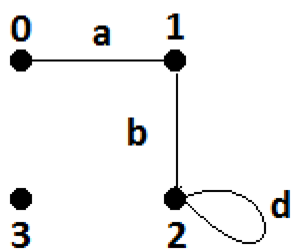
Chương 3. Cây

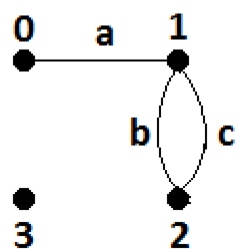
3.1 Cây :

3.1.1 Định nghĩa cây :

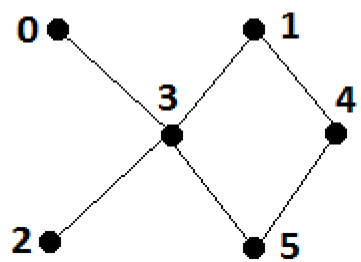
Cây T là một đồ thị đơn giản thỏa : Nếu v và w là hai đỉnh trong T , thì có một đường đi sơ cấp duy nhất nối v và w.



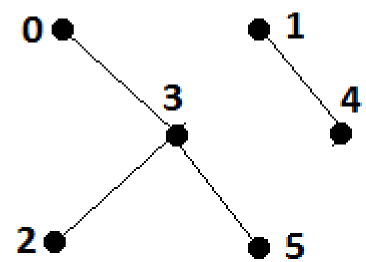
G1



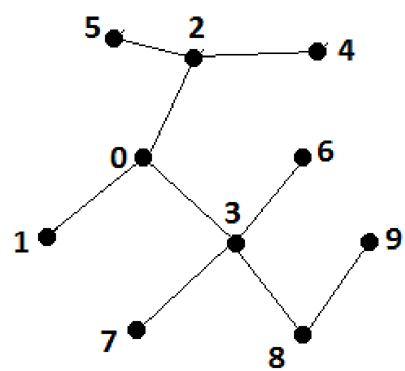
G2



G3



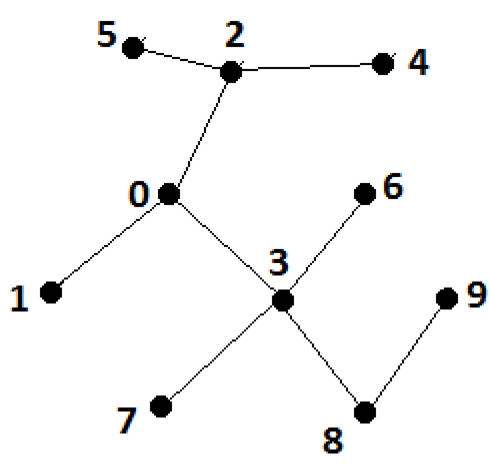
G4



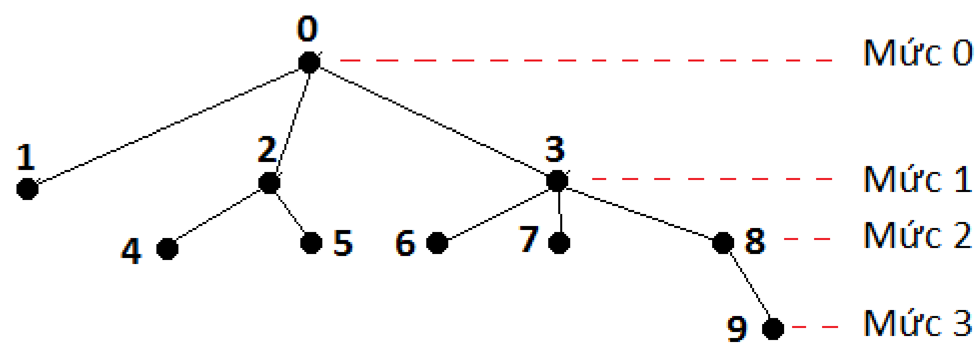
G5

3.1.2 Định nghĩa cây có gốc:

- Cây T có gốc là cây mà trong T một đỉnh v nào đó được chọn làm gốc (duy nhất).
- Độ dài (số cạnh) đường đi sơ cấp từ gốc đến đỉnh v được gọi mức của v. Gốc có mức 0.
- Mức cao nhất được gọi là chiều cao của cây.



H1



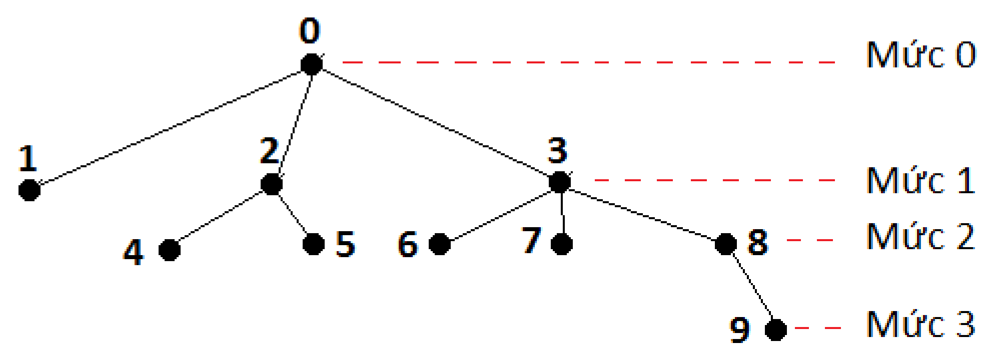
H2

3.1.3 Định nghĩa các thuật ngữ:

Cho T có gốc v_0. Giả sử x, y, và z các đỉnh trong T, và ($\mathbf{v_0}$, v_1, ..., v_{n-1}, v_n) đường sơ cấp trong T:

- v_{n-1} đỉnh (nút) cha của v_n.
- v_0, v_1, ..., v_{n-1} tổ tiên của v_n.
- v_n là con của v_{n-1}.
- Nếu x là tổ tiên của y, thì y là con cháu của x.
- Nếu x và y là các con của z, thì x và y anh em.
- Nếu x không có con , x là lá.
- Nếu x không là lá , x là đỉnh trong.

(**0**, 3, 8)

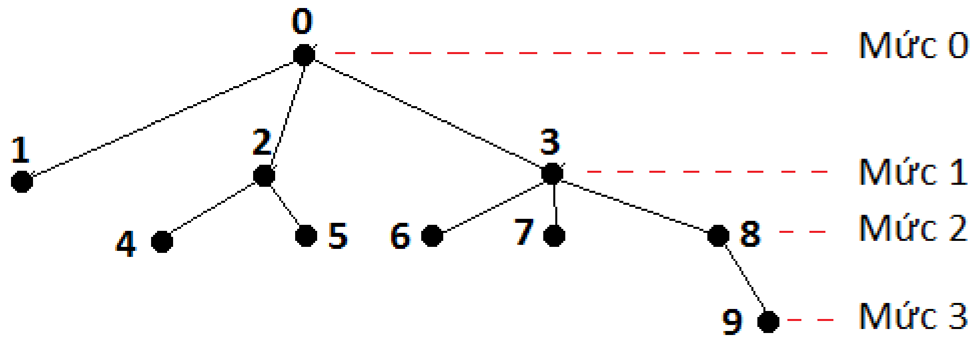


H2

- Cây con của *T có gốc x* là đồ thị với tập đỉnh V và tập cạnh E, trong đó

+ V gồm x và tất cả các con cháu của x,

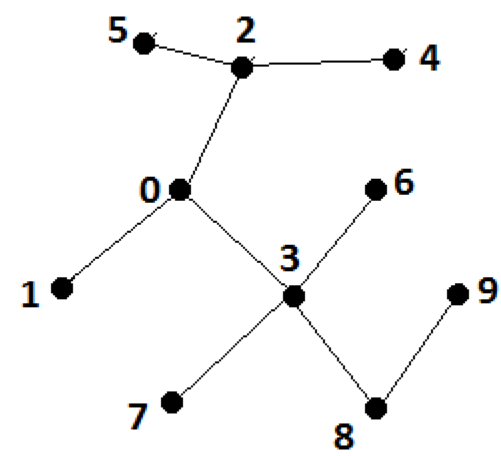
+ E = { e : e là một cạnh trên đường đi sơ cấp từ x tới một đỉnh nào đó thuộc V}.



H2

3.1.4 Định lý: Cho T là đồ thị có n đỉnh. Ta có :

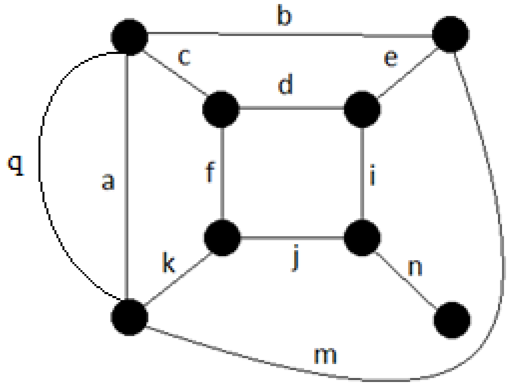
T là cây $\Leftrightarrow$ T liên thông và có n-1 cạnh.

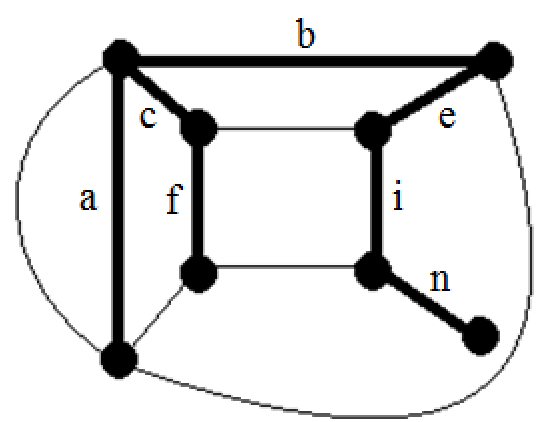


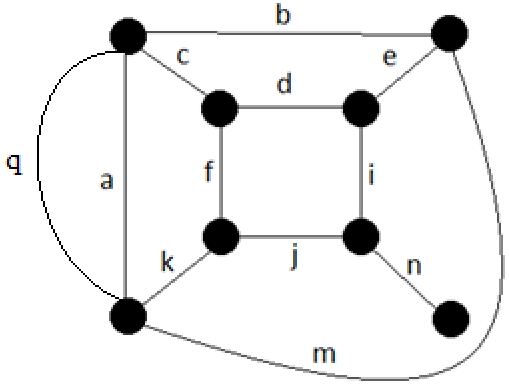
H1

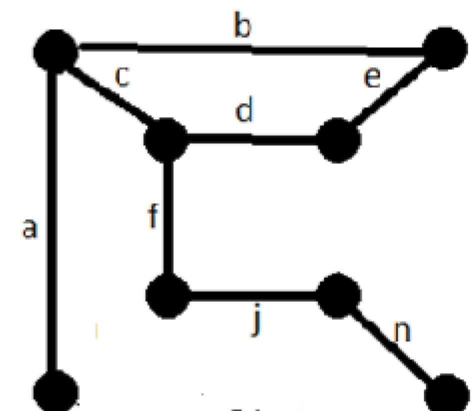
3.3 Cây khung (Spanning Tree):

3.3.1 Định nghĩa : *T là cây khung của một đồ thị G* =(V, E) nếu T là một **cây** có tập đỉnh là tập đỉnh V và tập cạnh là tập con của E của G.

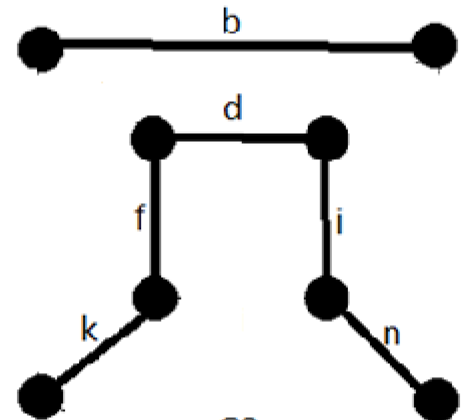




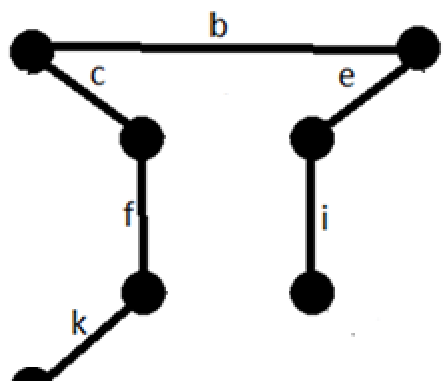




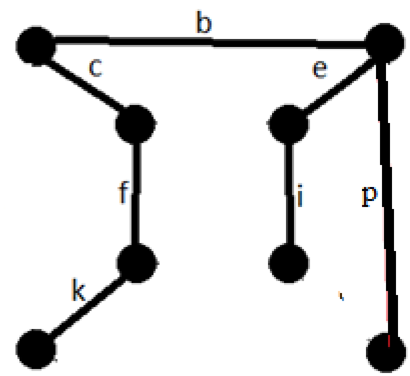
G1



G2



G3



G4

3.3.3 Thuật toán tìm cây khung:

Thuật toán 1: Breadth-First Search tìm spanning tree

Định nghĩa hàng đợi : Hàng đợi là một tập hợp có tính chất :

- Mỗi lần cho vào hàng đợi một phần tử,
- Mỗi lần chỉ lấy một phần tử ra khỏi hàng đợi,
- Phần tử vào trước sẽ lấy ra trước,

VD : Các phần tử lần lượt đưa vào hang đợi Q là 1, 4, 2, 3, thì thứ tự lấy ra là 1, 4, 2, 3. đỉnh hang đợi.

3	**2**	**4**	**1**

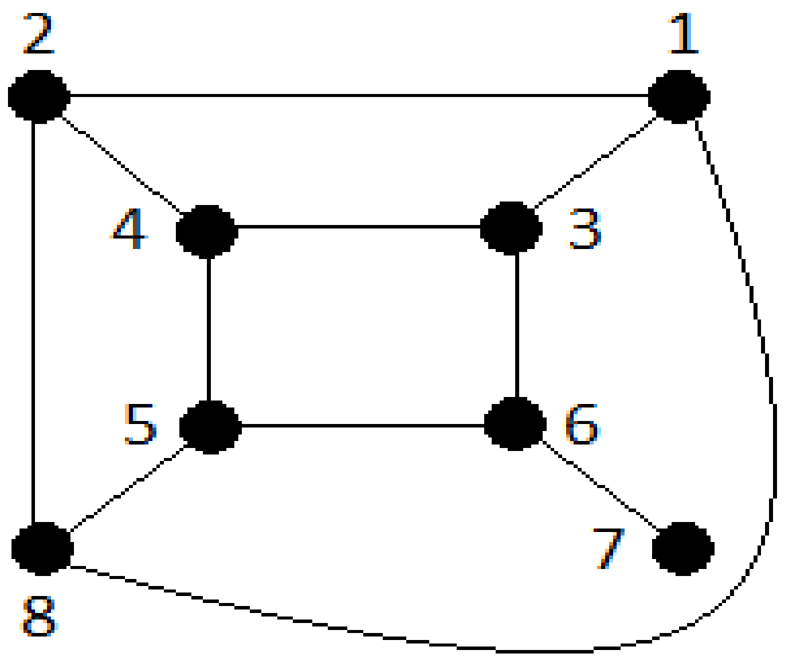
Tìm theo chiều rộng trước (Breadth-First Search, BFS) :

```
Tree_BFS(r)
{
  QUEUE = Ø;  QUEUE ⇐ r;  ChuaXet[r] = 0;
  while (QUEUE ≠ Ø)
  {
    v ⇐ QUEUE;
    for (u ∈ Ke(v)) /* u theo thứ tự từ nhỏ đến lớn */
      if (ChuaXet[u]==1)
      {
       QUEUE ⇐ u;  ChuaXet[u] = 0; T = T ∪ (v, u);
      }
   }/* Kết thúc while (QUEUE ≠ Ø)
  }/* Kết thúc Tree_BFS
```

```
main()
{
   Nhập đồ thị;
   for (v ∈ V)
   ChuaXet[v] = 1;
   T = ∅;
   Tree_BFS(root);
}
```

root : gốc cây khung

Ví dụ : root = 1

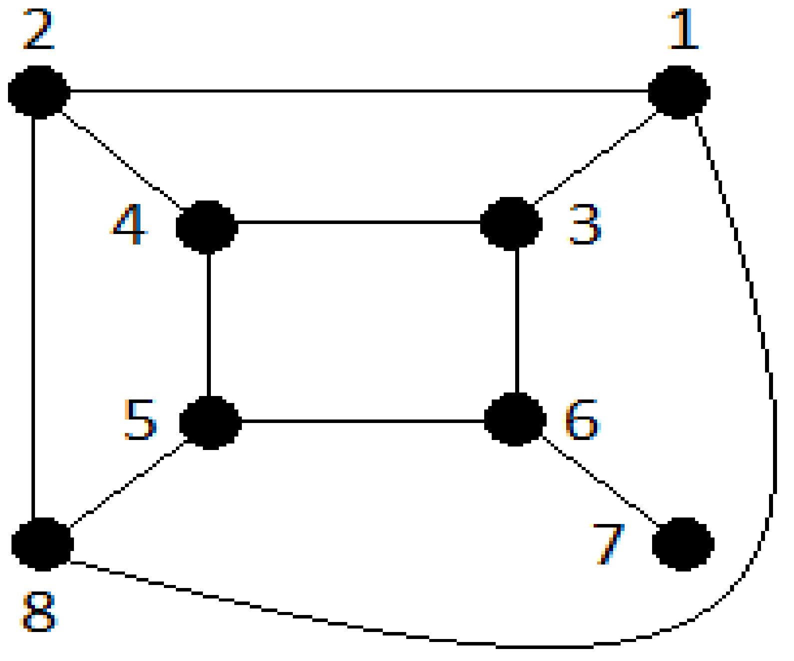


Tìm theo chiều sâu trước (Depth-First Search ,DFS)

```
void Tree_DFS(v);
{
    ChuaXet[v] = 0;
    for (u ∈ Ke(v))
   /* u theo thứ tự từ nhỏ đến lớn */
      if (ChuaXet[u])
      {
         T = T ∪ (v,u);
         Tree_DFS(u);
       };
} /* Kết thúc Tree_DFS */
```

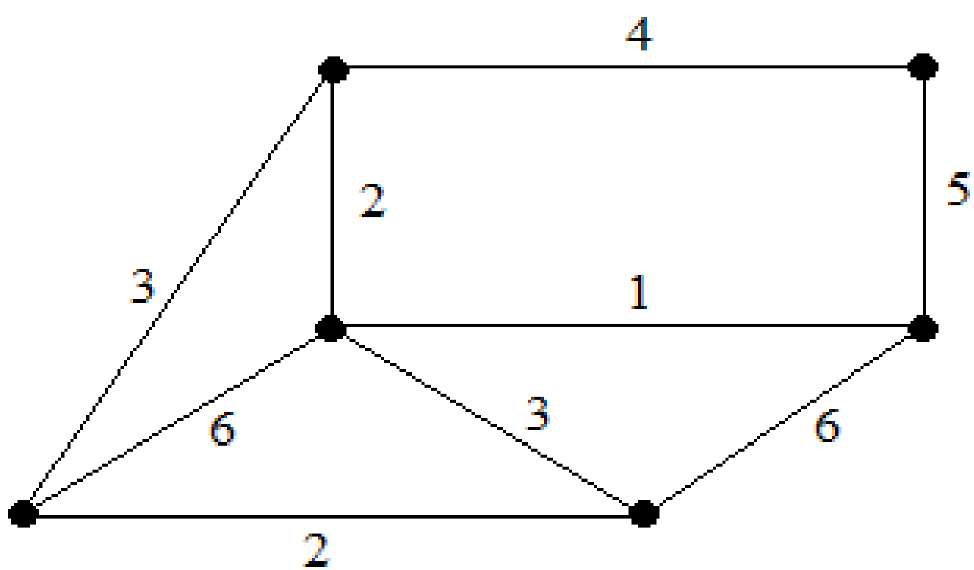
```
main()
{
   for (v ∈ V)
      ChuaXet[v] = 1;
   T = Ø;
   Tree_DFS(root);
}
 root : gốc cây khung
```

Ví dụ : root = 1



3.4 Cây khung bé nhất (Minium Spanning Tree):

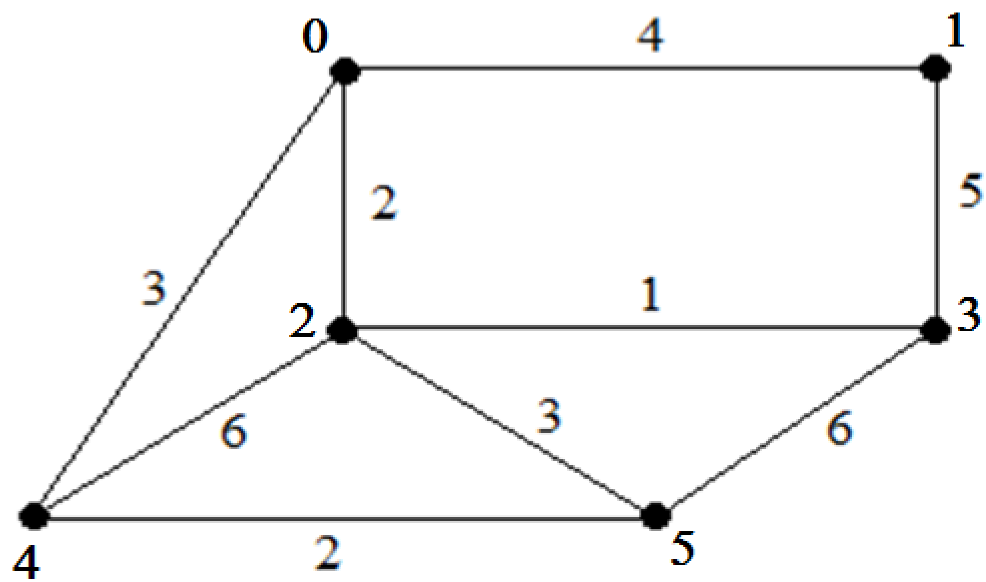
3.4.1 Định nghĩa: Cho G là đồ thị có trọng số. Cây khung bé nhất là cây khung có tổng các trọng số của các cạnh là bé nhất.



Thuật toán Kruskal :

```
Kruskal()
{
  1. T = ∅;
  2. for (v ∈ V)  MakeSet(v); //Tạo các tập hợp { v }
  3. xếp thứ tự các cạnh trong E tăng dần theo trọng số w;
  4. for (u,v) ∈ E (theo thứ tự đã sắp xếp) :
        if (FindSet(u) != FindSet(v))
          // Kiểm tra hai đỉnh u và v có khác nhãn ?
        {
          T = T ∪ {(u,v)};     //Chọn cạnh (u,v)
          Union(FindSet(u), FindSet(v));
                              // Gộp tập chứa v và tập chứ u thành một tập
        }
}
```

Ví dụ: Tìm cây khung bé nhất của đồ thị



Ví dụ: Tìm cây khung bé nhất của đồ thị

Bước	**Cạnh**	**T**	**Các tập**
1, 2, 3	Sắp xếp ↑	∅	{0}, {1}, {2}, {3}, {4}, {5}
4	**(2, 3)**		
	(0, 2)		
	(4, 5)		
	(0, 4)		
	(2, 5)		
	(0, 1)		
	(1, 3)		
	(2, 4)		
	(3, 5)		

Thuật toán Prim dạng 1:

+ Cho đồ thị G=(V,E), V là tập các đỉnh, E là tập các cạnh

+ V_T là tập các đỉnh đã được chọn.

+ F là tập các cạnh của cây khung cực tiểu.

+ w(u,v) là trọng số cạnh (u,v)

Bước 1: Khởi tạo

F = ∅;

V_T = {u}; // Gốc cây khung

Bước 2: Xây dựng cây khung

while (|F| < n-1)

{

B1. Chọn e = { w(u,v) bé nhất, với ($u \in V_T$) & ($v \notin V_T$)};

B2. $V_T = V_T \cup \{v\}$;

B3. $F = F \cup \{e\}$;

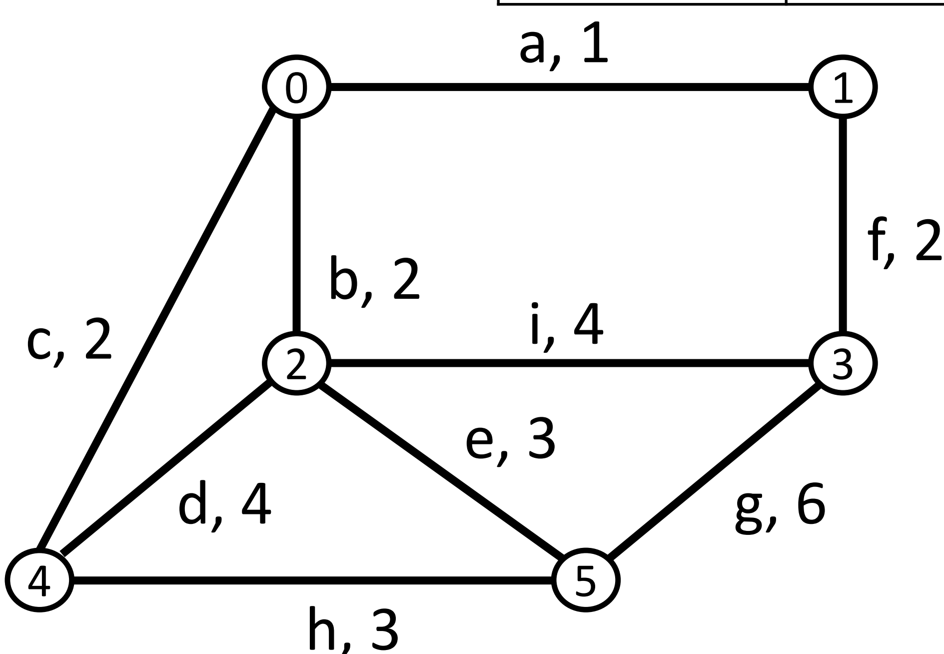
}

Lưu ý :

- u, v ở **B1** được chọn như sau : u là chỉ số bé nhất. Nếu có nhiều v kề u thỏa **B1** thì v bé nhất được chọn.
- Nếu các đỉnh được đánh nhãn là ký tự thì thứ tự theo thứ tự alphabet.

Bài giải: Tìm cây khung bé nhất của đồ thị với gốc = 0.

V_T	F



Thuật toán Prim (Dạng 2).

- Đồ thị không có cạnh song song.
- r : gốc.

Bước 1:

S = V

Bước 2:

Với mỗi u $\in$ S thực hiện:

{

key[u] = ∞

}

Bước 3 :

key[r] = 0

p[r] = -1

Bước 4:

while $S \neq \varnothing$

{ Tìm $u \in S$ với key[u] bé nhất

S=S-{u}

Với mỗi $v \in S$ kề với u thực hiện

if (w_{uv} < key[v]**)**

{ p[v] = u

key[v] = w_{uv}

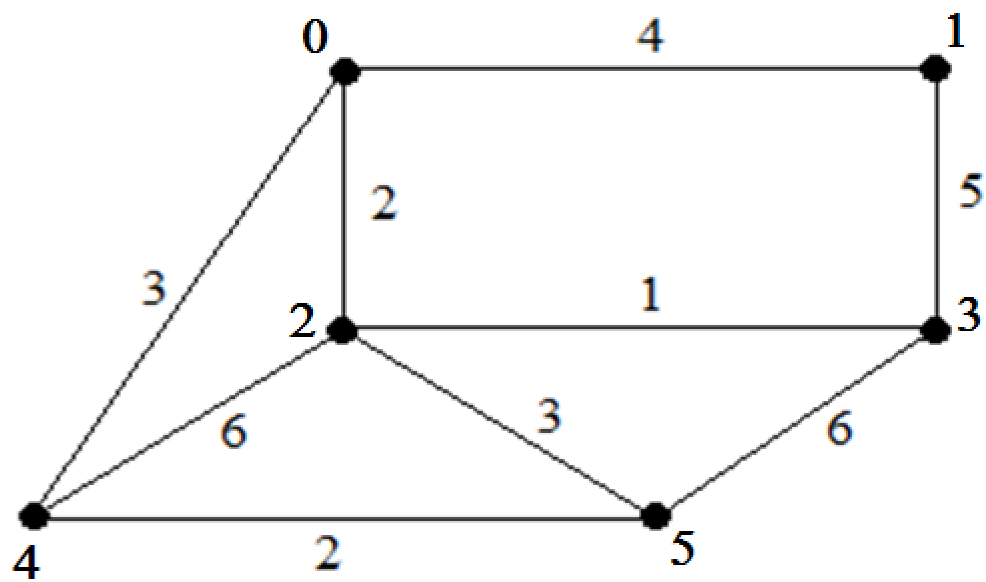
}

}

Bước 5:

Viết mảng p.

Ví dụ: Tìm cây khung bé nhất của đồ thị



r = 0

Bước	u	S	key[0]	key[1]	key[2]	key[3]	key[4]	key[5]
1		{0, 1, 2, 3, 4, 5}						
2, 3		{0, 1, 2, 3, 4, 5}	0, **-1**	∞	∞	∞	∞	∞
4	0	{1, 2, 3, 4, 5}		4, **0**	2, **0**		3, **0**	
	2	{1, 3, 4, 5}				1, **2**		3, **2**
	3	{1, 4, 5}						
	4	{1, 5}						2, **4**
	5	{1}						
	1	∅						
Kết quả			0, **-1**	4, **0**	2, **0**	1, **2**	3, **0**	2, **4**

p[2]

Tài liệu tham khảo:

1. Discrete Mathematics , Richard Johnsonbaugh
2. Algorithms, Thomas h. Cormen
3. Toán Rời Rạc Nâng Cao, Trần Ngọc Danh, ĐHQG TP HCM
4. Lý Thuyết Đồ Thị, Đặng Trường Sơn, Lê văn Vinh, ĐHSP Kỹ Thuật TP HCM