Chương 4

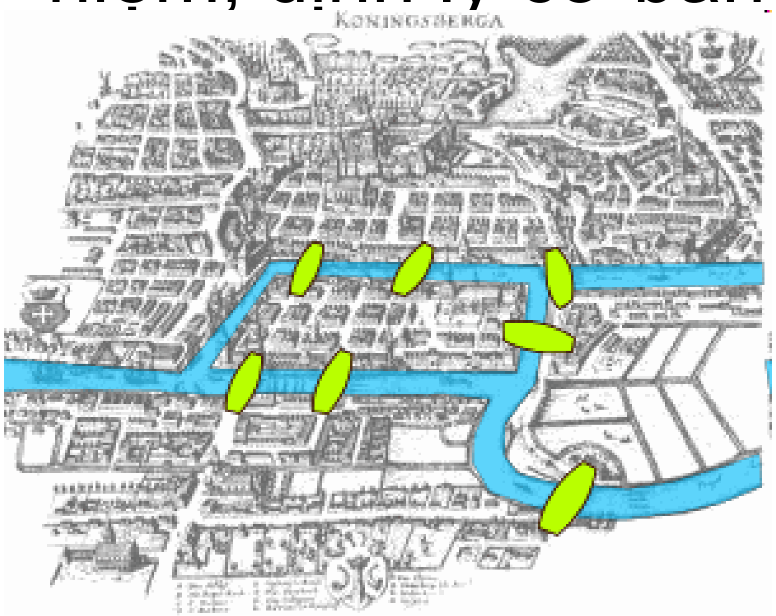
LÝ THUYẾT ĐỒ THỊ

TÀI LIỆU THAM KHẢO THƯ VIỆN

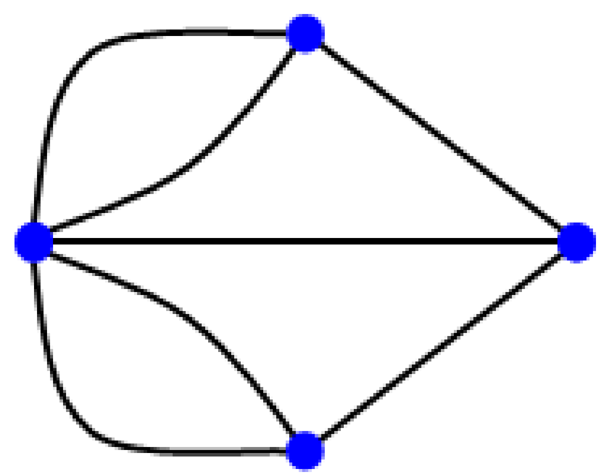
- Lý thuyết đồ thị và ứng dụng - Đặng Huy Ruận – Nhà xuất bản khoa học và kỹ thuật
- Giáo trình lý thuyết đồ thị - Nguyễn Thanh Hùng; Nguyễn Đức Nghĩa; - Nhà xuất bản ĐHQG HCM.
- Lý thuyết đồ thị - PGS. Nguyễn Cam; PTS. Chu Đức Khánh - Nhà xuất bản ĐHQG HCM.
- Giáo trình Lý thuyết đồ thị - Trương Mỹ Dung– Trường ĐH Bà Rịa Vũng Tàu.

4.1. Định nghĩa đồ thị

- Nhà toán học - vật lý học Thụy Sỹ Leonhard Euler là người đầu tiên đề xuất những khái niệm, định lý cơ bản



Bảy cầu ở Königsberg



Bài toán bảy cây cầu Euler

4.1. Định nghĩa đồ thị

- **Định nghĩa 1.1:** Đồ thị G là một cặp (V,E), trong đó:
  - V là tập hợp các đỉnh của đồ thị;
  - E là tập hợp các cạnh (cung) của đồ thị, $E \subseteq V \times V$

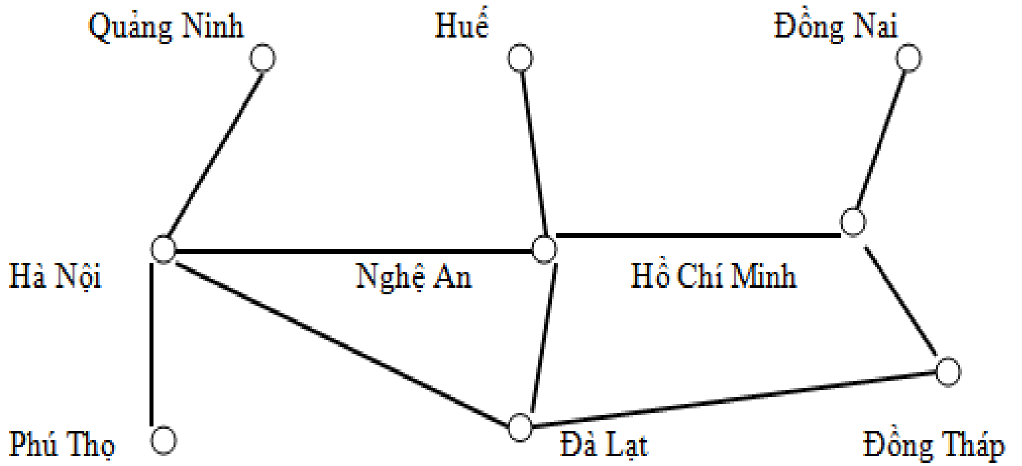
Một cung $e = (x_i, x_j)$, x_i là đỉnh đi ,x_j là đỉnh đến.

Cạnh, cung được biểu diễn hình học như sau:

Cạnh

Cung

4.1. Định nghĩa đồ thị



4.1. Định nghĩa đồ thị

- ***Định nghĩa 1.2:***
  - Đồ thị vô hướng: chỉ chứa cạnh vô hướng
  - Đồ thị có hướng: chỉ chứa cung
- ***Định nghĩa 1.3:***
  - Đơn đồ thị: mỗi cặp đỉnh được nối với nhau bởi không quá một cạnh
  - Đa đồ thị: tồn tại một cặp đỉnh được nối với nhau bởi nhiều hơn một cạnh

4.1. Định nghĩa đồ thị

- **Định nghĩa 1.4:**

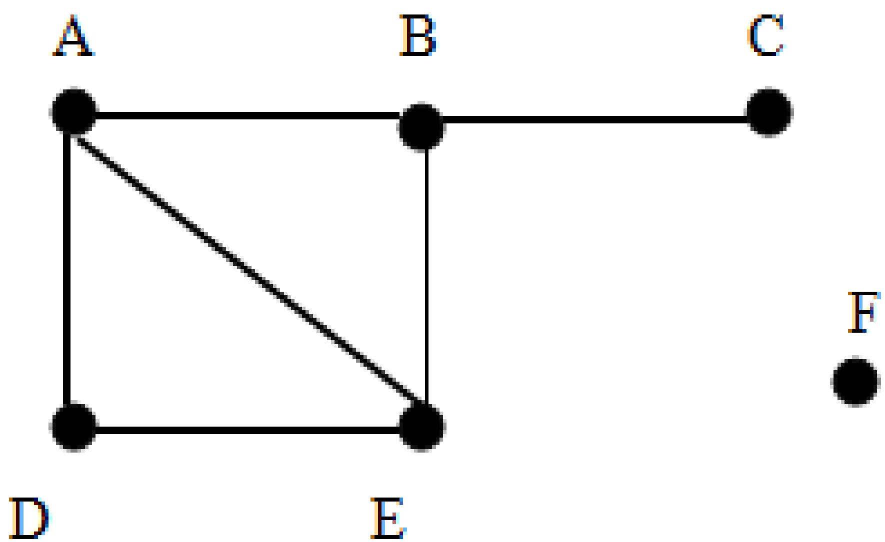
  Giả đồ thị vô hướng: Tồn tại cạnh là khuyên nối một đỉnh với chính nó
- **Định nghĩa 1.5:** Cho đồ thị G = (V, E)
  - Mỗi cạnh e của G được gán số c(e) (c(u,v) với u và v là đỉnh của cạnh): trọng số của cạnh
  - Đồ thị có trọng số: Các cạnh có trọng số

4.1. Định nghĩa đồ thị

- **Định nghĩa 1.6:**
  - Hai đỉnh kề nhau u và v: nếu (u,v) là cạnh của đồ thị G.
  - Cạnh liên thuộc với hai đỉnh u và v: Nếu e =(u,v) là cạnh của đồ thị

4.1. Định nghĩa đồ thị

- **Định nghĩa 1.7:** Bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với v, ký hiệu là deg(v)



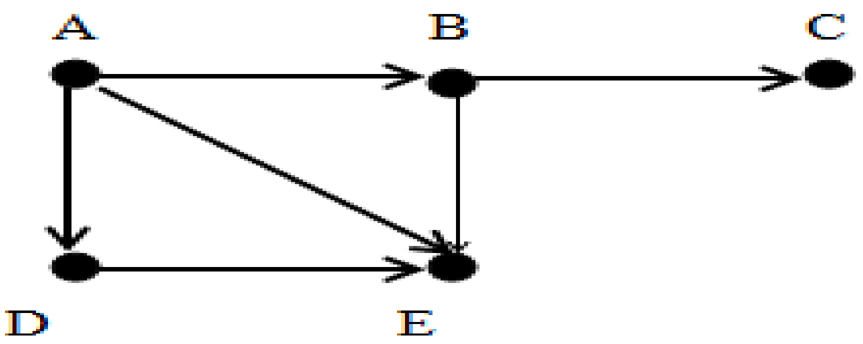
deg(A)= 3, deg(B)= 3,

deg(C)= 1, deg(D)= 2,

deg(E)= 3, deg(F)=0;

4.1. Định nghĩa đồ thị

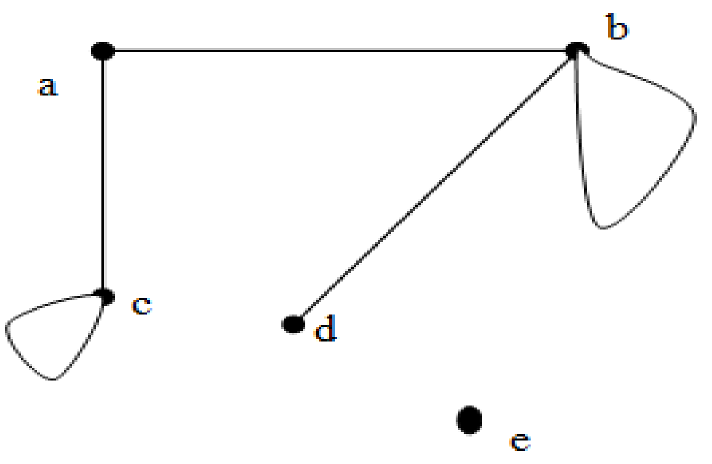
- ***Định nghĩa 1.8:***
  - Bán bậc ra (bán bậc vào) của đỉnh v trong đồ thị có hướng là số cung của đồ thị đi ra khỏi v (đi vào v) và ký hiệu $deg^+(v)$ ($deg^-(v)$)



$deg^+(A) = 3$, $deg^-(A) = 0$,
$deg^+(B) = 2$, $deg^-(B) = 1$,
$deg^+(C) = 0$, $deg^-(C) = 1$,
$deg^+(D) = 1$, $deg^-(D) = 1$,
$deg^+(E) = 0$, $deg^-(E) = 3$

4.1. Định nghĩa đồ thị

- Bậc của đỉnh có khuyên được cộng thêm 2 cho mỗi khuyên



deg(a) = 2;deg(b) = 4

deg(c) = 3; deg(d) = 1

deg(e) = 0

Đỉnh có bậc 0 gọi là đỉnh *cô lập*.

Đỉnh có bậc 1 gọi là đỉnh *treo*.

4. Một số tính chất của đồ thị

- ***Định lý 1.1:*** Giả sử G = (V, E) là đồ thị vô hướng m cạnh. Khi đó:2m= $\sum_{v \in V} \deg(v)$
- **Hệ quả:** Trong đồ thị vô hướng, số đỉnh có bậc là số lẻ (đỉnh bậc lẻ) là một số chẵn.
- **Định lý 1.2:** Giả sử G =(V, E) là đồ thị có hướng. Khi đó:
  $\sum_{v \in V} \deg^+(v)$ = $\sum_{v \in V} \deg^-(v)$ =m= $|E|$

4.2. Biểu diễn đồ thị trong máy tính

- **Ma trận kề. Ma trận trọng số**
- **Ma trận liên thuộc đỉnh – cạnh**
- **Danh sách kề**

4.2.1. **Ma trận kề. Ma trận trọng số**

Xét đơn đồ thị G = (V,E):

V = {1,2,3,…,n}, E ={e_1,e_2,…,e_m}

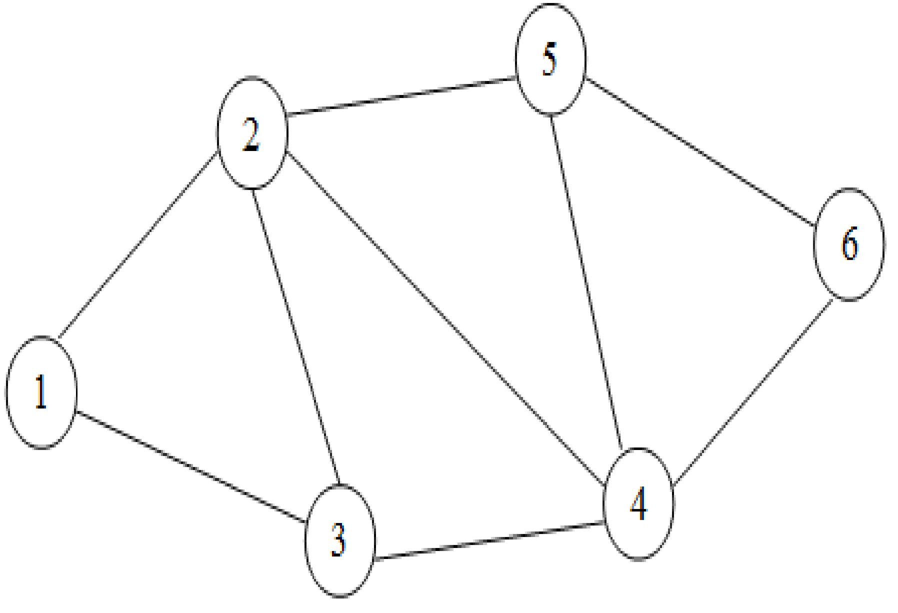
Ta gọi ma trận kề A của đồ thị G là (0,1) - ma trận:

$$A = \{a_{ij}: i, j = 1, 2, \ldots, n\}$$

với a_{ij} được xác định theo quy tắc sau đây:

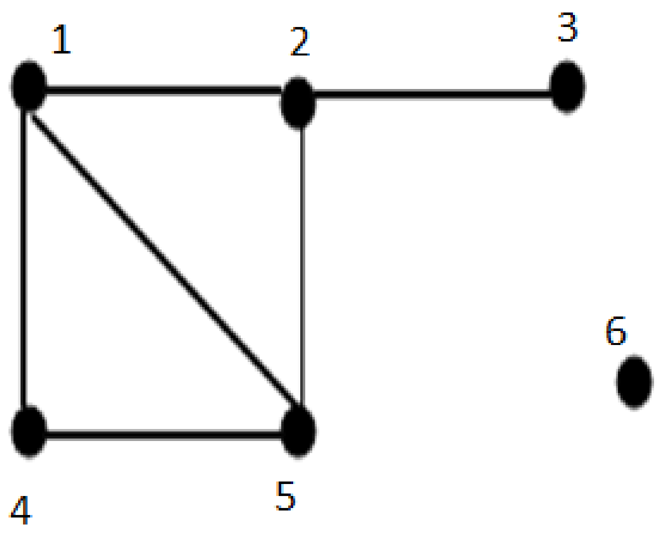
$$a_{ij} = \begin{cases} 0 \text{ nếu } (i, j) \notin E \\ 1 \text{ nếu } (i, j) \in E \end{cases} \quad \text{với } i, j = 1, 2, \ldots n$$

4.2.1. **Ma trận kề. Ma trận trọng số**



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

4.2.1. **Ma trận kề. Ma trận trọng số**



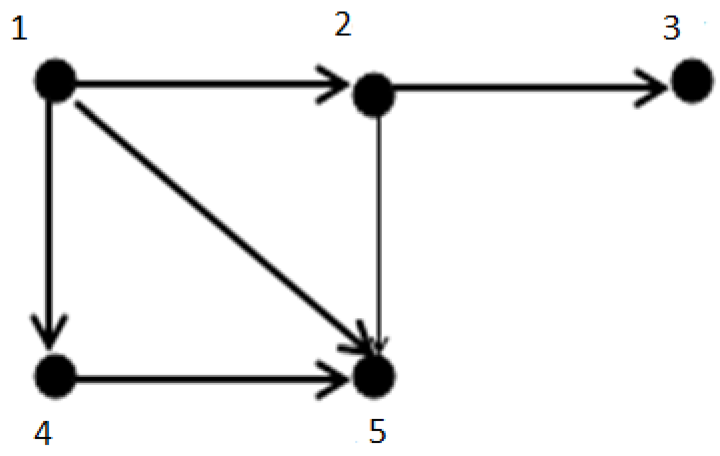
1	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	1	0
3	0	1	0	0	0	0
4	1	0	0	0	1	0
5	1	1	0	1	0	0
6	0	0	0	0	0	0

Nhận xét:

- Ma trận kề của đồ thị vô hướng G là ma trận đối xứng

-Tổng các dòng i (cột j) là bậc của đỉnh tại dòng i (cột j)

4.2.1. **Ma trận kề. Ma trận trọng số**



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

Nhận xét:

Ma trận kề của đồ thị có hướng không là ma trận đối xứng

Tổng các phần tử trên dòng i (cột j) là bán bậc ra (bán bậc vào) của đỉnh i (j)

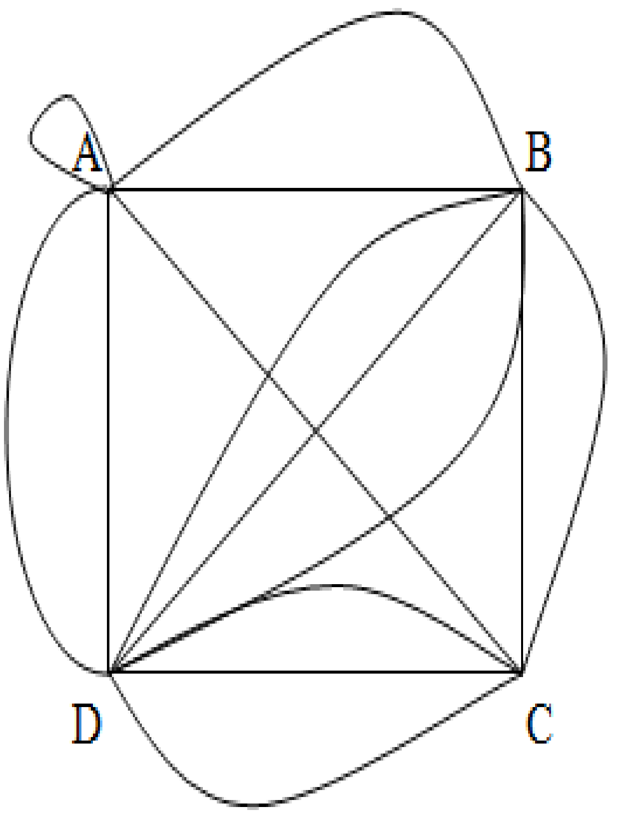
4.2.1. **Ma trận kề. Ma trận trọng số**

- Ma trận kề biểu diễn đa đồ thị, với (i, j) là cạnh của đồ thị thì ta chỉ cần thay số 1 vào ô a[i,j] bằng số k là số cạnh nối hai đỉnh i, j
- Trong trường hợp đồ thị trọng số, ta sử dụng ma trận trọng số C = c[i,j], i,j = 1,2,3,....,n với

  c[i,j] = c(i,j) nếu $(i, j) \in E$

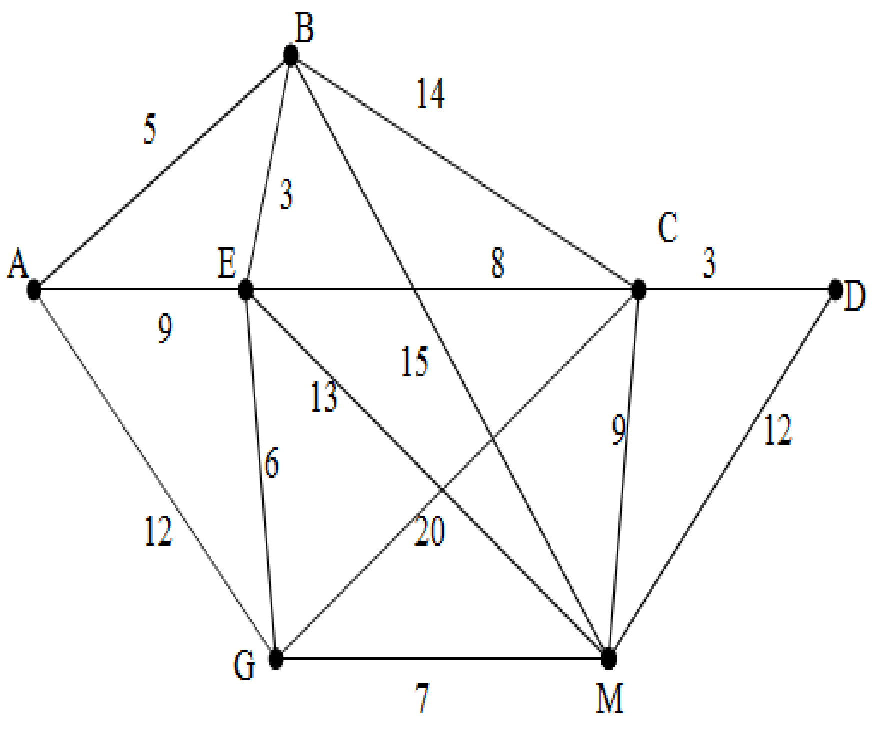
và c[i, j] = θ nếu $(i, j) \notin E$, $\theta = 0, -\infty, +\infty$

4.2.1. **Ma trận kề. Ma trận trọng số**



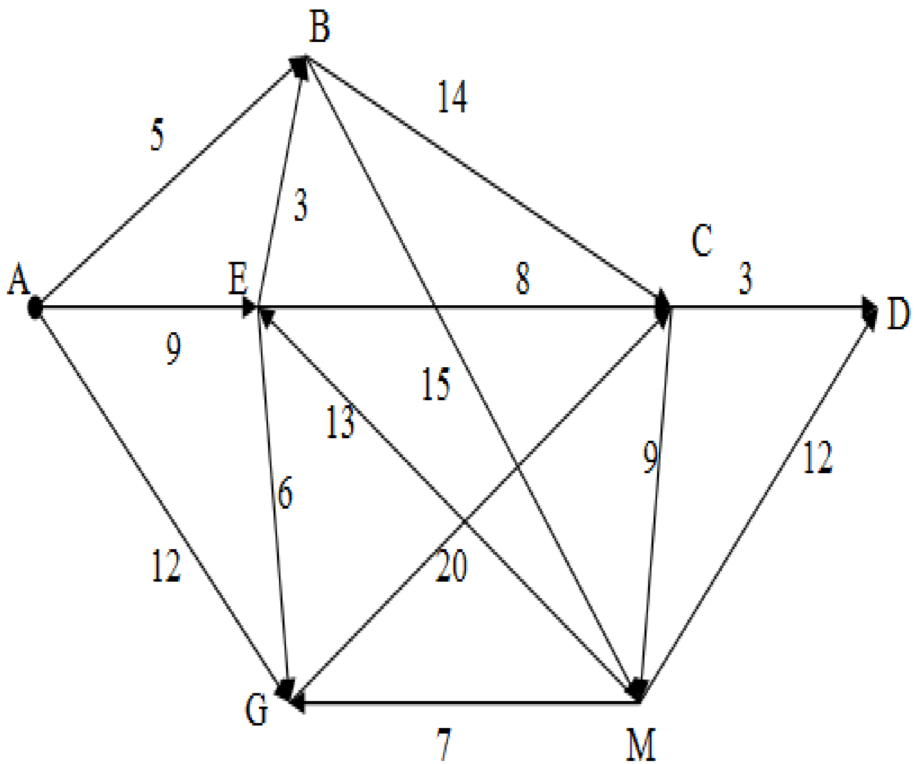
	A	B	C	D
A	1	2	1	2
B	2	0	2	3
C	1	2	0	3
D	2	3	3	0

4.2.1. **Ma trận kề. Ma trận trọng số**



	A	B	C	D	E	G	M
A	0	5	0	0	9	12	0
B	5	0	14	0	3	0	15
C	0	14	0	3	8	20	9
D	0	0	3	0	0	0	12
E	9	3	8	0	0	6	20
G	12	0	20	0	6	0	7
M	0	15	9	12	20	7	0

4.2.1. **Ma trận kề. Ma trận trọng số**



	A	B	C	D	E	G	M
A	0	5	0	0	9	12	0
B	0	0	14	0	0	0	0
C	0	0	0	3	0	0	9
D	0	0	0	0	0	0	0
E	0	3	8	0	0	6	0
G	0	0	20	0	0	0	0
M	0	0	0	12	20	7	0

4.2.2. Ma trận liên thuộc đỉnh – cạnh

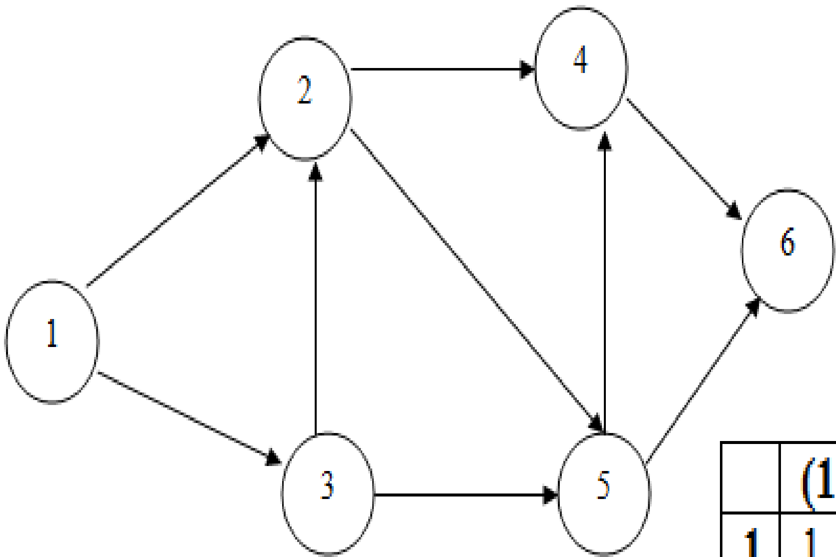
Xét đơn đồ thị có hướng G = (V, E)

V = {1,2,3,4,…}, E = {e_1,e_2,…,e_n}

Ma trận liên thuộc đỉnh – cạnh của đồ thị được xây dựng như sau:

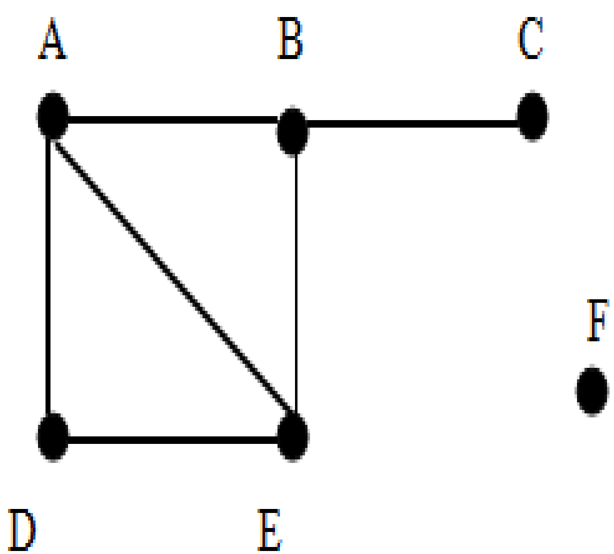
$$a_{ij} = \begin{cases} 1, \text{ nếu đỉnh i là đỉnh đầu của cung ej} \\ 0, \text{ nếu đỉnh i không là đỉnh của cung ej} \\ -1, \text{ nếu đỉnh i là đỉnh cuối của cung ej} \end{cases}$$

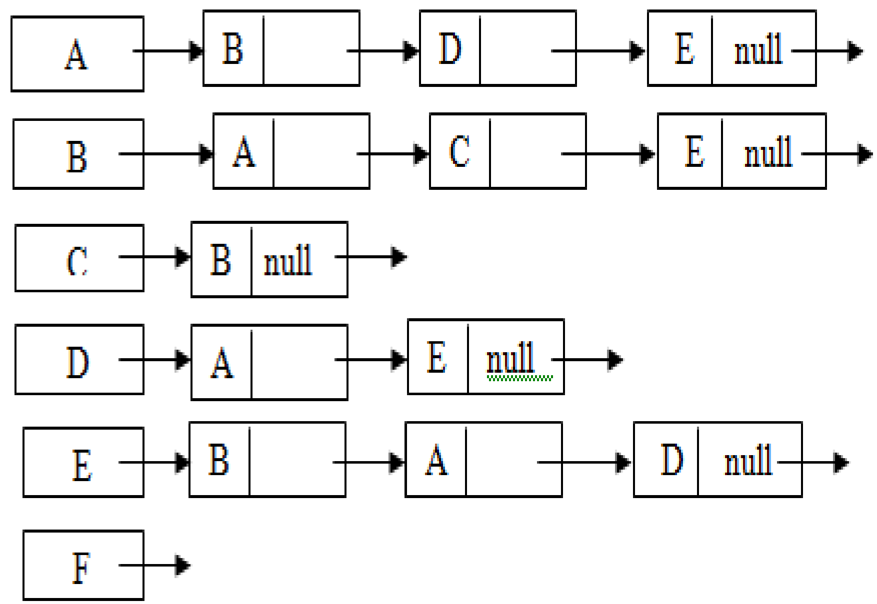
4.2.2. Ma trận liên thuộc đỉnh – cạnh



	(1,2)	(1,3)	(2,4)	(2,5)	(3,2)	(3,5)	(4,6)	(5,4)	(5,6)
1	1	1	0	0	0	0	0	0	0
2	-1	0	1	1	-1	0	0	0	0
3	0	-1	0	0	1	1	0	0	0
4	0	0	-1	-1	0	0	1	-1	0
5	0	0	0	-1	0	-1	0	1	1
6	0	0	0	0	0	0	-1	0	-1

4.2.3 Danh sách kề





4.3. Tìm kiếm đồ thị

- Tìm kiếm theo chiều sâu
- Tìm kiếm theo chiều rộng

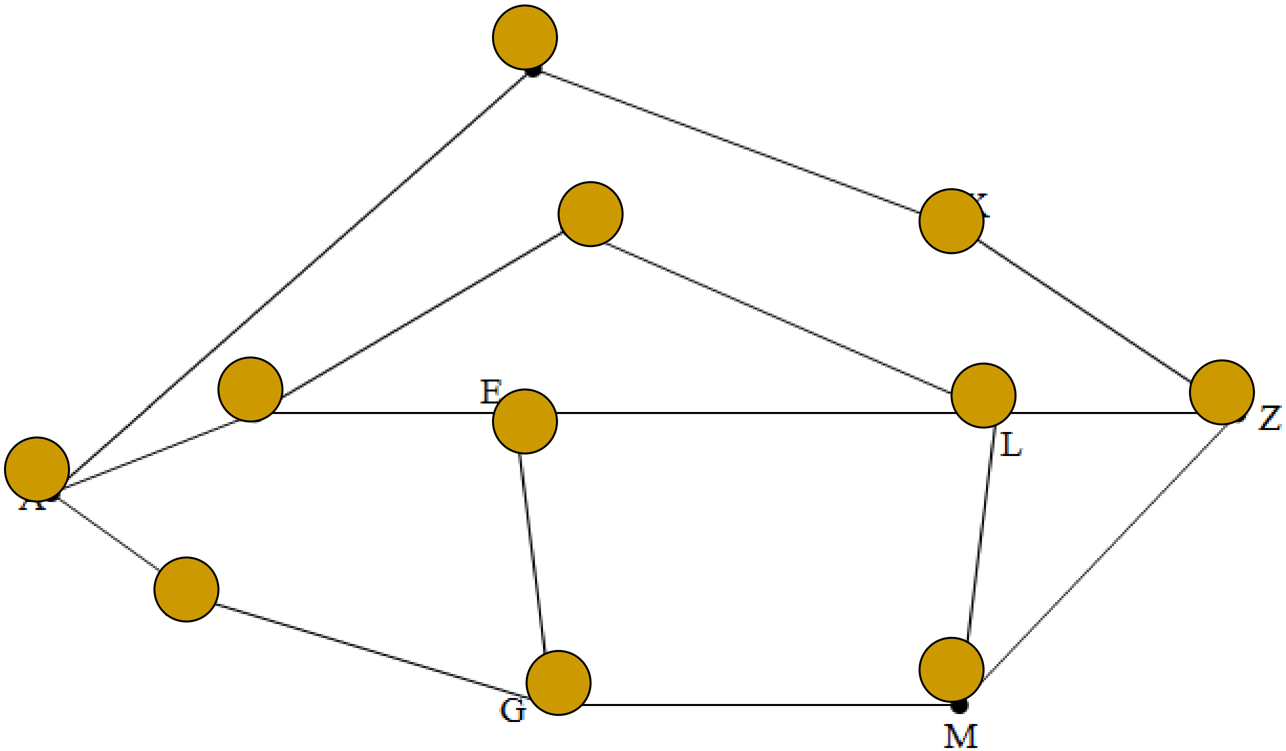
4.3.1.Tìm kiếm theo chiều sâu

- Ý tưởng:
  - Bắt đầu duyệt từ một đỉnh v_0 nào đó của đồ thị, sau đó chọn một đỉnh kề u chưa được duyệt bất kỳ của v_0
  - Lặp lại quá trình trên đối với đỉnh u

4.3.1. Tìm kiếm theo chiều sâu

- Tổng quát, giả sử ta đang xét đỉnh v
  - Nếu trong số các đỉnh kề của v tìm được w là chưa duyệt thì xét w, w trở thành đỉnh đã xét và bắt đầu từ w ta tiếp tục quá trình duyệt
  - Còn nếu không có đỉnh nào kề với v mà chưa xét thì ta nói đỉnh v đã được duyệt xong và quay trở lại tiếp tục duyệt đỉnh trước khi ta duyệt v cho tới khi đỉnh ta duyệt quay lại là v_0.

4.3.1. Tìm kiếm theo chiều sâu



4.3.1. Tìm kiếm theo chiều sâu

- Thuật toán
  - input: Đồ thị G = (V, E)
  - output: Thứ tự các đỉnh được duyệt của đồ thị theo chiều sâu
  - Chú ý:
    - Chuaxet là biến toàn cục
    - Khi đỉnh v chưa được thăm thì chuaxet[v] :=true
    - Khi đỉnh v được thăm chuaxet[v]:=false;

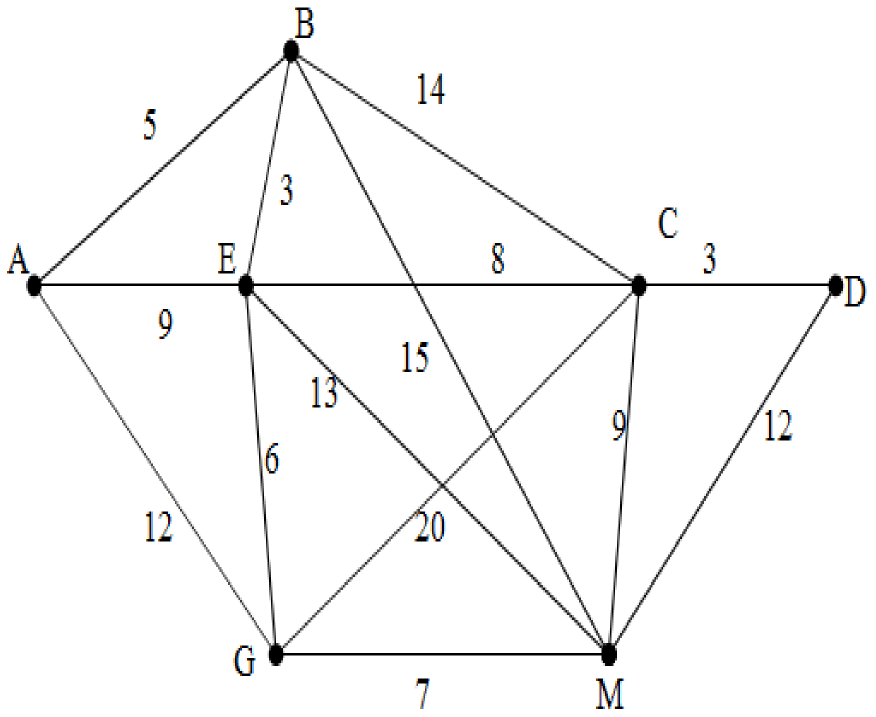
4.3.1. Tìm kiếm theo chiều sâu

```
Procedure DFS(v)
    begin
      Thamđinh(v);
      chuaxet[v] : =false;
      for u  ∈  Ke[v] do
      if  chuaxet[u] = true then DFS(u);
    end;
```

4.3.1. Tìm kiếm đồ thị theo chiều sâu

```
Program DuyetG
  Begin
    for v∈ V do chuaxet[v]:= true;
    for v∈ V do
        if chuaxet[v]= true then DFS(v);
  End.
```

4.3.1. Tìm kiếm đồ thị theo chiều sâu

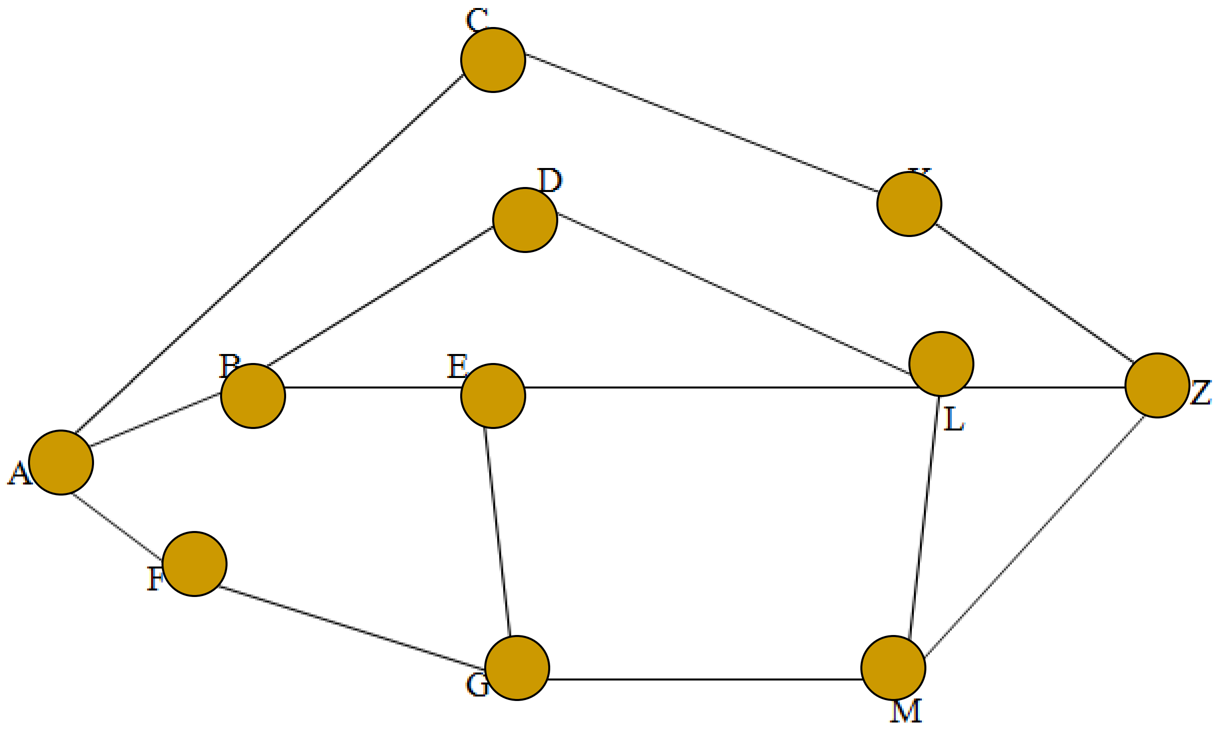


	A	B	C	D	E	G	M
A	0	5	0	0	9	12	0
B	5	0	14	0	3	0	15
C	0	14	0	3	8	20	9
D	0	0	3	0	0	0	12
E	9	3	8	0	0	6	20
G	12	0	20	0	6	0	7
M	0	15	9	12	20	7	0

4.3.2. Tìm kiếm theo chiều rộng

- **Ý tưởng**
  - Bắt đầu duyệt từ một đỉnh nào đó của đồ thị
  - Giả sử đỉnh đó là v_1
    - Khi duyệt đỉnh v_1 ta sẽ xét tới các đỉnh v_{11}, v_{12}, v_{13},…,v_{1k} mà ta chưa xét đến để ta lần lượt xét tới các đỉnh này
    - Khi duyệt đỉnh v_{1i} (i=1,…,k) ta lại để ý tới các đỉnh kề với nó mà chưa được xét đến để xét đến các đỉnh đó
    - Quá trình cứ như vậy cho tới khi tất cả các đỉnh được xét đến.

4.3.2. Tìm kiếm theo chiều rộng



4.3.2. Tìm kiếm theo chiều rộng

- Thuật toán
  - input: Đồ thị G = (V, E)
  - output: Thứ tự các đỉnh được duyệt của đồ thị theo chiều rộng
  - Chú ý:
    - Chuaxet là biến toàn cục
    - Khi đỉnh v chưa được thăm thì chuaxet[v] :=true
    - Khi đỉnh v được thăm chuaxet[v]:=false;
    - Hàng đợi Queue

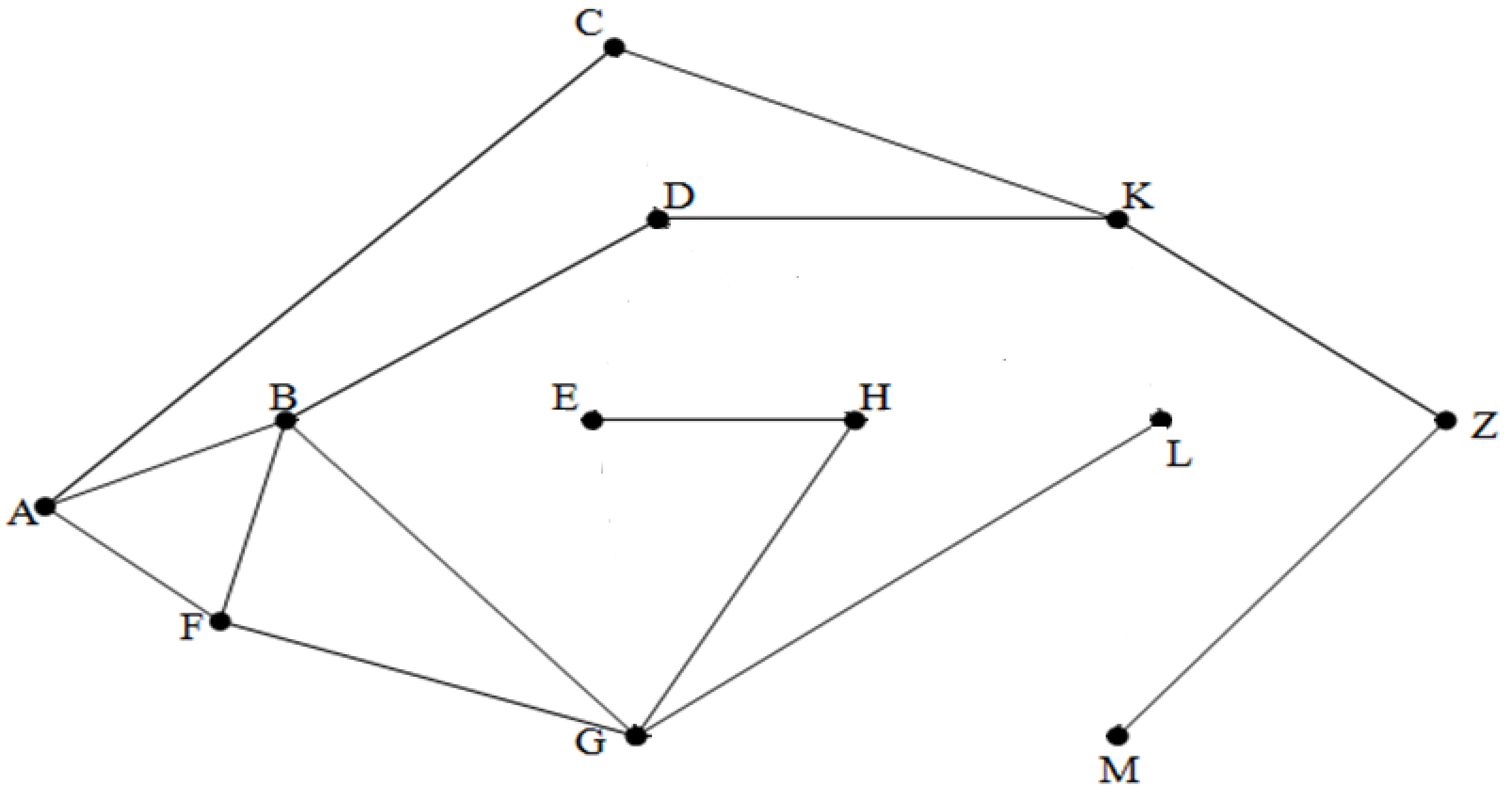
4.3.2. Tìm kiếm theo chiều rộng

```
Procedure BFS(v)
 Begin
     Queue: = ∅ ;   Queue ← v;  Chuaxet[v]:=flase;
     While Queue ≠ ∅  do
           begin
             p ←Queue;  Xet_dinh(p);
            for u ∈ Ke[p] do
              if Chuaxet[u] = true then
                         begin
                             Queue ← u; Chuaxet[u]:=false;
                         end;
            end;
end;
```

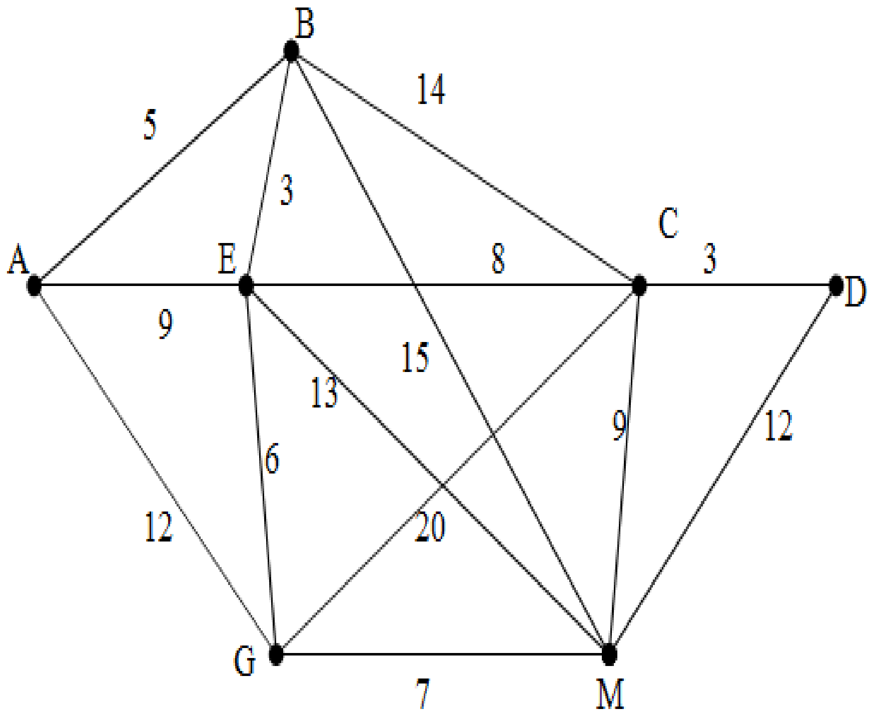
4.3.2. Tìm kiếm theo chiều rộng

```
Program DuyetG
  Begin
    for v∈ V do chuaxet[v]:= true;
    for v∈ V do
        if chuaxet[v]= true then BFS(v);
  End.
```

Tìm kiếm đồ thị



4.3.2. Tìm kiếm đồ thị theo chiều rộng



	A	B	C	D	E	G	M
A	0	5	0	0	9	12	0
B	5	0	14	0	3	0	15
C	0	14	0	3	8	20	9
D	0	0	3	0	0	0	12
E	9	3	8	0	0	6	20
G	12	0	20	0	6	0	7
M	0	15	9	12	20	7	0

4.4. Cây khung của đồ thị

4.4.1. Định nghĩa

- ***Định nghĩa 4.1:***

  Đường đi trên đồ thị G= (V, E) là một dãy các đỉnh <x_1, x_2, x_3,…,x_{k-1}, x_k>

  Mỗi đỉnh trong dãy tính từ đỉnh thứ hai kề với đỉnhtrước nó bằng một cạnh nào đó

  - Đường đi có đỉnh đầu x_1 và đỉnh cuối x_k
  - Số cạnh của đường đi gọi là độ dài của đường đi

4.4.1. Định nghĩa

- ***Định nghĩa 4.2:*** Chu trình của đồ thị G là một đường đi khép kín

$$[x_1, x_2, x_3,..., x_{k-1}, x_k]$$

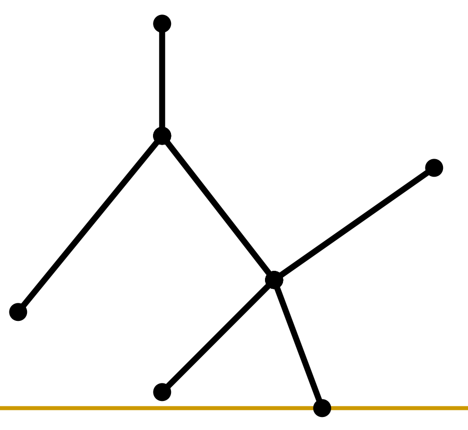
Trong đó $x_k = x_1$.

$$[x_1, x_2, x_3,..., x_{k-2}, x_{k-1}]$$

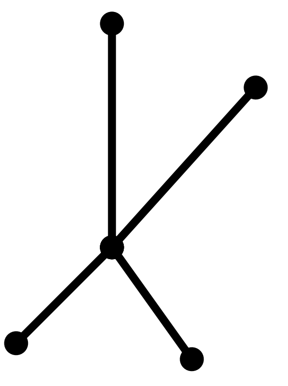
- Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.

4.4.1. Định nghĩa

- **Định nghĩa 4.3: Đồ thị liên thông** là đồ thị vô hướng G= (V,E) và luôn tìm được đường đi giữa hai đỉnh bất kỳ
- **Định nghĩa 4.4:** Cây là đồ thị vô hướng, liên thông và không có chu trình đơn



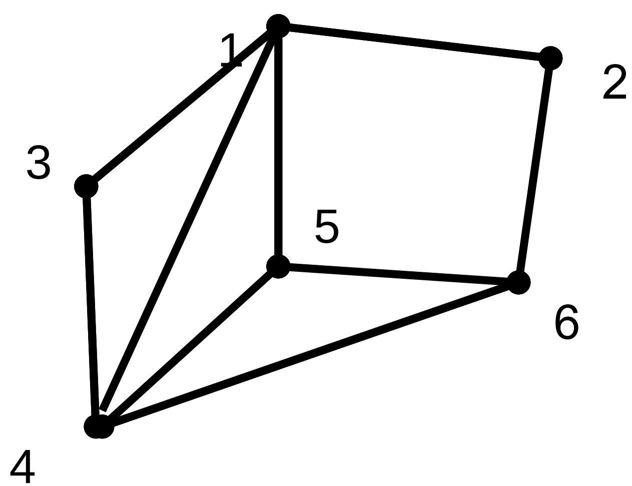
Cây T1



Cây T2

4.4.1. Định nghĩa

Giả sử G = (V, E) là đồ thị vô hướng liên thông. Cây T = (V, F) trong đó F⊂ E gọi là cây khung bao trùm (cây khung) của đồ thị G.



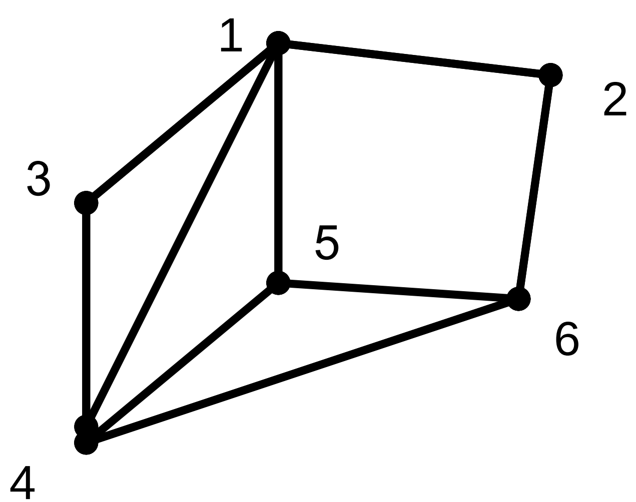
Đồ thị G

1 2
3
5 6
4

T3

4.4. Cây khung của đồ thị

- Đồ thị T_4 sau có là cây khung của G không?



Đồ thị G

1
2
3
5
6
4

T_4

4.4.2. Tìm cây khung của đồ thị

- Thuật toán tổng quát
  - Bước 1: Chọn một đỉnh $v \in V$ và khởi tạo X: = {v}; T:= $\varnothing$;
  - Bước 2: Chọn w = V/X sao cho cạnh e của G nối giữa w với một đỉnh nào đó trong X
  - Bước 3: Gán $X := X \cup \{w\}$; $T := T \cup \{e\}$
  - Bước 4: Nếu T đủ n- 1 phần tử thì dừng, ngược lại quay về bước 2.

4.4.2. Tìm cây khung của đồ thị

0	1	0	1	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1
1	0	1	0	1	0	1	1
1	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	1	0	1
0	0	1	1	0	0	1	0

4.4.2. Tìm cây khung của đồ thị

0	1	0	1	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0
1	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	1
0	0	0	0	0	1	1	0

4.4.2.1. Tìm kiếm theo chiều sâu

```
Procedure DFS_Tree(v)
   begin
     chuaxet[v]=false;
     for u ∈ Ke[v] do
          if chuaxet[u] = true then
            begin
        T:=T∪ (u,v)
        DFS_Tree(u);
   end;
```

```
Begin
   T := Φ; root:=1;
   for v ∈ V do chuaxet[v]:= true;
   DFS_Tree(root);
 End.
```

4.4.2.1. Tìm kiếm theo chiều sâu

0	1	0	1	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1
1	0	1	0	1	0	1	1
1	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	1	0	1
0	0	1	1	0	0	1	0

4.4.2.2. Tìm kiếm theo chiều rộng

```
Procedure BFS_Tree(v)
Begin
  Queue: =  Φ
  Queue ←— v;
  Chuaxet[v]:=flase;
While Queue do
   begin
      p ←— Queue;
 for u  Ke[p] do
       if Chuaxet[u] then
         begin
            Queue ←— u;
             Chuaxet[u]:=false;
             T:=T ∪ (u,v)
         end;
     end;
End;
```

4.4.2.2. Tìm kiếm theo chiều rộng

```
Begin
T := Φ; root:=1;
for v ∈ V do chuaxet[v]:= true;
BFS_Tree(root);
End.
```

4.4.2.2. Tìm kiếm theo chiều rộng

0	1	0	1	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1
1	0	1	0	1	0	1	1
1	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	1	0	1	0	1
0	0	1	1	0	0	1	0

4.2.2.3.Bài toán tìm cây khung nhỏ nhất của đồ thị trọng số

- Cho đồ thị vô hướng liên thông G =(V,E) với |V|=n, |E|=m. Mỗi cạnh E của đồ thị được gán trọng số c(e) gọi là độ dài của cạnh.
- Giả sử T = (V,F) là cây khung của đồ thị G. Ta gọi số c(T) là tổng độ dài các cạnh của cây khung, hay c(T) = $\sum_{e \in F} c(e)$

4.2.2.3.Bài toán tìm cây khung nhỏ nhất của đồ thị trọng số

- Bài toán đặt ra là trong tất cả các cây khung của đồ thị G thì cây khung nào có số c(T) nhỏ nhất
  - Thuật toán Kruskal
  - Thuật toán Prim

4.2.2.3.1.Thuật toán Kruskal

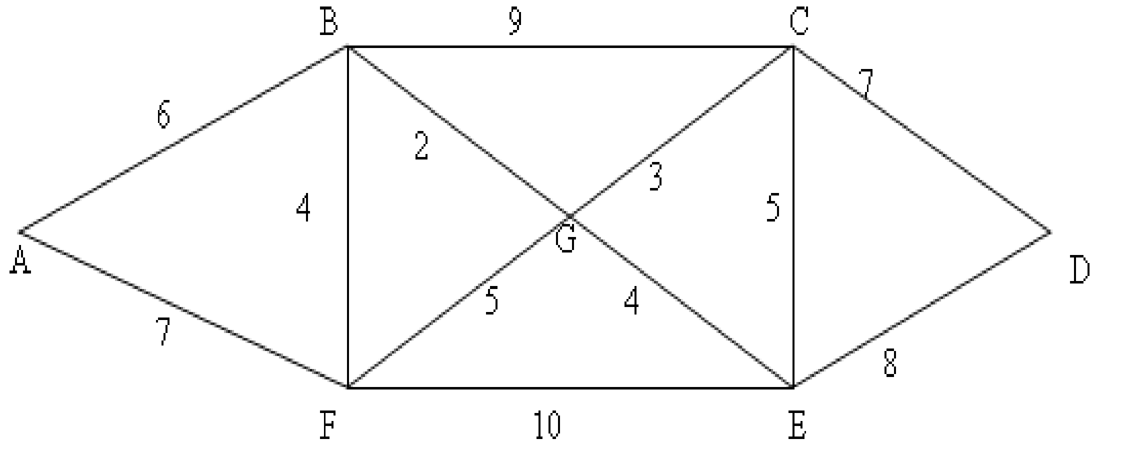
- Phương pháp: Xây dựng tập cạnh từ F của cây khung T =(V,F) theo từng bước sau:
  - Sắp xếp các cạnh của đồ thị theo thứ tự độ dài tăng dần
  - Sau đó ở mỗi bước ta sẽ lần lượt duyệt trong danh sách các cạnh đã sắp xếp để tìm ra một cạnh nhỏ nhất sao cho việc bổ xung cạnh đó vào F không tạo thành chu trình
  - Thuật toán kết thúc khi thu được tập F có n – 1 cạnh

4.2.2.3.1.Thuật toán Kruskal

```
Procedure Kruskal
Begin
  F:=Φ ;
While |F|<(n-1) and (E≠Φ)
begin
 chọn e có độ dài nhỏ nhất trong E;
 E:=E\{e};
 if F\{e} không chứa chu trình thì  F: =F∪{e}
End
if |F| < (n-1) then write(“Khong co cay khung “)
```

Ví dụ

- Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Kruskal



Ví dụ

- Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Kruskal

0	5	0	4	6	0	8	0
5	0	2	0	4	0	6	0
0	2	0	5	0	3	0	4
4	0	5	0	2	0	7	8
6	4	0	2	0	3	0	0
0	0	3	0	3	0	4	0
8	6	0	7	0	4	0	8
0	0	4	8	0	0	8	0

4.2.2.3.2.Thuật toán Prim

- Xây dựng cây khung T = (V,F) như sau:
  - Bắt đầu từ đỉnh s, nối s với đỉnh kề u gần s nhất
  - Trong số các trọng số các cạnh kề với hai đỉnh s, u ta tìm cạnh có độ dài nhỏ nhất, cạnh này dẫn tới đỉnh thứ ba v.
  - Quá trình tiếp tục cho tới khi thu được cây gồm n đỉnh và n – 1 cạnh

4.2.2.3.2.Thuật toán Prim

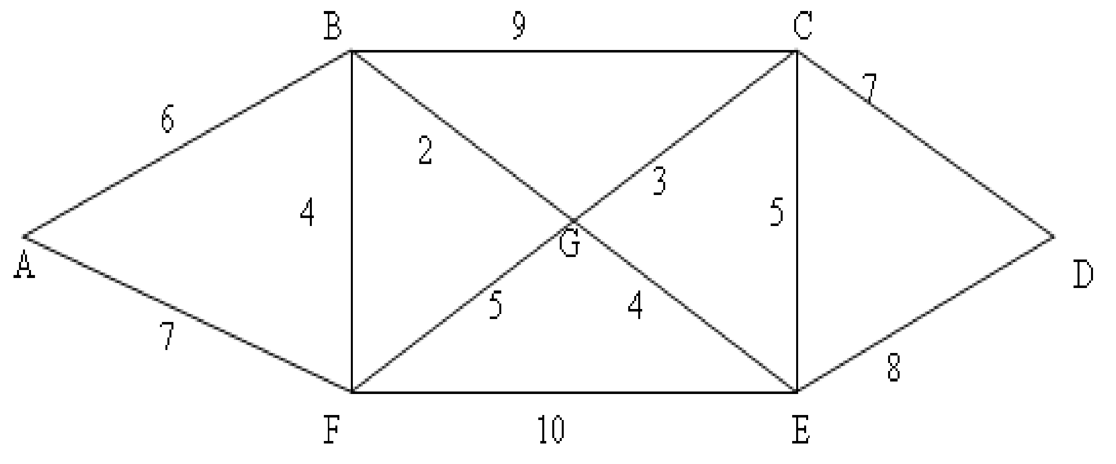
```
Procedure Prim
Begin
Chọn s là một đỉnh nào đó của đồ thị;
VT:={s}; F:=Φ ;
min(s):=0; near(s):=s;
For v ∈ V\VT do
  begin
     min(v):=c[s,v]; near(v):=s;
   end;
stop:=false;
```

4.2.2.3.2.Thuật toán Prim

```
While not stop do
Begin
        Tìm u ∈ V\VT có min(u) nhỏ nhất;
        VT:=VT ∪ {u}; F = F ∪ {(u,near(u))};
        if |VT |=n then
            begin
                T=(VT,F) là cây khung nhỏ nhất;
                 stop:=true;
            end
        else
        for v ∈ V/VT do
                if (min(v)>c[u,v]) then
                begin
                  min(v):=c[u,v];
                  near(v):=u;
                end;
    End;
    End;
```

Ví dụ

- Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Prim



Ví dụ

- Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Prim

0	5	0	4	6	0	8	0
5	0	2	0	4	0	6	0
0	2	0	5	0	3	0	4
4	0	5	0	2	0	7	8
6	4	0	2	0	3	0	0
0	0	3	0	3	0	4	0
8	6	0	7	0	4	0	8
0	0	4	8	0	0	8	0

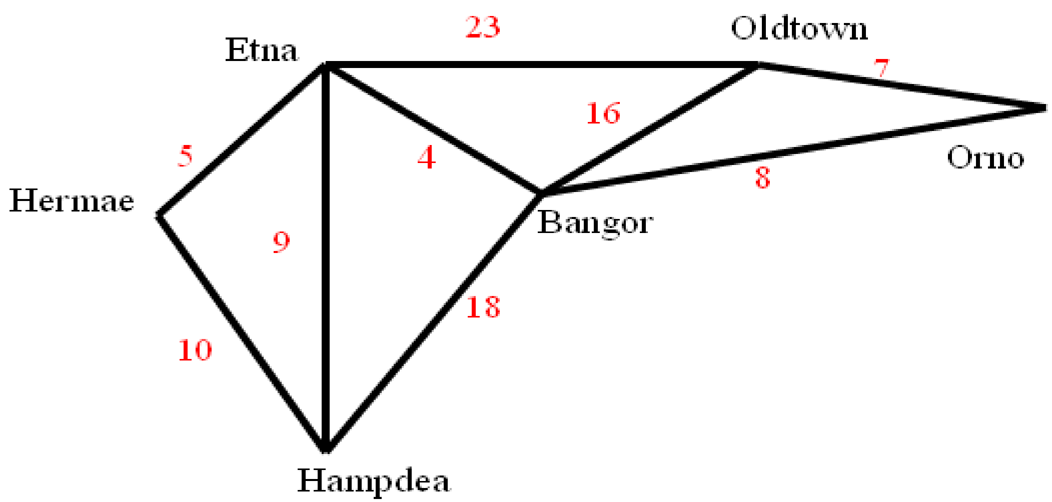
4.2.2.4.Ứng dụng của bài toán tìm cây khung nhỏ nhất

1. Bài toán xây dựng hệ thống đường sắt
2. Bài toán nối mạng máy tính

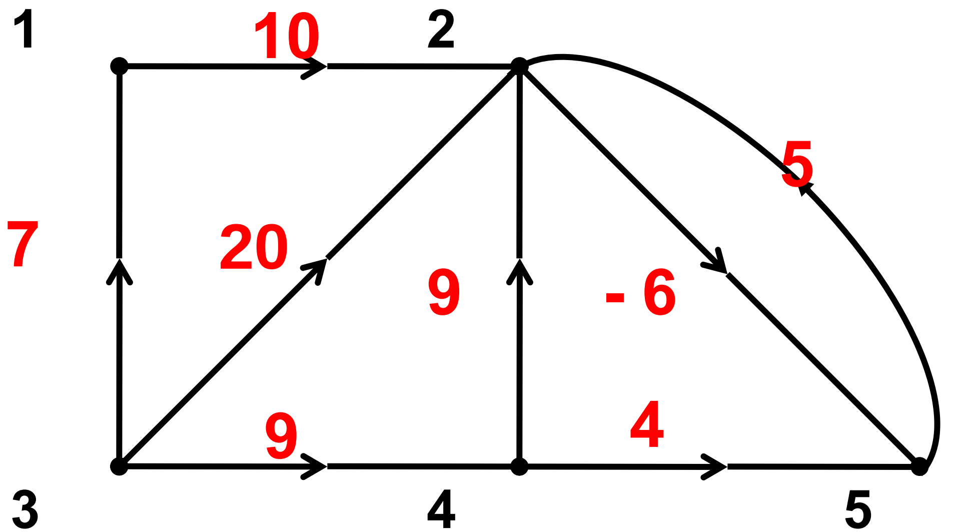
Bài toán Tìm đường đi ngắn nhất

4.5.1.Các khái niệm mở đầu

- **<u>Bài toán:</u>** Cho G = <V,E> là đồ thị có trọng số c(e), s và t là 2 đỉnh của đồ thị. Hãy tìm đường đi có tổng trọng số nhỏ nhất từ s đến t.



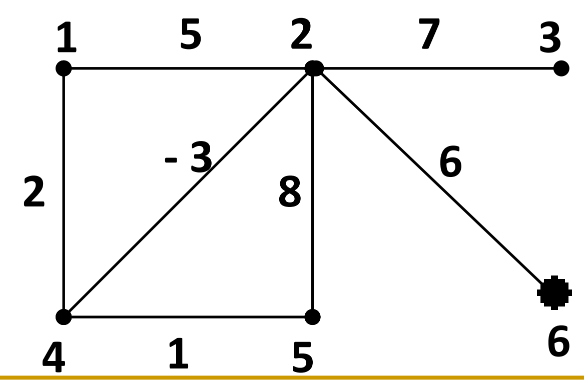
4.5.1.Các khái niệm mở đầu



- Tìm đường đi ngắn nhất từ đỉnh 3 đến đỉnh 5

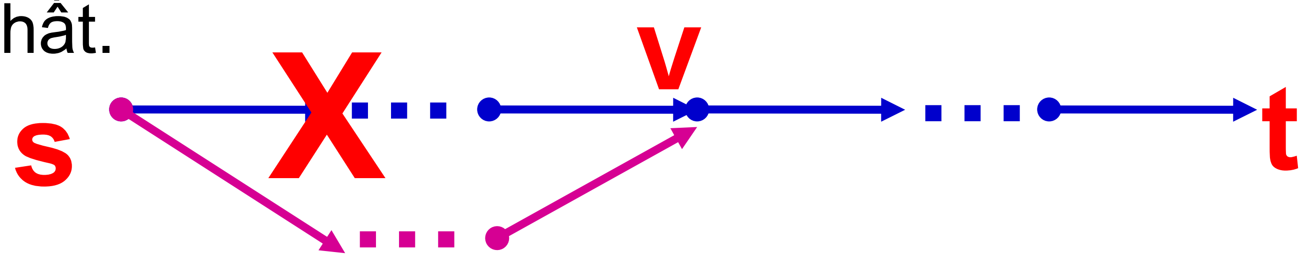
4.5.1.Các khái niệm mở đầu

- Điều kiện để bài toán có lời giải:
  - Phải tồn tại đường đi từ s đến t:
    - s và t nằm trong cùng một thành phần liên thông
    - Tồn tại đường đi từ s đến t
  - Trong đồ thị không tồn tại chu trình của đồ thị đi qua một cạnh có độ dài âm



Đường đi ngắn nhất xuất phát từ 1 đỉnh

- Nhận xét:
  - Nếu v là đỉnh trung gian trên đường đi ngắn nhất từ s đến t thì đường đi từ s đến v phải là ngắn nhất và đường đi từ v đến t cũng phải là ngắn nhất.



  - Do đó, để tối ưu, người ta mở rộng bài toán **tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại** của đồ thị.

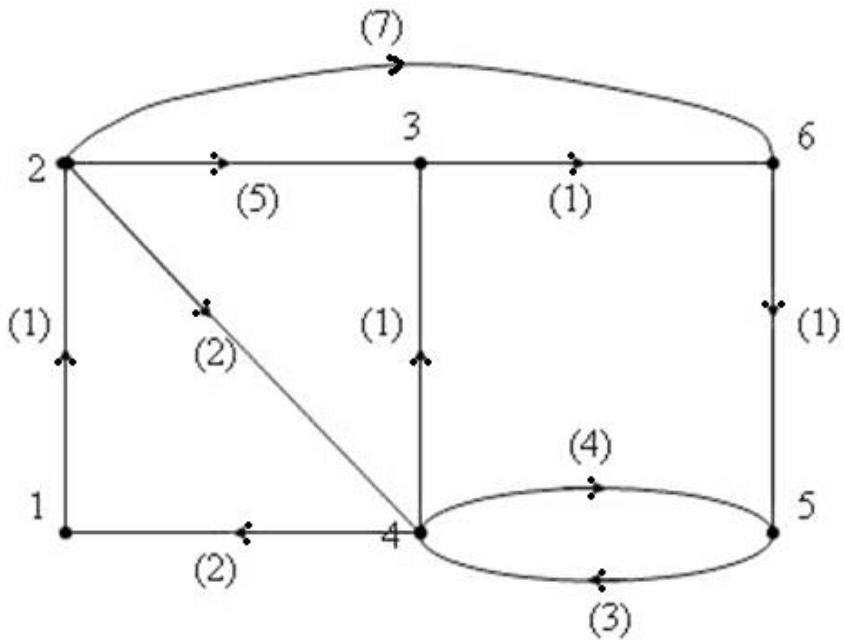
4.5.2. Thuật toán Dijkstra

- **Ý tưởng:**
  - Do không có cạnh âm nên tại mỗi bước, sẽ có đỉnh mà thông tin về nó sẽ không thay đổi về sau
  - Tại mỗi bước, ta không cần phải kiểm tra qua tất cả các đỉnh trung gian, mà chỉ thực hiện như sau:
    - Chọn một đỉnh u có giá trị d[u] nhỏ nhất
    - Chọn u làm đỉnh trung gian để xác định các bước kế tiếp

Thuật toán Dijsktra

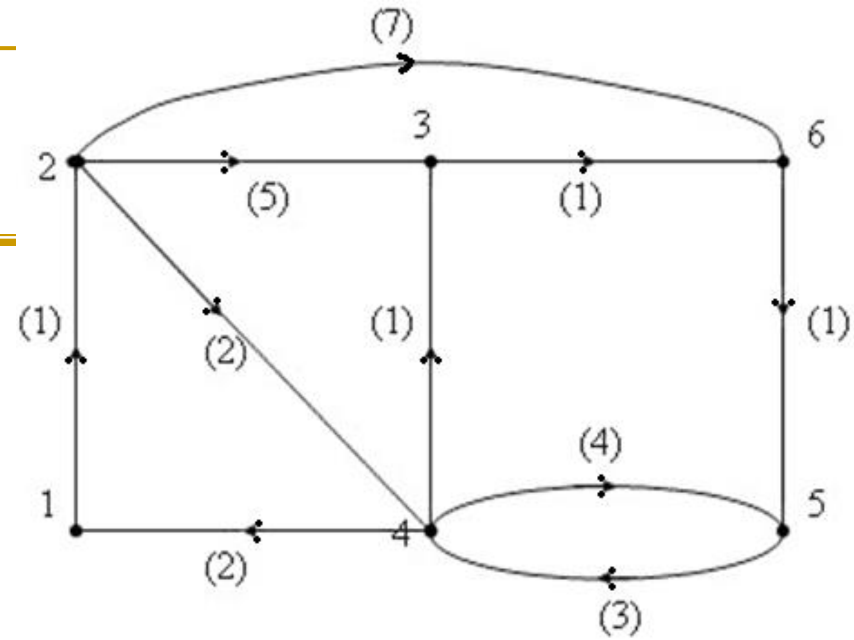
```
(* Khởi tạo *)
for v ∈ V do
    Begin
        d[v]:=a[s,v];
        Truoc[v]:=s;
    End;
d[s]:=0;  T:=V\{s} ;  (* T là tập các đỉnh chưa cố định *)
 (* Bước lặp *)
while T <> ∅ do
    Begin
        Tìm đỉnh u ∈ T thoả mãn d[u]=min{d[z]:z∈T};
        T:=T\{u} ;    (* Cố định nhãn của đỉnh u*)
        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;
```

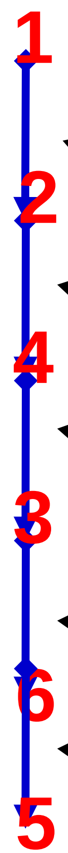
Thuật toán Dijsktra



k	1	2	3	4	5	6
	0,1	1,1*	∞ ,1	∞ ,1	∞,1	∞,1
1						
2						
3						
4						

Thuật toán Dijsktra





k	1	2	3	4	5	6
	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*	

HẾT