Lý thuyết đồ thị

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Lý thuyết đồ thị

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Các tác giả:

Khoa CNTT ĐHSP KT Hưng Yên

Phiên bản trực tuyến:
http://voer.edu.vn/c/cad51e6d

MỤC LỤC

Mở đầu

Lý thuyết đồ thị là một lĩnh vực đã có từ lâu và có nhiều ứng dụng hiện đại. Những tư tưởng cơ bản của lý thuyết đồ thị được đề xuất vào những năm đầu của thế kỷ 18 bởi nhà toán học lỗi lạc người Thụy Sỹ Lenhard Eurler. Chính ông là người đã sử dụng đồ thị để giải bài toán nổi tiếng về các cái cầu ở thành phố Konigsberg.

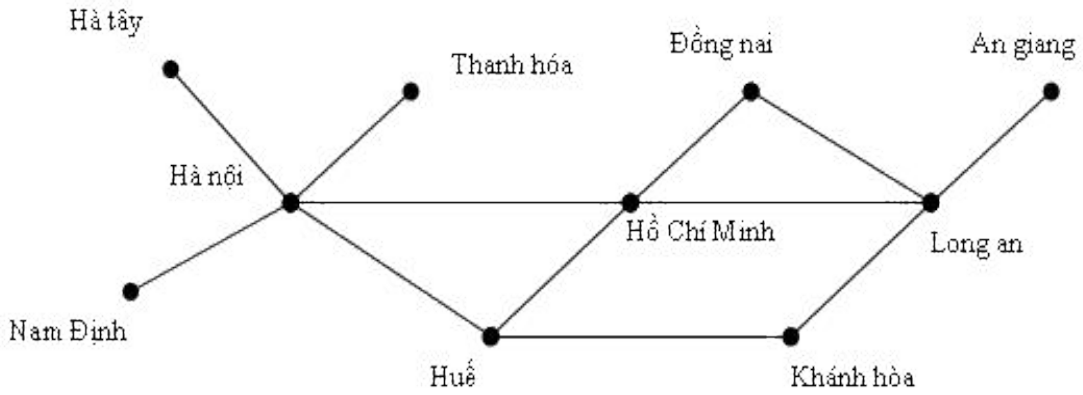
Đồ thị được sử dụng để giải các bài toán trong nhiều lĩnh vực khác nhau. Chẳng hạn, đồ thị có thể sử dụng để xác định các mạch vòng trong vấn đề giải tích mạch điện. Chúng ta có thể phân biệt các hợp chất hóa học hữu cơ khác nhau với cùng công thức phân tử nhưng khác nhau về cấu trúc phân tử nhờ đồ thị. Chúng ta có thể xác định hai máy tính trong mạng có thể trao đổi thông tin được với nhau hay không nhờ mô hình đồ thị của mạng máy tính. Đồ thị có trọng số trên các cạnh có thể sử dụng để giải các bài toán như: Tìm đường đi ngắn nhất giữa hai thành phố trong mạng giao thông. Chúng ta cũng còn sử dụng đồ thị để giải các bài toán về lập lịch, thời khóa biểu, và phân bố tần số cho các trạm phát thanh và truyền hình.

Bài 1: Các khái niệm cơ bản của Lý thuyết đồ thị.

Các khái niệm cơ bản của Lý thuyết đồ thị.

Định nghĩa cơ bản về đồ thị

Đồ thị là một cấu trúc rời rạc bao gồm các đỉnh và các cạnh nối các đỉnh này. Chúng ta phân biệt các loại đồ thị khác nhau bởi *kiểu* và *số lượng* cạnh nối hai đỉnh nào đó của đồ thị. Để có thể hình dung được tại sao lại cần đến các loại đồ thị khác nhau, chúng ta sẽ nêu ví dụ sử dụng chúng để mô tả một mạng máy tính. Giả sử ta có một mạng gồm các máy tính và các kênh điện thoại (gọi tắt là kênh thoại) nối các máy tính này. Chúng ta có thể biểu diễn các vị trí đặt náy tính bởi các điểm và các kênh thoại nối chúng bởi các đoạn nối, xem hình 1.



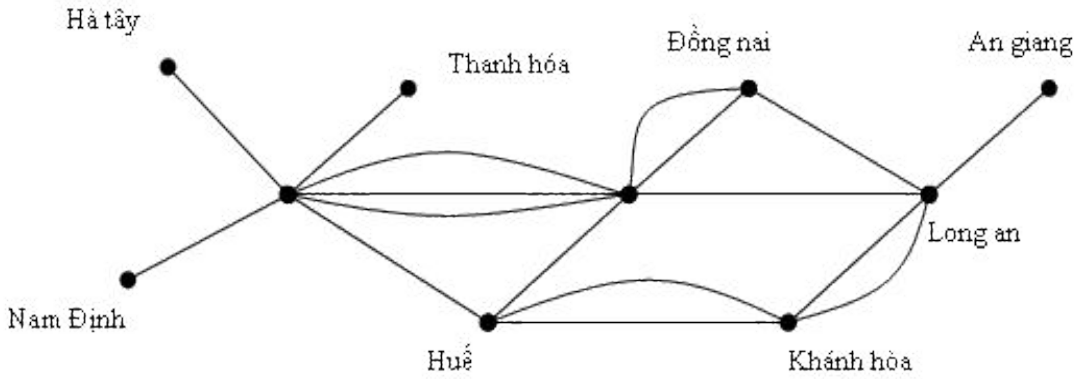
Hình 1Sơ đồ mạng máy tính.

Nhận thấy rằng trong mạng ở hình 1, giữa hai máy bất kỳ chỉ có nhiều nhất là một kênh thoại nối chúng, kênh thoại nay cho phép liên lạc cả hai chiều và không có máy tính nào lại được nối với chính nó. Sơ đồ mạng máy cho trong hình 1 được gọi là *đơn đồ thị vô hướng*. Ta đi đến định nghĩa sau

Định nghĩa 1

Đơn đồ thị vô hướng G = (V,E) bao gồm V là tập các đỉnh, và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh.

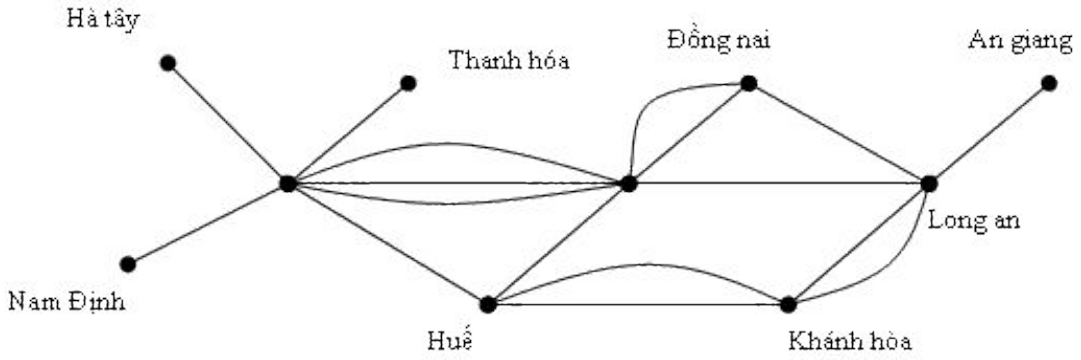
Trong trường hợp giữa hai máy tính nào đó thường xuyên phải truyền tải nhiều thông tin người ta phải nối hai máy nàu bởi nhiều kênh thoại. Mạng với đa kênh thoại giữa các máy được cho trong hình 2



Hình 2 Sơ đồ mạng máy tính với đa kênh thoại.

Định nghĩa 2

Đa đồ thị vô hướng G= (V, E) bao gồm V là tập các đỉnh, và E là tập các cặp không có thứ tự gồm hai phần tử khác nhau của V gọi là các cạnh. Hai cạnh e 1 và e 2 được gọi là cạnh lặp nếu chúng cùng tương ứng với một cặp đỉnh.



Hình 3 Sơ đồ mạng máy tính với kênh thoại thông báo.

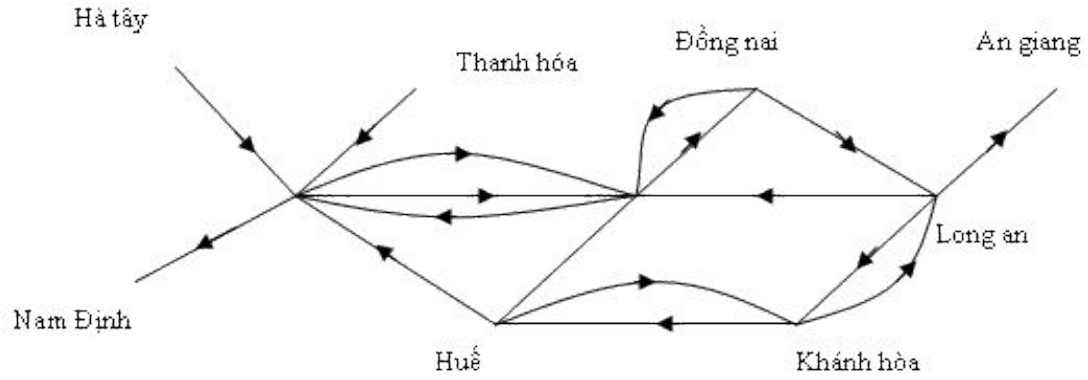
Rõ ràng mỗi đơn đồ thị đều là đa đồ thị, nhưng không phải đa đồ thị nào cũng là đơn đồ thị, vì trong đa đồ thị có thể có hai (hoặc nhiều hơn) cạnh nối một cặp đỉnh nào đó.

Trong mạng máy tính có thể có những kênh thoại nối một náy nào đó với chính nó (chẳng hạn vòi mục đính thông báo). Mạng như vậy được cho trong hình 3. Khi đó đa đồ thị không thể mô tả được mạng như vậy, bởi vì có những *khuyên* (cạnh nối một đỉnh

với chính nó). Trong trường hợp nàychúng ta cần sử dụng đến khái niệm *giả đồ thị vô hướng,* được định nghĩa như sau:

Định nghĩa 3

Giả đồ thị vô hướng G = (V, E) bao gồm V là tập các đỉnh và E là tập các cặp không có thứ tự gồm hai phần tử (không nhất thiết phải khác nhau) của V gọi là cạnh. Cạnh e được gọi là khuyên nếu nó có dạng e = (u, u).



Hình 4 Mạng máy tính với kênh thoại một chiều.

Các kênh thoại trong mạng máy tính có thể chỉ cho phép truyền tin theo một chiều. Chẳng hạn, trong hình 4 máy chủ ở Hà Nội chỉ có thể nhận tin từ các máy ở địa phương, có một số máy chỉ có thể gửi tin đi, còn các kênh thoại cho phép truyền tin theo cả hai chiều được thay thế bởi hai cạnh có hướng ngược chiều nhau.

Ta đi đến định nghĩa sau.

Định nghĩa 4

Đơn đồ thị có hướng G = (V, E) bao gồm V là tập các đỉnh và E là tập các cặp có thứ tự gồm hai phần tử khác nhau của V gọi là các cung.

Nếu trong mạng có thể có đa kênh thoại một chiều, ta sẽ phải sử dụng đến khái niệm *đa đồ thị có hướng*:

Định nghĩa 5

Đa đồ thị có hướng G = (V, E) bao gồm V là tập các đỉnh và E là tập các cặp có thứ tự gồm hai phần tử khác nhau của V gọi là các cung. Hai cung e 1 , e 2 tương ứng với cùng một cặp đỉnh được gọi là cung lặp.

Trong các phần tiếp theo chủ yếu chúng ta sẽ làm việc v?i đơn đồ thị vô hướng và đơn đồ thị có hướng. Vì vậy, để cho ngắn gọn, ta sẽ bỏ qua tính từ **đơn** khi nhắc đến chúng.

Đường đi. chu trình. Đồ thị liên thông

Định nghĩa 6

Đường đi độ dài n từ đỉnh u đến đỉnh v, trong đó n là số nguyên dương, trên đồ thị vô hướng G = (V, E) là dãy x 0 , x 1 ,..., x n-1 , x n

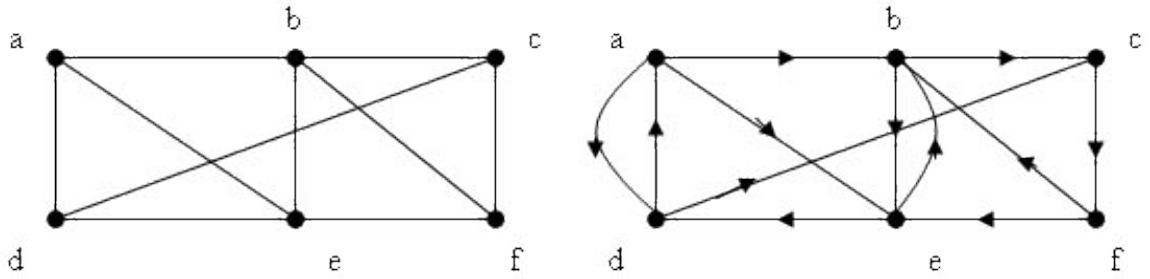
trong đó u = x 0 , v = x n , (x i , x i+1) ∈ E, i = 0, 1, 2,..., n-1.

Đường đi nói trên còn có thể biểu diễn dưới dạng dãy các cạnh:

(x 0 , x 1), (x 1 , x 2), ..., (x n-1 , x n)

Đỉnh u gọi là đỉnh đầu, còn đỉnh v gọi là đỉnh cuối của đường đi. Đường đi có đỉnh đầu trùng với đỉnh cuối (tức là u = v) được gọi là chu trình . Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.

Ví dụ 1 Trên đồ thị vô hướng cho trong hình 5: a, d, c, f, e là đường đi đơn độ dài 4. Còn d, e, c, a không là đường đi, do (c,e) không phải là cạnh của đồ thị. Dãy b, c, f, e, b là chu trình độ dài 5. Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cạnh (a, b) có mặt trong nó 2 lần.



Hình 5 Đường đi trên đồ thị

Khái niệm đường đi và chu trình trên đồ thị có hướng được định nghĩa hoàn toàn tương tự như trong trường hợp đồ thị vô hướng, chỉ khác là ta có chú ý đến hướng trên các cung.

Định nghĩa 7.

Đường đi độ dài n từ đỉnh u đến đỉnh v, trong đó, n là số nguyên dương, trên đồ thị có hướng G = (V, A) là dãy x 0 , x 1 ,..., x n-1 , x n

trong đó u = x 0 , v = x n , (xi, x i+1) ∈ E, i = 0, 1, 2,..., n-1.

Đường đi nói trên còn có thể biểu diễn dưới dạng dãy các cung:

(x 0 , x 1), (x 1 , x 2), ..., (x n-1 , x n)

Đỉnh u gọi là đỉnh đầu, còn đỉnh v gọi là đỉnh cuối của đường đi. Đường đi có đỉnh đầu trùng với đỉnh cuối (tức là u = v) được gọi là chu trình . Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.

Ví dụ 2 Trên đồ thị có hướng cho trong hình 1.6: a, d, c, f, e là đường đi đơn độ dài 4. Còn d, e, c, a không là đường đi, do (c,e) không phải là cạnh của đồ thị. Dãy b, c, f, e, b là chu trình độ dài 4. Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cạnh (a, b) có mặt trong nó 2 lần.

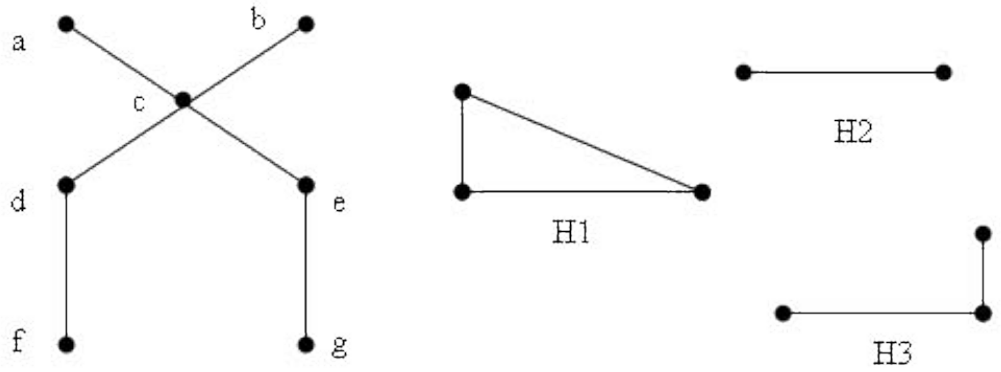
Xét một mạng máy tính. Một câu hỏi đặt ra là hai máy tính bất kỳ trong mạng này có thể trao đổi thông tin được với nhau hoặc là trực tiếp qua kênh nối chúng hoặc thông qua một hoặc vài máy tính trung gian trong mạng? Nếu sử dụng đồ thị để biểu diễn mạng máy tính này (trong đó các đỉnh của đồ thị tương ứng với các máy tính, còn các cạnh tương ứng với các kênh nối) câu hỏi đó được phát biểu trong ngôn ngữ đồ thị như sau: Tồn tại hay không đường đi giữa mọi cặp đỉnh của đồ thị.

Định nghĩa 8

Đồ thị vô hướng G = (V, E) được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Như vậy hai máy tính bất kỳ trong mạng có thể trao đổi thông tin được với nhau khi và chỉ khi đồ thị tương ứng với mạng này là đồ thị liên thông.

Ví dụ 3 Trong hình 6: Đồ thị G là liên thông, còn đồ thị H là không liên thông.



Hình 6 Đồ thị G và H.

Định nghĩa 9.

Ta gọi đồ thị con của đồ thị G = (V, E) là đồ thị H = (W, F), trong đó $W \subseteq V$ và $F \subseteq E$.

Trong trường hợp đồ thị là không liên thông, nó sẽ rã ra thành một số đồ thị con liên thông đôi một không có đỉnh chung. Những đồ thị con liên thông như vậy ta sẽ gọi là các *thành phần liên thông* của đồ thị.

Ví dụ 4. Đồ thị H trong hình 2 gồm 3 thành phần liên thông H_1, H_2, H_3.

Trong mạng máy tính có thể có những máy (Những kênh nối) mà sự hỏng hóc của nó sẽ ảnh hưởng đến việc trao đổi thông tin trong mạng. Các khái niệm tương ứng với tình huống này được đưa ra trong định nghĩa sau.

Định nghĩa 10.

Đỉnh v được gọi là đỉnh rẽ nhánh nếu việc loại bỏ v cùng với các cạnh liên thuộc với nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị. Cạnh e được gọi là cầu nếu việc loại bỏ nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị.

Ví dụ 5 Trong đồ thị G ở hình2, đỉnh d và e là đỉnh rẽ nhánh, còn các cạnh (d, g) và (e, f) là cầu.

Đối với đồ thị có hướng có hai khái niệm liên thông phụ thuộc vào việc ta có xét đến hướng trên các cung hay không.

Định nghĩa 11

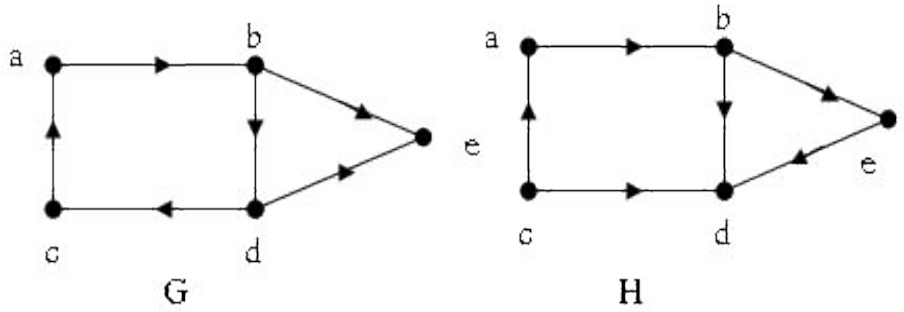
Đồ thị có hướng G = (V, A) được gọi là liên thông mạnh nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Định nghĩa 12

Đồ thị có hướng G = (V, A) được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là vô hướng liên thông.

Rõ ràng nếu đồ thị là liên thông mạnh thì nó cũng là liên thông yếu, nhưng điều ngược lại là không luôn đúng, như chỉ ra trong ví dụn dưới đây.

Ví dụ 6 Trong hình 7 đồ thị G là liên thông mạnh, còn H là liên thông yếu nhưng không là liên thông mạnh.



Hình 8 Đồ thị liên thông mạnh G và đồ thị liên thông yếu H.

Một câu hỏi đặt ra là khi nào có thể định hướng các cạnh của một đồ thị vô hướng liên thông để có thể thu được đồ thị có hướng liên thông mạnh? Ta sẽ gọi đồ thị như vậy là đồ thị định hướng được. Định lý dưới đây cho ta tiêu chuẩn nhận biết một đồ thị có là định hướng được hay không.

Định lý 1

Đồ thị vô hướng liên thông là định hướng được khi và chỉ khi mỗi cạnh của nó nằm trên ít nhất một chu trình.

Chứng minh.

Điều kiện cần. Giả sử (u,v) là một cạnh của một đồ thị. Từ sự tồn tại đường đi có hướng từ u đến v và ngược lại suy ra (u, v) phải nằm trên ít nhất một chu trình.

Điều kiện đủ. Thủ tục sau đây cho phép định hướng các cạnh của đồ thị để thu được đồ thị có hướng liên thông mạnh. Giả sử C là một chu trình nào đó trong đồ thị. Định hướng các cạnh trên chu trình này theo một hướng đi vòng theo nó. Nếu tất cả các cạnh của đồ thị là đã được định hướng thì kết thúc thủ tục. Ngược lại, chọn e là một cạnh chưa định hướng có chung đỉnh với ít nhất một trong số các cạnh đã định hướng. Theo giả thiết tìm được chu trình C’ chứa cạnh e. Định hướng các cạnh chưa được định hướng của C’ theo một hướng dọc theo chu trình này (không định hướng lại các cạnh đã có định hướng). Thủ tục trên sẽ được lặp lại cho đến khi tất cả các cạnh của đồ thị được định hướng. Khi đó ta thu được đồ thị có hướng liên thông mạnh.

Đồ thị vô hướng liên thông

Trong mục này chúng ta sẽ trình bày một số thuật ngữ cơ bản của lý thuyết đồ thị. Trước tiên, ta xét các thuật ngữ mô tả các đỉnh và cạnh của đồ thị vô hướng.

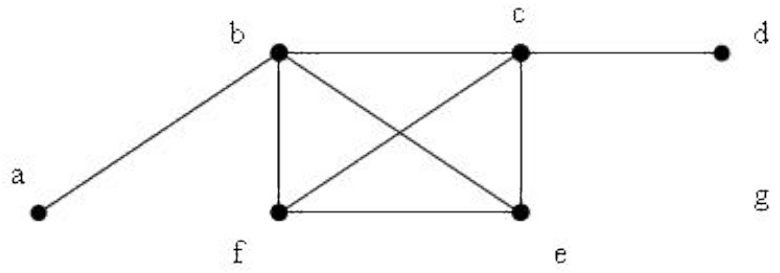
Định nghĩa 13

Hai đỉnh u và v của đồ thị vô hướng G được gọi là kề nhau nếu (u,v) là cạnh của đồ thị G. Nếu e = (u, v) là cạnh của đồ thị ta nói cạnh này là liên thuộc với hai đỉnh u và v, hoặc cũng nói là nối đỉnh u và đỉnh v, đồng thời các đỉnh u và v sẽ được gọi là các đỉnh đầu của cạnh (u, v).

Để có thể biết có vao nhiêu cạnh liên thuộc với một đỉnh, ta đưa vào định nghĩa sau

Định nghĩa 14

Ta gọi bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với nó và sẽ ký hiệu là deg(v).



Hình 8 Đồ thị vô hướng.

Ví dụ 7 Xét đồ thị cho trong hình 8, ta có

deg(a) = 1, deg(b) = 4, deg(c) = 4, deg(f) = 3,

deg(d) = 1, deg(e) = 3, deg(g) = 0

Đỉnh bậc 0 gọi là *đỉnh cô lập*. Đỉnh bậc 1 được gọi là đỉnh treo. Trong ví dụ trên đỉnh g là đỉnh cô lập, a và d là các đỉnh treo. Bậc của đỉnh có tính chất sau:

Định lý 2 *Giả sử G = (V, E) là đồ thị vô hướng với m cạnh. Khi đó tổng bậc của tất cả các đỉnh bằng hai lần số cung.*

Chứng minh. Rõ ràng mỗi cạnh e = (u, v) được tính một lần trong *deg(u)* và một lần trong *deg(v)*. Từ đó suy ra tổng tất cả các bậc của các đỉnh bằng hai lần số cạnh.

Ví dụ 8 Đồ thị với n đỉnh có bậc là 6 có bao nhiêu cạnh?

Giải: Theo định lý 2 ta có *2m = 6n*. Từ đó suy ra tổng các cạnh của đồ thị là 3n.

Hệ quả 3 *Trong đồ thị vô hướng, số đỉnh bậc lẻ (nghĩa là có bậc là số lẻ) là một số chẵn.*

Chứng minh. Thực vậy, gọi O và U tương ứng là tập đỉnh bậc lẻ và tập đỉnh bậc chẵn của đồ thị. Ta có

$2m = \sum_{v \in U} \deg(v) + \sum_{v \in O} \deg(v)$

Do *deg(v)* là chẵn với v là đỉnh trong U nên tổng thứ nhất ở trên là số chẵn. Từ đó suy ra tổng thứ hai (chính là tổng bậc của các đỉnh bậc lẻ) cũng phải là số chẵn, do tất cả các số hạng của nó là số lẻ, nên tổng này phải gồm một số chẵn các số hạng. Vì vậy, số đỉnh bậc lẻ phải là số chẵn.

Ta xét các thuật ngữ tương tự cho đồ thị vô hướng.

Đồ thị có hướng liên thông

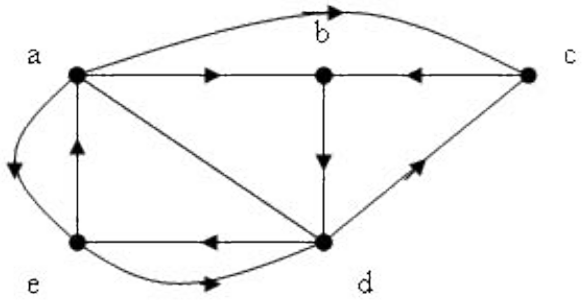
Định nghĩa 15

Nếu e = (u, v) là cung của đồ thị có hướng G thì ta nói hai đỉnh u và v là kề nhau, và nói cung (u, v) nối đỉnh u với đỉnh v hoặc cũng nói cung này là đi ra khỏi đỉnh u và vào đỉnh v. Đỉnh u(v) sẽ được gị là đỉnh đầu (cuối) của cung (u,v).

Tương tự như khái niệm bậc, đối với đồ thị có hướng ta có khái niệm bán bậc ra và bán bậc vào của một đỉnh.

Định nghĩa 16

Ta gọi bán bậc ra (bán bậc vào) của đỉnh v trong đồ thị có hướng là số cung của đồ thị đi ra khỏi nó (đi vào nó) và ký hiệu là deg + (v) (deg - (v))



Hình 9 Đồ thị có hướng.

Ví dụ 9 Xét đồ thị cho trong hình 9. Ta có

deg - (a)=1, deg - (b)=2, deg - (c)=2, deg - (d)=2, deg - (e) = 2.

deg + (a)=3, deg + (b)=1, deg + (c)=1, deg + (d)=2, deg + (e)=2.

Do mỗi cung (u, v) sẽ được tính một lần trong bán bậc vào của đỉnh v và một lần trong bán bậc ra của đỉnh u nên ta có:

Định lý 4. *Giả sử G = (V, E) là đồ thị có hướng. Khi đó*

2m =

Rất nhiều tính chất của đồ thị có hướng không phụ thuộc vào hướng trên các cung của nó. Vì vậy, trong nhiều trường hợp sẽ thuận tiện hơn nếu ta bỏ qua hướng trên các cung của đồ thị. Đồ thị vô hướng thu được bằng cách bỏ qua hướng trên các cung được gọi là *đồ thị vô hướng tương ứng* với đồ thị có hướng đã cho.

Bài 2: Các khái niệm cơ bản của lý thuyết đồ thị. Đồ thị Euler

Các khái niệm cơ bản của Lý thuyết đồ thị. Đồ thị Euler (phan 1)

Các khái niệm cơ bản của Lý thuyết đồ thị.Đồ thị Euler

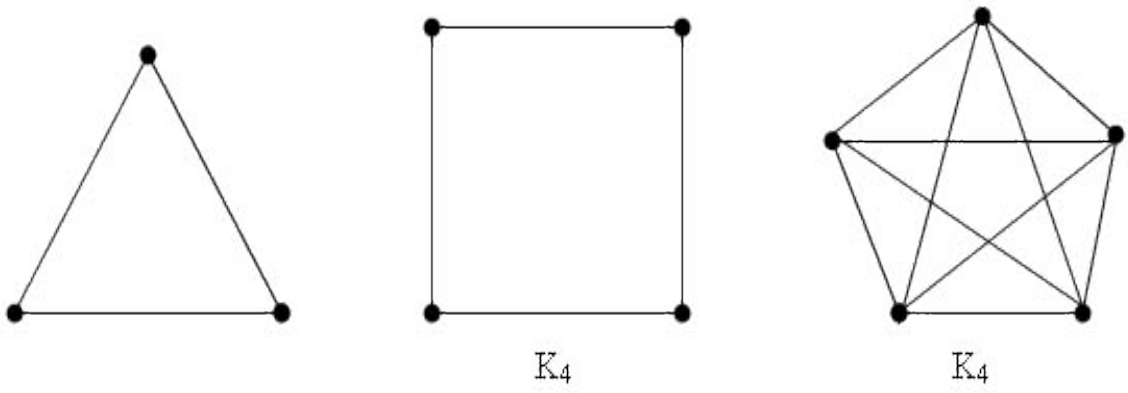
Các loại đồ thị đặc biệt

Trong mục này ta xét một số đơn đồ thị vô hướng dạng đặc biệt xuất hiện trong nhiều vấn đề ứng dụng thực tế.

Đồ thị đầy đủ.

Đồ thị đầy đủ n đỉnh, ký hiệu bởi K_n, là đơn đồ thị vô hướngmà giữa hai đỉnh bất kỳ của nó luôn có cạnh nối.

Các đồ thị K_3, K_4, K_5 cho trong hình dưới đây.



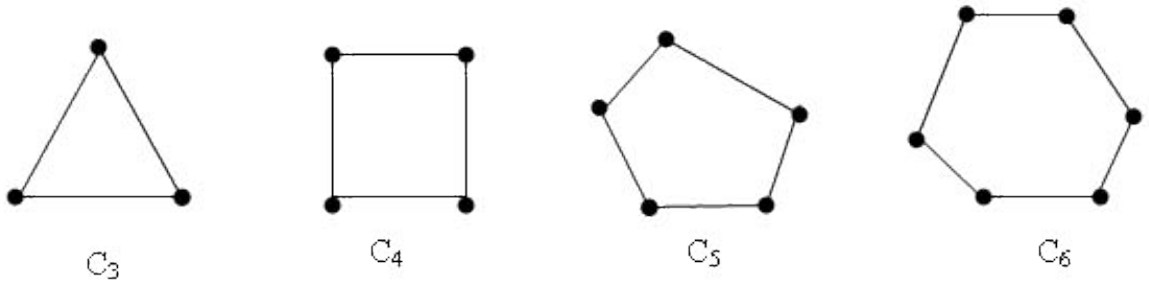
Hình 1 Đồ thị đầy đủ.

Đồ thị đầy đủ K_n có tất cả n(n-1)/2 cạnh, nó là đơn đồ thị có nhiều cạnh nhất.

Đồ thị vòng.

Đồ thị vòng C_n, $n \geq 3$. gồm n đỉnh v_1, v_2,. . . .v_n và các cạnh (v_1,v_2), (v_2,v_3) . . . (v_{n-1},v_n), (v_n,v_1).

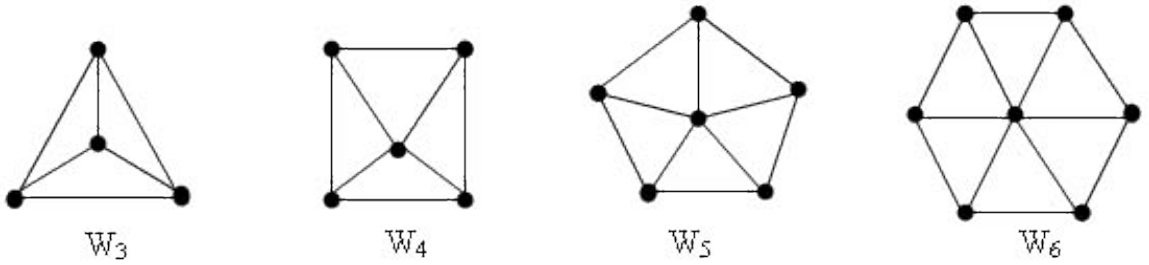
Đồ thị vòng C_3, C_4, C_5, C_6 cho trong hình 2



Hình 2 Đồ thị vòng C 3 , C 4 , C 5 , C 6 .

Đồ thị bánh xe.

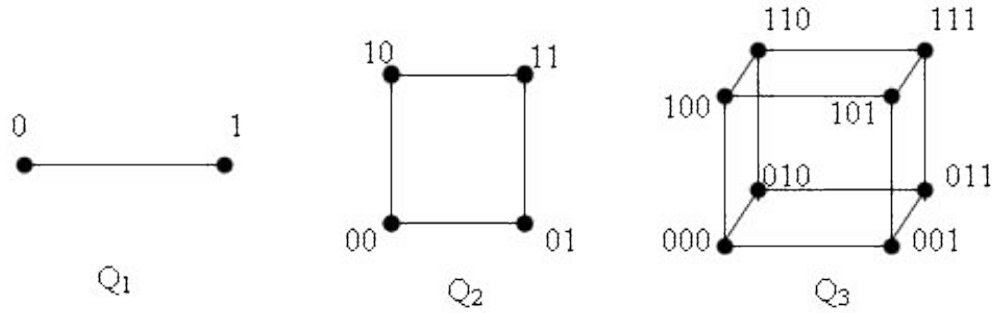
Đồ thị W_n thu được từ C_n bằng cách bổ sung vào một đỉnh mới nối với tất cả các đỉnh của C_n (xem hình 3).



Hình 3 Đồ thị bánh xe W 3 , W 4 , W 5 , W 6 .

Đồ thị lập phương.

Đồ thị lập phương n đỉnh Q_n là đồ thị với các đỉnh biểu diễn 2^n xâu nhị phân độ dài n. Hai đỉnh của nó gọi là kề nhau nếu như hai xâu nhị phân tương ứng chỉ khác nhau 1 bit. Hình 4 cho thấy Q_n với n=1,2,3.



Hình 4 Đồ thị lập phương Q 1 , Q 2 , Q 3 .

Đồ thị hai phía.

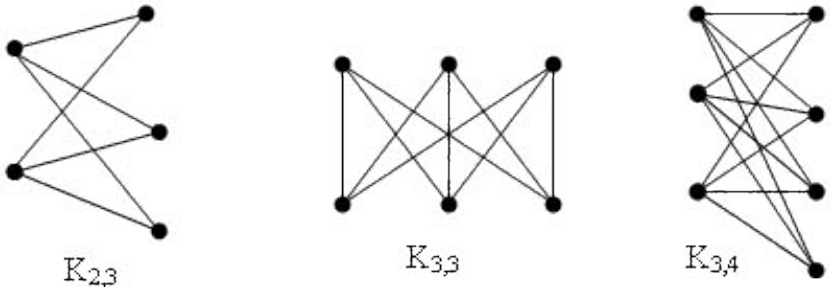
Đơn đồ thị G = (V, E) được gọi là hai phía nếu như tập đỉnh V của nó có thể phân hoạch thành hai tập X và Y sao cho mỗi cạnh của đồ thị chỉ nối một đỉnh nào đó trong X với một đỉnh nào đó trong Y. Khi đó ta sẽ sử dụng ký hiệu G = (X?Y, E) để chỉ đồ thị hai phía với tập đỉnh X?Y.

Định lý sau đây cho phép nhận biết một đơn đồ thị có phải là hai phía hay không.

Định lý 1 *Đơn đồ thị là đồ thị hai phía khi và chỉ khi nó không chứa chu trình độ dài lẻ.*

Để kiểm tra xem một đồ thị liên thông có phải là hai phía hay không có thể áp dụng thủ tục sau. Cho v là một đỉnh bất kỳ của đồ thị. Đặt X={v}, còn Y là tập các đỉnh kề của v. Khi đó các đỉnh kề của các đỉnh trong Y phải thuộc vào X. Ký hiệu tập các đỉnh như vậy là T. Vì thế nếu phát hiện T ? Y ≠ Ø thì đồ thị không phải là hai phía, kết thúc. ngược lại, đặt X = X ? T. Tiếp tục xét như vậy đối với T' là tập các đỉnh kề của T,. ..

Đồ thị hai phía G=(X ? Y, E) với |X|= m, |Y| = n được gọi là đồ thị hai phía đầy đủ và ký hiệu là $K_{2,3}$, $K_{3,3}$, $K_{3,4}$ được cho trong hình 5.



Hình 5 : Đồ thị hai phía.

Đồ thị phẳng.

Đồ thị được gọi là đồ thị phẳng nếu ta có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh. Cách vẽ như vậy sẽ được gọi là biểu diễn phẳng của đồ thị.

Ví dụ đồ thị K_4 là phẳng, vì có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh (xem hình 6).

Hình 6: Đồ thị K_4 là đồ thị phẳng.

Một điều đáng lưu ý nếu đồ thị là phẳng thì luôn có thể vẽ nó trên mặt phẳng với các cạnh nối là các đoạn thẳng không cắt nhau ngoài ở đỉnh (ví dụ xem cách vẽ K_4 trong hình 6).

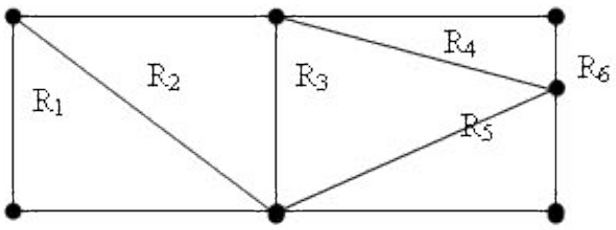
Để nhận biết xem một đồ thị có phải là đồ thị phẳng có thể sử dụng định lý Kuratovski, mà để phát biểu nó ta cần một số khái niệm sau: Ta gọi một phép chia cạnh (u,v) của đồ thị là việc loại bỏ cạnh này khỏi đồ thị và thêm vào đồ thị một đỉnh mới w cùng với hai cạnh (u,w), (w, u) . Hai đồ thị G(V,E) và H=(W,F) được gọi là đồng cấu nếu chúng có thể thu được từ cùng một đồ thị nào đó nhờ phép chia cạnh.

Định lý 2 (Kuratovski). *Đồ thị là phẳng khi và chỉ khi nó không chứa đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .*

Trong trường hợp riêng, đồ thị $K_{3,3}$ hoặc K_5 không phải là đồ thị phẳng. Bài toán về tính phẳng của đồ thị $K_{3,3}$ là bài toán đố nổi tiếng về ba căn hộ và ba hệ thống cung cấp năng lượng cho chúng: Cần xây dựng hệ thống đường cung cấp năng lượng với mỗi một căn hộ nói trên sao cho chúng không cắt nhau.

Đồ thị phẳng còn tìm được những ứng dụng quan trọng trong công nghệ chế tạo mạch in.

Biểu diễn phẳng của đồ thị sẽ chia mặt phẳng ra thành các miền, trong đó có thể có cả miền không bị chặng. Ví dụ, biểu diễn phẳng của đồ thị cho trong hình 7 chia mặt phẳng ra thành 6 miền $R_1, R_2, \ldots R_6$.



Hình 7: Các miền tương ứng với biểu diễn phẳng của đồ thị.

Euler đã chứng minh được rằng các cách biểu diễn phẳng khác nhau của một đồ thị đều chia mặt phẳng ra thành cùng một số miền. Để chứng minh điều đó, Euler đã tìm được mối liên hệ giữa số miền, số đỉnh của đồ thị và số cạnh của đồ thị phẳng sau đây.

Định lý 3 (Công thức Euler). *Giả sử G là đồ thị phẳng liên thông với n đỉnh, m cạnh. Gọi r là số miền của mặt phẳng bị chia bởi biểu diễn phẳng của G. Khi đó*

r = m-n + 2

Có thể chứng minh định lý bằng qui nạp. Xét Ví dụ minh hoạ cho áp dụng công thức Euler.

Ví dụ 1 Cho G là đồ thị phẳng liên thông với 20 đỉnh, mỗi đỉnh đều có bậc là 3. Hỏi mặt phẳng bị chia làm bao nhiêu phần bởi biểu diễn phẳng của đồ thị G?

Giải. Do mỗi đỉnh của đồ thị đều có bậc là 3, nên tổng bậc của các đỉnh là 3x20=60. Từ đó suy ra số cạnh của đồ thị m=60/20=30. Vì vậy, theo công thức Euler, số miền cần tìm là

r = 30-20+2=12.

Các phương pháp lưu trữ đồ thị

Để lưu trữ đồ thị và thực hiện các thuật toán khác nhau với đồ thị trên máy tính cần phải tìm những cấu trúc dữ liệu thích hợp để mô tả đồ thị. Việc chọn cấu trúc dữ liệu nào để biểu diễn đồ thị có tác động rất lớn đến hiệu quả của thuật toán. Vì vậy, việc chọn lựa cấu trúc dữ liệu để biểu diễn đồ thị phụ thuộc vào từng tình huống cụ thể (bài toán và thuật toán cụ thể). Trong mục này chúng ta sẽ xét một số phương pháp cơ bản được sử dụng để biểu diễn đồ thị trên máy tính, đồng thời cũng phân tích một cách ngắn gọn những ưu điểm cũng như những nhược điểm của chúng.

Ma trận kề. Ma trận trọng số

Xét đơn đồ thị vô hướng G =(V,E), với tập đỉnh V={1, 2,. . . ,n} , tập cạnh E= {e_1, e_2, . . . ,e_m }. Ta gọi ma trận kề của đồ thị G là ma trận.

A=($a_{i,j}$: i, j = 1, 2, . . . ,n)

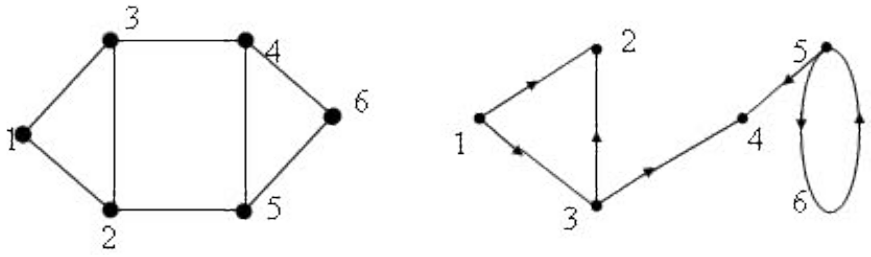
Với các phần tử được xác định theo qui tắc sau đây:

$a_{i, j}$ = 0, nếu (i, j) ∈ E và

$a_{i,j}$ = 1 , nếu (i, j) ∈ E, i, j =1, 2,. . ., n.

Ví dụ 2 Ma trận trận kề của đồ thị vô hướng G cho trong hình 2.8 là:

	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	1
6	0	0	0	1	1	0



Hình 2.8 Đồ thị vô hướng G và Đồ thị có hướng G 1 .

Các tính chất của ma trận kề:

1) Rõ ràng ma trận kề của đồ thị vô hướng là ma trận đối xứng, tức là

a[i,j]=a[j,i], i,j =1,2,. . .,n. Ngược lại, mỗi (0,1)-ma trận đối xứng cấp n sẽ tương ứng, chính xác đến cách đánh số đỉnh (còn nói là: chính xác đến đẳng cấu), với một đơn đồ thị vô hướng n đỉnh.

2) Tổng các phần từ trên dòng i (cột j) của ma trận kề chính bằng bậc của đỉnh i (đỉnh j).

3) nếu ký hiệu a_{ij}^p , i,j=1, 2,. . . ,n là phần tử của ma trận A^p =A.A. . .A p thừa số. Khi đó a_{ij}^p , i,j=1, 2,. . . ,n cho ta số đường đi khác nhau từ đỉnh i đến đỉnh j qua p-1 đỉnh trung gian.

Ma trận kề của đồ thị có hướng được định nghĩa một cách hoàn toàn tương tự.

Ví dụ 3 Đồ thị có hướng G_1 cho trong hình 8 có ma trận kề là ma trận sau:

	1	2	3	4	5	6

1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

Lưu ý rằng ma trận kề của đồ thị có hướng không phải là ma trận đối xứng.

Chú ý: Trên đây chúng ta chỉ xét đơn đồ thị. Ma trận kề của đa đồ thị có thể xây dựng hoàn toàn tương tự, chỉ khác là thay vì ghi 1 vào vị trí a[i,j] nếu (i,j) là cạnh của đồ thị, chúng ta sẽ ghi k là số cạnh nối hai đỉnh i, j.

Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, mỗi cạnh e=(u,v) của đồ thị được gán với một con số c(e) (còn viết là c(u,v) gọi là trọng số của cạnh e. Đồ thị trong trường hợp như vậy được gọi là đồ thị có trọng số. Trong trường hợp đồ thị có trọng số, thay vì mà trận kề, để biểu diễn đồ thị ta sử dụng ma trận trọng số.

C= {c[i,j], i, j = 1, 2,. . ., n}

với c[i,j]=c(i,j) nếu (i,j) ∈ E và c[i,j]= δ nếu (i, j) ∉ E

trong đó số δ , tuỳ từng trường hợp cụ thể, có thể được đặt bằng một trong các giá trị sau : 0, + , -.

Ưu điểm lớn nhất của phương pháp biểu diễn đồ thị bằng ma trận kề (hoặc ma trận trọng số) là để trả lời câu hỏi: Hai đỉnh u, v có kề nhau trên đồ thị hay không, chúng ta chỉ phải thực hiện một phép so sánh. nhược điểm lớn nhất của phương pháp này là : không phụ thuộc vào số cạnh của đồ thị, ta luôn phải sử dụng n^2 đơn vị bộ nhớ để lưu trữ ma trận kề của nó.

Danh sách cạnh (cung)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh m thoả mãn bất dẳng thức: m < 6n) người ta thường dùng cách biểu diễn đồ thị dưới dạng danh sách cạnh.

Trong cách biểu diễn đồ thị bởi danh sách cạnh (cung) chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Một cạnh (cung) e = (x,y) của đồ thị sẽ tương ứng với hai biến Dau[e], Cuoi[e]. như vậy, để lưu trữ đồ thị ta cần sử dụng 2m đơn vị bộ nhớ. Nhược điểm của cách biểu diễn này là để xác định những đỉnh nào

của đồ thị là kề với một đỉnh cho trước chúng ta phải làm cỡ m phép so sánh (khi duyệt qua danh sách tất cả các cạnh của đồ thị).

Chú ý: Trong trường hợp đồ thị có trọng số ta cần thêm m đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

Ví dụ 2.4 Danh sách cạnh (cung) của đồ thị G (G_1) cho trong hình 2.8 là:

Dau	Cuoi		Dau	Cuoi
1	2		1	2
1	3		1	3
1	5		3	2
2	3		3	4
2	5		5	4
3	4		5	6
4	5		6	5
4	6			
5	6			

Danh sách cạnh của G Danh sánh cung của G_1

Danh sách kề

Trong rất nhiều vấn đề ứng dụng của lý thuyết đồ thị, cách biểu diễn đồ thị dưới dạng danh sách kề là cách biểu diễn thích hợp nhất được sử dụng.

Trong cách biểu diễn này, với mỗi đỉnh v của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó, mà ta sẽ ký hiệu là

$Ke(v)= \{ u \in V: (v,u) \in E \}$

Khi đó vòng lặp thực hiện với mỗi một phần tử trong danh sách này theo thứ tự các phần tử được sắp xếp trong nó sẽ được viết như sau:

for $u \in Ke(v)$ do. . .

Chẳng hạn, trên PASCAL có thể mô tả danh sách này như sau (Gọi là cấu trúc **Forward Star**):

```
Const

m=1000; {m-so canh}

n= 100; {n-so dinh}

var

Ke: array[1..m] of integer;

Tro: array[1..n+1] of integer;
```

Trong đó Tro[i] ghi nhận vị trí bắt đầu của danh sách kề của đỉnh i, i=1, 2,. . .,n, Tro[n+1]=2m+1.

Khi đó dòng lệnh qui ước

for u ∈ Ke(v) do

```
begin . . . end.
```

Có thể thay thế bởi cấu trúc lệnh cụ thể trên PASCAL như sau

```
For i:=Tro[v] to Tro[v+1]-1 do

Begin

U:=Ke[i];

. . . .

End;
```

Trong rất nhiều thuật toán làm việc với đồ thị chúng ta thường xuyên phải thực hiện các thao tác: Thêm hoặc bớt một số cạnh. Trong trường hợp này cấu trúc dữ liệu dùng ở trên là không thuận tiện. Khi đó nên chuyển sang sử dụng danh sách kề liên kết (Linked Adjancency List) như mô tả trong chương trình nhập danh sách kề của đồ thị từ bàn phím và đưa danh sách đó ra màn hình sau đây:

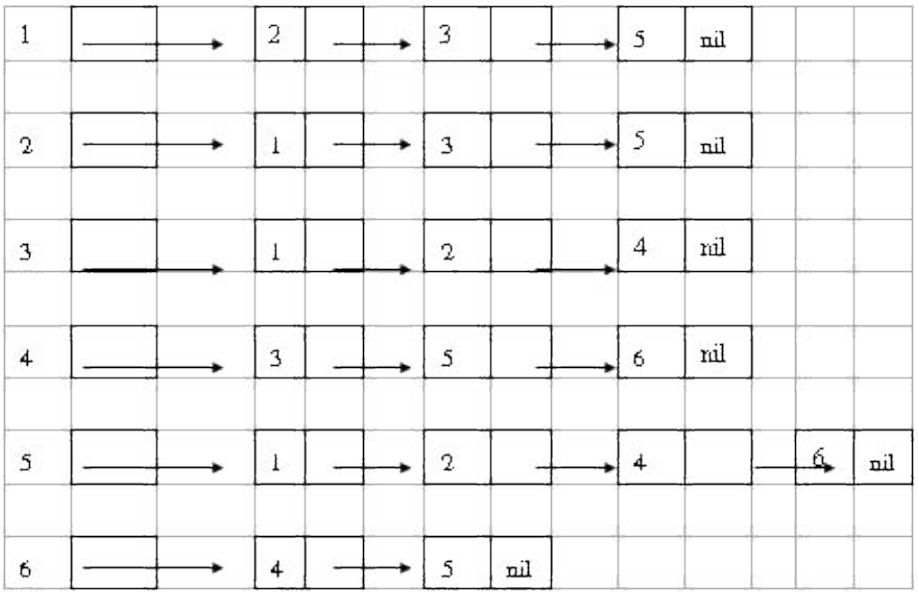
```
Program AdjList;

Const

maxV=100;
```

```
Type
link=^node;
node=record
v:integer;
next:link;
End;
Var
j,x,y,m,n,u,v:integer;
t:link;
Ke:array[1. .Vmax] of link;
Begin
Write('Cho so canh va dinh cua do thi:'); readln(m,n);
(*Khoi tao*)
for j:=1 to n do Ke[j]:=nil;
for j:=1 to m do
begin
write('Cho dinh dau va cuoi cua canh ',j,':');
readln(x,y);
new(t); t^.v:=x, t^.next:=Ke[y]; Ke[y]:=t;
new(t); t^.v:=y, t^.next:=Ke[x]; Ke[x]:=t;
end;
writeln('Danh sach ke cua cac dinh cua do thi:');
```

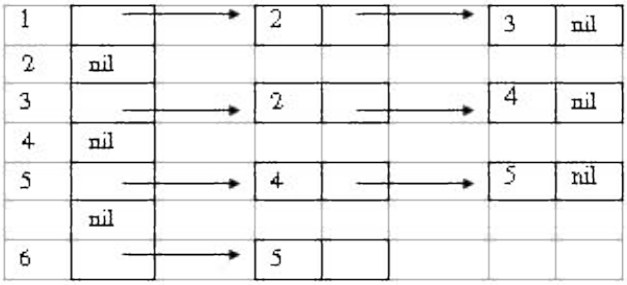
```
for J:=1 to m do
begin
writeln(‘Danh sachcac dinh ke cua dinh ‘,j,’:’);
t:=Ke[j];
while t^.next<>nil do
begin
write(t^.v:4);
t:=t^.next;
end;
end;
readln;
End.
```

Ví dụ 5 Danh sách kề của các đồ thị trong hình 2.8 được mô tả trong hình sau:

Đỉnh đầu



Đỉnh đầu



Hình 2.9 Danh sách kề của đồ thị vô hướng G và có hướng G 1 cho trong hình 2.8.

Để ý rằng trong cách biểu diễn này chúng ta cần phải sử dụng cỡ m+n đơn vị bộ nhớ.

Trong các thuật toán mô tả ở các phần tiếp theo hai cấu trúc danh sách kề và ma trận trọng số được sử dụng thường xuyên.

Các khái niệm cơ bản của Lý thuyết đồ thị. Đồ thị Euler (phan 2)

Các khái niệm cơ bản của Lý thuyết đồ thị. Đồ thị Euler

Đồ thị eule

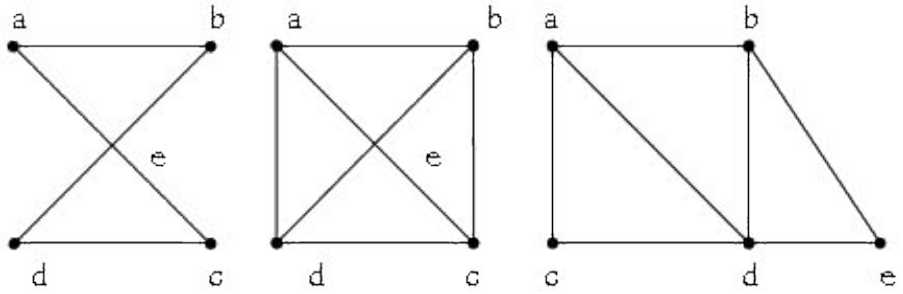
Định nghĩa.

Chu trình đơn trong đồ thị G đi qua mỗi cạnh của nó một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler, và gọi là đồ thị nửa Euler nếu nó có đường đi Euler.

Rõ ràng mọi đồ thị Euler luôn là nửa Euler, nhưng điều ngược lại không luôn đúng.

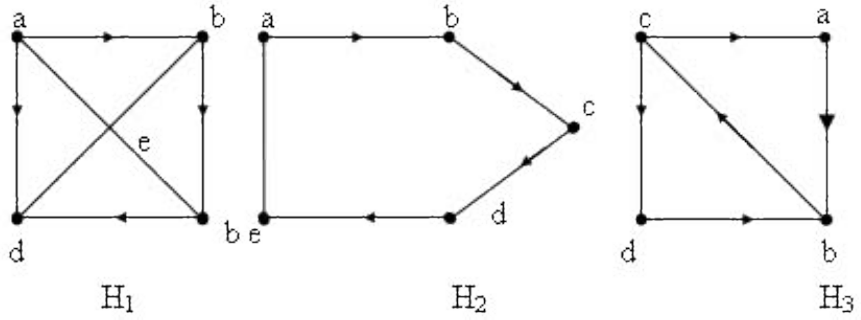
Các thí dụ

Ví dụ 6 Đồ thị G_1 trong hình 1 là đồ thị Euler vì nó có chu trình Euler a, e, c, d, e, b, a. Đồ thị G_3 không có chu trình Euler nhưng nó có đường đi Euler a, c, d, e, b, d, a, b, vì thế G_3 là đồ thị cửa Euler. Đồ thị G_2 không có chu trình cũng như đường đi Euler.



Hình 1 : Đồ thị G 1 , G 2 , G 3 .

Ví dụ 7 : Đồ thị H_2 trong hình 1 là đồ thị Euler vì nó có chu trình Euler a, b, c, d, e, a. Đồ thị H_3 không có chu trình Euler nhưng nó có đường đi Euler c, a, b, c, d, b vì thế H_3 là đồ thị nửa Euler. Đồ thị H_1 không có chu trình cũng như đường đi Euler.



Hình 1: Đồ thị H 1 , H 2 , H 3 .

Điều kiện cần và đủ để một đồ thị là một đồ thị Euler được Euler tìm ra vào năm 1736 khi ông giải quyết bài toán hóc búa nổi tiếng thế giới thời đó về bảy cái cầu ở thành phố Konigsberg và đây là định lý đầu tiên của lý thuyết đồ thị.

Định lý Euler và thuật toán Flor

Định lý 4 (Euler). *Đồ thị vô hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều có bậc chẵn.*

Để chứng minh định lý trước hết ta chứng minh bổ đề:

Bổ đề 5 Nếu bậc của mỗi đỉnh của đồ thị G không nhỏ hơn 2 thì G chứa chu trình.

Chứng minh.

Nếu G có cạnh lặp thì khẳng định của bổ đề là hiển nhiên. Vì vậy giả sử G là đơn đồ thị. Gọi v là một đỉnh nào đó của G. Ta sẽ xây dựng theo qui nạp đường đi

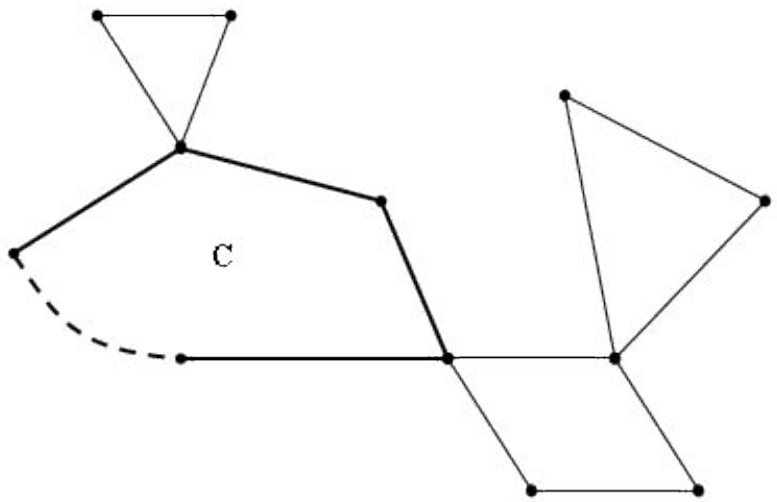
$v \rightarrow v_1 \rightarrow v_2 \rightarrow . . .$

trong đó v_1 là đỉnh kề với v, còn với $i \geq 1$ chọn $v_{i+1} \neq v_{i-1}$ (có thể chọn v_{i+1} như vậy là vì $deg(v_i) \geq 2$). Do tập đỉnh của G là hữu hạn , nên sau một số hữu hạn bước ta phải quay lại một đỉnh đã xuất hiện trước đó. Gọi đỉnh đầu tiên như thế là v_k. Khi đó, đoạn của đường đi xây dựng nằm giữa hai đỉnh v_k là 1 chu trình cần tìm.

Chứng minh định lý:

Cần. Giả sử G là đồ thị Euler tức là tồn tại chu trình Euler P trong G. Khi đó cứ mỗi lần chu trình P đi qua một đỉnh nào đó của G bậc của đỉnh đó tăng lên 2. mặt khác mỗi cạnh của đồ thị xuất hiện trong P đúng một lần, suy ra mỗi đỉnh của đồ thị điều có bậc chẵn.

Đủ.Quy nạp theo số đỉnh và số cạnh của G. Do G liên thông và deg(v) là số chẵn nên bậc của mỗi đỉnh của nó không nhỏ hơn 2. Từ đó theo bổ đề G phải chứa chu trình C. Nếu C đi qua tất cả các cạnh của G thì nó chính là chu trình Euler. Giả sử C không đi qua tất cả các cạnh của G. Khi đó loại bỏ khỏi G tất cả các cạnh thuộc C ta thu được một đồ thị mới H vẫn có bậc là chẵn. Theo giả thiết qui nạp, trong mỗi thành phần liên thông của H điều tìm được chu trình Euler. Do G là liên thông nên trong mỗi thành phần của H có ít nhất một đỉnh chung với chu trình C. Vì vậy, ta có thể xây dựng chu trình Euler trong G như sau: bắt đầu từ một đỉnh nào đó của chu trình C, đi theo các cạnh của C chừng nào chưa gặp phải đỉnh không cô lập của H. Nếu gặp phải đỉnh như vậy ta sẽ đi theo chu trình Euler của thành phần liên thông của H chứa đỉnh đó. Sau đó lại tiếp tục đi theo cạnh của C cho đến khi gặp phải đỉnh không cô lập của H thì lại theo chu trình Euler của thành phần liên thông tương ứng trong H v.v… (xem hình 12). Quá trình sẽ kết thúc khi ta trở về đỉnh xuất phát , tức là thu được chu trình đi qua mỗi cạnh của đồ thị đúng một lần.



Hình 12 Minh hoạ cho chứng minh định lý 4.

Hệ quả 6 Đồ thị vô hướng liên thông G là nửa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

Chứng minh. Thực vậy , nếu G có không quá 2 đỉnh bậc lẻ thì số đỉnh bậc lẻ của nó chỉ có thể là 0 hoặc 2. Nếu G không có đỉnh bậc lẻ thì theo định lý 4, nó là đồ thị Euler. Giả sử G có 2 đỉnh bậc lẻ là u và v. Gọi H là đồ thị thu được từ G bằng cách thêm vào G một đỉnh mới w và hai cạnh (w,u) và(w,v). Khi đó tất cả các đỉnh của H điều có bậc chẵn, vì thế theo định lý 4, nó có chu trình Euler C. Xoá bỏ khỏi chu trình này đỉnh w và hai cạnh kề nó ta thu được đường đi Euler trong đồ thị G.

Giả sử G là đồ thị Euler, từ chứng minh định lý ta có thủ tục sau để tìm chu trình Euler trong G.

```
Procedure Euler_Cycle;
Begin
STACK:= Ø ; CE:= Ø ;
Chon u la mot dinh nao do cua do thi;
STACK ? u;
While STACK<> Ø do
Begin
X:=top(STACK); (* x la phan tu dau STACK)
If Ke(x)<> Ø then
Begin
Y:=dinh dau tien trong danh sach Ke(x);
STACK ? y;
(* loai bo canh (x,y) khoi do thi *)
Ke(x):=Ke(x)\ { y} ;
Ke(y):=Ke(y)\{ x} ;
End
Else
Begin
x ⇐ STACK; CE ⇐ x;
End;
End;
End;
```

Giả sử G là đồ thị Euler, thuật toán đơn giản sau đây cho phép xác định chu trình Euler khi làm bằng tay.

Thuật toán Flor

Xuất phát từ một đỉnh u nào đó của G ta đi theo các cạnh của nó một cách tuỳ ý chỉ cần tuân thủ 2 qui tắc sau:

(1) Xoá bỏ cạnh đã đi qua đồng thời xoá bỏ cả những đỉnh cô lập tạo thành.

(2) Ở mỗi bước ta chỉ đi qua cầu khi không còn cách lựa chon nào khác.

Chứng minh tính đúng đắn của thuật toán.

Trước tiên ta chỉ ra rằng thủ tục trên có thể thực hiện ở mỗi bước. Giả sử ta đi đến một đỉnh v nào đó, khi đó nếu $v \neq u$ thì đồ thị con còn lại H là liên thông và chứa đúng hai đỉnh bậc lẻ là v và u. Theo hệ quả trong H có đường đi Euler P từ v tới u. Do việc xoá bỏ cạnh đầu tiên của đường đi P không làm mất tính liên thông của H, từ đó suy ra thủ tục có thể thực hiện ở mỗi bước. Nếu v=u thì lập luận ở trên sẽ vẫn đúng chừng nào vẫn còn cạnh kề với u.

Như vậy chỉ còn phải chỉ ra thủ tục trên dẫn đến đường đi Euler. Thực vậy trong G không thể còn cạnh chưa đi qua khi mà ta sử dụng cạnh cuối cùng kề với u (trong trường hợp ngược lại, việc loại bỏ một cạnh nào đó kề với một trong số những cạnh còn lại chưa đi qua sẽ dẫn đến một đồ thị không liên thông, và điều đó là mâu thuẫn với giả thiết ii).

Chứng minh tương tự như trong định lý 1 ta thu được kết quả sau đây cho đồ thị có hướng.

Định lý 7 Đồ thị có hướng liên thông mạnh là đồ thị Euler khi và chỉ khi

$Deg^{+}(v) = deg^{-}(v), \forall v \in V.$

Bài 3: Đồ thị hamilton

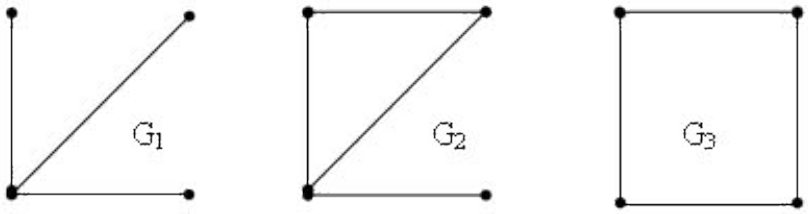
Đồ thị hamilton

Định nghĩa

Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Chu trình bắt đầu từ một đỉnh v nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần rồi quay trở về v được gọi là chu trình Hamilton. Đồ thị G được gọi là đồ thị Hamilton nếu nó chứa chu trình Hamilton và gọi là đồ thị nữa Hamilton nếu nó có đường đi Hamilton.

Rõ ràng đồ thị Hamilton là nửa Hamilton, nhưng điều ngược lại không còn đúng.

Ví dụ 1. Trong hình 1: G_3 là Hamilton, G_2 là nửa Hamilton còn G_1 không là nửa Hamilton.



Hình 1 Đồ thị Hamilton G 3 , nửa Hamilton G 2 , và G 1 .

Cho đến nay việc tìm một tiêu chuẩn nhận biết đồ thị Hamilton vẫn còn là mở, mặc dù đây là một vấn đề trung tâm của lý thuyết đồ thị. Hơn thế nữa, cho đến nay cũng chưa có thuật toán hiệu quả để kiểm tra một đồ thị có là Hamilton hay không. Các kết quả thu được phần lớn là điều kiện đủ để một đồ thị là đồ thị Hamilton. Phần lớn chúng điều có dạng "nếu G có số cạnh đủ lớn thì G là Hamilton". Một kết quả như vậy được phát biểu trong định lý sau đây.

Định lý và thuật toán liệt kê tất cả các chu trình Hamilton.

Định lý 1 (Dirak 1952)

Đơn đồ thị vô hướng G với n>2 đỉnh, mỗi đỉnh có bậc không nhỏ hơn n/2 là đồ thị Hamilton.

Chứng minh:

Thêm vào đồ thị G k đỉnh mới và nối chúng với tất cả các đỉnh của G. giả sử k là số nhỏ nhất các đỉnh cần thêm vào để cho đồ thị thu được G' là đồ thị Hamilton. Ta sẽ chỉ ra rằng k=0. Thực vậy, giả sử ngược lại là k >0. Ký hiệu

v, p, w, . . ., v là chu trình Hamilton trong G', trong đó v, w là đỉnh của G còn p là một trong số các đỉnh mới. Khi đó w không kề với v vì nếu ngược lại, ta không cần sử dụng p và điều đó là mâu thuẫn với giả thiết k nhỏ nhất. Hơn thế nữa đỉnh (w' chẳng hạn) kề với w không thể đi liền sau đỉnh v' (kề với v) vì rằng khi đó có thể thay

$$v \rightarrow p \rightarrow w \rightarrow \ldots \rightarrow v' \rightarrow w' \rightarrow \ldots \rightarrow v$$

bởi $v \rightarrow v' \rightarrow \ldots \rightarrow w \rightarrow w' \rightarrow \ldots \rightarrow v$ bằng cách đảo ngược đoạn của chu trình nằm giữa w và v'. Từ đó suy ra là số đỉnh của đồ thị G' không kề với w là không nhỏ hơn số đỉnh kề với v (tức là ít nhất cũng là bằng n/2+k), đồng thời số đỉnh của G' kề với w ít ra là phải bằng n/2+k. Do không có đỉnh nào của G' vừa không kề, lại vừa kề với w, cho nên tổng số đỉnh của đồ thị G' (G' có n+k đỉnh) không ít hơn n+2k. Mâu thuẫn thu được đã chứng minh định lý.

Định lý sau là tổng quát hoá của định lý Dirak cho đồ thị có hướng:

Định lý 2 Giả sử G là đồ có hướng liên thông với n đỉnh.

Nếu $\deg^+(v) \geq n/2$, $\deg^-(v) \geq n/2$, $\forall\, \forall v$ thì G là Hamilton.

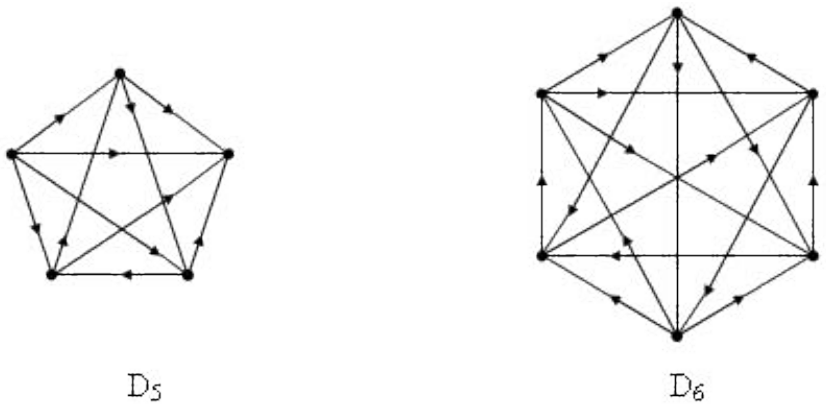
Có một số dạng đồ thị mà ta có thể biết khi nào là đồ thị Hamilton. Một ví dụ như vậy là đồ thị đấu loại. Đồ thị đấu loại là đồ thị có hướng mà trong đó hai đỉnh bất kỳ của nó được nối với nhau bởi đúng một cung. Tên đấu loại xuất hiện như vậy vì đồ thị như vậy có thể dùng để biểu diễn kết quả thi đấu bóng chuyền, bóng bàn hay bất cứ một trò chơi nào mà không cho phép hoà. Ta có định lý sau:

Định lý 3

i) Mọi đồ thị đấu loại là nửa Hamilton.

ii) Mọi đồ thị đấu loại liên thông mạnh là Hamilton.

Ví dụ 2 Đồ thị đấu loại D_5, D_6 được cho trong hình 3.2



Hình 2 Đồ thị đấu loại D 5 , đấu loại liên thông mạnh D 6 .

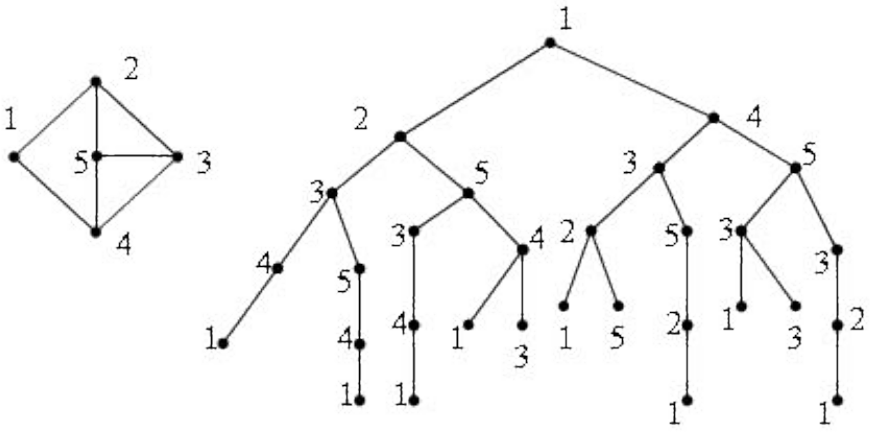
Thuật toán liệt kê tất cả các chu trình Hamilton của đồ thị

Thuật toán sau đây được xây dựng dựa trên cơ sở thuật toán quay lui cho phép liệt kê tất cả các chu trình Hamilton của đồ thị.

```
Procedure Hamilton(k);

(* liet ke cac chu trinh Hamilton thu duoc bang viec phat trien day dinh (X[1],. . . , X[k-1]) cua do thi G=(V,E) cho boi danh sach ke: Ke(v), v ∈ V *)

begin

for y ∈ Ke(X[k-1]) do

if (k =N+1) and (y=v 0 ) then Ghinhan(X[1],. . . , X[n], v 0 )

else

if Chuaxet[y] then

begin

X[k]:=y;

Chuaxet[y]:=false;

Hamilton(k+1);

Chuaxet[y]:=true;

end;
```

```
end;

(* Main program*)

begin

for v ∈ V do Chuaxet[v]:=true;

X[1]:=0; (* v0 la mot dinh nao do cua do thi *)

Chuaxet[v0]:=false;

Hamilton(2);

end.
```

Ví dụ 3 : Hình 3 dưới đây mô tả cây tìm kiếm theo thuật toán vừa mô tả.



Hình 3: Đồ thị và cây liệt kê chu trình Hamilton của nó theo thuật toán quay lui.

Trong trường hợp đồ thị có không quá nhiều cạnh thuật toán trên có thể sử dụng để kiểm tra đồ thị có phải là Hamilton hay không.

Bài tập

Bài tập 1: Hội nghị bàn tròn

Tổng thư ký Đại hội đồng Liên hợp quốc triệu tập một cuộc họp có N nhà ngoại giao của N tổ chức tham gia. Các đại diện ngoại giao được bố trí ngồi quanh một bàn tròn. Giữa một số tổ chức có quan hệ căng thẳng, vì vậy không thể xếp họ ngồi cạnh nhau được. Thông tin về quan hệ giữa các tổ chức được cho dưới dạng cặp số nguyên i, j nếu giữa 2 tổ chức này có quan hệ căng thẳng.

Hãy lập trình giúp Tổng thư ký Liên hợp quốc bố trí chỗ ngồi quanh bàn họp. Các tổ chức được đánh số từ 1 tới N, 0 < N <= 500.

Dữ liệu vào: từ file CONF.INP, dòng đầu tiên chứa số nguyên N, các dòng sau, mỗi dòng một cặp số i, j cho biết các đại diện i và j không ngồi cạnh nhau được. Kết thúc là một dòng chứa 2 số 0.

Kết quả: đưa ra file CONF.OUT. Nếu không có cách bố trí thỏa mãn yêu cầu thì đưa ra thông báo KHONG CO, trong trường hợp ngược lại – đưa ra dãy N số nguyên xác định vị trí ai ngồi cạnh ai quanh bàn tròn.

Ví dụ:

CONF.INP CONF.OUT

11 1 9 7 4 11 5 8 2 10 3 6

1 4

1 7

5 7

10 7

10 8

10 9

3 4

0 0

Bài tập 2: Domino

Cho trước N con cờ domino, hãy viết chương trình cho biết rằng có cách sắp N con cờ đó theo đúng luật domino sao cho các con cờ hình thành vòng tròn hay không, nếu có hãy chỉ ra một cách sắp.

Dữ liệu vào: file DOMINO.INP

Dòng đầu chứa số N

N dòng tiếp theo, mỗi dòng chứa 2 số từ 0 đến 6 mang giá trị đại diện cho 2 đầu của domino.

Dữ liệu ra: file DOMINO.OUT

Dòng đầu chứa số 1 hoặc 0 tương ứng với sắp được quân domino thành vòng tròn và không sắp được.

Nếu dòng đầu là 1 (tương ứng với sắp được) thì dòng thứ hai liệt kê ra N chỉ số của N con cờ domino theo thứ tự để sắp thành vòng tròn. Nếu dòng đầu là 0 thì không có dòng thứ hai.

Ví dụ:

DOMINO.INP

4

5 3

5 6

2 3

6 2

DOMINO.OUT

1

1 3 4 2

Bài tập 3: Kiểm tra đường

Một trạm quảng đường giao thông phải chịu trách nhiêm về tình trạng của một mạng lưới giao thông nối giữa các điểm dân cư. Hàng tháng, họ phải cử một đội đi kiểm tra một vòng qua khắp mạng lưới để xem xét tình trạng hiện thời của các đường giao thông nhằm báo sửa chữa kịp thời nếu có nhu cầu. Hãy viết chương trình nhập vào mạng lưới giao thông và giúp trạm quyết định lộ trình của đội kiểm tra sao cho có thể thăm tất cả các con đường mà tổng chiều dài đoạn đường đi qua là nhỏ nhất.

Bài tập 4: Mã đi tuần

Hãy cài đặt chương trình xác định lộ trình của con mã trên bàn cờ 8x8 ô bắt đầu từ ô (i, j) đi qua tất cả các ô của bàn cờ vàmỗi ô chỉ 1 lần duy nhất.

Mở rộng với trường hợp bàn cờ kích thước NxN.

Bài tập 5: Hội nghị bàn tròn

Có 12 người ngồi chung 1 bàn tiệc tròn. Mỗi người có ít nhất 6 người quen. Hãy chỉ ra cách sắp xếp sao cho mỗi người đều ngồi cạnh người mình quen . Tổng quát, hãy sắp N người ngồi chung quanh bàn tròn sao cho mỗi người đều ngồi cạnh người mình quen. Biết mỗi người có ít nhất (N + 1)/2 người quen.

Bài 4: Cây và cây khung của đồ thị

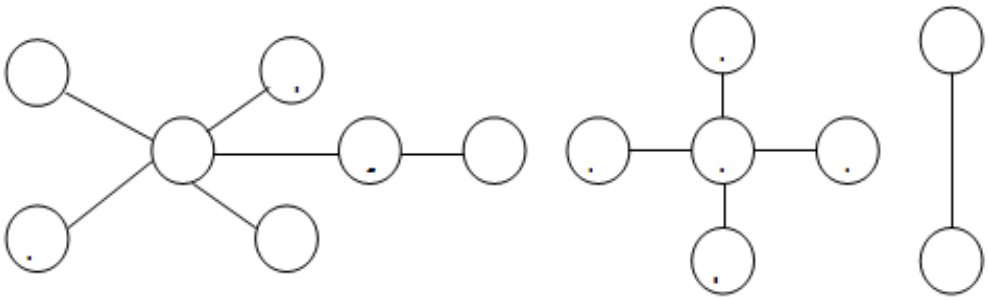
Cây và cây khung của đồ thị

Cây và các tính chất cơ bản của cây

Định nghĩa 1 Cây là một đồ thị vô hướng liên thông, không chứa chu trình và có ít nhất hai đỉnh.

Một đồ thị vô hướng không chứa chu trình và có ít nhất hai đỉnh gọi là một rừng. Trong một rừng, mỗi thành phần liên thông là một cây.

Ví dụ 1 Rừng sau có 3 cây:



Mệnh đề 1 Nếu T là một cây có n đỉnh thì T có ít nhất hai đỉnh treo.

Chứng minh: Lấy một cạnh (a,b) tuỳ ý của cây T. Trong tập hợp các đường đi sơ cấp chứa cạnh (a,b), ta lấy đường đi từ u đến v dài nhất. Vì T là một cây nên $u \neq v$. Mặt khác, u và v phải là hai đỉnh treo, vì nếu một đỉnh, u chẳng hạn, không phải là đỉnh treo thì u phải là đầu mút của một cạnh (u,x), với x là đỉnh không thuộc đường đi từ u đến v. Do đó, đường đi sơ cấp từ x đến v, chứa cạnh (a,b), dài hơn đường đi từ u đến v, trái với tính chất đường đi từ u đến v đã chọn.

Định lý 2: Cho T là một đồ thị có $n \geq 2$ đỉnh. Các điều sau là tương đương:

1) T là một cây.

2) T liên thông và có n?1 cạnh.

3) T không chứa chu trình và có n?1 cạnh.

4) T liên thông và mỗi cạnh là cầu.

5) Giữa hai đỉnh phân biệt bất kỳ của T luôn có duy nhất một đường đi sơ cấp.

6) T không chứa chu trình nhưng khi thêm một cạnh mới thì có được một chu trình duy nhất.

Chứng minh:1)⇒2) Chỉ cần chứng minh rằng một cây có n đỉnh thì có n?1 cạnh. Ta chứng minh bằng quy nạp. Điều này hiển nhiên khi n=2. Giả sử cây có k đỉnh thì có k?1 cạnh, ta chứng minh rằng cây T có k+1 đỉnh thì có k cạnh. Thật vậy, trong T nếu ta xoá một đỉnh treo và cạnh treo tương ứng thì đồ thị nhận được là một cây k đỉnh, cây này có k?1 cạnh, theo giả thiết quy nạp. Vậy cây T có k cạnh.

2)⇒3) Nếu T có chu trình thì bỏ đi một cạnh trong chu trình này thì T vẫn liên thông. Làm lại như thế cho đến khi trong T không còn chu trình nào mà vẫn liên thông, lúc đó ta được một cây có n đỉnh nhưng có ít hơn n?1 cạnh, trái với 2).

3)⇒4) Nếu T có k thành phần liên thông T_1, ..., T_k lần lượt có số đỉnh là n_1, ..., n_k (với $n_1+n_2+ \dots +n_k=n$) thì mỗi T_i là một cây nên nó có số cạnh là n_i?1. Vậy ta có

$$n-1=(n_1-1)+(n_2-1)+ \dots +(n_k-1)=(n_1+n_2+ \dots +n_k)-k=n-k.$$

Do đó k=1 hay T liên thông. Hơn nữa, khi bỏ đi một cạnh thì T hết liên thông, vì nếu còn liên thông thì T là một cây n đỉnh với n?2 cạnh, trái với điều đã chứng minh ở trên.

4)⇒5) Vì T liên thông nên giữa hai đỉnh phân biệt bất kỳ của T luôn có một đường đi sơ cấp, nhưng không thể được nối bởi hai đường đi sơ cấp vì nếu thế, hai đường đó sẽ tạo ra một chu trình và khi bỏ một cạnh thuộc chu trình này, T vẫn liên thông, trái với giả thiết.

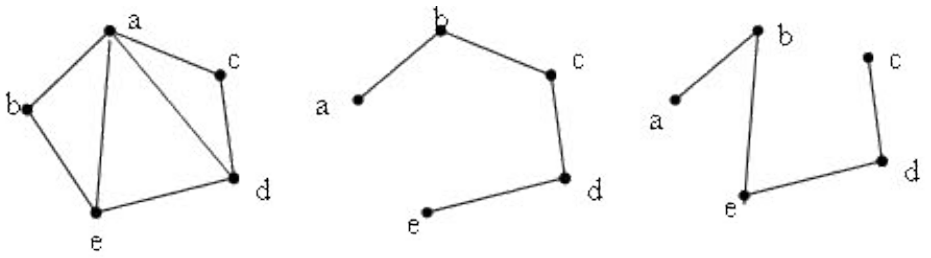
5)⇒6) Nếu T chứa một chu trình thì hai đỉnh bất kỳ trên chu trình này sẽ được nối bởi hai đường đi sơ cấp. Ngoài ra, khi thêm một cạnh mới (u,v), cạnh này sẽ tạo nên với đường đi sơ cấp duy nhất nối u và v một chu trình duy nhất.

6)⇒1) Nếu T không liên thông thì thêm một cạnh nối hai đỉnh ở hai thành phần liên thông khác nhau ta không nhận được một chu trình nào. Vậy T liên thông, do đó nó là một cây.

Cây khung của đồ thị

Định nghĩa 2. Giả sử G = (V, E) là đồ thị vô hướng liên thông. Cây T = (V, F) với F ⊂ E được gọi là **cây khung** của đồ thị G.

Ví dụ 2 Đồ thị G và cây khung của nó được cho trong hình 2



Hình 2 Đồ thị và các cây khung của nó.

Định lý sau đây cho biết số lượng cây khung của đồ thị đầy đủ K_n:

Định lý 3 (Cayley). *Số lượng cây khung của đồ thị K n là n n-2 .*

Định lý 3 cho thấy số lượng cây khung của đồ thị là một số rất lớn. Bây giờ ta xét áp dụng của thuật toán tìm kiếm theo chiều sâu và theo chiều rộng trên đồ thị để xây dựng cây khung của đồ thị vô hướng liên thông. Trong cả hai trường hợp mỗi khi ta đến được đỉnh mới u (tức Chuaxet[u]=true) từ đỉnh v thì cạnh (v, u) sẽ được kết nạp vào cây khung. Hai thuật toán tương ứng được trình bày trong hai thủ tục sau đây.

Procedure stree_DFS(v);

(tim kiem theo chieu sau ap dung vao tim tap canh cua cay khung T cua do thi vo huong lien thong G cho boi danh sach ke. Cac bien Chuaxet, Ke, T la toan cuc*)*

begin

Chuaxet[v]:=false;

For u ∈ Ke(v) do

If Chuaxet[u] then

Begin

T:=T ? (u,v);

STREE_DFS(u);

End;

end;

(* Main Program *)

```
begin

(* Initialition *)

for u ∈ V do Chuaxet[u]:=true;

T := Ø ; (* T la tap canh cua cay khung *)

STREE_DFS(root); ( root la dinh nao do cua do thi *)

end.
```

```
Procedure Stree_BFS(v);

(* tim kiem theo chieu rong ap dung tim tap canh cua cau khung T cua do thi vo huong
lien thong G cho boi danh sach Ke *)

begin

Queue:= Ø ;

Queue ⇐ r;

Chuaxet[r]:=false;

While queue <> Ø do

Begin

V ⇐ queue;

For r ∈ Ke(v) do

If Chuaxet[u] then

Begin

Queue ⇐ u;

Chuaxet[u]:=false;

T:= T ? (u,v);
```

```
End;

End;

end;

(* Main Program *);

begin

for u ∈ V do Chuaxet[u]:=true;

T := Ø ; (* T la tap canh cua cay khung *)

Stree_BFS(root); (* root la mot dinh tuy y cua do thi *)

end.
```

Chú ý:

1. Lập luận tương tự như trong phần trước có thể chỉ ra được rằng các thuật toán mô tả ở trên có độ phức tạp tính toán O(m+n).

2. Cây khung tìm được theo thủ tục Stree_BFS() là cây đường đi ngắn nhất từ gốc r đến tất cả các đỉnh còn lại của đồ thị.

Xây dựng tập các chu trình cơ bản của đồ thị

Bài toán xây dựng cây khung của đồ thị liên quan chặt chẽ đến một số bài toán ứng dụng khác của lý thuyết đồ thị: bài toán xây dựng tập các chu trình cơ bản của đồ thị mà ta sẽ xét trong mục này.

Giả sử G=(V, E) là đơn đồ thị vô hướng liên thông, H=(V, T) là cây khung của nó. Các cạnh của đồ thị thuộc cây khung ta sẽ gọi là các cạnh trong, còn các cạnh còn lại sẽ gọi là cạnh ngoài.

Định nghĩa 3 Nếu thêm một cạnh ngoài e ∈ E\T vào cây khung H chúng ta sẽ thu được đúng một chu trình trong H, ký hiệu chu trình này là C_e. Tập các chu trình ? = { C_e : e ∈ E\T } được gọi là tập các chu trình cơ bản của đồ thị G.

Giả sử A và B là hai tập hợp, ta đưa vào phép toán sau

A ? B = (A ?B) \ (A ? B).

Tập A ? B được gọi là hiệu đối xứng của hai tập A và B.

Tên gọi chu trình cơ bản gắn liền với sự kiện là mỗi chu trình của đồ thị đều có thể thu được từ các chu trình cơ bản như chỉ ra trong định lý sau đây:

Định lý 3 Giả sử G=(V,E) là đồ thị vô hướng liên thông, H=(V,T) là cây khung của nó. Khi đó mọi chu trình của đồ thị G điều có thể biểu diễn như là hiệu đối xứng của một số các chu trình cơ bản.

Việc tìm tập hợp chu trình cơ bản giữ một vai trò quan trọng trong vấn đề giải tích mạng điện. Cụ thể hơn, theo mỗi chu trình cơ bản của đồ thị tương ứng với mạng điện cần phân tích ta sẽ thiết lập được một phương trình tuyến tính theo định luật Kirchoff: tổng hiệu điện thế dọc theo một mạch vòng là bằng không. Hệ thống phương trình tuyến tính thu được cho phép tính toán hiệu điện thế trên mọi đường dây của lưới điện.

Ta sẽ xây dựng thuật toán xây dựng các chu trình cơ bản dựa trên thủ tục tìm kiếm theo chiều sâu trên đồ thị. Thuật toán có cấu trúc tương tự như thuật toán xây dựng cây khung theo thủ tục tìm kiếm theo chiều sâu mô tả trong mục trước.

Thuật toán xây dựng tập các chu trình cơ bản.

Giả thiết rằng đồ thị G=(V,E) được mô tả bằng danh sách Ke(v),v∈ V.

Procedure Cycle(v);

(* tim kiem cac chu trinh co ban cua thanh phan lien thong chua dinh v; cac bien d, num , stack, index la bien toan cuc *)

begin

d:=d+1; stack[d]:=v; num:=num+1;index[v]:=num;

for u ∈ Ke(v) do

if index[u]=0 then cycle(u)

else

if (u <> stack[d-1]) and (index[v]>index[u]) then

<Ghi nhan chu trinh voi cac dinh:

stack[d], stack[d-1],. . ., stack[c], voi stack[c]=u>

```
d:=d-1;
end;
(* Main Program *)
begin
for v ∈ V do Index[v]:=0;
num:=0; d:=0; stack[0]:=0;
for v ∈ V do
if Index[v]=0 then cycle(v);
end.
```

Chú ý: Độ phức tạp tính toán của thuật toán vừa mô tả là O(|E| |V|).

Bài 5: Bài toán cây khung nhỏ nhất

Bài toán tìm đường đi ngắn nhất

Trong các ứng dụng thực tế, vài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị liên thông có một ý nghĩa to lớn. Có thể dẫn về bài toán như vậy nhiều bài toán thực tế quan trọng. Ví dụ, bài toán chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn hoặc khoảng cách hoặc thời gian hoặc chi phí) trên một mạng giao thông đường bộ, đường thủy hoặc đường không; bài toán chọn một phương pháp tiết kiệm nhất để đưa ra một hệ thống động lực từ trạng thái xuất phát đến trạng một trạng thái đích, bài toán lập lịch thi công các công các công đoạn trong một công trình thi công lớn, bài toán lựa chọn đường truyền tin với chi phí nhỏ nhất trong mạng thông tin, v.v… Hiện nay có rất nhiều phương pháp để giải các bài toán như vậy. Thế nhưng, thông thường, các thuật toán được xây dựng dựa trên cơ sở lý thuyết đồ thị tỏ ra là các thuật toán có hiệu quả cao nhất. Trong chương này chúng ta sẽ xét một số thuật toán như vậy.

Các khái niệm mở đầu

Trong bài này chúng ta chỉ xét đồ thị có hướng G =(V,E), |V|=n, |E|=m với các cung được gán trọng số, nghĩa là, mỗi cung (u,v) ∈ E của nó được đặt tương ứng với một số thực a(u,v) gọi là trọng số của nó. Chúng ta sẽ đặt a(u,v) = ∞ , nếu (u,v) ∉ E. Nếu dãy $v_0, v_1, . . ., v_p$ là một đường đi trên G, thì độ dài của nó được định nghĩa là tổng sau

$$\sum_{i=1}^{p} a(v_{i-1},v_i)$$

tức là, độ dài của đường đi chính là tổng của các trọng số trên các cung của nó. (Chú ý rằng nếu chúng ta gán trọng số cho tất cả cung đều bằng 1, thì ta thu được định nghĩa độ dài của đường đi như là số cung của đường đi giống như trong các chương trước đã xét).

Bài toán tìm đường đi ngắn nhất trên đồ thị dưới dạng tổng quát có thể phát biểu như sau: tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát s ∈V đến đỉnh cuối (đích) t ∈ V. Đường đi như vậy ta sẽ gọi là đường đi ngắn nhất từ s đến t còn độ dài của nó ta sẽ ký hiệu là d(s,t) và còn gọi là khoảng cách từ s đến t (khoảng cách định nghĩa như vậy có thể là số âm). Nếu như không tồn tại đường đi từ s đến t thì ta sẽ đặt d(s,t) =∞ . Rõ ràng, nếu như mỗi chu trình trong đồ thị đều có độ dài dương, trong đường đi ngắn nhất không có đỉnh nào bị lặp lại (đường đi không có đỉnh lặp lại sẽ gọi là đường đi cơ bản). Mặt khác nếu trong đồ thị có chu trình với độ dài âm (chu trình như vậy để gọi ngắn gọn ta gọi là chu trình âm) thì khoảng cách giữa một số cặp đỉnh nào đó của đồ thị có thể là không xác định, bởi vì, bằng cách đi vòng theo chu trình này một số đủ lớn lần, ta có thể chỉ ra đường đi giữa các đỉnh này có độ dài nhỏ hơn bất cứ số thực cho trước nào. Trong

những trường hợp như vậy, có thể đặt vấn đề tìm đường đi cơ bản ngắn nhất, tuy nhiên bài toán đặt ra sẽ trở nên phức tạp hơn rất nhiều, bởi vì nó chứa bài toán xét sự tồn tại đường đi Hamilton trong đồ thị như là một trường hợp riêng.

Trước hết cần chú ý rằng nếu biết khoảng cách từ s đến t, thì đường đi ngắn nhất từ s đến t, trong trường hợp trọng số không âm, có thể tìm được một cách dễ dàng. Để tìm đường đi, chỉ cần để ý là đối với cặp đỉnh s, t ∈ V tuỳ ý (s <> t) luôn tìm được đỉnh v sao cho

$$d(s,t) = d(s,v) + a(v,t).$$

Thực vậy, đỉnh v như vậy chính là đỉnh đi trước đỉnh t trong đường đi ngắn nhất từ s đến t. Tiếp theo ta lại có thể tìm được đỉnh u sao cho d(s,v) = d(s,u) + a(u,v), . . . Từ giả thiết về tính không âm của các trọng số dễ dàng suy ra rằng dãy t, v, u, . . . không chứa đỉnh lặp lại và kết thúc ở đỉnh s. Rõ ràng dãy thu được xác định (nếu lật ngược thứ tự các đỉnh trong nó) đường đi ngắn nhất từ s đến t. Từ đó ta có thuật toán sau đây để tìm đường đi ngắn nhất từ s đến t khi biết độ dài của nó.

Procedure Find_Path;

(* Đầu vào:

D[v] - khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại v ∈ V;

- đỉnh đích;

a[u,v], u, v ∈ V – ma trận trọng số trên các cung.

Đầu ra:

Mảng Stack chứa dãy đỉnh xác định đường đi ngắn nhất từ s đến t

*)

begin

stack:= Ø ; stack ⇐ t; v:=t;

while v <> s do

begin

u:=đỉnh thoả mãn d[v]=d[u]+a[u,v];

stack ⇐ u;

v:=u;

end;

end;

Chú ý rằng độ phức tạp tính toán của thuật toán là $O(n^2)$, do để tìm đỉnh u ta phải xét qua tất cả các đỉnh của đồ thị. Tất nhiên, ta cũng có thể sử dụng kỹ thuật ghi nhận đường đi đã trình bày trong chương 3: dùng biến mảng Truoc[v], v∈ V, để ghi nhớ đỉnh đi trước v trong đường đi tìm kiếm.

Cũng cần lưu ý thêm là trong trường hợp trọng số trên các cạnh là không âm, bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng có thể dẫn về bài toán trên đồ thị có hướng, bằng cách thay đổi mỗi cạnh của nó bởi nó bởi hai cung có hướng ngược chiều nhau với cùng trọng số là trọng số của các cạnh tương ứng. Tuy nhiên, trong trường hợp có trọng số âm, việc thay như vậy có thể dẫn đến chu trình âm.

Đường đi ngắn nhất xuất phát từ một đỉnh. Thuật toán ford-bellman

Phần lớn các thuật toán tìm khoảng cách giữa hai đỉnh s và t được xây dựng nhờ kỹ thuật tính toán mà ta có thể mô tả đại thể như sau: từ ma trận trọng số a[u,v], u,v ∈ V, ta tính cận trên d[v] của khoảng cách từ s đến tất cả các đỉnh v ∈ V. Mỗi khi phát hiện

d[u] + a[u,v] < d[v] (1)

cận trên d[v] sẽ được làm tốt lên: d[v] + a[u,v].

Quá trình đó sẽ kết thúc khi nào chúng ta không làm tốt thêm được bất kỳ cận trên nào. Khi đó, rõ ràng giá trị của mỗi d[v] sẽ cho khoảng cách từ đỉnh s đến đỉnh v. Khi thể hiện kỹ thuật tính toán này trên máy tính, cận trên d[v] sẽ được gọi là nhãn của đỉnh v, còn việc tính lại các cận này sẽ được gọi là thủ tục gán. Nhận thấy rằng để tính khoảng cách từ s đến t, ở đây, ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị. Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đường đi ngắn nhất giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại.

Sơ đồ tính toán mà ta vừa mô tả còn chưa xác định, bởi vì còn phải chỉ ra thứ tự các đỉnh u và v để kiểm tra điều kiện (1). Thứ tự chọn này có ảnh hưởng rất lớn đến hiệu quả của thuật toán.

Bây giờ ta sẽ mô tả thuật toán Ford-Bellman tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại của đồ thị. Thuật toán làm việc trong trường hợp trọng số của các cung là tuỳ ý, nhưng giả thiết rằng trong đồ thị không có chu trình âm.

Procedure Ford_Bellman

(* Đầu vào:

Đồ thị có hướng G=(V,E) với n đỉnh,

s ∈ V là đỉnh xuất phát, A[u,v], u, v ∈ V, ma trận trọng số;

Giả thiết: Đồ thị không có chu trình âm.

Đầu ra:

Khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại d[v], v ∈ V.

Trước[v], v ∈ V, ghi nhận đỉnh đi trước v trong đường đi ngắn nhất từ s đến v.)*

begin

(* Khởi tạo *)

for v ∈ V do

begin

d[v]:=a[s,v];

Truoc[v]:=s;

end;

d[s]:=0;

for k:=1 to n-2 do

for v ∈ V\{ s} do

for u ∈ V do

if d[v] > d[u] +a[u,v] then

```
begin

d[v]:=d[u]+a[u,v];

Truoc[v]:=u;

end;

end;
```

Tính đúng đắn của thuật toán có thể chứng minh trên cơ sở trên nguyên lý tối ưu của quy hoạch động. Rõ ràng là độ phức tạp tính toán của thuật toán là $O(n^3)$. Lưu ý rằng chúng ta có thể chấm dứt vòng lặp theo k khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến d[v] nào bị đổi giá trị. Việc này có thể xảy ra đối với k<n-2, và điều đó làm tăng hiệu quả của thuật toán trong việc giải các bài toán thực tế. Tuy nhiên, cải tiến đó không thực sự cải thiện được đánh giá độ phức tạp của bản thân thuật toán. Đối với đồ thị thưa tốt hơn là sử dụng danh sách kề Ke(v), v∈ V, để biểu diễn đồ thị, khi đó vòng lặp theo u cần viết lại dưới dạng

For u ∈ Ke(v) do

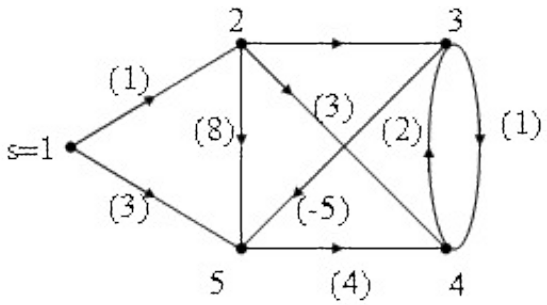
```
If d[v] > d[u] + a[u,v] thenBegin

D[v]:=d[u]+a[u,v];Truoc[v]:=u;

End;
```

Trong trường hợp này ta thu được thuật toán với độ phức tạp O(n,m).

A=

∞	2	∞	∞	∞
∞	∞	3	3	8
∞	∞	∞	1	-5
∞	∞	∞	∞	∞
∞	∞	∞	4	∞

Ví dụ 1 xét đồ thị trong hình 1. Các kết quả tính toán theo thuật toán được mô tả trong bảng dưới đây



Hình 1. Minh họa thuật toán Ford_Bellman.

k	d[1] Truoc[1]	d[2] Truoc[2]	d[3] Truoc[3]	d[4] Truoc[4]	d[5] Truoc[5]
	0, 1	1, 1	∞, 1	∞, 1	3, 1
1	0, 1	1, 1	4, 2	4, 2	-1, 3
2	0, 1	1, 1	4, 2	3, 5	-1, 3
3	0, 1	1, 1	4, 2	3, 5	-1, 3

Bảng kết quả tính toán theo thuật toán Ford_Bellman

Trong các mục tiếp theo chúng ta sẽ xét một số trường hợp riêng của bài toán tìm đường đi ngắn nhất mà để giải chúng có thể xây dựng những thuật toán hiệu quả hơn thuật toán Ford_Bellman. Đó là khi trọng số của tất cả các cung là các số không âm hoặc là khi đồ thị không có chu trình.

Trường hợp ma trận trọng số không âm. Thuật toán dijkstra

Trong trường hợp trọng số trên các cung là không âm thuật toán do Dijkstra đề nghị làm việc hữu hiệu hơn rất nhiều so với thuật toán trình bày trong mục trước. Thuật toán được xây dựng dựa trên cơ sở gán cho các đỉnh các nhãn tạm thời. Nhãn của mỗi đỉnh cho biết cận của độ dài đường đi ngắn nhất từ s đến nó. Các nhãn này sẽ được biến đổi theo một thủ tục lặp, mà ở mỗi bước lặp có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh nào đó trở thành một nhãn cố định thì nó sẽ cho ta không phải là cận trên mà là độ dài của đường đi ngắn nhất từ đỉnh s đến nó. Thuật toán được mô tả cụ thể như sau.

Procedure Dijstra;

(* Đầu vào:

Đồ thị có hướng G=(v,E) với n đỉnh,

s ∈ V là đỉnh xuất phát, a[u,v], u,v ∈ V, ma trận trọng số;

Giả thiết: a[u,v]≥0, u,v ∈ V.

Đầu ra:

Khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại d[v], v ∈ V.

*Truoc[v], v ∈ V, ghi nhận đỉnh đi trước v trong đường đi ngắn nhất từ s đến v *)*

Begin

(* Khởi tạo *)

for v ∈ V do

begin

d[v]:=a[s,v];

Truoc[v]:=s;

end;

d[s]:=0; T:=V\{ s} ; (T là tập các đỉnh cá nhãn tạm thời *)*

(* Bước lặp *)

while T <> Ø do

begin

tìm đỉnh u ∈ T thoả mãn d[u]=min{ d[z]:z ∈ T} ;

T:=T\{u} ; (Cố định nhãn của đỉnh u*)*

For v ∈ T do

If d[v]>d[u]+a[u,v] then

Begin

d[v]:=d[u]+a[u,v];

```
Truoc[v]:=u;

End;

end;

End;
```

Định lý 1 Thuật toán Dijkstra tìm được đường đi ngắn nhất trên đồ thị sau thời gian cỡ $O(n^2)$.

Chứng minh.

Trước hết ta chứng minh là thuật toán tìm được đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị. Giả sử ở một bước lặp nào đó các nhãn cố định cho ta độ dài các đường đi ngắn nhất từ s đến các đỉnh có nhãn cố định, ta sẽ chứng minh rằng ở lần gặp tiếp theo nếu đỉnh u* thu được nhãn cố định d(u*) chính là độ dài đường đi ngẫn nhất từ s đến u*.

Ký hiệu S_1 là tập hợp các đỉnh có nhãn cố định còn S_2 là tập các đỉnh có nhãn tạm thời ở bước lặp đang xét. Kết thúc mỗi bước lặp nhãn tạm thời $d(u^*)$ cho ta độ dài của đường đi ngắn nhất từ s đến u* không nằm trọng trong tập S_1, tức là nó đi qua ít nhất một đỉnh của tập S_2. Gọi $z \in S_2$ là đỉnh đầu tiên như vậy trên đường đi này. Do trọng số trên các cung là không âm, nên đoạn đường từ z đến u* có độ dài $L > 0$ và

$d(z) < d(u^*) – L < d(u^*)$.

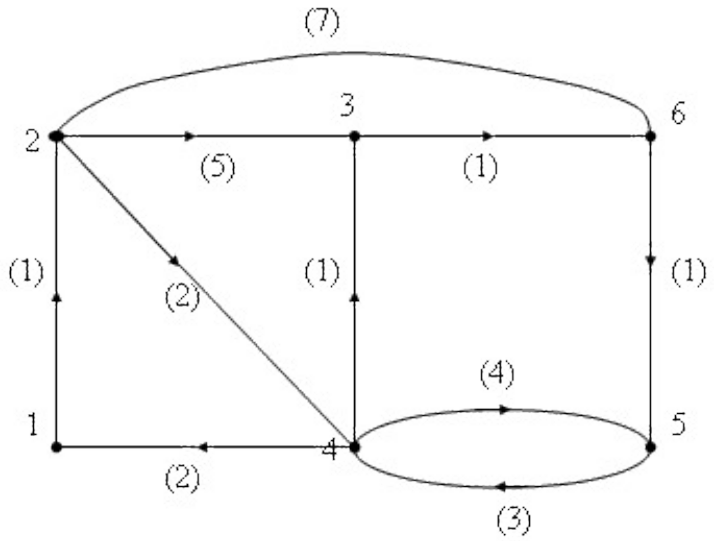
Bất đẳng thức này là mâu thuẫn với cách xác định đỉnh u* là đỉnh có nhãn tạm thời nhỏ nhất. Vậy đường đi ngắn nhất từ s đến u* phải nằm trọn trong S_1, và vì thế, d[u*] là độ dài của nó. Do ở lần lặp đầu tiên $S_1 = \{s\}$ và sau mỗi lần lặp ta chỉ thêm vào một đỉnh u* nên giả thiết là d(v) cho độ dài đường đi ngắn nhất từ s đến v với mọi $v \in S_1$ là đúng với bước lặp đầu tiên. Theo qui nạp suy ra thuật toán cho ta đường đi ngắn nhất từ s đến mọi đỉnh của đồ thị.

Bây giờ ta sẽ đánh giá số phép toán cần thực hiện theo thuật toán. Ơû mỗi bước lặp để tìm ra đỉnh u cần phải thực hiện O(n) phép toán, và để gán nhãn lại cũng cần thực hiện một số lượng phép toán cũng là O(n). thuật toán phải thực hiện n-1 bước lặp, vì vậy thời gian tính toán của thuận toán cỡ $O(n^2)$.

Định lý được chứng minh.

Khi tìm được độ dài của đường đi ngắn nhất d[v] thì đường đi này có thể tìm dựa vào nhãn Truoc[v], $v \in V$, theo qui tắc giống như chúng ta đã xét.

Ví dụ 2 Tìm đường đi ngắn nhất từ 1 đến các đỉnh còn lại của đồ thị ở hình 2.



Hình 2 Minh họa thuật toán Dijkstra.

Kết quả tính toán theo thuật toán được trình bày theo bảng dưới đây. Qui ước viết hai thành phần của nhãn theo thứ tự: d[v]. Đỉnh được đánh dấu * là đỉnh được chọn để cố định nhãn ở bước lặp đang xét, nhãn của nó không biến đổi ở các bước tiếp theo, vì thế ta đánh dấu -.

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	0,1	1,1*	∞ ,1	∞ ,1	∞ ,1	∞ ,1
1	-	-	6,2	3,2*	∞ ,1	8,2
2	-	-	4,4*	-	7,4	8,2
3	-	-		-	7,4	5,3*
4	-	-		-	6,6*	-

Chú ý:

1) Nếu chỉ cần tìm đường đi ngắn nhất từ s đến một đỉnh t nào đó thì có thể kết thúc thuật toán khi đỉnh t trở thành có nhãn cố định.

2) Tương tự như trong mục 2, dễ dàng mô tả thuật toán trong trường hợp đồ thị cho bởi danh sách kề. Để có thể giảm bớt khối lượng tính toán trong việc xác định đỉnh u ở mỗi bước lặp, có thể sử dụng thuật toán Heasort (tương tự như trong bài 5 khi thể hiện thuật toán Kruskal). Khi đó có thể thu được thuật toán với độ phức tạp tính toán là O(m log n).

Bài 6: Bài toán tìm đường đi ngắn nhất

Bài toán tìm đường đi ngắn nhất

Bài toán tìm đường đi ngắn nhất

Trong các ứng dụng thực tế, vài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị liên thông có một ý nghĩa to lớn. Có thể dẫn về bài toán như vậy nhiều bài toán thực tế quan trọng. Ví dụ, bài toán chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn hoặc khoảng cách hoặc thời gian hoặc chi phí) trên một mạng giao thông đường bộ, đường thủy hoặc đường không; bài toán chọn một phương pháp tiết kiệm nhất để đưa ra một hệ thống động lực từ trạng thái xuất phát đến trạng một trạng thái đích, bài toán lập lịch thi công các công các công đoạn trong một công trình thi công lớn, bài toán lựa chọn đường truyền tin với chi phí nhỏ nhất trong mạng thông tin, v.v… Hiện nay có rất nhiều phương pháp để giải các bài toán như vậy. Thế nhưng, thông thường, các thuật toán được xây dựng dựa trên cơ sở lý thuyết đồ thị tỏ ra là các thuật toán có hiệu quả cao nhất. Trong chương này chúng ta sẽ xét một số thuật toán như vậy.

Các khái niệm mở đầu

Trong bài này chúng ta chỉ xét đồ thị có hướng G =(V,E), |V|=n, |E|=m với các cung được gán trọng số, nghĩa là, mỗi cung (u,v) ∈ E của nó được đặt tương ứng với một số thực a(u,v) gọi là trọng số của nó. Chúng ta sẽ đặt a(u,v) = ∞ , nếu (u,v) ∉ E. Nếu dãy $v_0, v_1, . . ., v_p$ là một đường đi trên G, thì độ dài của nó được định nghĩa là tổng sau

$$\sum_{i=1}^{p} a(v_{i-1},v_i)$$

tức là, độ dài của đường đi chính là tổng của các trọng số trên các cung của nó. (Chú ý rằng nếu chúng ta gán trọng số cho tất cả cung đều bằng 1, thì ta thu được định nghĩa độ dài của đường đi như là số cung của đường đi giống như trong các chương trước đã xét).

Bài toán tìm đường đi ngắn nhất trên đồ thị dưới dạng tổng quát có thể phát biểu như sau: tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát s ∈V đến đỉnh cuối (đích) t ∈ V. Đường đi như vậy ta sẽ gọi là đường đi ngắn nhất từ s đến t còn độ dài của nó ta sẽ ký hiệu là d(s,t) và còn gọi là khoảng cách từ s đến t (khoảng cách định nghĩa như vậy có thể là số âm). Nếu như không tồn tại đường đi từ s đến t thì ta sẽ đặt d(s,t) =∞ . Rõ ràng, nếu như mỗi chu trình trong đồ thị đều có độ dài dương, trong đường đi ngắn nhất không có đỉnh nào bị lặp lại (đường đi không có đỉnh lặp lại sẽ gọi là đường đi cơ bản). Mặt khác nếu trong đồ thị có chu trình với độ dài âm (chu trình như vậy để gọi ngắn gọn ta gọi là chu trình âm) thì khoảng cách giữa một số cặp đỉnh nào đó của đồ thị có thể là

không xác định, bởi vì, bằng cách đi vòng theo chu trình này một số đủ lớn lần, ta có thể chỉ ra đường đi giữa các đỉnh này có độ dài nhỏ hơn bất cứ số thực cho trước nào. Trong những trường hợp như vậy, có thể đặt vấn đề tìm đường đi cơ bản ngắn nhất, tuy nhiên bài toán đặt ra sẽ trở nên phức tạp hơn rất nhiều, bởi vì nó chứa bài toán xét sự tồn tại đường đi Hamilton trong đồ thị như là một trường hợp riêng.

Trước hết cần chú ý rằng nếu biết khoảng cách từ s đến t, thì đường đi ngắn nhất từ s đến t, trong trường hợp trọng số không âm, có thể tìm được một cách dễ dàng. Để tìm đường đi, chỉ cần để ý là đối với cặp đỉnh s, t ∈ V tuỳ ý (s <> t) luôn tìm được đỉnh v sao cho

$$d(s,t) = d(s,v) + a(v,t).$$

Thực vậy, đỉnh v như vậy chính là đỉnh đi trước đỉnh t trong đường đi ngắn nhất từ s đến t. Tiếp theo ta lại có thể tìm được đỉnh u sao cho d(s,v) = d(s,u) + a(u,v), . . . Từ giả thiết về tính không âm của các trọng số dễ dàng suy ra rằng dãy t, v, u, . . . không chứa đỉnh lặp lại và kết thúc ở đỉnh s. Rõ ràng dãy thu được xác định (nếu lật ngược thứ tự các đỉnh trong nó) đường đi ngắn nhất từ s đến t. Từ đó ta có thuật toán sau đây để tìm đường đi ngắn nhất từ s đến t khi biết độ dài của nó.

Procedure Find_Path;

(* Đầu vào:

D[v] - khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại v ∈ V;

- đỉnh đích;

a[u,v], u, v ∈ V – ma trận trọng số trên các cung.

Đầu ra:

Mảng Stack chứa dãy đỉnh xác định đường đi ngắn nhất từ s đến t

*)

begin

stack:= Ø ; stack ⇐ t; v:=t;

while v <> s do

```
begin

u:=đỉnh thoả mãn d[v]=d[u]+a[u,v];

stack ⇐ u;

v:=u;

end;

end;
```

Chú ý rằng độ phức tạp tính toán của thuật toán là $O(n^2)$, do để tìm đỉnh u ta phải xét qua tất cả các đỉnh của đồ thị. Tất nhiên, ta cũng có thể sử dụng kỹ thuật ghi nhận đường đi đã trình bày trong chương 3: dùng biến mảng Truoc[v], v∈ V, để ghi nhớ đỉnh đi trước v trong đường đi tìm kiếm.

Cũng cần lưu ý thêm là trong trường hợp trọng số trên các cạnh là không âm, bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng có thể dẫn về bài toán trên đồ thị có hướng, bằng cách thay đổi mỗi cạnh của nó bởi nó bởi hai cung có hướng ngược chiều nhau với cùng trọng số là trọng số của các cạnh tương ứng. Tuy nhiên, trong trường hợp có trọng số âm, việc thay như vậy có thể dẫn đến chu trình âm.

Đường đi ngắn nhất xuất phát từ một đỉnh. Thuật toán ford-bellman

Phần lớn các thuật toán tìm khoảng cách giữa hai đỉnh s và t được xây dựng nhờ kỹ thuật tính toán mà ta có thể mô tả đại thể như sau: từ ma trận trọng số a[u,v], u,v ∈ V, ta tính cận trên d[v] của khoảng cách từ s đến tất cả các đỉnh v ∈ V. Mỗi khi phát hiện

d[u] + a[u,v] < d[v] (1)

cận trên d[v] sẽ được làm tốt lên: d[v] + a[u,v].

Quá trình đó sẽ kết thúc khi nào chúng ta không làm tốt thêm được bất kỳ cận trên nào. Khi đó, rõ ràng giá trị của mỗi d[v] sẽ cho khoảng cách từ đỉnh s đến đỉnh v. Khi thể hiện kỹ thuật tính toán này trên máy tính, cận trên d[v] sẽ được gọi là nhãn của đỉnh v, còn việc tính lại các cận này sẽ được gọi là thủ tục gán. Nhận thấy rằng để tính khoảng cách từ s đến t, ở đây, ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị. Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đường đi ngắn nhất giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại.

Sơ đồ tính toán mà ta vừa mô tả còn chưa xác định, bởi vì còn phải chỉ ra thứ tự các đỉnh u và v để kiểm tra điều kiện (1). Thứ tự chọn này có ảnh hưởng rất lớn đến hiệu quả của thuật toán.

Bây giờ ta sẽ mô tả thuật toán Ford-Bellman tìm đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại của đồ thị. Thuật toán làm việc trong trường hợp trọng số của các cung là tuỳ ý, nhưng giả thiết rằng trong đồ thị không có chu trình âm.

Procedure Ford_Bellman

(* Đầu vào:

Đồ thị có hướng G=(V,E) với n đỉnh,

s ∈ V là đỉnh xuất phát, A[u,v], u, v ∈ V, ma trận trọng số;

Giả thiết: Đồ thị không có chu trình âm.

Đầu ra:

Khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại d[v], v ∈ V.

Trước[v], v ∈ V, ghi nhận đỉnh đi trước v trong đường đi ngắn nhất từ s đến v.)*

begin

(* Khởi tạo *)

for v ∈ V do

begin

d[v]:=a[s,v];

Truoc[v]:=s;

end;

d[s]:=0;

for k:=1 to n-2 do

for v ∈ V\{ s} do

for u ∈ V do

if d[v] > d[u] +a[u,v] then

begin

d[v]:=d[u]+a[u,v];

Truoc[v]:=u;

end;

end;

Tính đúng đắn của thuật toán có thể chứng minh trên cơ sở trên nguyên lý tối ưu của quy hoạch động. Rõ ràng là độ phức tạp tính toán của thuật toán là $O(n^3)$. Lưu ý rằng chúng ta có thể chấm dứt vòng lặp theo k khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến d[v] nào bị đổi giá trị. Việc này có thể xảy ra đối với k<n-2, và điều đó làm tăng hiệu quả của thuật toán trong việc giải các bài toán thực tế. Tuy nhiên, cải tiến đó không thực sự cải thiện được đánh giá độ phức tạp của bản thân thuật toán. Đối với đồ thị thưa tốt hơn là sử dụng danh sách kề Ke(v), v∈ V, để biểu diễn đồ thị, khi đó vòng lặp theo u cần viết lại dưới dạng

For u ∈ Ke(v) do

If d[v] > d[u] + a[u,v] thenBegin

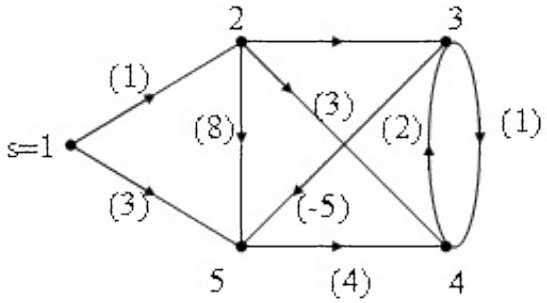
D[v]:=d[u]+a[u,v];Truoc[v]:=u;

End;

Trong trường hợp này ta thu được thuật toán với độ phức tạp O(n,m).

$$A=\begin{pmatrix} \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & 3 & 3 & 8 \\ \infty & \infty & \infty & 1 & -5 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty \end{pmatrix}$$

Ví dụ 1 xét đồ thị trong hình 1. Các kết quả tính toán theo thuật toán được mô tả trong bảng dưới đây



Hình 1. Minh họa thuật toán Ford_Bellman.

k	d[1] Truoc[1]	d[2] Truoc[2]	d[3] Truoc[3]	d[4] Truoc[4]	d[5] Truoc[5]
	0, 1	1, 1	∞, 1	∞, 1	3, 1
1	0, 1	1, 1	4, 2	4, 2	-1, 3
2	0, 1	1, 1	4, 2	3, 5	-1, 3
3	0, 1	1, 1	4, 2	3, 5	-1, 3

Bảng kết quả tính toán theo thuật toán Ford_Bellman

Trong các mục tiếp theo chúng ta sẽ xét một số trường hợp riêng của bài toán tìm đường đi ngắn nhất mà để giải chúng có thể xây dựng những thuật toán hiệu quả hơn thuật toán Ford_Bellman. Đó là khi trọng số của tất cả các cung là các số không âm hoặc là khi đồ thị không có chu trình.

Trường hợp ma trận trọng số không âm. Thuật toán dijkstra

Trong trường hợp trọng số trên các cung là không âm thuật toán do Dijkstra đề nghị làm việc hữu hiệu hơn rất nhiều so với thuật toán trình bày trong mục trước. Thuật toán được xây dựng dựa trên cơ sở gán cho các đỉnh các nhãn tạm thời. Nhãn của mỗi đỉnh cho biết cận của độ dài đường đi ngắn nhất từ s đến nó. Các nhãn này sẽ được biến đổi theo một thủ tục lặp, mà ở mỗi bước lặp có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh nào đó trở thành một nhãn cố định thì nó sẽ cho ta không phải là cận trên mà là độ dài của đường đi ngắn nhất từ đỉnh s đến nó. Thuật toán được mô tả cụ thể như sau.

Procedure Dijstra;

(* Đầu vào:

Đồ thị có hướng G=(v,E) với n đỉnh,

s ∈ V là đỉnh xuất phát, a[u,v], u,v ∈ V, ma trận trọng số;

Giả thiết: a[u,v]≥0, u,v ∈ V.

Đầu ra:

Khoảng cách từ đỉnh s đến tất cả các đỉnh còn lại d[v], v ∈ V.

*Truoc[v], v ∈ V, ghi nhận đỉnh đi trước v trong đường đi ngắn nhất từ s đến v *)*

Begin

(* Khởi tạo *)

for v ∈ V do

begin

d[v]:=a[s,v];

Truoc[v]:=s;

end;

d[s]:=0; T:=V\{ s} ; (T là tập các đỉnh cá nhãn tạm thời *)*

(* Bước lặp *)

while T <> Ø do

begin

tìm đỉnh u ∈ T thoả mãn d[u]=min{ d[z]:z ∈ T} ;

T:=T\{u} ; (Cố định nhãn của đỉnh u*)*

For v ∈ T do

If d[v]>d[u]+a[u,v] then

Begin

d[v]:=d[u]+a[u,v];

Truoc[v]:=u;

End;

end;

End;

Định lý 1 Thuật toán Dijkstra tìm được đường đi ngắn nhất trên đồ thị sau thời gian cỡ $O(n^2)$.

Chứng minh.

Trước hết ta chứng minh là thuật toán tìm được đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị. Giả sử ở một bước lặp nào đó các nhãn cố định cho ta độ dài các đường đi ngắn nhất từ s đến các đỉnh có nhãn cố định, ta sẽ chứng minh rằng ở lần gặp tiếp theo nếu đỉnh u* thu được nhãn cố định d(u*) chính là độ dài đường đi ngẫn nhất từ s đến u*.

Ký hiệu S_1 là tập hợp các đỉnh có nhãn cố định còn S_2 là tập các đỉnh có nhãn tạm thời ở bước lặp đang xét. Kết thúc mỗi bước lặp nhãn tạm thời $d(u^*)$ cho ta độ dài của đường đi ngắn nhất từ s đến u* không nằm trọng trong tập S_1, tức là nó đi qua ít nhất một đỉnh của tập S_2. Gọi $z \in S_2$ là đỉnh đầu tiên như vậy trên đường đi này. Do trọng số trên các cung là không âm, nên đoạn đường từ z đến u* có độ dài $L > 0$ và

$d(z) < d(u^*) – L < d(u^*)$.

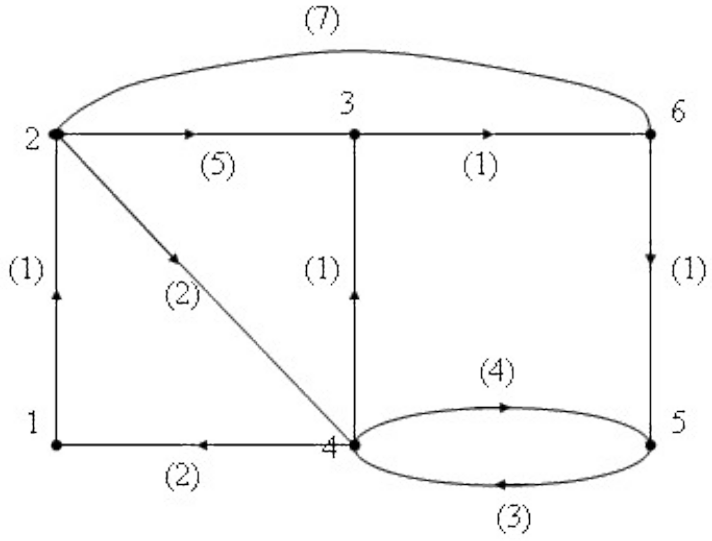
Bất đẳng thức này là mâu thuẫn với cách xác định đỉnh u* là đỉnh có nhãn tạm thời nhỏ nhất. Vậy đường đi ngắn nhất từ s đến u* phải nằm trọn trong S_1, và vì thế, d[u*] là độ dài của nó. Do ở lần lặp đầu tiên $S_1 = \{s\}$ và sau mỗi lần lặp ta chỉ thêm vào một đỉnh u* nên giả thiết là d(v) cho độ dài đường đi ngắn nhất từ s đến v với mọi $v \in S_1$ là đúng với bước lặp đầu tiên. Theo qui nạp suy ra thuật toán cho ta đường đi ngắn nhất từ s đến mọi đỉnh của đồ thị.

Bây giờ ta sẽ đánh giá số phép toán cần thực hiện theo thuật toán. Ơû mỗi bước lặp để tìm ra đỉnh u cần phải thực hiện O(n) phép toán, và để gán nhãn lại cũng cần thực hiện một số lượng phép toán cũng là O(n). thuật toán phải thực hiện n-1 bước lặp, vì vậy thời gian tính toán của thuận toán cỡ $O(n^2)$.

Định lý được chứng minh.

Khi tìm được độ dài của đường đi ngắn nhất d[v] thì đường đi này có thể tìm dựa vào nhãn Truoc[v], $v \in V$, theo qui tắc giống như chúng ta đã xét.

Ví dụ 2 Tìm đường đi ngắn nhất từ 1 đến các đỉnh còn lại của đồ thị ở hình 2.



Hình 2 Minh họa thuật toán Dijkstra.

Kết quả tính toán theo thuật toán được trình bày theo bảng dưới đây. Qui ước viết hai thành phần của nhãn theo thứ tự: d[v]. Đỉnh được đánh dấu * là đỉnh được chọn để cố định nhãn ở bước lặp đang xét, nhãn của nó không biến đổi ở các bước tiếp theo, vì thế ta đánh dấu -.

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6

Khởi tạo	0,1	1,1*	∞ ,1	∞ ,1	∞ ,1	∞ ,1
1	-	-	6,2	3,2*	∞ ,1	8,2
2	-	-	4,4*	-	7,4	8,2
3	-	-		-	7,4	5,3*
4	-	-		-	6,6*	-

Chú ý:

1) Nếu chỉ cần tìm đường đi ngắn nhất từ s đến một đỉnh t nào đó thì có thể kết thúc thuật toán khi đỉnh t trở thành có nhãn cố định.

2) Tương tự như trong mục 2, dễ dàng mô tả thuật toán trong trường hợp đồ thị cho bởi danh sách kề. Để có thể giảm bớt khối lượng tính toán trong việc xác định đỉnh u ở mỗi bước lặp, có thể sử dụng thuật toán Heasort (tương tự như trong bài 5 khi thể hiện thuật toán Kruskal). Khi đó có thể thu được thuật toán với độ phức tạp tính toán là O(m log n).

Bài toán tìm đường đi ngắn nhất (Tiếp)

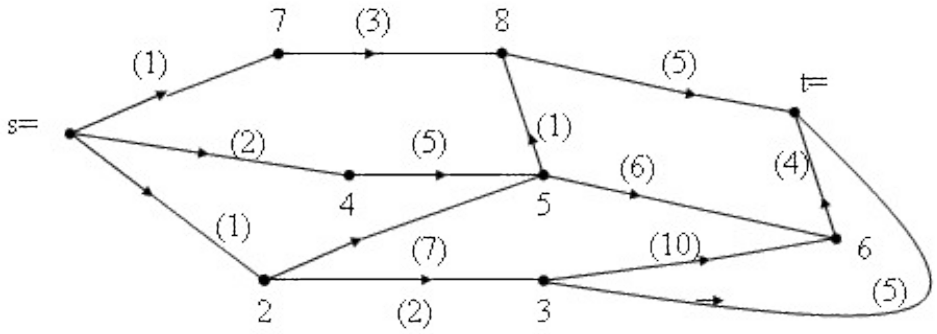
Bài toán tìm đường đi ngắn nhất (tiếp)

Đường đi trong đồ thị không có chu trình

Bây giờ ta xét trường hợp riêng thứ hai của bài toán đường đi ngắn nhất, mà để giải nó có thể xây dựng thuật toán với độ phức tạp tính toán $O(n^2)$, đó là khi đồ thị không có chu trình (còn trọng số trên các cung có thể là các số thực tuỳ ý). Trước hết ta chứng minh định lý sau.

Định lý 1. Giả sử G là đồ thị không có chu trình. Khi đó các đỉnh của nó có thể đánh số sao cho mỗi cung của đồ thị chỉ hướng từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn, nghĩa là mỗi cung của nó có sự biểu diễn dưới dạng (v[i], v[j]), trong đó i < j.

Ví dụ 1. Đồ thị trong hình 1 có các đỉnh số thoả mãn điều kiện nêu trong định lý.



Hình 1 Đồ thị không có chu trình.

Để chứng minh định lý ta mô tả thuật toán sau đây, cho phép tìm ra cách đánh số thoả mãn điều kiện định lý.

Procudure Numbering;

(Đầu vào: Đồ thị có hướng G=(V,E) với n đỉnh không chứa chu trình được cho bởi danh sách kề Ke(v), v ∈ V.*

Đầu ra:

Với mỗi đỉnh v ∈ V chỉ số NR [v] thoả mãn:

Với mọi cung (u,v) của đồ thị ta đều có NR [u] < NR [v] *)

Begin

For v ∈ V do Vao[v]:=0; (Tính Vao[v]=deg - (v) *)*

for u ∈ V do

for v ∈ Ke(u) do Vao[v]:=Vao[v]+1;

Queue:= Ø ;

For v ∈ V do

if Vao[v]=0 then Queue ⇐ v;

num:=0;

while queue <> Ø do

begin

u ⇐ queue; num:=num+1; NR[u]:=num;

for v ∈ Ke(u) do

begin

Vao[v]:=Vao[v]-1;

If Vao[v]=0 then queue ⇐ v;

end;

end;

End;

Thuật toán được xây dựng dựa trên ý tưởng rất đơn giản sau: rõ ràng trong đồ thị không có chu trình bao giờ cũng tìm được đỉnh có bán bậc vào bằng 0 (không có cung đi vào). Thực vậy, bắt đầu từ đỉnh v_1 nếu có cung đi vào nó từ v_2 thì ta lại chuyển sang xét đỉnh v_2 . Nếu có cung từ v_3 đi vào v_2, thì ta lại chuyển sang xét đỉnh v_3. . .Do đồ thị không có chu trình nên sau một số hữu hạn lần chuyển như vậy ta phải đi đến đỉnh không có cung đi vào. Thoạt tiên, tìm các đỉnh như vậy của đồ thị. Rõ ràng ta có thể đánh số chúng

theo thứ tự tuỳ ý bắt đầu từ 1. Tiếp theo, loại bỏ khỏi đồ thị những đỉnh đã được đánh số cùng các cung đi ra khỏi chúng, ta thu được đồ thị mới cũng không có chu trình, và thủ tục được lặp với đồ thị mới này. Quá trình đó sẽ được tiếp tục cho đến khi tất cả các đỉnh của đồ thị được đánh số.

Chú ý:

1) Rõ ràng trong bước khởi tạo ra phải duyệt qua tất cả các cung của đồ thị khi tính bán bậc vào của các đỉnh, vì vậy ở đó ta tốn cỡ O(m) phép toán, trong đó m là số cung của đồ thị. Tiếp theo, mỗi lần đánh số một đỉnh, để thực hiện việc loại bỏ đỉnh đã đánh số cùng với các cung đi ra khỏi nó, chúng ta lại duyệt qua tất cả các cung này. Suy ra để đánh số tất cả các đỉnh của đồ thị chúng ta sẽ phải duyệt qua tất cả các cung của đồ thị một lần nữa. Vậy độ phức tạp của thuật toán là O(m).

2) Thuật toán có thể áp dụng để kiểm tra xem đồ thị có chứa chu trình hay không? Thực vậy, nếu kết thúc thuật toán vẫn còn có đỉnh chưa được đánh số (num<n) thì điều đó có nghĩa là đồ thị chứa chu trình.

Do có thuật toán đánh số trên, nên khi xét đồ thị không có chu trình ta có thể giả thiết là các đỉnh của nó được đánh số sao cho mỗi cung chỉ đi từ đỉnh có chỉ số nhỏ đến đỉnh có chỉ số lớn hơn. Thuật toán tìm đường đi ngắn nhất trên đồ thị không có chu trình được mô tả trong sơ đồ sau đây.

Procedure Critical_Path;

(* Tìm đường đi ngắn nhất từ đỉnh nguồn đến tất cả các đỉnh còn lại trên đô thị không có chu trình *)

Đầu vào:

Đồ thị G=(V,E), trong đó V={ v[1], v[2], . . . , v[n]} .

Đối với mỗi cung (v[i], v[j]) ∈ E, ta có i<j.

Đồ thị được cho bởi danh sách kề Ke(v), v ∈ V.

Đầu ra:

Khoảng cách từ v[1] đến tất cả các đỉnh còn lại được ghi trong mảng d[v[i]], i= 2, 3, . . .,n *)

Begin

dv[1]]:=0;

for j:=2 to n do d[v[j]]:=a[v[1], v[j]];

for j:=2 to n do

for v ∈ Ke[v[j]] do d[v]:=min(d[v], d[v[j]]+a[v[j], v]);

End;

Độ phức tạp tính toán của thuật toán là O(m), do mỗi cung của đồ thị phải xét qua đúng một lần.

Các thuật toán được mô tả ở trên thường được ứng dụng vào việc xây dựng những phương pháp giải bài toán điều khiển việc thực hiện những dự án lớn, gọi tắt là PERT (Project Evaluation and Review Technique) hay CDM (Critical path Method). Một ví dụ đơn giản cho ứng dụng này được mô tả trong ví dụ dưới đây.

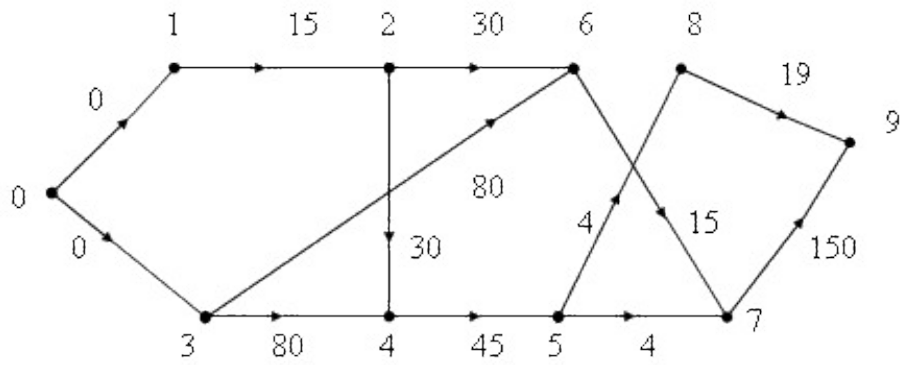
Công đoạn	t[i]	Các công đoạn phải được hoàn thành trước nó
1	15	Không có
2	30	1
3	80	Không có
4	45	2, 3
5	124	4
6	15	2, 3
7	15	5, 6
8	19	5

Ví dụ 2 Việc thi công một công trình lớn được chia thành n công đoạn, đánh số từ 1 đến n. Có một số công đoạn mà việc thực hiện nó chỉ được tiến hành sau khi một sô công đoạn nào đó đã hoàn thành. Đối với mỗi cong đoạn i biết t[i]] là thời gian cần thiết để hoàn thành nó (i=1, 2,. . .,n). Dữ liệu với n=8 được cho trong bảng dưới đây

Giả sử thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể được.

Ta có thể xây dựng đồ thị có hướng n đỉnh biểu diễn hạn chế về trình tự thực hiện các công việc như sau: Mỗi đỉnh của đồ thị tương ứng với một công việc, nếu công việc i

phải được thực hiện trước công đoạn j thì trên đồ thị có cung (i,j), trọng số trên cung này được gán bằng t[i], xem hình 2 dưới đây.



Hình 2 Đồ thị minh hoạ PERT.

Thêm vào đồ thị hai đỉnh 0 và n+1 tương ứng với hai sự kiện đặc biệt: đỉnh 0 tương ứng với công đoạn lễ khởi công, nó phải được thực hiện trước tất cả các công đoạn khác, và đỉnh n+1 tương ứng với công đoạn cắt băng khánh thành công trình, nó phải được thực hiện sau các công đoạn, với t[0]=t[n+1]=0 (trên thực tế chỉ cần nối đỉnh 0 với tất cả các đỉnh có bán bậc bằng 0 và nối tất cả các đỉnh có bán bậc ra bằng 0 với đỉnh n+1). Gọi đồ thị thu được là G. Rõ ràng bài toán đặt ra dẫn về bài toán tìm đường đi ngắn nhất từ đỉnh 0 đến tất cả các đỉnh còn lại trên đồ thị G. Do đồ thị G rõ ràng là không chứa chu trình, nên để giải bài toán đặt ra có thể áp dụng các thuật toán mô tả trên, chỉ cần đổi dấu tất cả các trọng số trên các cung thành dấu ngược lại, hoặc đơn giản hơn chỉ cần đổi toán tử Min trong thuật toán Critcal_Path thành toán tử Max. Kết thúc thuật toán, chúng ta thu được d[v] là độ dài đường đi dài nhất từ đỉnh 0 đến đỉnh v. Khi đó d[v] cho ta thời điểm sớm nhất có thể bắt đầu thực hiện công đoạn v, nói riêng d[n+1] là thời điểm sớm nhất có thể cắt băng khánh thành, tức là thời điểm sớm nhất có thể hoàn thành toàn bộ công trình.

Cây đường đi dài nhất của bài toán trong ví dụ 2 tìm được theo thuật toán được chỉ ra trong hình 2.

Đường đi ngắn nhất giữa tất cả các cặp đỉnh

Rõ ràng ta có thể giải bài toán tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị bằng cách sử dụng n lần thuật toán mô tả ở mục trước, trong đó ta sẽ chọn s lần lượt là các đỉnh của đồ thị. Rõ ràng, khi đó ta thu được thuật toán với độ phức tạp $O(n^4)$ (nếu sử dụng thuật toán Ford_Bellman) hoặc $O(n^3)$ đối với trường hợp trọng số không âm hoặc đồ thị không có chu trình. Trong trường hợp tổng quát, sử dụng thuật toán Ford_Bellman n lần không phải là cách làm tốt nhất. Ở đây ta sẽ mô tả một thuật toán giải bài toán trên

với độ phức tạp tính toán $O(n^3)$: thuật toán Floyd. Thuật toán được mô tả trong thủ tục sau đây.

Procedure Floyd;

(* Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh

Đầu vào: Đồ thị cho bởi ma trận trọng số a[i,j], i, j =1, 2,. . ,n.

Đầu ra:

Ma trận đường đi ngắn nhất giữa các cặp đỉnh

d[i,j]=, i,j = 1, 2. . .,n,

trong đó d[i,j] cho độ dài đường đi ngắn nhất từ đỉnh i đến đỉnh j.

Ma trận ghi nhận đường đi

p[i,j], i, j = 1, 2.. . , n,

trong đó p[i,j] ghi nhận đỉnh đi trước đỉnh j trong đường đi ngắn nhất từ i đến j. *)

```
begin
(* Khởi tạo *)
for i:=1 to n do
for j:=1 to n do
begin
d[i,j]:=a[i.j];
p[i.j]:=i;
end;
(* Bước lặp *)
for k:=1 to n do
for i:=1 to n do
```

```
for j:=1 to n do

if d[i,j]>d[i,k]+d[k,j] then

begin

d[i,j]+d[i,k]+d[k,j];

p[i,j]>p[k,j];

end;

end;
```

Rõ ràng độ phức tạp tính toán của thuật toán là $O(n^3)$.

Kết thúc phần này chúng ra trình bày một cách thể hiện thuật toán Dijkstra trên ngôn ngữ Pascal:

```
(* CHƯƠNG TRÌNH TÌM ĐƯỜNG ĐI NGẮN NHẤT TỪ ĐỈNH S ĐẾN ĐỈNH T THEO THUẬT TOÁN DIJKSTRA *)

uses crt;

const max=50;

var

n, s, t:integer;

chon:char;

Truoc:array[1..max] of byte;

d: array[1..max] of integer;

a: array[1..max,1..max] of integer;

final: array[1..max] of boolean;

procedure Nhapsolieu;
```

```
var
f:text;
fname:string;
i,j:integer;
begin
write('Vao ten file du lieu can doc:');
readln(fname);
assign(f,fname);
reset(f);
readln(f,n);
for i:=1 to n do
for j:=1 to n do read(f, a[i,j];
close(f);
end;
procedure Insolieu;
var
i,j:integer;
begin
writeln('So dinh cua do thi:',n);
writeln('Ma tran khoang cach:');
for i:=1 to n do
begin
```

```
for j:=1 to n do write(a[i,j]:3,' ');
writeln;
end;
end;
Procedure Inketqua;
Var
i,j:integer;
begin
writeln('Duong di ngan nhat tu ',s,' den ',t);
write(t,' → ');
while i<> s do
begin
i:=Truoc[i];
write(i,' → ');
end;
end;
Procedure Dijkstra;
Var
U,v,minp:integer;
Begin
Write('Tim duon di tu s=');Readln(s);Write(' den t=');Readln(t);
For v:=1 to n doBegin
```

```
d[v]:=a[s,v];
Truoc[v]:=s;
Filal[v]:=false;
End;
Truoc[s]:=0;D[s]:=0;Final[s]:=true;
While not final[t] do (* Buoc lap *)Begin
{ Tim u la dinh co nhan tam thoi nho nhat }
minp:=maxint;
for v:=1 to n do
if (not final[v]) ans minp>d[v]) then
begin
u:=v; minp:=d[v];
end;
final[u]:=true;
if not final[t] then
for v:=1 to n do
if (not final[v]) and (d[u]+a[u,v]<d[v]) then
begin
d[v]:=d[u]+a[u,v];
Truoc[v]:=u;
end;
End;
```

```
end;
Procedure Menu;
Begin
Clrscr;
Writeln(‘1. Nhap du lieu tu file’);
Writeln(‘2. Giai bai toan’);
Writeln(‘3. Ket thuc’);
Writeln(‘----------‘);
Write(‘Hay chon chuc nang:’):
End;
(* Chuong trinh chinhs *)
Begin
Repeat
Menu;
Chon:=readkey;
Writeln(chon);
Case chon of
‘1’:Nhapsolieu;
‘2’: begin
Insolieu;
Dijkstra;
Inketqua;
```

‘3’:exit;

end;

Until false;

End.

Bài tập

Bài tập 1: Di chuyển trên các hình tròn

Cho N hình tròn (đánh số từ 1 đến N). Một người muốn đi từ hình tròn này sang hình tròn khác cần tuân theo qui ước:

- Nếu khoảng cách giữa 2 điểm gần nhất của 2 hình tròn không quá 50 cm thì có thể bước sang.

- Nếu khoảng cách này hơn 50cm và không quá 80cm thì có thể nhảy sang.

- Các trường hợp khác không thể sang được.

Một đường đi từ hình tròn này sang hình tròn khác được gọi là càng "tốt" nếu số lần phải nhảy là càng ít. Hai đường đi có số lần nhảy bằng nhau thì đường đi nào có số hình tròn đi qua ít hơn thì đường đi đó "tốt" hơn.

Các hình tròn được cho trong một file văn bản, trong đó dòng thứ i mô tả hình tròn số hiệu i (i = 1, 2,..., N) bao gồm 3 số thực: hoành độ tâm, tung độ tâm, độ lớn bán kính (đơn vị đo bằng mét).

Lập trình đọc các hình tròn từ một file văn bản (tên file vào từ bàn phím), sau đó cứ mỗi lần đọc số hiệu hình tròn xuất phát S và hình tròn kết thúc T từ bàn phím, chương trình sẽ đưa ra đường đi từ S đến T là "tốt nhất" theo nghĩa đã nêu (hoặc thông báo là không có).

Yêu cầu đường đi được viết dưới dạng một dãy các số hiệu hình tròn lần lượt cần được đi qua trong đó nói rõ tổng số các bước nhảy, tổng số các hình tròn đi qua và những bước nào cần phải nhảy.

Giới hạn số hình tròn không quá 100.

Bài tập 2: Tìm hành trình tốn ít xăng nhất

Trên một mạng lưới giao thông, một người muốn đi từ điểm A đến điểm B bằng xe máy. Xe chứa được tối đa 3 lít xăng và chạy 100km hết 2,5 lít. Các trạm xăng chỉ được đặt ở các điểm dân cư, không đặt ở giữa đường và người này không mang theo bất kỳ thùng chứa xăng nào khác. Hãy viết chương trình nhập vào mạng lưới giao thông và xác định giúp người này tuyến đường đi từ A đến B sao cho ít tốn xăng nhất.

Bài tập 3: Di chuyển giữa các đảo

Trên một đảo quốc, có N hòn đảo. Giả sử tất cả các đảo đều có hình dạng là hình chữ nhật nằm ngang. Trên mỗi hòn đảo có thể có sân bay nằm ở trung tâm đảo, có thể có cảng nằm ở 4 góc đảo. Trên mỗi đảo đều có tuyến đường xe buýt nối 4 góc đảo với nhau và với trung tâm đảo. Giữa 2 đảo có thể đi lại bằng máy bay nếu cả 2 đảo đều có sân bay và có thể đi lại bằng tàu nếu cả 2 đảo đều có cảng.

Giả sử rằng:

- Các tuyến đường (bộ, không, thủy) đều là đường thẳng.

- Chi phí cho mỗi km và tốc độ của mỗi loại phương tiện là:

Phương tiện	Tốc độ (km/h)	Chi phí (đ/km)
Máy bay	1000	1000
Xe buýt	70	100
Tàu thủy	30	50

Hãy viết chương trình xác định tuyến đường và cách di chuyển giữa 2 hòn đảo trong đảo quốc sao cho:

- Thời gian di chuyển ít nhất.

- Chi phí di chuyển ít nhất.

- Thời gian di chuyển ít nhất nhưng với một số tiền chi phí không quá Đ đồng.

- Chi phí di chuyển ít nhất nhưng với thời gian di chuyển không vượt quá T giờ.

Bài tập 4: Hành trinh tới y

Các ô tô đi từ các thành phố khác nhau x_1, x_2,…., x_n và cùng tới một địa điểm thống nhất **y**. Nếu tồn tại đường đi từ x_i đến x_j thì ta ký hiệu t_{ij} là thời gian cần thiết để đi từ x_i đến x_j, c_{ij} là lượng ô tô có thể đi trên con đường đó trong một đơn vị thời gian (c_{ij} =

0 nếu không có đường đi), c_{ii} là lượng ô tô có thể nghỉ đồng thời ở thành phố x_i, a_i là số lượng xe ban đầu có ở x_i. Hãy tổ chức hành trình sao cho trong khoảng thời gian t số ô tô tới **y** là nhiều nhất.

Bài tập 5: Tìm đường ngắn nhất

Giả sử X là tập các khu dân cư, U là tập các đường sá nối liền các khu đó. Ta giả sử mọi chỗ giao nhau của các con đường đều thuộc X. Với con đường u, số l(u) là độ dài của u tính bằng km. Hãy chỉ ra tuyến đường đi từ một khu i sang khu j sao cho tổng chiều dài là nhỏ nhất.

Bài tập 6: Đường đi trên lưới

Cho 1 ma trận A[M, N], mỗi phần tử của nó chứa 1 số tự nhiên. Từ 1 ô (i, j) ta có thể đi sang ô kề nó (có chung 1 cạnh) nếu giá trị của ô kề này nhỏ hơn giá trị lưu trong (i, j). Hãy tìm 1 đường đi từ ô (i, j) tới ô (k, l) trên ma trận sao cho phải đi qua ít ô nhất. Hãy tìm 1 đường đi từ ô (i, j) tới ô (k, l) trên ma trận sao cho tổng giá trị các ô phải đi qua nhỏ nhất.

Bài tập 7 : Tìm đường với chi phí phải trả cho phép

Có N thành phố được đánh số từ 1..N nối với nhau bằng các đoạn đường *một chiều*. Mỗi đoạn đường bao gồm 2 thông số : Độ dài và chi phí đi của đoạn đường.

A sống tại thành phố 1 và A muốn di chuyển đến thành phố N nhanh nhất có thể.

Bạn hãy giúp A tìm ra *đường đi ngắn nhất* từ thành phố 1 đến thành phố N mà A có *khả năng chi trả* tiền.*Dữ liệu vào*: ROADS.IN

- Dòng đầu tiên chứa số nguyên K, 0 <= K <= 10000, số tiền mà A có.

- Dòng thứ 2 chứa số nguyên N, 2 <= N <= 100, số thành phố.

- Dòng thứ 3 chứa số nguyên R, 1 <= R <= 10000, tổng số đoạn đường.

- Mỗi dòng trong số R dòng tiếp theo mô tả một đoạn đường bằng các số S, D, L và T cách nhau bởi ít nhất một khoảng trắng.

+ S là thành phố khởi hành, 1 <= S <= N.

+ D là thành phố đến, 1 <= D <= N.

+ L là độ dài của đoạn đường, 1 <= L <= 100.

+ T là lộ phí, 0 <= T <= 100.

Chú ý rằng giữa 2 thành phố có thể có nhiều đoạn đường nối 2 thành phố này.

Dữ liệu ra: ROADS.OUT

Chỉ có duy nhất 1 dòng chứa tổng độ dài của đường đi ngắn nhất từ 1->N và nhỏ hơn K.

ROADS.IN5671 2 2 32 4 3 33 4 2 41 3 4 14 6 2 13 5 2 05 4 3 2ROADS.OUT11	ROADS.IN0441 4 5 21 2 1 02 3 1 13 4 1 0ROADS.OUT-1

Bài 7: Bài toán luồng cực đại trong mạng

Bài toán luồng cực đại trong mạng

Bài toán luồng cực đại trong mạng là một trong số bài toán tối ưu trên đồ thị tìm được những ứng dụng rộng rãi trong thực tế cũng như những ứng dụng thú vị trong lý thuyết tổ hợp. Bài toán được đề xuất vào đầu năm 1950, và gắn liên với tên tuổi của hai nhà toán học Mỹ là Ford và Fulkerson. Trong chương này chúng ra sẽ trình bày thuật toán Ford và Fulkerson để giải bài toán đặt ra và nêu một sô ứng dụng của bài toán.

Mạng. Luồng trong mạng. Bài toán luồng cực đại

Định nghĩa 1 Ta gọi mạng là đồ thị có hướng G = (V,E), trong đó duy nhất một đỉnh s không có cung đi vào gọi là đỉnh phát, duy nhất một đỉnh t không có cung đi ra gọi là điểm thu và mỗi cung e=(v,w) ∈ E được gán với một số không âm c(e) =c(v,w) gọi là khả năng thông qua của cung e.

Để thuận tiện cho việc trình bày ta sẽ qui ước rằng nếu không có cung (v,w) thì khả năng thông qua c(v,w) được gán bằng 0.

Định nghĩa 2 Giả sử cho mạng G=(V,E). Ta gọi mạng f trong mạng G=(V,E) ;là ánh xạ f: E→R_+ gán cho mỗi cung e =(v,w) ∈ E một số thực không âm f(e)=f(v,w), gọi là luồng trên cung e, thoả mãn các điều kiện sau:

- Luồng trên cung e ∈ E không vượt quá khả năng thông qua của nó:

$$0 \leq f(e) \leq c(e),$$

- Điều kiện cân bằng luồng trên mỗi đỉnh của mạng: Tổng luồng trên các cung đi vào đỉnh v bằng tổng luồng trên các cung đi ra khỏi đỉnh v, nếu v ≠ s, t:

$$Div_f(v) = \sum_{w \in N^-(v)} f(w,v) - \sum_{w \in N^+(v)} f(v,w) = 0$$

$$N^-(v) = \{ w \in V : (w,v) \in E \},$$

$$N^+(v) = \{ w \in V : (v,w) \in E \}.$$

$$Val(f) = \sum_{w \in N^+(s)} f(s,w) = \sum_{w \in N^-(t)} f(w,t).$$

trong đó $N^-(v)$ – tập các đỉnh của mạng mà từ đó có cung đến v, $N^+(v)$ - tập các đỉnh của mạng mà từ v có cung đến nó:

Giá trị của luồng f là số

Bài toán luồng cực đại trong mạng:

Cho mạng G(V,E). Hãy tìm luồng f* trong mạng với giá trị luồng val(f*) là lớn nhất. Luồng như vậy ta sẽ gọi là luồng cực đại trong mạng.

Bài toán như vậy có thể xuất hiện trong rất nhiều ứng dụng thực tế. Chẳng hạn khi cần xác định cường độ lớn nhất của dòng vận tải giữa hai nút của một bản đồ giao thông. Trong ví dụ này lời giải của bài toán luồng cực đại sẽ chỉ cho ta các đoạn đường đông xe nhất và chúng tạo thành "chỗ hẹp" tương ứng với dòng giao thông xét theo hai nút được chọn. Một ví dụ khác là nếu xét đồ thị tương ứng với một hệ thống đường ống dẫn dầu. Trong đó các ống tương ứng với các cung, điểm phát có thể coi là tầu chở dầu, điểm thu là bể chứa, còn những điểm nối giữa các ống là các nút của đồ thị. Khả năng thông qua của các cung tương ứng với tiết diện của các ống. Cần phải tìm luồng dầu lớn nhất có thể bơm từ tàu chở dầu vào bể chứa

Lát cắt. đường tăng luồng. Định lý ford_fulkerson

Định nghĩa 3.

Ta gọi lát cắt (X, X^*) là một cách phân hoạch tập đỉnh V của mạng ra thành hai tập X và $X^* = V\backslash X$, trong đó $s \in X$, $t \in X^*$. Khả năng thông qua của lát cắt (X, X^*) là số

$c(X,X^*) = ?c(v,w)$ $v \in X$ $w \in X^*$

Lát cắt với khả năng thông qua nhỏ nhất được gọi là lát cắt hẹp nhất.

Bổ đề 1. Giá trị của luồng f trong mạng luôn nhỏ hơn hoặc bằng khả năng thông qua của lát cắt (X, X^*) bất kỳ trong nó: $val(f) \le c(X, X^*)$.

Chứng minh. Cộng các điều kiện cân bằng luồng $Div_f(v)=0$ với mọi $v \in X$. Khi đó ta có

$$\sum_{v \in X} \left(\sum_{w \in N^-(v)} f(w,v) - \sum_{w \in N^+(v)} f(v,w) \right) = -Val(f)$$

$$- \sum_{\substack{v \in X \\ w \in X^*}} f(v,w) + \sum_{\substack{v \in X^* \\ w \in X}} f(v,w) = -val(f),$$

$$val(f) = \sum_{\substack{v \in X \\ w \in X^*}} f(v,w) - \sum_{\substack{v \in X^* \\ w \in X}} f(v,w)$$

tổng này sẽ gồm các số hạng dạng f(u,v) với dấu cộng hoặc dấu trừ mà trong đó có ít nhất một trong hai đỉnh u,v phải thuộc tập X. Nếu cả hai đỉnh u,v đều trong tập X, thì f(u,v) xuất hiện với dấu cộng trong $Div_f(v)$ và với dấu trừ trong $Div_f(u)$, vì thế, chúng triệt tiêu lẫn nhau. Do đó, sau khi giản ước các số hạng như vậy ở vế trái, ta thu được

hay là

Mặt khác, từ điều kiện 1 rõ ràng là

$$\sum_{v \in X,\ w \in X^*} f(v, w) \leq \sum_{v \in X^*,\ w \in X} c(v, w)$$

$$-\sum_{v \in X^*,\ w \in X} f(v,w) \leq 0$$

còn

suy ra val(f)≤c(X, X^*). Bổ đề được chứng minh.

Hệ quả 2. Giá trị luồng cực đại trong mạng không vượt quá khả năng thông qua của lát cắt hẹp nhất trong mạng.

Ford và Fulkerson đã chứng minh rằng giá trị luồng cực đại trong mạng đúng bằng khả năng thông qua của lát cắt hẹp nhất. Để có thể phát biểu và chứng minh kết quả này chúng ra sẽ cần thêm một số khái niệm.

Giả sử f là một luồng trong mạng G = (V, E). Từ mạng G =(V, E) ta xây dựng đồ thị có trọng số trên cung G_f = (V, E_f), với tập cung E_f và trọng số trên các cung được xác định theo qui tắc sau :

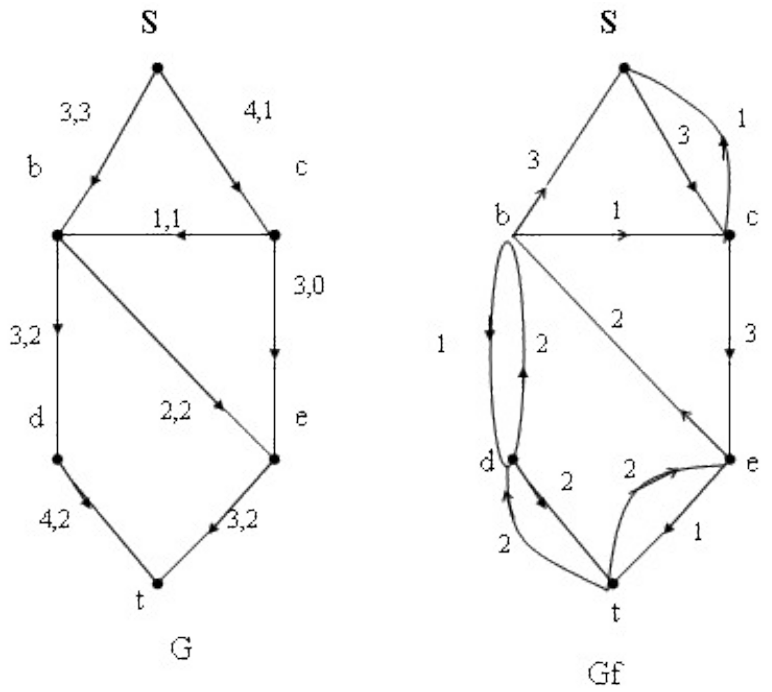
- Nếu e = (v,w) ∈ E với f(v,w) = 0, thì (v,w) ∈ E_f với trọng số c(v,w);

- Nếu e = (v,w) ∈ E với f(v,w) =c(v,w), thì (w,v) ∈ E_f với trọng số f(v,w);

- Nếu e = (v,w) ∈ E với 0 < f(v,w) < c(v,w), thì (v,w) ∈ E_f với trọng số c(v,w) - f (v,w) và (w,v) ∈ E_f với trọng số f(v,w).

Các cung của G_f đồng thời cũng là cung của G được gọi là cung thuận, các cung còn lại được gọi là cung nghịch. Đồ thị G_f được gọi là đồ thị tăng luồng.

Ví dụ 1: Các số viết cạnh các cung của G ở hình 9.1 theo thứ tự là khả năng thông qua và luồng trên cung.



Hình 9.1 Mạng G và luồng f. Đồ thị có trọng số G f tương ứng.

Giả sử P = (s = v_0, v_1, . . . , v_k = t) là một đường đi từ s đến t trên đồ thị tăng luồng G_f. Gọi ε là giá trị nhỏ nhất của các trọng số của các cung trên đường đi P. Xây dựng luồng f' trên mạng theo qui tắc sau:

	f(u,v) + ε , nếu (u,v) ∈ P là cung thuận
f'(u,v) =	f(u,v) - ε , nếu (v,u) ∈ P là cung nghịch
	f(u,v), nếu (u,v) ∈ P

Dễ dàng kiểm tra được rằng f' được xây dựng như trên là luồng trong mạng và val(f') = val(f) + ε . Ta sẽ gọi thủ tục biến đổi luồng vừa nêu là tăng luồng dọc theo đường P.

Định nghĩa 4. Ta gọi đường tăng luồng f là mọi đường đi từ s đến t trên đồ thị tăng luồng G(f).

Định lý 3 Các mệnh đề dưới đây là tương đương:

(i) f là luồng cực đại trong mạng;

(ii) không tìm được đường tăng luồng f;

(iii) val(f) = $c(X,X^*)$ với một lát cắt (X, X^*) nào đó.

Chứng minh.

(i) ⇒ (ii). Giả sử ngược lại, tìm được đường tăng luồng P. Khi đó ta có thể tăng giá trị luồng bằng cách tăng luồng dọc theo đường P. Điều đó mâu thuẫn với tính cực đại của luồng f.

(ii) ⇒ (iii). Giả sử không tìm được đường tăng luồng. Ký hiệu X là tập tất cả các đỉnh có thể đến được từ đỉnh s trong đồ thị G_f, và đặt $X^*=V\backslash X$. Khi đó (X, X^*) là lát cắt, và $f(v,w) = 0$ với mọi $v \in X^*$, $w \in X$ nên

val(f) = ? f(v,w) - ? f(v,w) = ? f(v,w) $v \in X$ $v \in X^*$ $v\in X$ $w\in X^*$ $w\in X$ $w\in X^*$

Với $v \in X$, $w \in X^*$, do $(v,w) \in G_f$, nên $f(v,w) = c(v,w)$. Vậy

val(f) = ? f(v,w) = ? c(v,w) = $c(X,X^*)$ $v \in X$ $v \in X$ $w\in X^*$ $w\in X^*$

(iii) ⇒ (i). Theo bổ đề 9.1, $val(f) \le c(X,X^*)$ với mọi luồng f và với mọi lát cắt (X,X^*). Vì vậy, từ đẳng thức $val(f) = c(X,X^*)$ suy ra luồng f là luồng cực đại trong mạng.

Thuật toán tìm luồng cực đại

Định lý 3 là cơ sở để xây dựng thuật toán lặp sau đây để tìm luồng cực đại trong mạng: Bắt đầu từ luồng với luồng trên tất cả các cung bằng 0 (ta sẽ gọi luồng như vậy là luồng không), và lặp lại bước lặp sau đây cho đến khi thu được luồng mà đối với nó không còn đường tăng:

Bước lặp tăng luồng (Ford-Fulkerson): Tìm đường tăng P đối với luồng hiện có. Tăng luồng dọc theo đường P.

Khi đã có luồng cực đại, lát cắt hẹp nhất có thể theo thủ tục mô tả trong chứng minh định lý 3. Sơ đồ của thuật toán Ford-Fulkerson có thể mô tả trong thủ tục sau đây:

```
Procedure Max_Flow;

(* Thuật toán Ford-Fulkerson *)

Begin

(* Khởi tạo: Bắt đầu từ luồng với giá trị 0 *)
```

for u ∈ V do

for v ∈ V do f(u,v) := 0;

stop := false;

while not stop do

if <Tìm được đường tăng luồng P> then <Tăng luồng dọc theo P>

else stop:=true;

End;

Để tìm đường tăng luồng trong G(f) có thể sử dụng thuật toán tìm kiếm theo chiều rộng (hay tìm kiếm theo chiều sâu), bắt đầu từ đỉnh s trong đó không cần xây dựng tường minh đồ thị G(f). Ford-Fulkerson đề nghị thuật toán gán nhãn chi tiết sau đây để giải bài toán luồng cực đại trong mạng. Thuật toán bắt đầu từ luồng chấp nhận được nào đó trong mạng (có thể bắt đầu từ luồng không) , sau đó ta sẽ tăng luồng bằng cách tìm các đường tăng luồng. Để tìm đường tăng luồng ta sẽ áp dụng phương pháp gán nhãn cho các đỉnh. Mỗi đỉnh trong quá trình thực hiện thuật toán sẽ ở một trong ba trạng thái: chưa có nhãn, có nhãn chưa xét, có nhãn đã xét. Nhãn của một đỉnh v gồm hai phần và có một trong hai dạng sau : [,] hoặc []. Phần thứ nhất +p(v) (-p(v)) chỉ ra là cần tăng giảm luồng theo cung (p(v),v)(cung (v, p(v)) còn phần thứ hai chỉ ra lượng lớn nhất có thể tăng hoặc giảm luồng theo cung này. Đầu tiên chỉ có đỉnh s được khởi tạo nhãn và nhãn của nó là chưa xét, còn tất cả các đỉnh còn lại đều chưa có nhãn. Từ s ta gán nhãn cho tất cả các đỉnh kề với nó và nhãn của đỉnh s sẽ trở thành đã xét. Tiếp theo, từ một đỉnh v có nhãn chưa xét ta lại gán nhãn cho tất cả các đỉnh chưa có nhãn kề với nó và nhãn của đỉnh v trở thành đã xét. Quá trình sẽ được lặp lại cho đến khi hoặc là đỉnh t trở thành có nhãn hoặc là nhãn của tất cả các đỉnh có nhãn đầu là đã xét nhưng đỉnh t vẫn không có nhãn. Trong trường hợp thứ nhất ta tìm được đường tăng luồng, còn trong trường hợp thứ hai đối với luồng đang xét không tồn tại đường tăng luồng (tức là luồng đã cực đại). Mỗi khi tìm được đường tăng luồng, ta lại tăng luồng theo đường tìm được, sau đó xoá tất cả các nhãn và đối với luồng mới thu được lại sử dụng phép gán nhãn các đỉnh để tìm đường tăng luồng. Thuật toán sẽ kết thúc khi nào đối với luồng đang có trong mạng không tìm được đường tăng luồng.

Hai thủ tục tìm đường tăng luồng có thể mô tả như sau :

Procedure Find-path;

{ Thủ tục gán nhãn đường tăng luồng

p[v], [v] là nhãn của đỉnh v;

VT là danh sách các đỉnh có nhãn chưa xét ;

c[u,v] là khả năng thông qua của cung (u,v),u,v V;

f[u,v] là luồng trên cung (u,v), (u,v V); }

BEGIN

p[s] := s ; [s] :=; VT := {s};

Pathfound := true;

While VT <> {} do

BEGIN

u VT ; (* lấy u từ VT *)

For vV do

If (v chưa có nhãn) then

Begin

If (c[u,v] >0) and (f[u,v] < c[u,v]) then

Begin

P[v] := u ; [v] := min {[u],c[u,v]-f[u,v] };

VT:=VT{v}; (* nạp v vào danh sách các đỉnh có nhãn *)

If v = t then exit;

End

Else

If (c[v,u] > 0) and (f[v,u] < 0) then

Begin

P[v] := u ;

```
[v] := min {[u] , f[u,v] };
VT:=VT{v};(* nạp v vào danh sách các đỉnh có nhãn *)
If v = t then exit;
End;
End;
End;
PathFound :=false;
End;
Procedure Inc_flow ;
{ thuật toán tăng luồng theo đường tăng }
Begin
v := t ; u := t ; tang := [t];
while u <> s do
begin
v := p[u];
if v > 0 then f[v,u] := f[v,u] + tang
else
begin
v := -v;
f[u,v] :=f[u,v] –tang;
end;
u := v ;
```

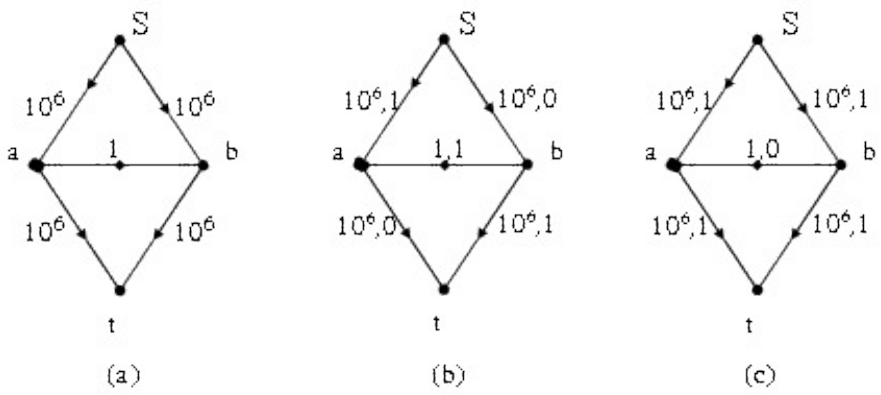
```
end;

Procedure FF;

{ thủ tục thể hiện thuật toán Ford_fulkerson }

Begin

(* khởi tạo bắt đầu từ luồng với giá trị 0 *)

For u V do

For v V do f[u,v] :=0;

Stop := false;

While not Stop do

begin

find_path;

If pathfound then

Inc_flow

Else Stop:=true;

End;

< Luồng cực đại trong mạng là f[u,v], u,v V >

< Lát cắt hẹp nhất là (VT , V\ VT) >

End;
```

Giả sử là khả năng thông qua của tất cả các khung của đồ thị là các số nguyên. Khi đó sau mỗi lần tăng luồng, giá trị luồng sẽ tăng lên ít nhất là 1. Từ đó suy ra thuật toán Ford_Fulkerson sẽ dừng sau không quá val(f*) lần tăng luồng và cho ta luồng cực đại trong mạng. Đồng thời, rõ ràng $f^*(u,v)$ sẽ là số nguyên đối với mỗi cung $(u,v) \in E$. Từ đó ra có các kết quả sau:

Định lý 4 (Định lý về luồng cực đại trong mạng và lát cắt hẹp nhất). Luồng cực đại trong mạng bằng khả năng thông qua của lát cắt hẹp nhất.

Định lý 5 (Định lý về tính nguyên). Nếu tất cả các khả năng thông qua là các số nguyên thì luồng tìm được cực đại với luồng trên các cung là các số nguyên.

Tuy nhiên, nếu các khả năng thông qua là các số rất lớn thì giá trị của luồng cực đại cũng có thể là rất lớn và khi đó thuật toán mô tả ở trên sẽ đòi hỏi thực hiện rất nhiều bước tăng luồng. Ví dụ trong hình 2 sẽ minh hoạ cho điều này. Hình 2(a) mô tả mạng cần xét với các khả năng thông qua trên các cung. Hình 2(b) mô tả luồng trên các cung (số thứ hai bên cạnh cung) sau khi thực hiện tăng luồng dọc theo đường tăng luồng (s, a, b, t). Hình 2(c) mô tả luồng trên các cung sau khi thực hiện tăng luồng dọc theo đường tăng luồng (s, b, a, t). Rõ ràng, sau 2.10^6 lần tăng luồng theo đường (s, a, b, t) và (s, b, a, t) một cách luân phiên ta thu được luồng cực đại.



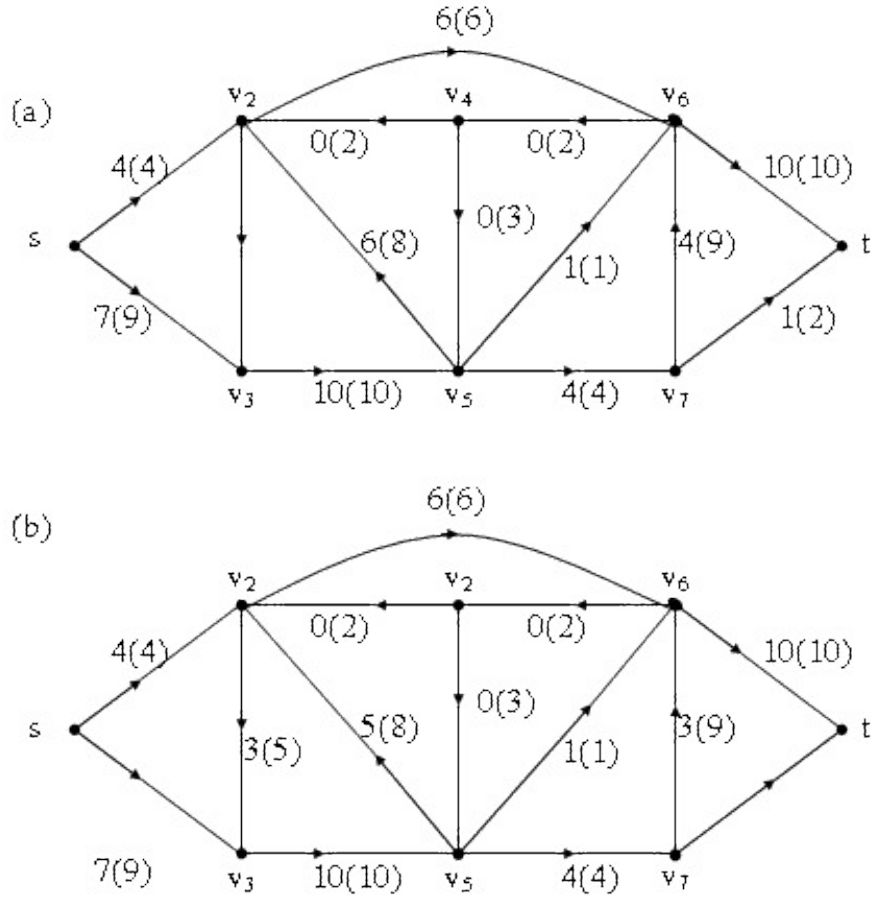
Hình 2 Ví dụ tồi tệ đối với thuật toán Ford_Fulkerson.

Hơn thế nữa, nếu các khả năng thông qua là các số vô tỉ, người ta còn xây dựng được ví dụ để cho thuật toán không dừng, và tệ hơn nếu dãy các giá trị luồng xây dựng theo thuật toán hội tụ thì nó còn không hội tụ đến giá trị luồng cực đại. Như vậy, muốn thuật toán làm việc hiệu quả, việc lựa chọn đường tăng luồng cần được tiến hành hết sức cẩn thận.

Edmonds và Karp chỉ ra rằng nếu đường tăng luồng được chọn là đường ngắn nhất từ s đến t trên đồ thị G_f. Điều đó có thể thực hiện, nếu trong thủ tục tìm đường tăng Find_Path mô tả ở trên, danh sách VT được tổ chức dưới dạng QUEUE (nghĩa là ta thực hiện tìm đường tăng bởi thủ tục tìm kiếm theo chiều rộng) thì thuật toán sẽ kết thúc sau không quá mn/2 lần sử dụng đường tăng luồng. Nếu để ý rằng, tìm kiếm theo chiều rộng trên đồ thị đòi hỏi thời gian O(m+n), thì thuật toán thu được sẽ có độ phức tạp tính toán là $O(nm^2)$.

Nhờ cách tổ chức tìm đường tăng khéo léo hơn, người ta đã xây dựng được thuật toán với độ phức tạp tính toán tốt hơn như: $O(n^2m)$ (Dinic, 1970). $O(n^3)$ (Karzanov, 1974), $O(n^2m^2)$, (Cherkasky, 1977), O(nm log n) (Sleator, - Tarrjan, 1980).

Ta kết thúc mục này bởi ví dụ minh hoạ cho thuật toán Ford_Fulkerson sau đây. Hình 3(a) cho mạng G cùng với thông qua của tất cả các cung và luồng giá trị 10 trong nó. Hai số viết bên cạnh mỗi cung là khả năng thông qua của cung (số trong ngoặc) và luồng trên cung. Đường tăng luồng có dạng (s, v_3, v_2, v_6, v_7, t). Ta tính được ε (t) = 1, giá trị luồng tăng từ 10 lên 1. Hình 3 (b) cho luồng thu được sau khi tăng luồng theo đường tăng tìm được.



Hình 3 Tăng luồng dọc theo đường tăng.

Luồng trong hình 3(b) đã là cực đại. Lát cắt hẹp nhất

X = { s, v_2, v_3, v_5}, X* = { v_4, v_6, v_7, t} .

Giá trị luồng cực đại là 11.

Bài 8: Một số ứng dụng của đồ thị

Một số ứng dụng của đồ thị(phần 1)

Một số ứng dụng trong đồ thị

Bài toán luồng cực đại có rất nhiều ứng dụng trong việc giải bài toán tổ hợp. Khó khăn chính ở đây là phải xây dựng mạng tương ứng sao cho việc tìm luồng cực đại trong nó sẽ tương đương với việc giải bài toán tổ hợp đặt ra. Mục này sẽ giới thiệu một số bài toán như vậy.

Bài toán đám cưới vùng quê

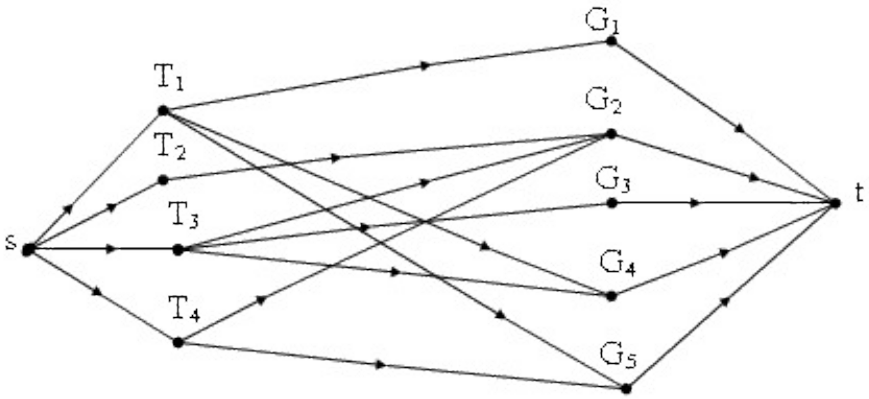
Có m chàng trai ở một vùng quê nọ. Đối với mỗi chàng trai ta biết các cô gái mà anh ta vừa ý. Hỏi khi nào thì có thể tổ chức các đám cưới trong đó chàng trai nào cũng sánh duyên với các cô gái mà mình vừa ý.

Ta có thể xây dựng đồ thị với các đỉnh biểu thị các chàng trai và các cô gái, còn các cung biểu thị sự vừa ý của các chàng trai với các cô gái. Khi đó ta thu được một đồ thị hai phía.

Ví dụ. Có 4 chàng trai { T_1, T_2, T_3,T_4}và 5 cô gái { G_1, G_2, G_3,G_4, G_5}. Sự vừa ý cho trong bảng sau

Chàng trai	Các cô gái mà chàng trai ưng ý
T_1	G_1, G_4, G_5
T_2	G_2
T_3	G_2, G_3,G_4
T_4	G_2, G_4

Đồ thị tương ứng được cho trong hình 1.



Hình 1 Mạng tương ứng với bài toán đám cưới vùng quê.

Đưa vào điểm phát s và điểm thu t. Nối s với tất cả các đỉnh biểu thị các chàng trai, và nối t với tất cả các đỉnh biểu thị các cô gái. Tất cả các cung của đồ thị đều có khả năng thông qua bằng 1. Bắt đầu từ luồng 0, ta tìm luồng cực đại trong mạng xây dựng được theo thuật toán Ford-Fulkerson. Từ định lý về tính nguyên, luồng trên các cung là các số hoặc 1. Rõ ràng là nếu luồng cực đại trong đồ thị có giá trị $V_{max} = m$, thì bài toán có lời giải, và các cung với luồng bằng 1 sẽ chỉ ra cách tổ chức đám cưới thoả mãn điều kiện đặt ra. Ngược lại, nếu bài toán có lời giải thì $V_{max} = m$. Bài toán về đám cưới vùng quê là một trường hợp riêng của bài toán về cặp ghép trên đồ thị hai phía mà để giải nó có thể xây dựng thuật toán hiệu quả hơn.

Bài toán về hệ thống đại diện chung

Cho tập m phần tử $X=\{ z_1, z_2, \ldots ,z_m\}$. Giả sử $<A_1, A_2, \ldots, A_n>$ và $<B_1, B_2, \ldots, B_n>$ là hai dãy các tập con của X. Dãy gồm n phần tử khác nhau của X: $<a_1, a_2, \ldots ,a_n>$ được gọi là hệ thống các đại diện chung của hai dãy đã cho nếu như tìm được một hoán vị s của tập {1, 2,. . .,n} sao cho $< a_1, a_2, \ldots ,a_n>$ là hệ thống các đại diện phân biệt của hai dãy $<A_1, A_2, \ldots, A_n>$ và $<Bs_{(1)}, Bs_{(2)}, \ldots, Bs_{(n)}>$, tức là điều kiện sau được thoả mãn: $a_i \in A_i$? Bs $_{(i)}$, i = 1, 2, . . ,n. Xây dựng mạng G = (V, E) với tập đỉnh

V = { s, t} ? { $x_1, x_2, \ldots ,x_n$}? {$u_1, u_2, \ldots ,u_n$}? { $v_1, v_2, \ldots ,v_n$}? { $y_1, y_2, \ldots ,y_n$}.

$$E = \{ (s, x_i): 1 \le i \le n\} \cup \{ (x_i,u_j): \text{với } z_j \in A_i,\ 1 \le i \le n,\ 1 \le j \le m\} \cup$$

$$\{ (u_j,v_j): 1 \le j \le m\} \cup \{(v_j, y_i): \text{với } z_j \in B_i,\ 1 \le i \le n,\ 1 \le j \le m\} \cup \{ (y_i, t): 1 \le i \le n\}$$

trong đó đỉnh x_i tương ứng với tập A_i, đỉnh y_i tương ứng với tập B_i, các phần tử u_j, y_j tương ứng với phần tử z_j. Tập các cung của mạng G được xác định như sau

Khả năng thông qua của tất cả các cung được đặt bằng 1. Dễ dàng thấy rằng hệ thống đại diện chung của hai dãy và tồn tại khi và chỉ khi trong mạng G=(V,E) tìm được luồng với giá trị n. Để xét sự tồn tại của luồng như vậy có thể sử dụng thuật toán tìm luồng cực đại từ s đến t trong mạng G=(V, E).

Bài toán tối ưu rời rạc

Trong mục này ta sẽ trình bày thuật toán được xây dựng dựa trên thuật toán tìm luồng cực đại để giải một bài toán tối ưu rời rạc là mô hình toán học cho một số bài toán tối ưu tổ hợp.

Xét bài toán tối ưu rời rạc:

$$f(x_1,x_2,\ldots,x_n) = \max_{1\le i\le m} \sum_{j=1}^{n} x_{ij} \to \min \quad (1)$$

$$\sum_{j=1}^{n} a_{ij}\, x_{ij} = p_i,\ i=1, 2,\ldots,m \quad (2)$$

$$x_{ij}=0 \text{ hoặc } 1,\ j=1, 2,\ldots,n \quad (3)$$

với điều kiện

□□

trong đó $a_{ij} \in \{0,1\}$, i = 1, 2, . . . , m; j=1, 2, . . . n, p_i –nguyên dương, i = 1, 2, . . .,m.

Bài toán (1)-(3) là mô hình toán học cho nhiều bài toán tối ưu tổ hợp thực tế. Dưới đây ta dẫn ra một vài ví dụ điển hình.

Bài toán phân nhóm sinh hoạt. Có m sinh viên và n nhóm sinh hoạt chuyên đề. Với mỗi sinh viên i, biết

+ a_{ij} =1, nếu sinh viên i có nguyện vọng tham gia vào nhóm j,

+ a_{ij} =0, nếu ngược lại,

+ và p_{ij} là số lượng nhóm chuyên đề mà sinh viên i phải tham dự,

i = 1, 2, . . . ,m; j=1, 2,. . . ,n.

Trong số các cách phân các sinh viên vào nhóm chuyên đề mà họ có nguyện vọng tham gia và đảm bảo mỗi sinh viên i phải tham gia đúng pi nhóm, hãy tìm cách phân phối với số người trong nhóm có nhiều sinh viên tham gia nhất là nhỏ nhất có thể được.

Đưa vào biến số

x_{ij} = 1, nếu sinh viên i tham gia vào nhóm j,

x_{ij} = 0, nếu ngược lại,

i = 1, 2, . . .,m, j=1, 2,. . .,n, khi đó dễ thấy mô hình toán học cho bài toán đặt ra chính là bài toán (1)-(3).

Bài toán lập lịch cho hội nghị. Một hội nghị có m tiểu ban, mỗi tiểu ban cần sinh hoạt trong một ngày tại phòng họp phù hợp với nó. Có n phòng họp dành cho việc sinh hoạt của các tiểu ban. Biết

a_{ij} = 1, nếu phòng họp i là thích hợp với tiểu ban j,

a_{ij}=0, nếu ngược lại,

i = 1, 2, . . .,m, j =1, 2,. . .,n. Hãy bố trí các phòng họp cho các tiểu ban sao cho hội nghị kết thúc sau ít ngày làm việc nhất.

Đưa vào biến số

x_{ij} = 1, nếu bố trí tiểu ban i làm việc ở phòng j,

x_{ij} =0, nếu ngược lại,

i =1, 2, . . .,m, j =1, 2,. . .,n, khi đó dễ thấy mô hình toán học cho bài toán đặt ra chính là bài toán (1)-(3), trong đó p_i=1, i =1, 2, . . .,m.

Một số bài toán liên quan đến việc tổ chức mạng vận chuyển bưu chính.

Các bài toán tối ưu trên mạng, một phần của lý thuyết đồ thị hữu hạn là một lý thuyết toán học được ứng dụng rộng rãi trong kinh tế, quân sự.

Người đặt nền móng đầu tiên cho lý thuyết đồ thị là nhà toán học Euler, với “bài toán bảy cái cầu” nổi tiếng vào năm 1736.

Trong quá trình khai thác các khía cạnh khác nhau của bài toán, các nhà toán học đã dần dần đặt cơ sở lý luận cho một lý thuyết toán học mới ra đời, đó là lý thuyết đồ thị hữu hạn (*lý thuyết Graph)*.

Đến nay lý thuyết đồ thị hữu hạn đã được nghiên cứu ứng dụng trong hầu hết các lĩnh vực của hoạt động kinh tế xã hội, là công cụ toán học sắc bén trong nghiên cứu các hệ thống kỹ thuật -công nghệ, hệ thống kinh tế -xã hội, hệ thống quân sự, hệ thống bưu chính viễn thông v.v...

Trong những năm gần đây nhờ sự hỗ trợ của công nghệ thông tin và máy tính điện tử, lý thuyết graph càng trở thành công cụ hiệu quả, năng động giải quyết nhiều bài toán liên quan đến nghiên cứu phân tích hệ thống.

Mô hình định tuyến mạng đường thư cấp 1

Mạng đường thư cấp một thực chất là một đồ thị có các đỉnh là các nút trung tâm Bưu chính và các Bưu điện trung tâm. Vận chuyển giữa các nút mạng có thể qua đường trực tiếp hoặc qua các nút trung gian. Do vậy xuất hiện bài toán lựa chọn tuyến đường vận chuyển. Tức là phải chỉ ra cách vận chuyển từ một nút bất kỳ tới một nút bất kỳ khác cần phải qua nút trung gian nào. Giữa 2 đỉnh sẽ có cung liên kết nếu chúng có đường vận chuyển trực tiếp với nhau.

Để giải bài toán xác định đường vận chuyển bưu chính cần có các khái niệm sau:

1. Lưu lượng (luồng) vận chuyển bưu gửi: Số lượng bưu gửi xuất hiện tại một nút mạng bất kỳ cần phải chuyển tới một nút mạng khác. Đại lượng này tính trong một đơn vị thời gian (giờ, ngày, tuần, tháng), được gọi là tải trọng. Do đặc điểm không đồng đều của tải trọng theo ngày trong tuần và tháng trong năm nên ta chỉ xét tải trọng trung bình ngày để lập kế hoạch vận chuyển (thống kê một tháng tiêu biểu và chia trung bình cho một ngày). Trong mô hình, tải trọng giữa các nút mạng được biểu diễn dưới dạng ma trận mà phần tử (ij) được hiểu là tải trọng trong một ngày từ nút mạng i tới nút mạng j.

2. Khả năng lưu thoát của nút mạng là số lượng bưu gửi có thể được khai thác tại một nút mạng trong một ngày. Khả năng lưu thoát phụ thuộc vào nhiều yếu tố như diện tích mặt bằng, mức độ cơ giới hoá, tự động hoá, tổ chức sản xuất, mức độ không đồng đều của tải trọng, tần số và thời gian khởi hành của các phương tiện vận chuyển...

3. Giá trị của các cung (chiều dài các cung) là giá thành vận chuyển một đơn vị sản phẩm theo từng cung liên kết, hoặc thời gian vận chuyển giữa các nút mạng. Đơn vị sản phẩm có thể là một túi thư, một container hoặc một bưu kiện tuỳ vào bài toán cụ thể. Giá thành vận chuyển một đơn vị sản phẩm được biểu diễn qua chiều dài cung hoặc thời gian vận chuyển giữa các nút mạng.

Bài toán lập kế hoạch vận chuyển bưu gửi

Trước tiên, xét hai nút mạng cần trao đổi bưu gửi, một nút mạng là nguồn *ws*, một nút là đích *wt*, luồng tải trọng từ nguồn tới đích sẽ là: *(x1 ,...,xj ,...,xn)*

các cung d_j với *xj*> 0 tạo thành tuyến vận chuyển tải trọng *xj*từ nguồn *ws*tới đích *wt* , tuyến vận chuyển này được xác định là một tập hợp các nút mạng (hay tập hợp các cung) tham gia vào tuyến vận chuyển này.

Như vậy, bài toán định tuyến mạng vận chuyển bưu gửi là cần xác định luồng bưu gửi từ một nút mạng tới một nút mạng khác cần phải qua các nút trung gian nào để tối thiểu hoá chi phí vận chuyển của toàn bộ mạng, đồng thời việc lựa chọn tuyến đường cần thoả mãn các điều kiện ràng buộc về thời gian toàn trình và khả năng lưu thoát của từng nút mạng.

Trong mạng vận chuyển bưu chính, trên mạng đồng thời thực hiện nhiều luồng trao đổi, mỗi một luồng có nút khởi đầu và nút kết thúc. Do vậy, cần đưa vào ký hiệu từng luồng là véc tơ *xq*:

$x^q = (x^q_1,..., x^q_j,...,x^q_n)$

*Xq*cần thoả mãn điều kiện không âm và điều kiện bảo toàn luồng nghĩa là:

AXq =Vq *Xq ≥ 0*Trong đó:V*q*: véctơ tất cả các phần tử đều bằng 0, ngoại trừ hai phần tử tương ứng với nút mạng khởi đầu và nút kết thúc có giá trị là -v*q*và V*q*(v*R*là lưu lượng cần vận chuyển của mỗi luồng);

A: Ma trận liên kết cung nút chứa m dòng và n cột trong đó m là số nút mạng và n là số cung. Ma trận A chính là mô hình định tuyến mạng vận chuyển của từng luồng cần xác định.

Ma trận liên kết cung nút của graph G = (W,D), ký hiệu A=[a*ij*] có kích thước m x n với các phần tử được xác định như sau:

$$a_{ij} = \begin{cases} 1, \text{ nếu } w_i \text{ là đỉnh đầu của cung } d_j \\ -1, \text{ nếu } w_i \text{ là đỉnh cuối của cung } d_j \\ 0, \text{ nếu } w_i \text{ không là đỉnh đầu hoặc cuối của cung } d_j \end{cases}$$

Ngoài ra luồng X*q*còn phải thoả mãn điều kiện ràng buộc không được vượt quá khả năng khai thác của từng nút mạng w*i*.

Xét véc tơ P*i*(P*i*$_1$, P*i*$_2$, ... P*in*) trong đó P*i*= l nếu *dj*hướng tới đỉnh w*i*và P*i* = 0 nếu ngược lại. Véc tơ P*i*chính là dòng i của ma trận liên kết cung nút A mà trong đó tất cả các phần tử -l được thay bằng 0.

Như vậy, điều kiện ràng buộc về khả năng lưu thoát của các nút mạng là:

$$P_i \sum_{q=1}^{r} X^q \le h_i$$

Trong đó:

h*i*: Khả năng Lưu thoát của nút mạng W

r: Số đôi nút mạng trong mạng có trao đổi bưu gửi

Tiêu chí tối ưu của bài toán vận chuyển bưu gửi như sau:

Giả sử C (C$_1$,C$_2$,... C*n*) là Véctơ chi phí vận chuyển trong đó C$_j$ là cước vận chuyển l đơn vị sản phẩm qua cung d*j*(chiều dài cung d$_j$). Khi đó chi phí vận chuyển sẽ là:

$Z = CX^1 + ... + CXq + ... + CXr$

$= C(X^1 + ... + Xq + ... + Xr) \rightarrow \min$

Vậy mô hình vận chuyển tối ưu là:

$$\min[Z = C(X^1 + ... + X^q + ... + X^r)]$$

$$P_1(X^1 + ... + X^q + ... + X^r) \le h_1$$

.....

$$P_m(X^1 + ... + X^q + ... + X^r) \le h_m$$

$$AX^1 = V^1$$

$$AX^q = V^q$$

$$AX^r = V^r$$

Trong trường hợp không có điều kiện hạn chế về khả năng lưu thoát của các nút mạng, bài toán định tuyến mạng bưu chính chỉ đơn giản là bài toán tìm đường đi ngắn nhất

giữa từng đôi nút mạng. Bài toán tìm đường ngắn nhất được giải bằng thuật toán dán nhãn của Dijkstra.

Mô hình mạng đường thư trong thành phố

Mạng đường thư trong thành phố là một đồ thị đỉnh của nó là các bưu cục. Hai đỉnh của đồ thị sẽ được nối kết với nhau bằng cung liên kết nếu giữa chúng có tuyến đường đi. Trong thành phố giữa các bưu cục bao giờ cũng có các đường thư, nên đồ thị được kết nối theo kiểu điểm nối điểm. Đồ thị mạng đường thư trong thành phố là một đồ thị có hướng vì khoảng cách i tới j và j tới i có thể không trùng nhau (đường một chiều). Giá trị của cung được biểu diễn bằng khoảng cách hoặc thời gian vận chuyển giữa các nút mạng hoặc chi phí vận chuyển giữa các nút mạng.

Ta có chi phí vận chuyển giữa các nút mạng là:

$c_{ij} = kr_{ij}$

r_{ij} : Khoảng cách giữa các nút i và nút j (cần được xác định theo khoảng cách thực tế và phải lựa chọn r_{ij} là đường ngắn nhất, tức là phải thoả mãn điều kiện $r_{ij} \leq r_{ik} + r_{kj}$ do một cạnh tam giác nhỏ hơn tổng 2 cạnh còn lại).

k: Chi phí vận chuyển l km bằng ô tô.

Thời gian vận chuyển trên cung ij

$$t_{ij} = \frac{r_{ij}}{V_{ij} + t_{0j}}$$

V_{ij} : Vận tốc vận chuyển ô tô từ nút i tới nút j.

t_{0j} : Thời gian trao đổi tại nút mạng j.

Khi tổ chức mạng đường thư có thể sử dụng phương thức đường thẳng, đường vòng hoặc hỗn hợp.

Mạng đường vòng có ưu điểm là sử dụng các phương tiện vận chuyển hiệu quả hơn. Do vậy trong thành phố thường sử dụng đường vòng do tính kinh tế của nó.

Bài toán

Bài toán tổ chức mạng đường thư trong thành phố là xác định hành trình của từng chiếc ô tô phải qua các nút mạng nào, theo trình tự nào để đảm bảo chi phí vận chuyển toàn mạng là nhỏ nhất (hoặc tổng quãng đường hay tổng thời gian vận chuyển nhỏ nhất) đồng

thời thoả mãn các ràng buộc về thời gian vận chuyển của từng ô tô và dung lượng vận chuyển của từng ô tô.

Trong hệ thống khai thác tập trung tồn tại một Bưu điện trung tâm duy nhất. Nếu chia các nút mạng ra làm hai loại nguồn và đích, nút mạng trung tâm sẽ là nguồn, các nút mạng còn lại sẽ là đích hoặc ngược lại.

Trong mô hình vận chuyển bưu gửi trong thành phố, các đỉnh đồ thị được đặc trưng bởi số lượng bưu gửi mà nó cần nhận được từ q_i hoặc ngược lại cần gửi đi r_i.

$$r_0 = \sum_{i=1}^{N} q_i$$

Trong đó:

0: nút nguồn.

i = l ÷ N : đích

Trong hệ thống khai thác phân tán khi đó đồ thị sẽ được chia thành các đồ thị con, mỗi một đồ thị chỉ có một nút mạng nguồn duy nhất và việc giải bài toán thực tế là giải từng bài toán con.

Nếu mạng vận chuyển trong thành phố chủ yếu bằng ô tô, ta giả sử:

M: số ô tô toàn mạng

Q_j - dung lượng của j ô tô, phụ thuộc vào loại ô tô

T - thời gian vận chuyển tối đa cho phép trên một đường thư.

T được xác định dựa trên các định mức (T = 2 giờ).

$Q_j = \min (P_j / b, V_j / d)$

Trong đó P_j : tải trọng của ô tô;

V_j : thể tích vận chuyển của ô tô;

b: Khối lượng trung bình của túi thư;

d: thể tích trung bình của túi thư.

Mô hình toán học

Giả sử gọi x*ijk* là ẩn cần tìm, x*ijk* = 1 nếu trong tuyên vòng k, đỉnh j sẽ được tới ngay sau đỉnh i, X*ijk* = 0 trong trường hợp ngược lại, khi đó mô hình toán học của bài toán mạng đường thư trong thành phố là:

$$Z = \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij} \sum_{k=1}^{M} x_{ijk} \rightarrow \min$$

Biến X*ijk* cần thoả mãn các ràng buộc sau:

$\sum_{i=0}^{N} \sum_{k=1}^{M} x_{ijk} = 1$	j = 0 ÷ N	(1)
$\sum_{i=0}^{N} X_{ipk} - \sum_{j=0}^{N} x_{pjk} = 0$	k = 1 ÷ M, p = 0 ÷ N	(2)
$\sum_{i=0}^{N} q_i \sum_{j=0}^{N} x_{ijk} \le Q_k$	k = 1 ÷ M	(3)
$\sum_{i=0}^{N} \sum_{j=0}^{N} t_{ij} x_{ijk} + \sum_{i=1}^{N} t_{0i} \sum_{j=0}^{N} x_{ijk} \le T$	k = 1 ÷ M	(4)

Trong đó t_{0i} : thời gian trao đổi tại nút mạng i.

(l): Do mỗi đỉnh đồ thị chỉ thuộc một tuyến đường vòng;

(2): Đối với mỗi một đỉnh, số lượng các cung đi vào và đi ra phải bằng nhau tại một đỉnh;

(3): Ràng buộc về dung lượng của ô tô;

(4): Ràng buộc về thời hạn vận chuyển của từng ô tô.

Đây là bài toán tổng quát về mạng vận chuyển thư trong thành phố. Bài toán tìm hành trình của bưu tá qua n điểm là trường hợp riêng khi chỉ có 1 tuyến đường vòng qua n điểm (M=1), còn bài toán này cần xác định M tuyến đường cho M ô tô và cần thoả mãn các hạn chế (ràng buộc) về chỉ tiêu thời gian và dung lượng vận chuyển của ô tô.

Một số ứng dụng của đồ thị(phần 2)

Một số ứng dụng trong đồ thị

Bài tập

Bài tập 1 :Cho G=(V,E) đồ thị có hướng trong đó không có cung (s,t). Chứng minh rằng số đường đi cơ bản nối hai đỉnh s và t là bằng số ít nhất các đỉnh của đồ thị cần loại bỏ để trong đồ thị không còn đường đi nối s với t.

Bài tập 2 :Xây dựng thuật toán tìm tập E_1 tất cả các cung của đồ thị mà việc tăng khả năng thông qua của bất kỳ cung nào trong E đều dẫn đến tăng giá trị của luồng cực đại trong mạng.

Bài tập 3 :Cho hai dãy số nguyên dương $\{p_i, i=1,2,\ldots,m\}$ và $\{q_j, j=1,2,\ldots,n\}$. Hãy xây dựng ma trận $A=\{a_{ij} : i=1,2,\ldots,m; j=1,2,\ldots n\}$ với các phần tử$a_{i\ j} \in \{0,1\}$ có tổng các phần tử trên dòng i là p_i , tổng các phần tử trên cột j là q_j.

Bài tập 4 :Có m chàng trai, n cô gái và k bà mối, mỗi bà mối p (p=1,2,…,k) có một danh sách L_p một số chàng trai và cô gái trong số các chàng trai và cô gái nói trên là khách hàng của bà ta. Bà mối p có thể se duyên cho bất cứ cặp trai gái nào là khách hàng của bà ta, nhưng không đủ sức tổ chức quá d_p đám cưới. Hãy xây dựng thuật toán căn cứ vào danh sách L_p, d_p, p=1,2,…,k; đưa ra cách tổ chức nhiều nhất các đám cưới giữa m chàng trai và n cô gái với sự giúp đỡ của các bà mối.

Bài tập 5 : **Chuyển bi**

Cậu bé vẽ N (N<=100) vòng tròn, đánh số từ 1 tới N và tô màu các vòng tròn đó (có thể có các vòng tròn có màu giống nhau), sau đó nối từng cặp các cung định hướng, mỗi cung có một màu nhất định. Các màu (của cung và vòng tròn) được đánh số từ 1 đến 100.

Cậu bé chọn 3 số nguyên khác nhau L, K và Q nằm trong phạm vi từ 1 tới N, đặt vào trong các vòng tròn số L và K mỗi vòng một hòn bi, sau đó bắt đầu di chuyển bi theo nguyên tắc sau:

- Bi chỉ được chuyển theo cung có màu trùng với màu của vòng tròn chứa viên bi thứ 2.

- Bi chỉ được chuyển theo chiều cung

- Hai viên bi không được đồng thời ở cùng một vòng tròn;

- Không nhất thiết phải di chuyển lần lượt các viên bi,

- Quá trình di chuyển kết thúc, khi một trong hai viên bi tới vòng tròn Q.

Hãy lập trình xác định cách di chuyển để chấm dứt quá trình sau một số ít nhất các bước chuyển.

Dữ liệu vào từ file BL.INP:

- Dòng đầu: 4 số nguyên N L K Q

- Dòng thứ 2: N số nguyên C_1, C_2,…,C_n; C_i là màu vòng tròn i

- Dòng thứ 3: số nguyên M (0 < M < 10000)

- M dòng sau: mỗi dòng 3 số nguyên A_i B_i D_i; xác định cung màu D_i từ vòng tròn A_i tới vòng tròn B_i.

Các số trên một dòng cách nhau một dấu cách.

Kết quả đưa ra file BL.OUT:

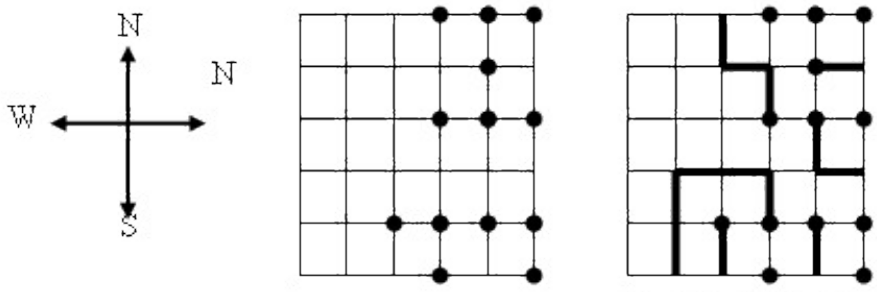
- Dòng đầu: CO hoặc KHONG, cho biết quá trình có kết thúc được hay không,

- Nếu dòng đầu là CO thì dòng 2 chứa số nguyên xác định số bước chuyển tối thiểu .

BL.INP	BL.OUT
5 3 4 12 3 2 1 482 1 24 1 54 5 24 5 25 1 33 2 22 3 45 3 13 5 1	CO3

Bài tập 6 : Bảng điện

Một lưới ô vuông được phủ trên một bảng điện hình vuông. Vị trí nằm trên giao của 2 đường kề của lưới sẽ được gọi là nút. Tất cả có nxn nút trên lưới.



Hình 2a. Bảng điện

Hình 2b. Lời giải

Có một số nút chứa tiếp điểm. Nhiệm vụ của bạn là cần nối các tiếp điểm với các nút ở trên biên của bảng bởi các đoạn dây dẫn (gọi là các mạch). Các mạch chỉ được chạy dọc theo các đường kẻ của lưới (nghĩa là không được chạy theo đường chéo). Hai mạch không được phép có điểm chung, vì vậy hai mạch bất kỳ không được phép cùng chạy qua cùng một đoạn đường kẻ của lưới cũng như không được chạy qua cùng một nút của lưới. Các mạch cũng không được chạy dọc theo các đoạn kẻ của lưới ở trên biên (mạch phải kết thúc khi nó gặp biên) và cũng không được chạy qua nút chứa tiếp điểm khác.

Ví dụ: Bảng điện và các tiếp điểm được cho trong hình 2a. Nút tô đậm trong hình vẽ thể hiện vị trí các tiếp điểm.

Yêu cầu: Viết chương trình cho phép nối được một số nhiều nhất các tiếp điểm với biên. Các tiếp điểm ở trên biên đã thỏa mãn đòi hỏi đặt ra, vì thế không nhất thiết phải thực hiện mạch nối chúng. Nếu như có nhiều lời giải thì chi cần đưa ra một trong số chúng.

Dữ liệu vào: file văn bản ELE.INP:

- Dòng đầu tiên chứa số nguyên n (3 <= n <= 15).

- Mỗi dòng trong số n dòng tiếp theo chứa n ký tự phân cách nhau bởi một dấu cách. Mỗi ký tự chỉ là 0 hoặc 1. Ký tự 1 thể hiện tiếp điểm, ký tự 0 thể hiện nút không có tiếp điểm trên vị trí tương ứng của lưới. Các nút được đánh số từ 1 đến n*n theo thứ tự từ trái sang phải, từ trên xuống dưới. Chỉ số của nút chứa tiếp điểm sẽ là chỉ số của tiếp điểm.

Kết quả: ghi ra file ELE.OUT:

- Dòng đầu tiên chứa k là số tiếp điểm lớn nhất có thể nối với biên bởi các mạch.

- Mỗi dòng trong số k dòng tiếp theo mô tả một mạch nối một trong số k tiếp điểm với biên theo qui cách sau: đầu tiên là chỉ số của tiếp điểm được nối, tiếp đến là dãy các ký tự mô tả hướng của mạch nối: E: đông, W: tây, N: bắc, S: nam. Giữa chỉ số và dãy ký tự phải có đúng một dấu cách, còn giữa các ký tự trong dãy ký tự không được có dấu cách.

Kết quả phải được đưa ra theo thứ tự tăng dần của chỉ số tiếp điểm.

Ví dụ:

ELE.IN	ELE.OUT
60 0 0 1 1 10 0 0 0 1 00 0 0 1 1 10 0 0 0 0 00 0 1 1 1 10 0 0 1 0 1	11 E16 NWN17 SE27 S28 NWWSS29 S

Bài tập 7:

Một khóa học gồm N môn học, môn học i phải học trong ti ngày. Giữa các môn học có mối quan hệ trước/sau: có môn học chỉ học được sau khi đã học một số môn học khác. Mối quan hệ đó được thể hiện bởi một mảng hai chiều A[i, j]; i, j = 1, …, N trong đó A[i, j] = 1/0 và A[i, i] bằng 0 với mọi i, A[i, j] = 1 khi và chỉ khi môn học i phải được dạy xong trước khi học môn j (ngày kết thúc môn i phải trước ngày bắt đầu môn j). Môn học i phải dạy trước môn học j nếu có một dãy môn học i_1, i_2, …, ik sao cho a[it, it_{+1}] = 1, 1 <= t <= k-1, i_1=i và ik=j. Nếu có một nhóm các môn học từng đôi một không có quan hệ trước/sau thì trong mỗi ngày, về nguyên tắc, ta có thể học đồng thời tất cả những môn học này (nếu không vi phạm quan hệ với các môn học khác). Mảng A[i, j] được gọi là bế tắc nếu có một dãy các môn học i_1, i_2,…, ik, k > 1, mà môn i_1 phải dạy trước môn i_2, môn i_2 phải dạy trước môn i_3, …, môn ik_{-1} phải dạy trước môn ik, môn ik phải dạy trước môn i_1.

Hãy viết chương trình làm các việc sau:

1. Hãy xét xem mảng A có bế tắc hay không.

2. Nếu mảng A không bế tắc, hãy tính xem khóa học có thể kết thúc trong thời gian nhanh nhất là bao nhiêu ngày.

3. Theo các học bảo đảm thời gian hoàn thành ngắn nhất ở câu 2, hãy tính xem một học sinh trong quá trình học phải học đồng thời trong một ngày nhiều nhất bao nhiêu môn.

Dữ liệu vào được cho bởi file text có tên MH.DAT trong đó số N ghi ở dòng thứ nhất, trong nhóm N dòng tiếp theo, dòng thứ i ghi N số A[i, 1], …, A[i, N] dòng cuối cùng ghi N số nguyên dượng ti không lớn hơn 30, 1 <= i <= N; N <= 30.

Kết quả ghi ra file TKB.DAT như sau: dòng thứ nhất ghi số 1/0 tùy theo mảng A bế tắc / không bế tắc. Nếu dòng thứ nhất ghi số 0, ta mới ghi tiếp kết quả câu 2 và 3.

Kết quả câu 2 ghi tiếp vào file TKB.DAT N+1 dòng như sau: dòng dầu ghi số T là số ngày tối thiểu có thể hoàn thành khóa học, tiếp theo là N dòng trong đó dòng thứ i ghi 2

số X, Y với ý nghĩa môn học thứ i học từ ngày thứ X đến ngày thứ Y (chú ý rằng Y – X = ti – 1).

Kết quả câu 3 ghi tiếp vào file TKB.DAT như sau: dòng thứ nhất ghi 2 số Z, W với ý nghĩa trong ngày Z phải học W môn (W là số nhiều nhất các môn học phải học đồng thời trong một ngày), tiếp theo là một dòng ghi tên các môn học phải học đồng thời trong ngày Z.

Trong các câu 2 và 3, có thể có nhiều lời giải tương đương chỉ cần đưa ra một lời giải.

Ví dụ 1

MH.DAT	TKB.DAT
4 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1	1

Ví dụ 2

MH.DAT	TKB.DAT
7 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 2 2 8 4 10 2 3	0 22 1 2 3 4 1 8 12 13 22 13 14 15 17 1 2 1 3

Bài tập 8: Giám đốc một công ty quyết định tổ chức buổi tiệc trà gặp gỡ toàn thể nhân viên trong công ty. Công ty được tổ chức theo mô hình phân cấp lãnh đạo và mối quan hệ *thủ trưởng – nhân viên* tạo thành cây có gốc là giám đốc. Để đảm bảo không khí tự nhiên, giám đốc quyết định không để thủ trưởng và nhân viên dưới quyền ngồi cùng một bàn. P gọi là thủ trưởng của Q, nếu P là thủ trưởng trực tiếp của Q hoặc tồn tại dãy P_1, P_2, …, P_k (1 < k), sao cho P = P_1, Q = P_k và P_i là thủ trưởng trực tiếp của P_{i+1} (i = 1, 2, … , k -1). Tất cả mọi người trong công ty được đánh số từ 1 đến N (N là tổng số người trong công ty với giám đốc bắt đầu từ 1).

+ **Yêu cầu:** tính số lượng bàn ít nhất cần thiết để có thể bố trí cho mọi người ngồi theo yêu cầu nêu trên và cho một phương án bố trí người ở mỗi bàn.

+ **Dữ liệu vào:** file text COMPANY.INP, dòng đầu tiên là số nguyên m – số ghế tối đa cho một bàn, dòng thứ 2 – số nguyên N – số người trong công ty, dòng thứ ba (và các dòng sau nếu cần) là dãy số nguyên, các số cách nhau ít nhất một dấu cách hoặc nhóm ký tự xuống dòng, số nguyên thứ i trong dãy cho biết ai là thủ trưởng trực tiếp của nhân viên i. Giám đốc không có thủ trưởng nên số này bằng 0. 2 <= m <= 10, 1 <= N <= 200.

+ **Kết quả:** đưa ra file text COMPANY.OUT, dòng đầu tiên là số bàn ít nhất cần thiết, các dòng sau mỗi dòng tương ứng với một bàn và chứa dãy số nguyên xác định ai được bố trí ngồi sau bàn đó.

Ví dụ:

COMPANY.INP

4

13

0 1 9 9 9 2 2 1 1 7 8 8 10

File kết quả COMPANY.OUT có thể có nội dung:

5

13 3 4 5

10 6 8

7 9 11 12

2

1

Bài tập 9: Trên bàn cờ NxN, hãy tìm cách sắp xếp số lượng tối đa các con hậu sao cho không con nào có thể ăn con nào.

Bài tập 10: Cho 1 đồ thị có hướng G, hãy tìm một tập hợp X_0 ít nhất các đỉnh của G sao cho mọi đỉnh i của G hoặc thuộc X_0 hoặc i nối trực tiếp với định j thuộc X_0.

Bài tập 11: Trên bàn cờ NxN, hãy tìm cách sắp xếp số lượng tối thiểu các con hậu sao cho mọi ô cờ trên bàn cờ bị chiếu bởi ít nhất 1 con.

Bài tập 12: Một ký túc xá có 15 cô gái. Hàng ngày các cô đi chơi với nhau theo bộ 3. Hỏi có thể đưa các cô đi chơi trong tối đa bao nhiêu ngày để không có 2 cô nào đi chung trong một bộ 3 quá 1 lần.

Hãy tổng quát hóa bài toán.

Bài tập 13: Một số hải cảng x_1, x_2, x_3… có các mặt hàng mà các hải cảng y_1, y_2, y_3…cần đến. Lượng hàng có ở x_i là s_i và yêu cầu hàng hóa của y_i là d_i. Nếu có tàu đi từ x_i tới y_j thì ta ký hiệu c_{ij} là tổng lượng hàng mà các tàu có thể vận chuyển từ x_i tới y_j. Vậy có thể thỏa mãn mọi yêu cầu không? Tổ chức vận chuyển ra sao? Hãy viết chương trình giải quyết bài toán trên.

Bài tập 14: Trong một cuộc du lịch, m gia đình phân nhau đi trên n xe. Các gia đình tương ứng có r_1, r_2, …, r_m người và các xe tương ứng có s_1, s_2, …, s_n chỗ ngồi. Hãy tìm cách phân phối sao cho 2 người cùng gia đình không ngồi chung một xe hoặc cho biết không thể làm như vậy.

Bài tập15: Trong một trường trung học, mỗi học sinh nữ có m bạn nam và mỗi học sinh nam có m bạn nữ. Hãy chỉ ra cách sắp xếp để mỗi cô gái có thể lần lượt khiêu vũ với các bạn trai của mình và các chàng trai có thể lần lượt khiêu vũ với các bạn gái của mình.

Bài 16: Một nhà in phải sản xuất n cuốn sách bằng 2 máy: một để in, một để đóng sách. Gọi a_k là thời gian cần cho việc in cuốn thứ k và b_k là thời gian cần cho việc đóng cuốn đó. Tất nhiên là sách phải in xong mới đóng, do đó máy đóng có thể phải chờ đợi lâu hay chóng. Vậy tiến hành theo thứ tự nào để có thể xong việc sớm nhất.

Bài tập 17: Ổn định

Trong một lớp có N dãy bàn và mỗi dãy có M chỗ ngồi. Trong lớp có K cán sự lớp. Mỗi một cán sự cầm một đề bài tập. Các cán sự này có nhiệm vụ chuyển đề bài tập đến các học sinh khác ngồi kề mình ở phía trước, sau, trái và phải. Sau khi các cán sự làm xong công việc của mình, mỗi học sinh thông báo số lượng đề bài tập mình đã nhận được. Dựa trên thông tin này hãy xác định vị trí của các cán sự trong lớp.

Bài tập 18: Có một máy thu và một máy phát tín hiệu. Giả sử máy phát có thể phát đi 5 loại tín hiệu khác nhau a, b, c, d, e. Ở máy thu mỗi tín hiệu có thể được hiểu theo 2 cách khác nhau: tín hiệu a có thể hiểu p hay q, tín hiệu b có thể hiểu q hay r, ... 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.

Tài liệu tham khảo

TÀI LIỆU THAM KHẢO

Sách giáo trình

[1]**.** *Toán rời rạc 2:Lý thuyếtĐồ thị và ứng dụng*, Khoa CNTT, trường ĐHSPKT Hưng Yên, 2008.

Sách tham khảo

[1]. *Lý thuyết tổ hợp và đồ thị*, Ngô Đắc Tân, Viện Toán Học, NXB Đại học Quốc Gia Hà Nội, 2003.

[2]. **Toán rời rạc**, **Nguyễn Đức Nghĩa***,* **Nguyễn**Tô Thành, NXB Giáo **dục**, Hà nội 1997.

[3]. *Discrete Mathematics and Its Applications*, 6th Edition, Kenneth H. Rosen, McGraw Hill, 2007.

[4]. *Handbook of Discrete and Combinatorial Mathematics*, Kenneth H.Rosen (chief of editor), CRC Press, 2000.

Tham gia đóng góp

Tài liệu: Lý thuyết đồ thị

Biên tập bởi: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://voer.edu.vn/c/cad51e6d

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Mở đầu

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/8607420a

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài 1: Các khái niệm cơ bản của Lý thuyết đồ thị.

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/6687aa1e

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các khái niệm cơ bản của Lý thuyết đồ thị. Đồ thị Euler (phan 1)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/e8e422d5

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Các khái niệm cơ bản của Lý thuyết đồ thị. Đồ thị Euler (phan 2)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/d9962958

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài 3: Đồ thị hamilton

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/cd42b631

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài 4: Cây và cây khung của đồ thị

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/09902196

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài 5: Bài toán cây khung nhỏ nhất

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/92b6ca71

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài toán tìm đường đi ngắn nhất

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/257dc760

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài toán tìm đường đi ngắn nhất (Tiếp)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/809c6969

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Bài 7: Bài toán luồng cực đại trong mạng

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/1ea96ed5

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Một số ứng dụng của đồ thị(phần 1)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/d7d2dc5a

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Một số ứng dụng của đồ thị(phần 2)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/8fd871a0

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Module: Tài liệu tham khảo

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: http://www.voer.edu.vn/m/84d50ee5

Giấy phép: http://creativecommons.org/licenses/by/3.0/

Chương trình Thư viện Học liệu Mở Việt Nam

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER (www.voer.edu.vn) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER) . Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.