ti}\right) + \left(\sum_{i\in V} f_{is} - \sum_{i\in V} f_{si}\right) + \sum_{j\in V, j\neq s,t}\left(\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji}\right) \\ &= \sum_{i\in V} f_{it} - \sum_{i\in V} f_{si} \end{aligned}$$

do $f_{ti} = 0 = f_{is}$ với mọi $v_i \in V$, và (7.1). ◁

Định nghĩa 7.2.7. Giả sử F là luồng trên mạng vận tải G. Đại lượng

$$\sum_i f_{si} = \sum_i f_{it}$$

gọi là *giá trị của luồng F*.

Bài toán trên mạng vận tải G có thể phát biểu:

Bài toán 7.2.8. *Tìm một luồng chấp nhận được lớn nhất trên đồ thị G; tức là trong số tất cả các luồng trên G, tìm luồng F có giá trị lớn nhất.*

Thuật toán gán nhãn của Ford và Fulkerson [27] giải bài toán này dựa trên Định lý 7.2.10. Trước hết ta có một số khái niệm

Định nghĩa 7.2.9. Nếu các đỉnh của đồ thị $G = (V, E)$ được phân hoạch thành hai tập con V_0 và $\tilde{V}_0$ (trong đó $V_0 \subset V$ và $\tilde{V}_0$ là phần bù của V_0 trong V), thì tập các cung của G với đỉnh xuất phát thuộc V_0 và đỉnh kết thúc thuộc $\tilde{V}_0$ gọi là *thiết diện* của G. Tập các cung của thiết diện thường được ký hiệu bởi $(V_0 \to \tilde{V}_0)$.

Giả sử G là mạng vận tải. Thiết diện $(V_0 \to \tilde{V}_0)$ tách s khỏi t nếu $s \in V_0$ và $t \in \tilde{V}_0$. *Khả năng thông qua* hay *giá trị* của thiết diện là tổng tất cả các khả năng của tất cả các cung của G với đỉnh xuất phát thuộc V_0 và đỉnh kết thúc thuộc $\tilde{V}_0$; tức là

$$v(V_0 \to \tilde{V}_0) := \sum_{(v_i,v_j)\in(V_0\to\tilde{V}_0)} q_{ij}.$$

Thiết diện nhỏ nhất là thiết diện có khả năng thông qua nhỏ nhất.

Định lý 7.2.10. (Định lý thiết diện nhỏ nhất-luồng lớn nhất) *Giá trị của luồng lớn nhất từ s đến t bằng khả năng thông qua của thiết diện nhỏ nhất* $(V_m \to \tilde{V}_m)$ *tách s khỏi t.*

Chứng minh. Hiển nhiên rằng, luồng lớn nhất từ s đến t không thể lớn hơn $v(V_m \to \tilde{V}_m)$ do tất cả các đường đi từ s đến t đều sử dụng ít nhất một cung của thiết diện này. Do đó chỉ cần chứng minh rằng tồn tại một luồng đạt giá trị này. Ta xem luồng đã cho F tương ứng với một vector m chiều và định nghĩa thiết diện $(V_0 \to \tilde{V}_0)$ bằng đệ quy theo thủ tục sau:

1. Khởi tạo, đặt $V_0 = \{s\}$.
2. Nếu $v_i \in V_0$, và hoặc $f_{ij} < q_{ij}$ hoặc $f_{ij} > 0$ thì đặt v_j vào trong tập V_0.
3. Lặp lại Bước 2 cho đến khi không thể thêm đỉnh nào vào V_0.

Có hai trường hợp xảy ra: hoặc $t \in V_0$ hoặc $t \notin V_0$.

Trường hợp 1. $t \in V_0$.

Theo Bước 2 trên, tồn tại một dây chuyền từ s đến t sao cho mọi cung (v_i, v_j) được sử dụng bởi dây chuyền theo hướng thuận (các cung định hướng thuận) thỏa $f_{ij} < q_{ij}$ và các cung (v_k, v_l) được định hướng ngược, tức là hướng từ v_l đến v_k thỏa $f_{lk} > 0$. Dây chuyền này gọi là *dây chuyền điều chỉnh.*

Đặt

$$\begin{array}{rcll} \delta_f & := & \min_{(v_i,v_j)}[q_{ij} - f_{ij}] & \text{nếu } (v_i, v_j) \text{ thuận hướng,} \\ \delta_b & := & \min_{(v_k,v_i)}[f_{kl}] & \text{nếu } (v_k, v_l) \text{ ngược hướng,} \end{array}$$

và

$$\delta := \min[\delta_f, \delta_b].$$

Nếu ta cộng thêm δ vào luồng trên tất cả các cung định hướng thuận và trừ đi δ trên tất cả các cung định hướng ngược trong dây chuyền điều chỉnh thì luồng thu được vẫn là luồng chấp nhận được có giá trị nhỏ hơn luồng ban đầu một lượng δ. Điều này là hiển nhiên vì thêm một lượng δ vào các cung theo chiều thuận không vượt quá khả năng của các cung này (do $\delta < \delta_f$) và trừ đi một lượng δ vào các cung theo chiều ngược thì luồng vẫn không âm (do $\delta < \delta_b$).

Áp dụng lại với luồng mới theo các Bước 1-3 trên ta lại thu được một thiết diện mới $(V_0 \to \tilde{V}_0)$.

Trường hợp 2. $t \notin V_0$.

Theo Bước 2, $f_{ij} = q_{ij}$ với mọi cung $(v_i, v_j) \in (V_0 \to \tilde{V}_0)$ và $f_{kl} = 0$ với mọi cung $(v_k, v_l) \in (\tilde{V}_0 \to V_0)$.

Do đó

$$\sum_{(v_i,v_j)\in(V_0\to\tilde{V}_0)} f_{ij} = \sum_{(v_i,v_j)\in(V_0\to\tilde{V}_0)} q_{ij}$$

và

$$\sum_{(v_k,v_l)\in(\tilde{V}_0\to V_0)} f_{kl} = 0;$$

tức là giá trị của luồng bằng

$$\sum_{(v_i,v_j)\in(V_0\to\tilde{V}_0)} f_{ij} - \sum_{(v_k,v_l)\in(\tilde{V}_0\to V_0)} f_{kl}$$

và bằng khả năng thông qua của thiết diện $(V_0 \to \tilde{V}_0)$.

Do Trường hợp 1, luồng tăng ít nhất một đơn vị, nên nếu giả thiết tất cả các khả năng q_{ij} là những số nguyên thì luồng lớn nhất có thể nhận được sau một số hữu hạn bước khi Trường hợp 2 xảy ra. Luồng này có giá trị bằng khả năng thông qua của thiết diện hiện hành $(V_0 \to \tilde{V}_0)$ nên là luồng lớn nhất và thiết diện tương ứng có khả năng thông qua nhỏ nhất. ◁

Phương pháp xây dựng được sử dụng để chứng minh định lý trên cho chúng ta thuật toán để tìm luồng lớn nhất từ s đến t. Thuật toán này sẽ được trình bày dưới đây.

Xuất phát từ luồng chấp nhận được tùy ý (có thể sử dụng luồng có giá trị bằng không) và sau đó tăng luồng bằng cách tìm các dây chuyền điều chỉnh luồng từ s đến t. Việc tìm một dây chuyền điều chỉnh luồng được thực hiện bằng cách gán nhãn. Khi tìm được một dây chuyền điều chỉnh luồng, ta sẽ tăng giá trị của luồng. Sau đó xóa

tất cả các nhãn và luồng mới được sử dụng để gán nhãn lại. Nếu không tồn tại dây chuyền điều chỉnh luồng thì thuật toán kết thúc, luồng chấp nhận được là lớn nhất. Thuật toán cụ thể như sau:

7.2.1 Thuật toán tìm luồng lớn nhất

A. Quá trình gán nhãn

Mỗi đỉnh chỉ có thể có một trong ba khả năng:

1. được gán nhãn và được kiểm tra (tức là nó được gán nhãn và tất cả các đỉnh liên thuộc với nó đã được xử lý); hoặc
2. được gán nhãn và chưa được kiểm tra (tức là nó được gán nhãn và tồn tại đỉnh liên thuộc với nó chưa được xử lý); hoặc
3. chưa được gán nhãn.

Nhãn của đỉnh v_i gồm hai thành phần và có một trong hai dạng hoặc $(+v_j, \delta)$ hoặc $(-v_j, \delta)$. Trong trường hợp đầu, có thể tăng luồng dọc theo cung (v_i, v_j); trong trường hợp thứ hai, có thể giảm luồng dọc theo cung (v_i, v_j). Đại lượng δ trong cả hai trường hợp là lượng hàng nhiều nhất có thể thêm hoặc bớt giá trị của luồng trên các cung thuộc *dây chuyền điều chỉnh* (trong quá trình xây dựng) từ s đến v_i. Nhãn của đỉnh v_i cho phép xác định dây chuyền điều chỉnh luồng từ s đến v_i.

Khởi tạo tất cả các đỉnh chưa được gán nhãn và $f_{ij} = 0$ với mọi cung $(v_i, v_j) \in E$.

1. Gán nhãn của đỉnh s là $(+s, \delta(s) = \infty)$. Đỉnh s được gán nhãn và chưa được kiểm tra; tất cả các đỉnh khác chưa được gán nhãn.
2. Chọn đỉnh $v_i \in V$ đã được gán nhãn và chưa được kiểm tra. Nếu không tồn tại, thuật toán dừng, luồng $F = (f_{ij})$ là lớn nhất. Ngược lại, giả sử $(\pm v_k, \delta(v_i))$ là nhãn của đỉnh v_i.

- Gán nhãn $(+v_i, \delta(v_j))$ cho tất cả các đỉnh $v_j \in \Gamma(v_i)$ chưa được gán nhãn sao cho $f_{ij} < q_{ij}$, trong đó

$$\delta(v_j) := \min\{\delta(v_i), q_{ij} - f_{ij}\}.$$

- Gán nhãn $(-v_i, \delta(v_j))$ cho tất cả các đỉnh $v_j \in \Gamma^{-1}(v_i)$ chưa được gán nhãn sao cho $f_{ji} > 0$, trong đó

$$\delta(v_j) := \min\{\delta(v_i), f_{ji}\}.$$

(Đỉnh v_i đã được gán nhãn và đã được kiểm tra; các đỉnh v_j xác định trên đã được gán nhãn và chưa được kiểm tra).

3. Nếu đỉnh t được gán nhãn, chuyển sang Bước 4; ngược lại chuyển sang Bước 2. Cần chú ý rằng, nếu V_0 là tập các đỉnh đã được gán nhãn và $\tilde{V}_0$ là tập các đỉnh chưa được gán nhãn thì $(V_0 \to \tilde{V}_0)$ là thiết diện nhỏ nhất.

B. Quá trình tăng luồng

4. Đặt $c = t$ và chuyển sang Bước 5.

   - Nếu nhãn của đỉnh c có dạng $(+z, \delta(c))$ thì thay luồng trên cung (z, c) là f_{zc} bởi $f_{zc} + \delta(t)$;
   - Nếu nhãn của đỉnh c có dạng $(-z, \delta(c))$ thì thay luồng trên cung (x, z) là f_{cz} bởi $f_{cz} - \delta(t)$;

5. Nếu $z = s$ thì xóa tất cả các nhãn từ các đỉnh và chuyển sang Bước 1; ngược lại (tức là $z \neq c$) đặt $c = z$ và trở lại Bước 5.

7.2.2 Đồ thị điều chỉnh luồng

Quá trình tìm một dây chuyền để tăng giá trị của luồng F trong đồ thị $G = (V, E)$ có thể đưa về tìm một đường đi từ s đến t trên đồ thị điều chỉnh luồng $G^\mu(F) = (V^\mu, E^\mu)$, $V^\mu = V, E^\mu = E_1^\mu \cup E_2^\mu$, trong đó

$$E_1^\mu := \{(v_i^\mu, v_j^\mu) \mid f_{ij} < q_{ij}, (v_i, v_j) \in E\},$$

với khả năng của mỗi cung $(v_i^\mu, v_j^\mu) \in E_1^\mu$ là $q_{ij}^\mu = q_{ij} - f_{ij}$, và

$$E_2^\mu := \{(v_j^\mu, v_i^\mu) \mid f_{ij} > 0, (v_i, v_j) \in E\}$$

với khả năng của mỗi cung $(v_j^\mu, v_i^\mu) \in E_2^\mu$ là $q_{ji}^\mu = f_{ij}$.

Khi đó thủ tục gán nhãn của thuật toán tìm luồng lớn nhất trong Phần 7.2.1 chính là thuật toán xác định tập phạm vi $R(s)$ trong đồ thị điều chỉnh luồng $G^\mu(F)$. Nếu $t \in R(s)$ (tức là nếu đỉnh t được gán nhãn) thì có thể xác định một đường đi từ s đến t trong đồ thị $G^\mu(F)$. Trong trường hợp này, dây chuyền điều chỉnh luồng của G là đường đi P mà các cung của P thuộc E_1^μ tương ứng cung định hướng thuận và các cung của P thuộc E_2^μ được định hướng ngược.

7.2.3 Phân tích luồng

Trong một số trường hợp ta muốn phân tích một luồng phức tạp thành tổng của những luồng đơn giản hơn. Điều này không những có ý nghĩa thực tiễn mà còn góp phần hiểu tốt hơn bản chất của luồng trên mạng vận tải và phục vụ một số thuật toán về luồng.

Ký hiệu $h \circ (S)$ là luồng trong đồ thị G mà các cung $(v_i, v_j) \in S$ có $f_{ij} = h$ và các cung $(v_i, v_j) \notin S$ có $f_{ij} = 0$. Chú ý rằng $h \circ (S)$ không nhất thiết là một luồng với tập S tùy ý. Hiển nhiên rằng $h \circ (S)$ là một luồng thì tập S các cung hoặc tạo thành một đường đi từ s đến t hoặc là một mạch trong G.

Với hai luồng F và H ta ký hiệu $F + H$ là luồng mà luồng trên cung (v_i, v_j) là $f_{ij} + h_{ij}$.

Định lý 7.2.11. *Nếu F là luồng từ s đến t có giá trị nguyên v trong G thì F có thể phân tích thành*

$$F = 1 \circ (P_1) + 1 \circ (P_2) + \cdots + 1 \circ (P_v) + 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \cdots + 1 \circ (\Phi_\kappa),$$

trong đó $P_1, P_2, \ldots, P_v$ là các đường đi sơ cấp từ s đến t và $\Phi_1, \Phi_2, \ldots, \Phi_\kappa$ là các mạch sơ cấp của G. (P_i và Φ_i không nhất thiết phân biệt).

Chứng minh. Từ $G = (V, E)$ với luồng F cho trước ta xây dựng đồ thị unitary $G^e = (V^e, E^e)$ như sau: Tập V^e các đỉnh của G^e chính là tập V các đỉnh của G. Nếu f_{ij} là

luồng trên cung (v_i, v_j) của G thì ta thay bằng f_{ij} cung song song giữa các đỉnh tương ứng v_i^e và v_j^e của G^e. Nếu $f_{ij} = 0$ thì không có cung nào được đặt giữa v_i^e và v_j^e. Ta được G^e là đa đồ thị trong đó mỗi cung của nó tương ứng với một đơn vị luồng trên cung tương ứng trong G; nói cách khác, G^e biểu diễn luồng F trong G.

Từ điều kiện về luồng F suy ra các đỉnh của đồ thị G^e cần thỏa mãn

$$\begin{array}{rcll} d^+(v_i^e) & = & d^-(v_i^e), & \text{với mọi } v_i^e \neq s^e \text{ hoặc } t^e, \\ d^+(s^e) & = & d^-(^e) \quad = & v. \end{array}$$

Suy ra nếu ta trả lại v cung được thêm vào G^e từ đỉnh t^e đến đỉnh s^e thì đồ thị G^e sẽ có một mạch Euler (xem Phần 5.1). Loại bỏ v cung này khỏi mạch Euler, ta được v đường đi từ s^e đến t^e qua mỗi cung của G^e đúng một lần. Ký hiệu các đường đi này là $P_1', P_2', \ldots, P_v'$. Các đường đi P_i' không nhất thiết sơ cấp (mặc dù chúng là đơn giản). Tuy nhiên, mỗi đường đi không sơ cấp có thể xem như tổng của một đường đi sơ cấp từ s^e đến t^e và một số các mạch sơ cấp rời nhau. Do vậy,

$$F = 1 \circ (P_1) + 1 \circ (P_2) + \cdots + 1 \circ (P_v) + 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \cdots + 1 \circ (\Phi_\kappa),$$

trong đó P_i là các đường đi sơ cấp từ s^e đến t^e và Φ_i là các mạch sơ cấp. ◁

Nói chung, các đường đi và các chu trình có thể trùng nhau. Nếu chỉ có v' đường đi và κ' mạch đầu tiên khác nhau, với đường đi P_i xuất hiện h_i lần trong danh sách $P_1, P_2, \ldots, P_v$ và mạch Φ_i xuất hiện l_i lần trong danh sách $\Phi_1, \Phi_2, \ldots, \Phi_\kappa$ thì F có thể viết dưới dạng

$$F = \sum_{i=1}^{v'} h_i \circ (P_i) + \sum_{i=1}^{\kappa'} l_i \circ (\Phi_i).$$

Nói chung hai luồng F và H là thích ứng nếu $f_{ij}.h_{ij} = 0$ với mọi cung (v_i, v_j).

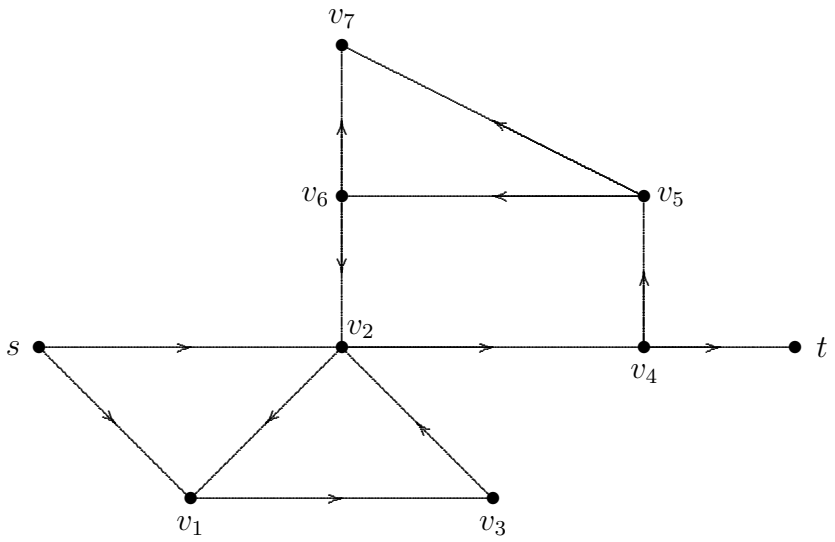
Ví dụ 7.2.12. Luồng F trong Hình 7.3 được phân tích thành các đường đi (từ s đến t) và các mạch sơ cấp:

$$F = 2 \circ P_1 + 1 \circ P_2 + 1 \circ \Phi_1 + 1 \circ \Phi_2 + 1 \circ \Phi_3,$$

trong đó

$$P_1 \quad = \quad \{s, v_2, v_4, t\},$$

$$
\begin{aligned}
P_2 &= \{s, v_1, v_3, v_2, v_4, t\},\\
\Phi_1 &= \{v_1, v_3, v_2, v_1\},\\
\Phi_2 &= \{v_2, v_4, v_5, v_6, v_2\},\\
\Phi_3 &= \{v_5, v_6, v_7, v_5\}.
\end{aligned}
$$



Hình 7.3: Luồng F.

7.3 Những cải biên đơn giản của bài toán luồng lớn nhất

Phần này chúng ta nêu một số kết quả nhận được từ bài toán luồng lớn nhất.

7.3.1 Đồ thị có nhiều nguồn và nhiều đích

Xét đồ thị với n_s đỉnh vào và n_t đỉnh ra và giả sử luồng có thể chuyển từ nguồn đến đích tuỳ ý. Bài toán tìm luồng lớn nhất từ tất cả các nguồn đến tất cả các đích có thể đưa về bài toán luồng lớn nhất bằng cách thêm một đỉnh nguồn nhân tạo s và một

đỉnh ra nhân tạo t với các cung được thêm từ s đến các đỉnh vào ban đầu và từ các đỉnh ra thực tế đến đỉnh t. Khả năng của các cung thêm vào từ s đến các nguồn có thể đặt bằng vô cùng, hoặc trong trường hợp lượng hàng cung cấp tại một nguồn s_k tối đa là a_k thì khả năng của cung (s, s_k) có thể đặt bằng giá trị này. Tương tự, khả năng của các cung dẫn tới đỉnh ra t có thể đặt bằng nhu cầu tại các đỉnh ra t_k hoặc bằng vô hạn nếu có nhu cầu là vô hạn.

Nếu bài toán đặt ra ở đó có đỉnh ra chỉ được cung cấp bởi những nguồn nào đó và ngược lại, thì bài toán không phải là cải biên đơn giản của bài toán luồng lớn nhất từ s đến t mà có thể đưa về bài toán đa luồng như đã đề cập trong phần mở đầu.

7.3.2 Đồ thị với ràng buộc tại các cung và đỉnh

Nếu ngoài khả năng q_{ij} của các cung, ta thêm khả năng của các đỉnh $w_j, j = 1, 2, \ldots, n$, sao cho tổng số lượng hàng đi đến đỉnh v_j nhỏ hơn w_j, tức là

$$\sum_{v_i \in \Gamma^{-1}(v_j)} f_{ij} \leq w_{ij}$$

với mọi v_j.

Ta cần tìm luồng lớn nhất từ s đến t với giả thiết thêm tại các đỉnh. Xét đồ thị G_0 sao cho mọi đỉnh v_j của đồ thị G tương ứng hai đỉnh v_j^+ và v_j^- trong đồ thị G_0 sao cho mọi cung (v_i, v_j) của G đến đỉnh v_j tương ứng một cung (v_i^-, v_j^+) đến đỉnh v_j^+, và mọi cung (v_j, v_k) của G xuất phát từ v_j tương ứng một cung (v_j^-, v_k^+) của G_0 xuất phát từ v_j^-. Ngoài ra ta thêm một cung giữa v_j^+ và v_j^- với khả năng thông qua w_j, tức là bằng khả năng của đỉnh v_j.

Vì tổng số lượng hàng đến đỉnh v_j^+ phải được chuyển dọc theo cung (v_j^+, v_j^-) với khả năng thông qua w_j nên luồng lớn nhất trong đồ thị G với ràng buộc tại các đỉnh bằng luồng lớn nhất trong đồ thị G_0 với ràng buộc chỉ tại các cung. Cần chú ý rằng nếu thiết diện nhỏ nhất của G_0 không chứa các cung dạng (v_j^+, v_j^-) thì ràng buộc tại đỉnh v_j tng ng trong G không "tích cực" và trở thành vô dụng; nếu ngược lại, thiết diện nhỏ nhất của G_0 chứa các cung loại này thì các đỉnh tương ứng của G là bảo hoà bởi luồng lớn nhất.

7.3.3 Đồ thị có cận trên và cận dưới về luồng

Xét đồ thị G trong đó mỗi cung (v_i, v_j) ngoài cận trên q_{ij} còn có cận dưới về luồng là r_{ij}. Bài toán đặt ra là *có tồn tại một luồng chấp nhận giữa s và t sao cho $r_{ij} \leq f_{ij} \leq q_{ij}$ với mọi cung (v_i, v_j)*?

Từ G, ta thêm một đỉnh vào nhân tạo s_a và đỉnh ra nhân tạo t_a để nhận được G_a. Mỗi cung (v_i, v_j) mà $r_{ij} \neq 0$ ta thêm một cung (s_a, v_j) với khả năng r_{ij} và cận dưới bằng không và thêm cung thứ hai (v_i, t_a) với khả năng r_{ij} và cận dưới bằng không. Thay q_{ij} bởi $q_{ij} - r_{ij}$ và r_{ij} bởi 0. Ngoài ra thêm cung (t, s) với $q_{ts} = \infty, r_{ts} = 0$.

Bây giờ ta tìm luồng lớn nhất từ s_a đến t_a trong đồ thị G_a. Nếu giá trị của luồng lớn nhất bằng $\sum_{r_{ij} \neq 0} r_{ij}$ (tức là, nếu tất cả các cung đi ra từ s_a và tất cả các cung đi đến t_a bảo hoà) và ký hiệu lượng hàng trên cung (t, s) là f_{ts} thì tồn tại luồng chấp nhận được với giá trị f_{ts} trong đồ thị ban đầu. Thật vậy, nếu ta trừ r_{ij} lượng hàng trên các cung (v_i, t_a) và (s_a, v_j) và cộng thêm r_{ij} vào lượng hàng trên cung (v_i, v_j) thì tổng lượng hàng từ s_a đến t_a giảm một lượng là r_{ij}, luồng trên các cung (v_i, t_a) và (s_a, v_j) giảm xuống không, còn luồng trên cung (v_i, v_j) là $f_{ij} \in [r_{ij}, q_{ij}]$. (Giá trị cuối của f_{ij} bằng r_{ij} nếu giá trị ban đầu của f_{ij} tương ứng luồng lớn nhất bằng không, và giá trị cuối của f_{ij} bằng q_{ij} nếu giá trị ban đầu của f_{ij} bằng $q_{ij} - r_{ij}$). Bước trừ luồng lớn nhất ngược với bước điều chỉnh luồng trong thuật toán tìm luồng lớn nhất. Vì ta giả thiết giá trị của luồng lớn nhất bằng $\sum_{r_{ij} \neq 0} r_{ij}$ nên cuối cùng, tiến trình trừ luồng sẽ cho luồng từ s_a đến t_a có giá trị bằng không (do đó sẽ khiến hai đỉnh nhân tạo và các cung liên thuộc chúng trở thành vô dụng), và luồng trên tất cả các cung với $r_{ij} \neq 0$ sẽ thay đổi trong phạm vi $[r_{ij}, q_{ij}]$. Kết quả là ta có một luồng "lưu thông" trong đồ thị với giá trị bằng f_{ts}.

Mặt khác ta có

Định lý 7.3.1. *Nếu giá trị của luồng lớn nhất từ s_a đến t_a trong đồ thị G_a khác $\sum_{r_{ij} \neq 0} r_{ij}$ thì không tồn tại luồng chấp nhận được trong G.*

Chứng minh. Bài tập. ◁

7.4 Luồng với chi phí nhỏ nhất

Trong Phần 7.2 chúng ta đã xét bài toán tìm luồng lớn nhất từ s đến t mà không đề cập đến chi phí được gắn trên mỗi cung. Phần này khảo sát bài toán tìm luồng với giá trị v cho trước từ s đến t sao cho chi phí của luồng là nhỏ nhất. Cụ thể là:

Bài toán 7.4.1. *Cho mạng vận tải $G := (V, E)$ với đỉnh vào s, đỉnh ra t, khả năng thông qua của cung $(i, j) \in E$ là q_{ij}. Giả sử c_{ij} là chi phí vận chuyển một đơn vị hàng trên cung $(i, j) \in E$. Tìm luồng $F := (f_{ij})$ có giá trị v trên G với chi phí nhỏ nhất; tức là giải bài toán*

$$\sum_{(v_i,v_j)\in E} f_{ij}c_{ij} \to \min$$

với điều kiện

$$\begin{cases} \sum_{(v_i,v_j)\in E} f_{ij} & = v, \\ 0 \le \quad f_{ij} & \le q_{ij}. \end{cases}$$

Hiển nhiên, bài toán không có lời giải nếu v lớn hơn giá trị lớn nhất của luồng từ s đến t. Tuy nhiên, nếu v nhỏ hơn hoặc bằng giá trị này thì sẽ có một số luồng có giá trị v và bài toán có lời giải. Ford và Fulkerson [27] đã xây dựng một thuật toán "không có thứ tự" để tìm luồng với chi phí nhỏ nhất. Thuật toán trình bày dưới đây dựa theo những kết quả của Klein [38], Busacker và Gowen [10]. Thuật toán này đơn giản hơn phương pháp của Ford-Fulkerson và sử dụng những kỹ thuật đã giới thiệu trước.

7.4.1 Thuật toán Klein-Busacker-Gowen

Thuật toán này dựa vào việc xác định mạch có độ dài âm. Chúng ta hãy giả thiết rằng tồn tại luồng chấp nhận được F với giá trị v và F đã được xác định. Luồng này có thể nhận được bằng cách áp dụng thuật toán tìm luồng lớn nhất và chúng ta tăng luồng cho đến khi nhận được luồng có giá trị v.

Với luồng chấp nhận này, ta định nghĩa đồ thị $G^\mu(F) := (V^\mu, E^\mu)$ như đã giải thích trong Phần 7.2 và có chi phí trên các cung như sau:

- với mỗi cung $(v_i^\mu, v_j^\mu) \in E_1^\mu$, đặt $c_{ij}^\mu := c_{ij}$.

- với mỗi cung $(v_j^\mu, v_i^\mu) \in E_2^\mu$, đặt $c_{ji}^\mu := -c_{ij}$.

Thuật toán dựa trên định lý sau:

Định lý 7.4.2. *F là luồng giá trị v với chi phí nhỏ nhất nếu và chỉ nếu không tồn tại mạch Φ trong $G^\mu(F)$ sao cho tổng của các chi phí của các cung thuộc Φ âm.*

Chứng minh. Đặt $c[F]$ là chi phí của luồng F trong đồ thị G và $c[\Phi|G^\mu(F)]$ là tổng của các chi phí của các cung thuộc mạch Φ tương ứng với đồ thị $G^\mu(F)$.

Điều kiện cần. Giả sử $c[\Phi|G^\mu(F)] < 0$ với mạch Φ nào đó trong $G^\mu(F)$. Thêm một đơn vị vào luồng trên mỗi cung thuộc mạch Φ sẽ tạo ra một luồng mới chấp nhận được $F + 1 \circ (\Phi)$ có giá trị v. Chi phí của luồng $F + 1 \circ (\Phi)$ bằng $c[F] + c[\Phi|G^\mu(F)] < c[F]$-mâu thuẫn với giả thiết F là luồng với giá trị v và có chi phí nhỏ nhất.

Điều kiện đủ. Giả sử $c[\Phi|G^\mu(F)] \geq 0$ với mọi mạch Φ trong $G^\mu(F)$ và $F^*(\neq F)$ là luồng giá trị v có chi phí nhỏ nhất.

Ta ký hiệu $F^* - F$ là luồng mà giá trị trên cung (v_i, v_j) bằng $f_{ij}^* - f_{ij}$.

Vì F^* và F có thể phân tích thành tổng của các luồng dọc theo (từ s đến t) các đường đi trong G, theo cách xây dựng của đồ thị G^e trong Phần 7.2.3 đối với luồng $F^* - F$, suy ra với mọi đỉnh $v_i \in V$ ta có

$$d_{G^*}^+(v_i) = d_{G^*}^-(v_i).$$

Do đó theo Phần 7.2.3, dễ dàng kiểm tra rằng

$$F^* - F = 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \cdots + 1 \circ (\Phi_\kappa).$$

Hơn nữa, luồng $F^* = F + 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \cdots + 1 \circ (\Phi_\kappa)$ là chấp nhận được nên tổng $F + 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \cdots + 1 \circ (\Phi_l)$ chấp nhận được với mọi $1 \leq l \leq \kappa$. Do đó với luồng F ta có

$$\begin{aligned} c[F + 1 \circ (\Phi_1)] &= c[F] + c[\Phi_1|G^\mu(F)] \\ &\geq c[F]. \end{aligned}$$

Mặt khác

$$c[\Phi_l|G^\mu(F + 1 \circ (\Phi_1))] \geq c[\Phi_l|G^\mu(F)]$$

với mọi $l = 1, 2, \ldots, k$.

Vậy chi phí của luồng $F + 1 \circ (\Phi_1) + 1 \circ (\Phi_2)$ là

$$\begin{aligned} c[F + 1 \circ (\Phi_1) + 1 \circ (\Phi_2)] &= c[F + 1 \circ (\Phi_1)] + c[\Phi_2 | G^\mu(F + 1 \circ (\Phi_1))] \\ &\geq c[F + 1 \circ (\Phi_1)] + c[\Phi_2 | G^\mu(F)] \\ &\geq c[F + 1 \circ (\Phi_1)] \\ &\geq c[F]. \end{aligned}$$

Tiếp tục quá trình trên ta được $c[F^*] \geq c[F]$. Điều này mâu thuẫn với giả thiết F^* là luồng có chi phí nhỏ nhất. ◁

Do đó theo Định lý 7.4.2, để tìm luồng chấp nhận được có giá trị v với chi phí nhỏ nhất ta bắt đầu từ luồng chấp nhận được có giá trị v, thiết lập đồ thị $G^\mu(F)$ và kiểm tra có tồn tại mạch có độ dài âm không. Nếu không có thì luồng nhận được có chi phí nhỏ nhất. Ngược lại nếu Φ là mạch có độ dài âm thì ta thêm δ vào luồng trên mạch này. Khi đó luồng mới vẫn có giá trị v và có chi phí giảm một lượng $\delta.c(\Phi)$, trong đó $c(\Phi)$ là chi phí của mạch có dồ dài âm Φ. Hiển nhiên δ cần được chọn sao cho các khả năng thông qua của các cung trong $G^\mu(F)$ không bị vi phạm; tức là

$$\delta = \min_{(v_i^\mu, v_j^\mu)} q_{ij}^\mu.$$

Theo cách chọn ban đầu của các khả năng của đồ thị $G^\mu(F)$ luồng mới nhận được từ luồng cũ (bằng cách cộng δ vào luồng trên mạch độ dài âm) là chấp nhận được. Quá trình lại được lặp lại với luồng mới và vân vân. Chi tiết của thuật toán như sau.

Thuật toán tìm luồng có chi phí nhỏ nhất

1. Sử dụng thuật toán luồng lớn nhất, tìm luồng chấp nhận được F với giá trị v.

2. Đặt

$$\begin{aligned} E_1^\mu &:= \{(v_i^\mu, v_j^\mu) \mid f_{ij} < q_{ij}, (v_i, v_j) \in E\}, \\ E_2^\mu &:= \{(v_j^\mu, v_i^\mu) \mid f_{ij} > 0, (v_i, v_j) \in E\}. \end{aligned}$$

Xây dựng đồ thị có trọng số $G^\mu(F) := (V^\mu, E^\mu)$, trong đó

$$\begin{aligned} V^\mu &:= V, \\ E^\mu &:= E_1^\mu \cup E_2^\mu, \end{aligned}$$

- Với mỗi cung $(v_i^\mu, v_j^\mu) \in E_1^\mu$, đặt $c_{ij}^\mu := c_{ij}$.
- Với mỗi cung $(v_j^\mu, v_i^\mu) \in E_2^\mu$, đặt $c_{ji}^\mu := -c_{ij}$.

3. Nếu tồn tại mạch có độ dài âm Φ trên đồ thị $G^\mu(F)$ có trọng số $W := (w_{ij})$, chuyển sang Bước 5. Ngược lại, F là luồng với chi phí nhỏ nhất; thuật toán dừng.

4. Tính δ theo công thức sau:

$$\delta := \min\{q_{ij}^\mu \mid (v_i^\mu, v_j^\mu) \in \Phi\}.$$

- Với mọi cung $(v_i^\mu, v_j^\mu) \in \Phi$ sao cho $c_{ij}^\mu < 0$ thay đổi luồng f_{ji} trên cung $(v_j, v_i) \in E$ bởi $f_{ji} - \delta$.
- Với mọi cung $(v_i^\mu, v_j^\mu) \in \Phi$ sao cho $c_{ij}^\mu > 0$ thay đổi luồng f_{ij} trên cung $(v_i, v_j) \in E$ bởi $f_{ij} + \delta$.

5. Với luồng mới F, chuyển sang Bước 2.

7.5 Cặp ghép

7.5.1 Các bài toán về cặp ghép

Các bài toán về cặp ghép trong các đồ thị (nói chung không phải đồ thị hai phần) được quan tâm vì hai lý do. Thứ nhất, có thể dẫn đến các bài toán này từ việc tổng quát hóa bài toán phân công, và chúng là một phần trong những bài toán khác của đồ thị: các bài toán tìm hành trình tối ưu (như bài toán người đưa thư Trung Hoa), xác định dây chuyền ngắn nhất trong đồ thị vô hướng, v.v.

Mối quan tâm thứ hai về khía cạnh lý thuyết, do mối liên hệ với lớp các bài toán quy hoạch nguyên mà có thể giải bằng thuật toán với độ phức tạp đa thức theo m và n (số các cạnh và các đỉnh của đồ thị).

Xét bài toán sau xây dựng đồ thị con bộ phận G_p của đồ thị vô hướng G trong đó bậc của các đỉnh của đồ thị G_p thỏa mãn tính chất cho trước.

Bài toán 7.5.1. (Bài toán đồ thị bộ phận có ràng buộc về đỉnh) *Giả sử $G = (V, E)$ là đồ thị vô hướng với chi phí c_j tương ứng với mỗi cạnh $e_j \in E$. Ngoài ra, cho trước các số nguyên dương $\delta_i, i = 1, 2, \ldots, n$. Vấn đề đặt ra là tìm một đồ thị bộ phận G_p^* của G sao cho bậc của các đỉnh v_i tương ứng đồ thị G_p^* bằng δ_i, và tổng của các cạnh trong G_p^* lớn nhất (hoặc nhỏ nhất).*

Hiển nhiên, với đồ thị G và các số δ_i cho trước, có thể không tồn tại đồ thị bộ phận G_p thỏa mãn các ràng buộc về đỉnh. Hai điều kiện cần (nhưng không đủ-tại sao?) để tồn tại đồ thị bộ phận chấp nhận được là

$$\delta_i \leq d_G(v_i), \quad \text{với mọi đỉnh } v_i \in V$$

và

$$\sum_{i=1}^{n} \delta_i \quad \text{chẵn.}$$

Điều kiện sau suy trực tiếp từ tính chất: tổng các bậc của các đỉnh bằng hai lần số các cạnh.

Tập con $M \subset E$ gọi là một *cặp ghép* nếu hai cạnh bất kỳ trong M không kề nhau. *Chi phí* của cặp ghép M định nghĩa bởi

$$c(M) = \sum_{e_j \in M} c_j.$$

Ta có bài toán sau:

Bài toán 7.5.2. (Bài toán đối sánh với chi phí lớn nhất) *Tìm cặp ghép M^* có chi phí lớn nhất.*

Bài toán đối sánh với chi phí lớn nhất có thể phát biểu dạng bài toán quy hoạch nguyên:

$$z = \sum_{j=1}^{m} c_j x_j \to \max$$

sao cho

$$\begin{cases} \sum_{j=1}^{m} b_{ij} x_j & \leq & 1, \quad i = 1, 2, \ldots, n, \\ x_j & \in & \{0, 1\}, \end{cases}$$

trong đó b_{ij} là ma trận liên thuộc của G và $x_j = 1$ (hoặc 0) phụ thuộc vào e_j có được ghép cặp hay không.

Hiển nhiên rằng, bài toán đối sánh với chi phí lớn nhất đối với đồ thị $\hat{G}$ nào đó là trường hợp đặc biệt của bài toán có ràng buộc về bậc của các đỉnh. Nếu số các đỉnh của $\hat{G}$ chẵn, ta thêm các cạnh với chi phí bằng $-\infty$ vào $\hat{G}$ để thu được một đồ thị đầy đủ G. Khi đó bài toán đưa về Bài toán 7.5.1 trên đồ thị G trong đó tất cả các $\delta_i = 1$. Nghiệm của bài toán ban đầu dễ dàng suy ra từ bài toán sau bằng cách bỏ qua tất cả các cạnh có chi phí bằng $-\infty$. Nếu số các đỉnh của $\hat{G}$ lẻ thì ta thêm một đỉnh cô lập vào $\hat{G}$ trước khi xây dựng đồ thị G và áp dụng lý luận trên.

Tương ứng với bài toán tìm cặp ghép có chi phí lớn nhất là *bài toán phủ có chi phí nhỏ nhất,* tức là: Tìm phủ E^* của G sao cho tổng chi phí $\sum_{e_j \in E^*} c_j$ nhỏ nhất. Bài toán này có thể phát biểu dạng quy hoạch nguyên như sau:

$$z = \sum_{j=1}^{m} c_j x_j \to \min$$

sao cho

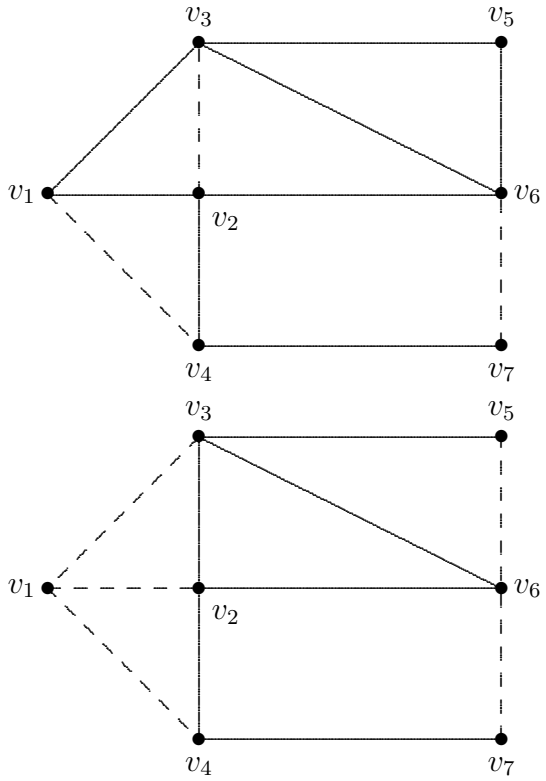
$$\begin{cases} \sum_{j=1}^{m} b_{ij} x_j & \geq & 1, \quad i = 1, 2, \ldots, n, \\ x_j & \in & \{0, 1\}, \end{cases}$$

trong đó $x_j = 1$ (hoặc 0) phụ thuộc vào e_j có thuộc phủ E^* hay không.

Hình 7.4(a) minh họa đồ thị với cặp ghép được vẽ bằng đoạn nét đứt và Hình 7.4(b) minh họa phủ được vẽ bằng đoạn nét đứt trong cùng đồ thị.

Trong trường hợp tất cả các cạnh có chi phí bằng nhau (chẳng hạn 1) thì bài toán đối sánh với chi phí lớn nhất và bài toán tìm phủ với chi phí nhỏ nhất đưa về *bài toán tìm cặp ghép lớn nhất* tức là tìm cặp ghép có số cạnh nhiều nhất và *bài toán tìm phủ nhỏ nhất* tương ứng. Nếu đồ thị G có n đỉnh, khi đó số phần tử của cặp ghép lớn nhất không vượt quá $[n/2]$. Tuy nhiên, số này không phải lúc nào cũng đạt được; chẳng hạn, đồ thị "hình sao" trong Hình 7.5 có cặp ghép lớn nhất với số phần tử 1.

Trường hợp đặc biệt khi các chi phí c_j tùy ý nhưng đồ thị là hai phần, thì bài toán tìm cặp ghép có chi phí nhỏ nhất đưa về *bài toán phân công công việc*, một bài toán quen thuộc của Vận trù học. Với cấu trúc đồ thị đặc biệt này, Bài toán 7.5.1 trở thành *bài toán vận tải.*



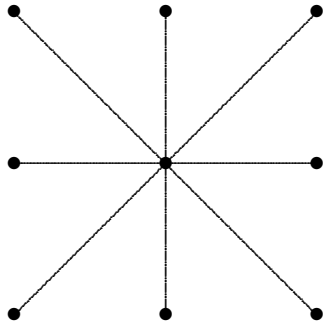
Hình 7.4: (a) Cặp ghép (nét đứt). (b) Phủ (nét đứt).

Mục đích chính của phần này là trình bày bài toán về cặp ghép lớn nhất của đồ thị hai phần trong mối liên hệ với bài toán luồng lớn nhất. Về các thuật toán giải quyết các bài toán cặp ghép trong trường hợp tổng quát có thể xem tài liệu dẫn [14], [30].

7.5.2 Cặp ghép lớn nhất trong đồ thị hai phần

Trước hết ta bắt đầu bằng một ví dụ.

Ví dụ 7.5.3. (Phân công công việc, cặp ghép trong đồ thị hai phần) Một nhà máy có p máy và q công việc cần thực hiện. Giả sử $G = (V, E)$ là đồ thị hai phần mà các đỉnh là $V = V_1 \cup V_2$, với $V_1 = \{1, 2, \ldots, p\}$ và $V_2 = \{1, 2, \ldots, q\}$ và có một cạnh liên thuộc (i, j) nếu máy i có thể thực hiện công việc j. Vấn đề đặt ra là sắp xếp mỗi máy với một công việc mà nó có thể thực hiện. Điều đó có nghĩa rằng, tìm trong G một cặp ghép có số phần tử bằng p.



Hình 7.5: Đồ thị hình sao.

Bài toán cặp ghép có thể đưa về bài toán mạng vận tải như sau. Ta thêm đỉnh nguồn và đỉnh ra nhân tạo s, t sau đó nối từ s đến các đỉnh thuộc tập V_1 và từ các đỉnh thuộc tập V_2 đến t. Gán mỗi cung trong đồ thị thu được, ký hiệu G^M, có khả năng thông qua bằng 1. Ta có G^M là một mạng vận tải.

Định lý 7.5.4. *Giả sử G là đồ thị hai phần định hướng với các tập đỉnh rời nhau V_1 và V_2 mà trong đó các cạnh được định hướng từ các đỉnh trong tập V_1 đến các đỉnh trong tập V_2.*

1. *Luồng F trong mạng vận tải G^M cho ta một cặp ghép trong G. Đỉnh $v_i \in V_1$ được đối sánh với đỉnh v_j trong V_2 nếu và chỉ nếu luồng F trên cung (v_i, v_j) bằng 1.*
2. *Luồng lớn nhất tương ứng với cặp ghép lớn nhất.*
3. *Luồng có giá trị bằng $\#V_1$ tương ứng với cặp ghép hoàn hảo.*

Chứng minh. Giả sử $f_{ij} = 1$. Có đúng một cung đến đỉnh v_i là (s, v_i). Do đó $f_{si} = 1$. Suy ra luồng đến đỉnh v_i bằng 1. Do luồng ra khỏi đỉnh v_i bằng 1 nên có đúng một cung có dạng (v_i, x) có $f_{ix} = 1$ là (v_i, v_j). Tương tự chỉ có một cung dạng (x, v_j) có $f_{xj} = 1$ là (v_i, v_j). Vậy nếu M là tập các cung (v_i, v_j) sao cho $f_{ij} = 1$ thì hai cạnh bất kỳ trong M không kề nhau. Nói cách khác M là cặp ghép của G.

Các phần còn lại suy từ số các đỉnh của V_1 được đối sánh bằng giá trị của luồng tương ứng. ◁

Định lý trên chỉ ra rằng có thể áp dụng thuật toán tìm luồng lớn nhất để xác định cặp ghép lớn nhất của đồ thị hai phần.

7.5.3 Cặp ghép hoàn hảo trong đồ thị hai phần

Ta có định nghĩa sau

Định nghĩa 7.5.5. *Cặp ghép hoàn hảo* trong đồ thị hai phần $G = (V_1 \cup V_2, E)$ là cặp ghép mà mỗi đỉnh $a \in V_1$ tồn tại $b \in V_2$ sao cho $(a, b) \in E$.

Nếu $S \subset V_1$, ta đặt

$$\Gamma(S) := \{v_j \in V_2 \mid \text{tồn tại } v_i \in S \text{ sao cho } (v_i, v_j) \in E\}.$$

Giả sử G có cặp ghép hoàn hảo. Nếu $S \subset V_1$ ta cần có

$$\#S \leq \#\Gamma(S).$$

Ta sẽ chỉ ra rằng nếu $\#S \leq \#\Gamma(S)$ với mọi tập con S của V_1 thì G có một cặp ghép hoàn hảo. Kết quả này được chứng minh bởi Hall và gọi là *Định lý đám cưới của Hall* do V_1 là tập những người đàn ông và V_2 là tập những người đàn bà và tồn tại cạnh nối $v_i \in V_1$ đến $v_j \in V_2$ nếu v_i và v_j ưng thích nhau; định lý cho một điều kiện để người đàn ông có thể cưới người mình thích.

Định lý 7.5.6. *Tồn tại cặp ghép hoàn hảo nếu và chỉ nếu*

$$\#S \leq \#\Gamma(S) \tag{7.2}$$

với mọi tập con S của V_1.

Chứng minh. Ta chỉ cần chứng minh nếu điều kiện (7.2) đúng thì tồn tại cặp ghép hoàn hảo. Đặt $n_1 = \#V_1$ và $(P, \tilde{P})$ là thiết diện nhỏ nhất trong mạng vận tải. Nếu ta chứng minh rằng khả năng của thiết diện này bằng n_1 thì luồng lớn nhất có giá trị bằng n_1. Cặp ghép tương ứng với luồng lớn nhất sẽ là cặp ghép hoàn hảo.

Chứng minh bằng phản chứng. Giả sử ngược lại, khả năng của thiết diện nhỏ nhất $(P, \tilde{P})$ nhỏ hơn n_1. Nhận xét rằng khả năng của thiết diện này bằng số các cạnh trong tập

$$M = \{(x, y) \mid x \in P, y \in \tilde{P}\}.$$

Một phần tử của M có một trong ba dạng:

Loại 1: $(s, v_i), v_i \in V_1$.

Loại 2: $(v_i, v_j), v_i \in V_1, v_j \in V_2$.

Loại 3: $(v_j, t), v_j \in V_2$.

Chúng ta sẽ đếm số các cạnh trong mỗi loại.

Nếu $V_1 \in \tilde{P}$ thì khả năng của thiết diện là n_1; do đó

$$V_1^* = V_1 \cap P$$

khác trống. Suy ra tồn tại $n_1 - \#V_1^*$ cạnh loại 1 trong E.

Ta phân hoạch $R(V_1^*)$ thành các tập hợp

$$X = R(V_1^*) \cap P \quad \text{và} \quad Y = R(V_1^*) \cap \tilde{P}.$$

Khi đó tồn tại ít nhất $\#V_1^*$ cạnh loại 3 trong E. Do đó tồn tại ít hơn

$$n_1 - (n_1 - \#V_1^*) - \#X = \#V_1^* - \#X$$

cạnh loại 2 trong E. Mà mỗi phần tử của Y đóng góp nhiều nhất một cạnh loại 2, nên

$$\#Y < \#V_1^* - \#X.$$

Vậy

$$\#R(V_1^*) = \#X + \#Y < \#V_1^*.$$

Điều này mâu thuẫn với (7.2). Do đó tồn tại cặp ghép hoàn hảo. ◁

Ví dụ 7.5.7. Có n máy tính và n ổ đĩa. Mỗi máy tính tương thích với m ổ đĩa và mỗi ổ đĩa tương thích với m máy tính. Có thể ghép mỗi máy tính với một ổ đĩa mà nó tương thích?

Đặt V_1 là tập các máy tính và V_2 là tập các ổ đĩa. Ta cho tương ứng cạnh (v_i, v_j) nếu máy tính $v_i \in V_1$ tương thích với ổ đĩa $v_j \in V_2$. Chú ý rằng mọi đỉnh có bậc bằng m. Đặt $S = \{v_1, v_2, \ldots, v_k\} \subset V_1$. Khi đó có km cạnh xuất phát từ S. Nếu $l := \#\Gamma(S)$ thì $\Gamma(S)$ nhận nhiều nhất lm cạnh đến từ S. Do đó

$$km \leq lm.$$

Nên

$$\#S = k \leq l = \#\Gamma(S).$$

Theo Định lý đám cưới Hall, tồn tại cặp ghép hoàn hảo. Vậy có thể ghép mỗi máy tính với một ổ đĩa tương thích.

Phần phụ lục A

Thư viện Graph.h

Dưới đây là thư viện gồm các cấu trúc dữ liệu và các thủ tục cần thiết hỗ trợ việc cài đặt các thuật toán trong giáo trình.

```
#if !defined(graph_h)
#define graph_h

#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <string.h>


/****** Phan dinh nghia cac hang *****/

#define TRUE                    1
#define FALSE                   0
#define INFTY                   32767
#define MAXEDGES        50 // So cuc dai cac canh
#define MAXVERTICES  25 // So cuc dai cac dinh
#define MAXSTRINGS    16 // Chieu dai cuc dai xau ky tu
```

```
/****** Phan dinh nghia cac kieu du lieu *****/

typedef unsigned char byte;
typedef byte Boolean;
typedef char DataType[MAXSTRINGS + 1]; // Them mot ma ket thuc chuoi

/*******************************
Cau truc du lieu: don lien ket
*******************************/

typedef struct VertexNode *AdjPointer;
struct VertexNode
{
  byte Vertex;
  int Length;
  int Flow;
  AdjPointer Next;
};

typedef struct
{
  DataType Data;
  AdjPointer Next;
} HeadNode;
typedef HeadNode *HeadPointer;

typedef HeadPointer ArrayOfPointer[MAXVERTICES];

typedef struct QueueType *QueueNode;
struct QueueType
{
  byte Vertex;
  QueueNode Next;
```

```
};
typedef struct
{
  QueueNode Head, Tail;
} Queue;

typedef byte Path[MAXVERTICES];

typedef byte
      SetOfVertices[(MAXVERTICES%8)?((MAXVERTICES/8)+1):(MAXVERTICES/8)];


/**********************************
Danh sach da lien ket cho cac canh
**********************************/

typedef struct EdgeNode *EdgePointer;
struct EdgeNode
{
  byte Vertex[2];
  EdgePointer Link[2];
};

typedef struct
{
  char Data;
  EdgePointer Next;
} ListEdge;
typedef ListEdge *ListEdgePointer;

typedef ListEdgePointer ArrayOfEdge[MAXVERTICES];

/***** Phan khai bao prototype ham *****/
```

```
void Create(AdjPointer *List);
Boolean Empty(AdjPointer List);
void Push(AdjPointer *List, byte Item);
void Pop(AdjPointer *List, byte *Item);

void CreatQueue(Queue *Q);
Boolean EmptyQueue(Queue Q);
void PushQueue(Queue *Q, byte Item);
void PopQueue(Queue *Q, byte *Item);

Boolean EmptySet(SetOfVertices S);
Boolean InSet(SetOfVertices S, byte Value);
void InitSet(SetOfVertices S, byte MaxValue);
void AddSet(SetOfVertices S, byte Value);
void SubSet(SetOfVertices S, byte Value);

void MakeV_out(char *FileName, ArrayOfPointer V_out, byte *NumVertices,
               Boolean Weight);
void MakeV_in(char *FileName, ArrayOfPointer V_in, ArrayOfPointer V_out,
              byte *NumVertices);
void BuildGraph(char *FileName, ArrayOfEdge E, byte *NumVertices);

void DisplayV_out(ArrayOfPointer V_out, byte NumVertices, Boolean Weight);
void DisplayV_in(ArrayOfPointer V_in, byte NumVertices);

void DFS(ArrayOfEdge E, byte Start, SetOfVertices S);

void PathTwoVertex(Path Pred, byte Start, byte Terminal);
void PopHeadPtr(byte NumVertices, ArrayOfPointer V_out);

/***** Phan cai dat cac ham *****/
void Create(AdjPointer *List)
{
  (*List) = NULL;
```

```
}

Boolean Empty(AdjPointer List)
{
  return((List == NULL) ? TRUE : FALSE);
}

void Push(AdjPointer *List, byte Item)
{
  AdjPointer TempPtr;

  TempPtr = (AdjPointer) malloc(sizeof(struct VertexNode));
  TempPtr->Vertex = Item;
  TempPtr->Next = (*List);
  (*List) = TempPtr;
}

void Pop(AdjPointer *List, byte *Item)
{
  AdjPointer TempPtr;

  if (Empty(*List))
  {
    printf(" Thao tac con tro khong hop le ");
    return;
  }
  TempPtr = (*List);
  (*Item) = TempPtr->Vertex;
  (*List) = TempPtr->Next;
  free(TempPtr);
}

void CreatQueue(Queue *Q)
{
```

```
  (*Q).Head = NULL;
}

Boolean EmptyQueue(Queue Q)
{
  return((Q.Head == NULL) ? TRUE : FALSE);
}

void PushQueue(Queue *Q, byte Item)
{
  QueueNode TempPtr;

  TempPtr = (QueueNode) malloc(sizeof(struct QueueType));
  TempPtr->Vertex = Item;
  TempPtr->Next = NULL;
  if ((*Q).Head == NULL) (*Q).Head = TempPtr;
  else (*Q).Tail->Next = TempPtr;
  (*Q).Tail = TempPtr;
}

void PopQueue(Queue *Q, byte *Item)
{
  QueueNode TempPtr;

  if (EmptyQueue(*Q))
  {
    printf(" Thao tac con tro khong hop le ");
    return;
  }
  TempPtr = (*Q).Head;
  (*Item) = TempPtr->Vertex;
  (*Q).Head = TempPtr->Next;
  free(TempPtr);
}
```

```
Boolean EmptySet(SetOfVertices S)
{
  int i;
  int k = (MAXVERTICES %8 ) ? ((MAXVERTICES / 8) + 1) : (MAXVERTICES / 8);

  for (i = 0; i < k; i++)
    if (S[i] != 0) return(FALSE);
  return(TRUE);
}

Boolean InSet(SetOfVertices S, byte Value)
{
  if ((Value < 1) || (Value > MAXVERTICES))
  return(FALSE);

  return((S[(Value - 1) / 8] & (0x80 >> ((Value - 1) % 8))) ? TRUE : FALSE);
}

void InitSet(SetOfVertices S, byte MaxValue)
{
  int i;

  if (MaxValue>MAXVERTICES)
  {
    printf(" Gia tri khong thuoc tap hop ");
    return;
  }

  for (i = 0; i < MaxValue; i++) S[i/8] |= 0x80 >> (i%8);
}

void AddSet(SetOfVertices S, byte Value)
{
```

```
  int i;

  if ((Value < 1) || (Value > MAXVERTICES))
  {
    printf(" Gia tri khong thuoc tap hop ");
    return;
  }

  S[(Value-1)/8] |= 0x80 >> ((Value-1)%8);
}

void SubSet(SetOfVertices S, byte Value)
{
  if ((Value < 1) || (Value > MAXVERTICES))
  {
    printf(" Gia tri khong thuoc tap hop ");
    return;
  }

  S[(Value-1)/8] &= ~(0x80 >> ((Value-1)%8));
}

int feoln(FILE *fp)
{
  char c;

  if ((c=fgetc(fp))==10)
  {
    fseek(fp, -2, 1);
    return(TRUE);
  }
  else
    if (c==EOF) return(TRUE);
    else
```

```
    {
      fseek(fp, -1, 1);
      return(FALSE);
    }
}

void freadln(FILE *fp)
{
  char c;

  while (((c=fgetc(fp))!=10)&&(c!=EOF));
}

void MakeV_out(char *FileName, ArrayOfPointer V_out, byte *NumVertices,
               Boolean Weight)
{
  byte NumVert;
  HeadPointer HeadPtr;
  AdjPointer VerPtr;
  FILE  *FileData;

  if ((FileData = fopen(FileName, "rt")) == NULL)
  {
    printf(" File khong tim thay ");
    return;
  }

  NumVert = 0;
  while (!feof(FileData))
  {
    HeadPtr = (HeadPointer) malloc(sizeof(HeadNode));
    HeadPtr->Next = NULL;
    fgets(HeadPtr->Data, MAXSTRINGS+1, FileData);
    // Ham fgets(char *s, int n, FILE *fp) chi doc n-1 ky tu
```

```
    while (!feoln(FileData))
    {
      VerPtr = (AdjPointer) malloc(sizeof(struct VertexNode));
      fscanf(FileData, "%d", &(VerPtr->Vertex));
      if (Weight)
      {
        fscanf(FileData, "%d", &(VerPtr->Length));
        VerPtr->Flow = 0;
      }
      VerPtr->Next = HeadPtr->Next;
      HeadPtr->Next = VerPtr;
    }
    freadln(FileData);
    ++NumVert;
    V_out[NumVert] = HeadPtr;
  }
  (*NumVertices) = NumVert;
  fclose(FileData);
}

void MakeV_in(char *FileName, ArrayOfPointer V_in,
                          ArrayOfPointer V_out, byte *NumVertices);
{
  byte NumVert;
  int    i, j;
  HeadPointer HeadPtr;
  AdjPointer CurrPtr, VerPtr;

  MakeV_out(FileName, V_out, &NumVert, TRUE);
  (*NumVertices) = NumVert;
  for (i = 1; i <= NumVert; i++)
  {
    HeadPtr = (HeadPointer) malloc(sizeof(HeadNode));
    strcpy(HeadPtr->Data, V_out[i]->Data);
```

```
    HeadPtr->Next = NULL;
    for (j = 1; j <= NumVert; j++)
    {
      CurrPtr = V_out[j]->Next;
      while (CurrPtr!=NULL)
      {
        if (CurrPtr->Vertex == i)
        {
          VerPtr = (AdjPointer)malloc(sizeof(struct VertexNode));
          VerPtr->Vertex = j;
          VerPtr->Length = CurrPtr->Length;
          VerPtr->Flow = 0;
          VerPtr->Next = HeadPtr->Next;
          HeadPtr->Next = VerPtr;
        }
        CurrPtr = CurrPtr->Next;
      }
    }
    V_in[i] = HeadPtr;
  }
}

void BuildGraph(char *FileName, ArrayOfEdge E, byte *NumVertices)
{
  byte EndPt, NumVert;
  int i;
  char ch[2];
  EdgePointer EdgePtr;
  FILE  *FileData;

  if ((FileData = fopen(FileName, "rt")) == NULL)
  {
    printf(" File khong tim thay ");
    return;
```

```
  }

  NumVert = 0;
  while (!feoln(FileData))
  {
    ++NumVert;
    E[NumVert] = (ListEdgePointer) malloc(sizeof(ListEdge));
    fscanf(FileData, "%s", ch);
    E[NumVert]->Data = ch[0];
    E[NumVert]->Next = NULL;
  }
  (*NumVertices) = NumVert;

  for (i = 1;  i<= NumVert; i++) printf("%c ", E[i]->Data);
  printf("\n");

  freadln(FileData);
  while (!feof(FileData))
  {
    EdgePtr = (EdgePointer) malloc(sizeof(struct EdgeNode));
    for (i=0; i<2; i++)
    {
      fscanf(FileData, "%d", &EndPt);
      printf("%d ", EndPt);
      EdgePtr->Vertex[i] = EndPt;
      EdgePtr->Link[i] = E[EndPt]->Next;
      E[EndPt]->Next = EdgePtr;
    }
    printf("\n");
    freadln(FileData);
  }
  fclose(FileData);
}
```

```
void DFS(ArrayOfEdge E, byte Start, SetOfVertices Unvisited)
{
  EdgePointer Ptr;
  byte StartEnd, OtherEnd, NewStart;

  SubSet(Unvisited, Start);
  Ptr = E[Start]->Next;
  while ((!EmptySet(Unvisited))&&(Ptr!=NULL))
  {
    StartEnd = 0;
    OtherEnd = 1;
    if (Ptr->Vertex[0]!=Start)
    {
      StartEnd = 1;
      OtherEnd = 0;
    }
    NewStart = Ptr->Vertex[OtherEnd];
    if (InSet(Unvisited, NewStart)) DFS(E, NewStart, Unvisited);
    Ptr = Ptr->Link[StartEnd];
  }
}

void DisplayV_out(ArrayOfPointer V_out, byte NumVertices, Boolean Weight)
{
  int  i;
  AdjPointer CurrPtr;

  for (i = 1; i <= NumVertices; i++)
  {
    printf("(%d) %s ", i, V_out[i]->Data);
    CurrPtr = V_out[i]->Next;
    while (CurrPtr!=NULL)
    {
      printf("%d ", CurrPtr->Vertex);
```

```
      if (Weight) printf("%d ", CurrPtr->Length);
      CurrPtr = CurrPtr->Next;
    }
    printf("\n");
  }
  printf("\n");
}

void DisplayV_in(ArrayOfPointer V_in, byte NumVertices)
{
  int i;
  AdjPointer CurrPtr;

  for (i = 1; i <= NumVertices; i++)
  {
    printf("%s ", V_in[i]->Data);
    CurrPtr = V_in[i]->Next;
    while (CurrPtr!=NULL)
    {
      printf(" %d %d", CurrPtr->Vertex, CurrPtr->Length);
      CurrPtr = CurrPtr->Next;
    }
    printf("\n");
  }
}

void PathTwoVertex(Path Pred, byte Start, byte Terminal)
{
  if (Terminal != Start)
  {
    PathTwoVertex(Pred, Start, Pred[Terminal]);
    printf(" ---> %2d", Terminal);
  }
}
```

```
void PopHeadPtr(byte NumVertices, ArrayOfPointer V_out)
{
  byte Item;
  int i;
  AdjPointer CurrPtr;

  for (i = 1; i <= NumVertices; i++)
  {
    CurrPtr = V_out[i]->Next;
    while (CurrPtr != NULL) Pop(&CurrPtr, &Item);
    free(V_out[i]);
  }
}

#endif
```

Tài liệu tham khảo

[1] Appel K., *The proof of the four-colour problem,* New Scientist. **72**, 154-155 (1976).

[2] Arlinghaus S., Arlinghaus W., Nystuen J., *The Hedetniemi matrix sum: an algorithm for shortest path and shortest distance,* Geographical Analysis, Vol. **22**, No. 4, Oct., 351-360 (1990).

[3] Bellman R., *On a routing problem,* Quart. of Applied Mathematics, **16**, 87 (1958).

[4] Berge C., *Lý thuyết đồ thị và ứng dụng,* NXB Khoa học và Kỹ thuật Hà Nội, 1971.

[5] Berge C., *Two theorems in graph theory,* Proc. Nat. Ac. Sc., **43**, 842 (1957).

[6] Berry R. C., *A constrained shortest path,* Paper presented at the 39th National ORSA Metting, Dallas, Texas (1971).

[7] Bondy J. A., *Properties of graphs with constraints on degrees,* Studia Sci. Math. Hung. **4** 473-475 (1969).

[8] Bondy J. A., Chvatal V., *A method in graph theory,* Discrete Math. **15**, 111-135 (1976).

[9] Brooks R. L., *On coloring the nodes of a network,* Proc. Cambridge Phil. Soc., Vol. **37**, 194-197 (1941).

[10] Busacker R. G., Gowen P. J., *A procedure for determining a family of minimal-cost network flow patterns,* Operations Research Office, Technical paper 15 (1961).

[11] Cayley A., *Collected papers,* Quart. Jl. of Mathematics, **13** Cambridge, 26 (1897).

[12] Chase S. M., *Analysis of algorithms for finding all spanning trees of a graph,* Report No. 401, Department of Computer Science, University of Illinois, Urbana, Oct. (1970).

[13] Chvatal V., *On Hamilton's ideals,* J. Combinat. Theory B **12** 163-168 (1972).

[14] Christofides N., *Graph theory an algorithmic approach,* Academic Press INC. (1975).

[15] Coxeter H. S. M., *Introduction to geometry,* Wiley, New York (1961).

[16] Danzig G. B., *All shortest routes in a graph,* in Théorie des graphes, Proceedings of the International Symposium, Rome 1966, Dunod, Paris, 91-92 (1967).

[17] Dirac G. A., *Some theorems on abstract graphs,* Proc. London Math. Soc. **2**, 68-81 (1952).

[18] De Freisseix H., Rosenstiehl P., *The depth-search theorem for planarity,* in Proceedings of the Cambridge Combinatorial Conference, North Holland, Amsterdam (1982).

[19] Deo N., *Graph theory with applications to engineering and computer science,* Prentice-Hall Inc. (1974).

[20] Dijkstra, E. W., *A note on two problems in connection with graphs,* Numerische Mathematik, **1**, 269 (1959).

[21] Dreyfus S. E., Wagner R. A., *The Steiner problem in graphs,* Networks, **1**, 195 (1972).

[22] Euler L., *Solutio problematis ad geometriam situs pertinentis,* Commun. Acad. Sci. Imp. Petropol. **8**, Opera Omnia (1), Vol. 7, 128-140 (1736).

[23] Euler L., *Commentationes Arithmeticae collectae,* St. Petersburg, (1766).

[24] Fary I., *On straight line representation of planar graphs,* Acta Sci. Math. Szeged, Vol. **11**, 229-293 (1948).

[25] Floyd R. W., *Algorithm 97-Shortest path,* Comm. of ACM, **5**, 345 (1962).

[26] Ford L. R. Jr., *Network flow theory,* Rand Corporation Report 923 (1946).

[27] Ford L. R., Fulkerson D. R., *Flows in networks,* Princeton University Press, Princeton (1962).

[28] Gilbert E. N., Pollack H. O., *Steiner minimal trees,* Jl. of SIAM (Appl. Math.), **16** 1 (1968).

[29] Gomory R. E., Hu T. C., *Synthesis of a communication network,* Jl. of SIAM (App. Math.), **12** 348 (1964).

[30] Gondran M., Minoux M., Vajda S., *Graphs and algorithms,* John Wiley & Sons (1990).

[31] Hamming R. W., *Coding and information theory,* Prentice Hall (1980).

[32] Hanan M., *On Steiner's problem with rectilinear distance,* Jl. of SIAM (Appl. Math.), **14** 255 (1966).

[33] Hopcroft J. E., Tarjan R. E., *Isomorphism of planar graphs,* in Complexity of Computer Computations, Plenum, New York (1972).

[34] Hopcroft J. E., Tarjan R. E., *Efficient planarity testing,* J. ACM **21**, 549-568 (1974).

[35] Hu T. C., *Integer programming and network flows,* Addison-Wesley, Reading, Massachusetts (1969).

[36] Kerchenbaum A., Van Slyke R., *Computing minimum spanning trees efficiently,* Proc. of the Ann. Conf. of ACM, Boston, 518 (1972).

[37] Kirchhoff G., in "Annalen der Physik and Chemie" **72**, 497 (1847).

[38] Klein M., *A primal method for minimal cost flows with applications to the assignment and transportation problems,* Man. Sci., **14**, 205 (1967).

[39] Kruskal J. B. Jr., *On the shortest spanning subtree of a graph and the traveling salesman problem,* Proc. American Mathematical Soc., **7**, 48 (1956).

[40] Kraitchik M., *Le problème des Reines,* Bruxelles (1926).

[41] Kevin V., Whitney M., *Algorithm 422-Minimum spanning tree,* Comm. of ACM, **15**, 273 (1972).

[42] Las Vergnas M., *Problèmes de couplage et problèmes hamiltoniens en théorie des graphes,* Doctoral thesis, Univsité de Paris VI (1972).

[43] Larry N., Sanford L., *Lập trình nâng cao bằng Pascal với các cấu trúc dữ liệu,* Scitec.

[44] Mei-Ko Kwan, *Graphic programming using odd or even points,* Chinese Math. **1**, 273 (1962).

[45] Moore E. F., *The shortest path through a maze,* Proc. Int. Symp. on the Theory of Switching, Path II, 285 (1957).

[46] Murchland J. D., *A new method for finding all elementary paths in a complete directed graph,* London School of Economics, Report LSE-TNT-22 (1965).

[47] Ore O., *Note on Hamilton circuits,* Amer. Math. Mothly, **67**, 55 (1960).

[48] Ore O., *The four colour problem,* Academic Press, New York (1967).

[49] Prim R. C., *Shortest connection networks and some generalizations,* Bell Syst. Tech. Jl., **36**, 1389 (1957).

[50] Paton K., *An algorithm for finding a fundamental set of cycles of a graph,* Comm. ACM, Vol. **12**, No. 9, Sept., 514-518 (1969).

[51] Pósa L., *A theorem concerning Hamiltonian lines,* Magyar Tud. Akad. Mat. Kutató Inst. Kozl **7** 255-226 (1962).

[52] Shirey R. W., *Implementation and analysis of efficient graph planarity testing algorithms,* Ph.D. Dissertation, Computer Sciences, University of Wisconsin, Madison, Wisc. (1969).

[53] Tarjan R., *Depth-first search and linear graph algorithms,* SIAM J. Computer **1** 146-160 (1972).

[54] Tutte W. T., *How to draw graph,* Proc. London Math. Soc., Ser. 3, Vol. 13, 743-768 (1963).

[55] Whitney H., 2−*Isomorphic graphs,* Am. J. Math. Vol. 55 245-254 (1933).