Chương 5: Đồ thị

1. Các khái niệm

1.1. Định nghĩa đồ thị

Đồ thị G(V,E) bao gồm một tập hữu hạn V các đỉnh (hay nút) và một tập hữu hạn E các cặp đỉnh mà ta gọi là cung (hay cạnh).

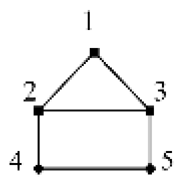
Ví dụ 1: Một mạng gồm các máy tính và các kênh điện thoại nối các máy tính này là một đồ thị.

Ví dụ 2: Một mạng gồm các thành phố, thị xã và các đường bộ nối các thành phố, thị xã là một đồ thị.

1.2. Định nghĩa đồ thị vô hướng

Đồ thị vô hướng G=(V,E) bao gồm V là tập các đỉnh và E là tập các cặp đỉnh không có thứ tự gọi là các cung.

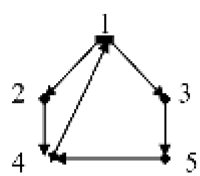
Ví dụ:



1.3. Định nghĩa đồ thị có hướng

Đồ thị có hướng G=(V,E) bao gồm V là tập các đỉnh và E là tập các cặp đỉnh có thứ tự gọi là các cung.

Ví dụ:



* Nếu (v1, v2) là một cung trong tập E(G) thì v1 và v2 gọi là lân cận của nhau.

Ví dụ trên 1,2 là lân cân, 1,3 là lân cận.

* Một đường đi từ đỉnh u đến đỉnh v trong đồ thị là một dãy các đỉnh

u=x0, x1, ..., xn-1, xn=v mà dãy các cạnh (x0, x1), (x1, x2), ..., (xn-1, xn) là các cung thuộc E(G) .

* Số lượng cung trên đường đi gọi là độ dài của đường đi.

Ví dụ đường đi từ 1 đến 4 có độ dài là 2.

* Đường đi đơn: Là đường đi mà mọi đỉnh trên đó, trừ đỉnh đầu và đỉnh cuối đều khác nhau.

* Một chu trình là một đường đi đơn mà đỉnh đầu và đỉnh cuối trùng nhau.

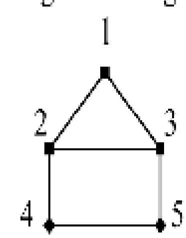
Ví dụ: 1→ 3 → 5→ 4→1

* Trong đồ thị G hai đỉnh u và v gọi là liên thông nếu có một đường đi từ u đến v .

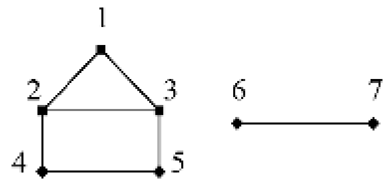
Ví dụ: 1, 4 là liên thông. 1, 5 là liên thông.

* Đồ thị G gọi là liên thông nếu với mọi cặp đỉnh phân biệt v_i, v_j trong V(G) đều có một đường đi từ v_i với v_j.

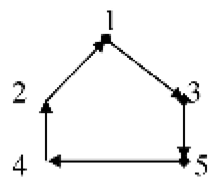
Ví dụ: Đồ thị sau là đồ thị vô hướng liên thông



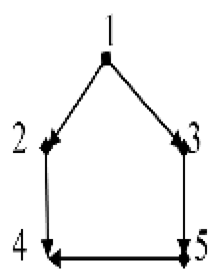
Ví dụ: Đồ thị sau là đồ thị vô hướng không liên thông



Ví dụ: Đồ thị sau là đồ thị có hướng liên thông



Ví dụ: Đồ thị sau là đồ thị có hướng không liên thông



* Đồ thị mà mỗi cung gắn với 1 giá trị nào đó được gọi là đồ thị có trọng số.

Ví dụ: mạng lưới giao thông, mỗi tuyến đường có độ dài, nên đó là đồ thị có trọng số.

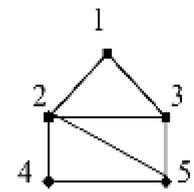
* Đồ thị vô hướng G=(V,E), bậc của đỉnh v là số cung liền thuộc với đỉnh đó.

Kí hiệu là: deg(v)

Ví dụ: Đồ thị vô hương sau

deg(1)=2

deg(2)=4



2. Biểu diễn đồ thị

2.1. Biểu diễn bằng ma trận lân cận (ma trận kề)

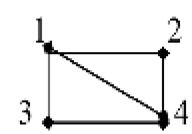
Xét đồ thị G(V,E), V gồm n đỉnh (n>=1), giả sử các đỉnh được đánh số thứ tự theo một quy định nào đó. Ma trận lân cận A biểu diễn G là một ma trận vuông kích thước nxn. Các phần tử của ma trận có giá trị 0 hoặc 1.

A_{ij} = 1 khi tồn tại cung (v_i, v_j) trong E.

A_{ij} = 0 khi không tồn tại cung (v_i, v_j) trong E.

* Nếu đồ thị vô hướng thì ma trận A đối xứng qua đường chéo chính.

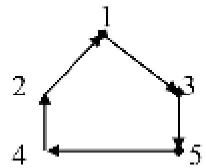
Ví dụ 1: Đồ thị vô hương sau



Ma trận lân cận A là ma trân vuông cấp 4 có giá trị như sau :

	1	2	3	4
1	0	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	0

Ví dụ 2: Đồ thị đồ thị có hướng sau



Ma trận lân cận A là ma trân vuông cấp 5 có giá trị như sau :

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

* Với đồ thị có trọng số ma trận lân cận thay giá trị 1 bằng trọng số tương ứng của cung đó.

2.2. Biểu diễn bằng danh sách lân cận (danh sách kề)

Trong cách biểu diễn này n hàng của ma trận thay bằng n danh sách móc nối. Mỗi đỉnh của G có một danh sách tương ứng. Các nút trong danh sách i biểu diễn các đỉnh lân cận của nút i.

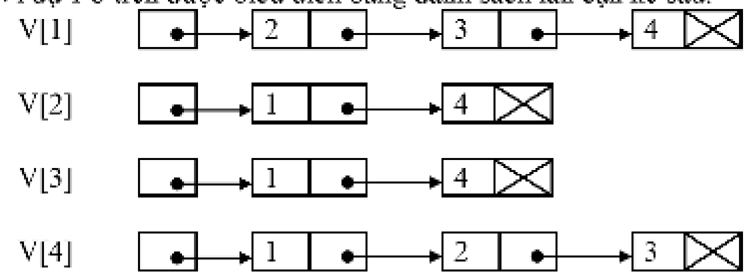
Mỗi nút trong danh sách có chứa 2 trường :

VERTEX	LINK

VERTEX: Là thông tin của đỉnh lân cận đỉnh i.

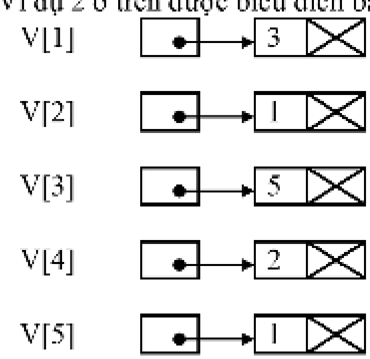
LINK : Là địa chỉ nút tiếp theo.

Ví dụ 1 ở trên được biểu diễn bằng danh sách lân cận kề sau:



Với V[1], V[2], V[3], V[4] là các phần tử của véc tơ V chứa các địa chỉ đầu danh sách (trỏ tới các danh sách).

Ví dụ 2 ở trên được biểu diễn bằng danh sách lân cận kề sau:



Với V[1], V[2], V[3], V[4], V[5] là các phần tử của véc tơ V chứa các địa chỉ đầu danh sách (trỏ tới các danh sách).

* Mỗi danh sách có nút đầu danh sách, các nút này được tổ chức lưu trữ kế tiếp (mảng) để truy nhập được nhanh.

* Đồ thị vô hướng có n đỉnh, e cung thì cần n nút đầu danh sách và 2e nút danh sách.

* Đồ thị vô hướng có n đỉnh, e cung thì cần n nút đầu danh sách và e nút danh sách.

3. Phép duyệt đồ thị

* Xét đồ thị vô hướng G(V,E) và một đỉnh v∈V. Ta cần thăm tất cả các đỉnh của G mà có thể " với tới" từ đỉnh v (nghĩa là đồ thị liên thông).

Có 2 cách duyệt đồ thị:

- Phép tìm kiếm theo chiều sâu (Depth first search)
- Phép tìm kiếm theo chiều rộng (Breadth first search)

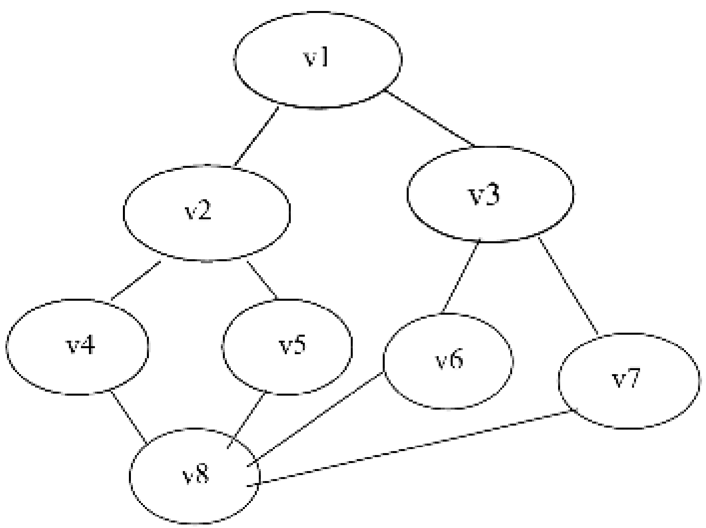
3.1. Phép tìm kiếm theo chiều sâu (Depth first search)

Xét đồ thị vô hướng. Phép tìm kiếm theo chiều sâu thể hiện như sau:

- Đỉnh xuất phát v được thăm.
- Tiếp theo đó ta thăm đỉnh w là đỉnh chưa được thăm và là lân cận của v. Phép tìm kiếm theo chiều sâu xuất phát từ w lại được thực hiện.

Trong trường hợp đỉnh u đã được thăm mà mọi đỉnh lân cận của nó đã được thăm rồi thì ta quay lại đỉnh cuối cùng vừa được thăm (mà đỉnh này còn đỉnh w là lân cận của nó chưa được thăm) và phép tìm kiếm theo chiều sâu xuất phát từ w lại được thực hiện.

Ví dụ 1: Cho đồ thị vô hướng sau:



Phép duyệt theo chiều sâu đi theo trình tự sau:

v1 → v2 → v4 → v8 → v5 → v6 → v3 → v7

* Thủ tục phép duyệt theo chiều sâu như sau:

Cho một đồ thị G(V,E) vô hướng có n đỉnh và véc tơ Visited(n) gồm n phần tử, ban đầu véc tơ này có giá trị =0. Thuật giải này thực hiện thăm mọi đỉnh " với tới được " từ đỉnh v.

```
Procedure  DFS(v)
   1. Visited(v):=1 { đánh dấu v  được thăm  }
   2.  FOR  mỗi đỉnh w lân cận với v   DO
          If  Visited(w) = 0  then    CALL  DFS(w);
Return
```

* Đánh giá thuật toán:

+ Trường hợp biểu diễn đồ thị dùng danh sách móc nối: G có e cung, mỗi nút với tới 1 lần, nên thời gian tìm kiếm là O(e).

+ Trường hợp biểu diễn đồ thị dùng ma trận lân cận : thì thời gian xác định mọi điểm lân cận của v là O(n). Có n đỉnh nên thời gian tìm kiếm là $O(n^2)$.

3.2. Phép tìm kiếm theo chiều rộng (Breadth first search)

Xét đồ thị vô hướng. Phép tìm kiếm theo chiều rộng thể hiện như sau:

- Đỉnh xuất phát v được thăm.
- Tiếp theo các đỉnh chưa được thăm mà là lân cận của v sẽ được thăm, rồi đến các đỉnh chưa được thăm là lân cận làn lượt của các đỉnh này và cứ tương tự như vậy.

Ví dụ 1 ở trên: Phép duyệt theo chiều rông đi theo trình tự sau:

v1 → v2 → v3 → v4 → v5 → v6 → v7 → v8

* Thủ tục phép duyệt theo chiều rong như sau:

Cho một đồ thị G(V,E) vô hướng có n đỉnh và véc tơ Visited(n) gồm n phần tử, ban đầu véc tơ này có giá trị =0. Thuật giải này thực hiện thăm mọi đỉnh " với tới được " từ đỉnh v. Bắt đầu từ đỉnh v. Mọi đỉnh i được thăm đánh dấu bằng Visited(i):=1.

Dùng hàng đợi Q có kích thước n; F, R là lối trước và lối sau của hàng đợi. Khi thăm 1 đình thì loại bỏ khỏi hàng đợi; khi chưa thăm thì bổ sung vào hàng đợi

```
Procedure   BFS(v)
   1. Khởi tạo hàng đợi Q với v được đưa vào.
   2. Visited(v):=1 { đánh dấu v  được thăm  }
   3.  While  Q không rỗng  DO
        Begin
          Call  CQDELETE(v,Q)     { loại bỏ v ra khỏi Q}
          FOR  mỗi đỉnh w lân cận với v   DO
            Begin
              If  Visited(w)=0   then
                 Begin
                   CALL   CQINSERT(w,Q); { Bổ sung w vào Q}
                   Visited(w):=1
                 End
            End
        End
   4. Return
```

* Đánh giá giải thuật: Vòng lặp While lặp lại n lần .

- Nếu biểu diễn đồ thị bằng ma trận lân cận thì thời gian thực hiện là $O(n^2)$.

- Nếu biểu diễn đồ thị bằng danh sách lân cận thì thời gian thực hiện là O(e).

4. Cây khung và cây khung với giá trị cực tiểu

4.1. Cây khung

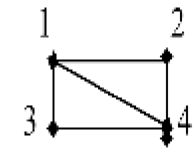
* Nếu G là đồ thị liên thông thì phép tìm kiếm theo chiều sâu hoặc theo chiều rộng xuất phát từ 1 đỉnh thăm mọi đỉnh. Như vậy các cung trong G phân thành 2 tập:

- Tập T chứa các cung đã được duyệt qua.

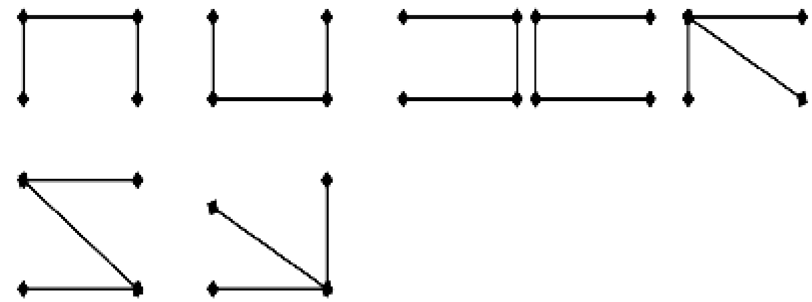
- Tập b gồm các cung còn lại.

* Tất cả các cung và các đỉnh trong T sẽ tạo thành một cây con bao gồm mọi đỉnh của G. Cây con như vậy gọi là cây khung của G.

Ví dụ 1: Cho đồ thị sau

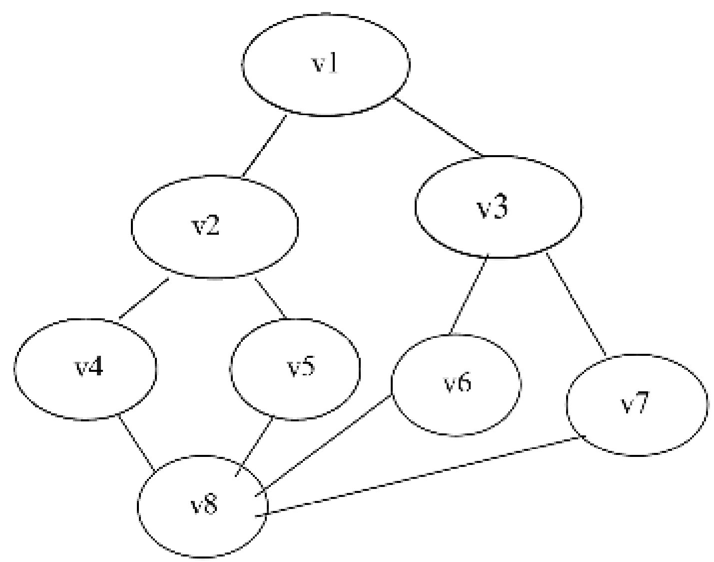


Các cây khung của nó là:

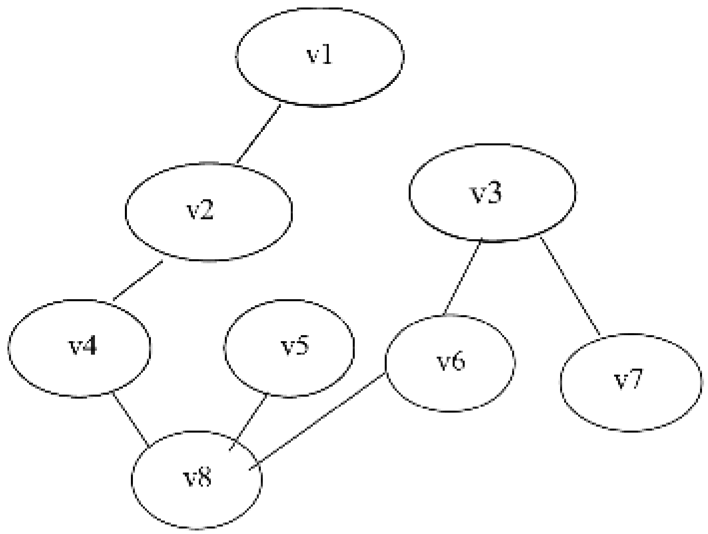


* Phép duyệt cây theo chiều sâu (DFS) cho cây khung theo chiều sâu.
* Phép duyệt theo chiều rộng cho cây khung theo chiều rộng.

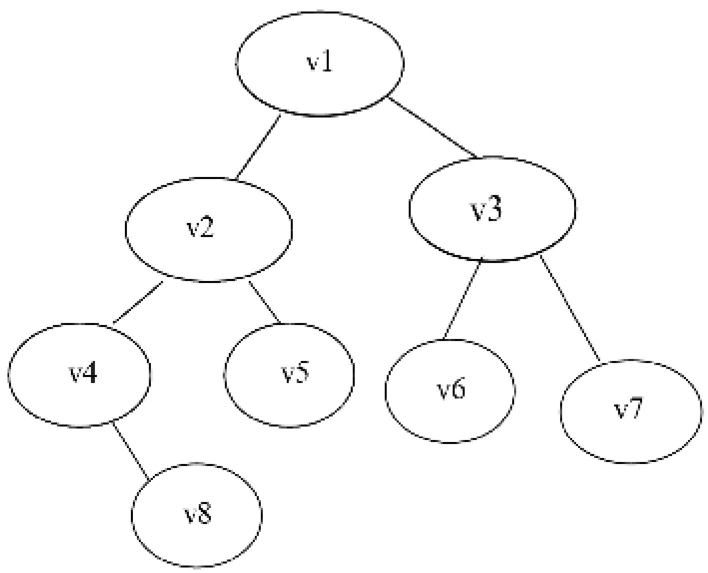
Ví dụ 2: cho đồ thị sau



+ Cây khung theo chiều sâu:



+ Cây khung theo chiều rộng:



4.2. Cây khung với giá trị cực tiểu

* Bài toán: Xác định cây khung với giá trị cực tiểu của đồ thị liên thông có trọng số.

Gía trị của cây khung là tổng các trọng số ứng với các cạnh của cây khung.

* Có nhiều giải thuật xác định cây khung với giá trị cực tiểu nhưng trong phần này ta chỉ xét giải thuật Kruskal. Với giải thuật này cây khung T sẽ được xây dựng dần từng cung một. Các cung đưa vào T thoả mãn:

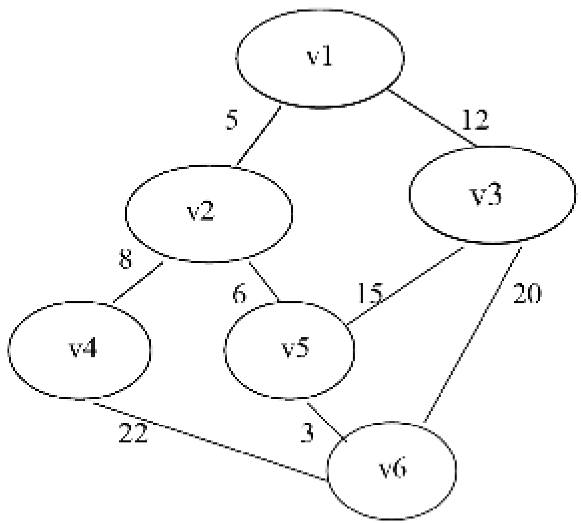
- Cung có giá trị cực tiểu trong các cung còn lại.
- Không tạo ra chu trình với các cung đã có của T.

* Giải thuật Kruskal được viết như sau:

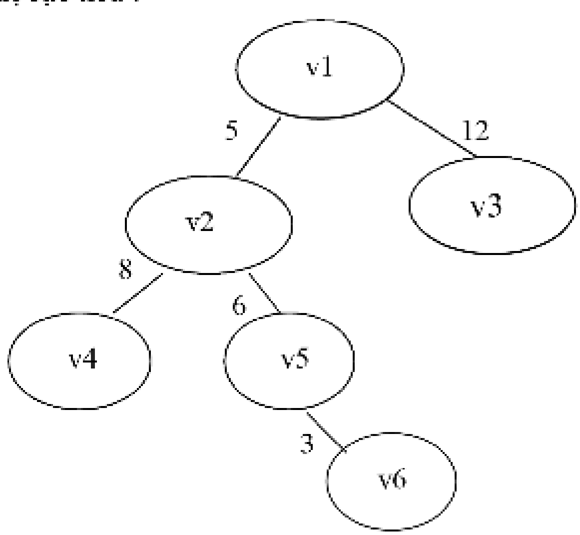
1. T=Φ { T rỗng
2. While T chứa ít hơn (n-1) cung Do
3. Begin Chọn 1 cung (v,w) từ E có giá trị nhỏ nhất.
4. Loại (v,w) ra khỏi E
5. If (v,w) không tạo nên chu trình trong T Then đưa (v,w) vào T.

   End;
6. Return

Ví dụ: Cho đồ thị sau:



Cây khung với giá trị cực tiểu :



* Đánh giá giải thuật:

Thời gian thực hiện giải thuật xác định qua thực hiện bước 3 và 4.

Trường hợp xấu nhất sẽ là O(e.log e) trong đó e là số cung của đồ thị G.

5. Bài toán tìm đường đi ngắn nhất

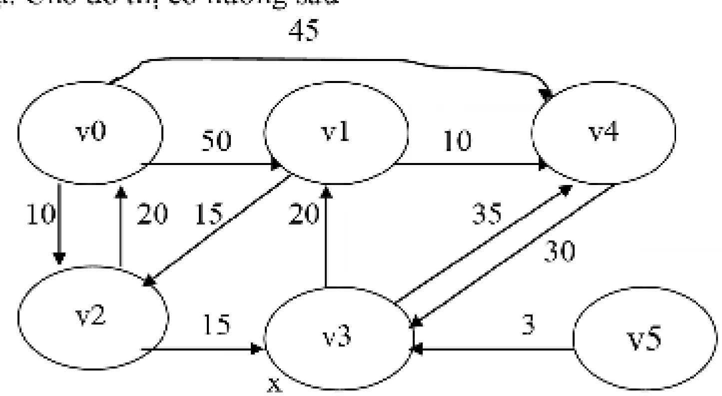
(Bài toán một nguồn mọi đích)

(Single source all destination)

* Cho đồ thị có hướng G(V,E), một hàm trọng số w(e) cho các cung e của G và một đỉnh nguồn v0 .

Bài toán đặt ra là: Xác định đường đi ngắn nhất từ v0 đến mọi đỉnh còn lại của G (độ dài đường đi là tổng các trọng số trên các cung đường đi đó và các trọng số đều dương)

Ví dụ: Cho đồ thị có hướng sau



Xuất phát từ V0, các đường đi ngắn nhất từ v0 tới v1,v2,v3,v4 sắp theo thứ tự tăng dần (không có đường đi từ vo đến v5):

1, v0,v2	độ dài	10
2, v0,v2,v3	độ dài	25
3, v0,v2,v3,v1	độ dài	45
4, v0,v4	dộ dài	45

* Gọi S là tập các đỉnh kể cả v0 mà đường đi ngắn nhất xác lập.

Đối với 1 đỉnh w ∈ S, gọi Dist(w) là độ dài của đường đi ngắn nhất từ v0 qua các đỉnh trong S và kết thúc ở w thì sẽ có một số nhận xét sau:

1. Nếu đường đi ngắn nhất tới w thì đường đi đó bắt đầu từ v0 kết thúc ở w và chỉ đi qua những đỉnh thuộc S.

2. Đích của đường đi sinh ra tiếp theo phải là một đỉnh w nào đó ∉ S mà có Dist(w) ngắn nhất so với mọi đỉnh ∉ S .

3. Nếu đã chọn được một đỉnh w như trong nhận xét 2 ở trên và sinh ra một đường đi ngắn nhất từ v0 đến w thì w sẽ trở thành 1 phần tử của S.

* Dựa trên các quan điểm như các nhận xét nêu trên Diskstra đưa ra giải thuật tìm đường đi ngắn nhất như sau:

- Giả thiết n đỉnh của G được đánh số từ 1 tới n.
- Tập S được thể hiện bằng véc tơ bít: S[i] = 0 nếu đỉnh i ∉ S
  S[i] = 1 nếu đỉnh i ∈ S
- Độ dài trọng số biểu diễn bằng ma trận lân cận Cost:
  Cost[i,j] là trọng số cung (i,j)
  Cost[i,j] = + ∞ nếu cung (i,j) không có.
  Cost[i,j] = 0 nếu i=j

Ví dụ trên thì ma trận lân cân Cost như sau:

	v0	v1	v2	v3	v4	v5
v0	0	50	10	+ ∞	45	+ ∞
v1	+ ∞	0	15	+ ∞	10	+ ∞
v2	20	+ ∞	0	15	+ ∞	+ ∞
v3	+ ∞	20	+ ∞	0	35	+ ∞
v4	+ ∞	+ ∞	+ ∞	30	0	+ ∞
v5	+ ∞	+ ∞	+ ∞	3	+ ∞	0

* Giải thuật:

```
Procedure   Shortest_Path(v,Cost,Dist,n)
{ Dist(j) 1<= j <=n là độ dài đường đi ngắn nhất từ v đến j trong đồ thị có hướng G có n đỉnh, Dist(v)=0. G được biểu diễn bởi ma trận lân cận Cost có kích thước n x n. }
1. For  i:=1 To n Do Begin
         S[i]:=0;  Dist[i]:= Cost[v,i];
       End;
2. S[v]:=1; Dist[v]:=0; k:=1;  { đưa v vào S }
3. While  k<n  { xác định n-1 đường đi từ đỉnh v }
4. Begin
       Chọn u sao cho Dist[u]= min(Dist[i]))  với  S[i]=0;
5. S[u]:=1; k:=k+1;  { đưa u vào S}
6. For  mọi w với S[w]=0 Do
7.    Dist[w]:= min(Dist[w], Dist[u]+Cost[u,w])  { tính lại khoảng cách theo đường đi ngắn nhất }
  End;
8. Return
```

Bài tập

1. Nêu khái niệm đồ thị, đồ thị vô hướng, đồ thị có hướng, đường đi, cây khung, cây khung với gía trị cực tiểu.
2. Cho đồ thị sau đây.

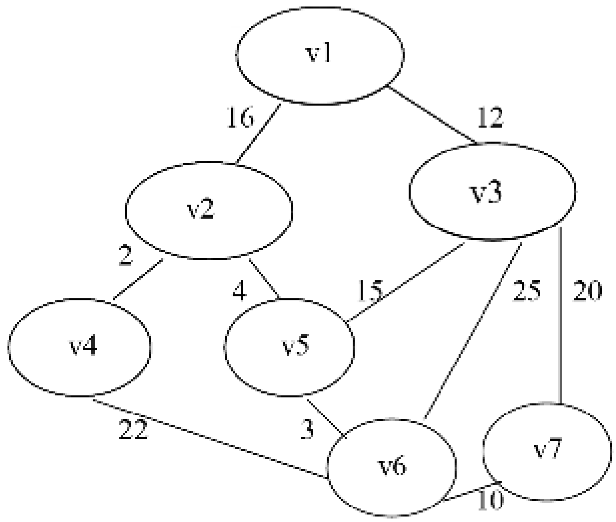
   a- Hãy biểu diễn đồ thị bằng ma trận lân cận, bằng danh sách lân cận

   b- Duyệt đồ thị theo chiều sâu, duyệt đồ thị theo chiều rộng.

   c- Tìm cây khung theo chiều sâu, cây khung theo chiều rộng.

   d-Tìm cây khung với giá trị cực tiểu.

   e- Viết chương trình trong Pascal tìm cây khung với giá trị cực tiểu.



3. cho đồ thị có hướng sau.

a- Xây dựng đồ thị Cost.

b- Tìm đường đi ngắn nhất từ v0 đến các đỉnh còn lại trong đồ thị

c- Viết chương trình trong Passcal tìm đường đi ngắn nhất từ đỉnh v0 đến mọi đỉnh.