CHƯƠNG 3

Cây và cây khung

Cây và cây khung

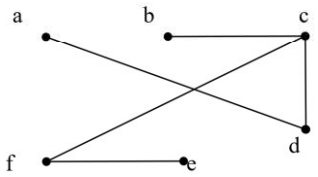
- Giới thiệu
- Cây là gì?
- Tính chất cơ bản của cây
- Cây khung
- Cây khung nhỏ nhất

Cây là gì?

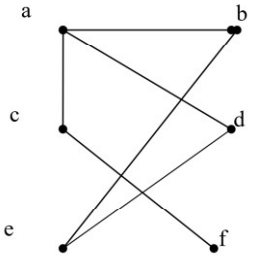
- ĐN1: Cây là đồ thị vô hướng liên thông không có chu trình.
- ĐN2: Rừng là ĐT mà mỗi thành phần liên thông của nó là một cây.

Cây là gì? (tt)

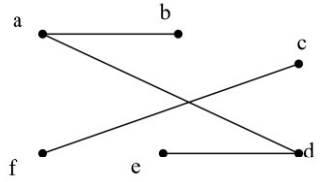
- *Ví dụ*: *Đồ thị nào dưới đây là cây?*



G1



G3



G2

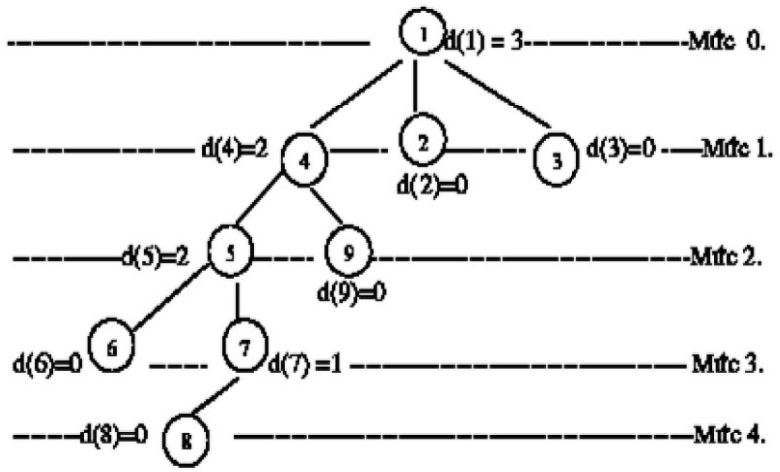
Cấu trúc cây (cây có gốc)

- Là ĐT có hướng, mỗi đỉnh đều có 1 đỉnh trước trừ 1 phần tử duy nhất không có gọi là *GỐC*.
- Xét một đỉnh x của một cây T có gốc là r :
  - Đỉnh y nằm trên đường hướng từ gốc đến x, gọi là *ĐỈNH TRƯỚC* của x, và x là *ĐỈNH SAU* của y.
  - Nếu (x,y) là một cạnh của T, ta gọi x là *CHA* của y và y là *CON* của x.
  - Hai đỉnh cùng cha gọi là *ANH EM*.
  - Đỉnh không có con gọi là *LÁ*.
  - Đỉnh không là lá gọi là *ĐỈNH TRONG*.

Cấu trúc cây (tt)

- Mức của đỉnh là khoảng cách từ gốc đến đỉnh.
  - Mức của nút gốc = 0.
- Chiều cao của cây = mức lớn nhất của đỉnh.
- Bậc của nút & bậc của cây .
  - Bậc của nút là số con của nút đó.
  - Bậc của cây là bậc lớn nhất của các nút của cây.
  - Nếu cây có bậc là k, ta gọi là cây k-phân.
  - Nếu mỗi đỉnh trong cây có tối đa hai con, thì ta gọi đó là cây nhị phân.

Cấu trúc cây (tt)



Các tính chất cơ bản của cây

- Cho G=(V,E) là đồ thị vô hướng n đỉnh, **các mệnh đề sau đây là tương đương**:
  - (1). T là cây.
  - (2). T không chứa chu trình và có n-1 cạnh.
  - (3). T liên thông và có n-1 cạnh.
  - (4). T liên thông và mỗi cạnh của nó đều là cầu.
  - (5). 2 đỉnh bất kỳ của T được nối với nhau bởi đúng 1 đường đi đơn.
  - (6). T không chứa chu trình nhưng nếu thêm vào 1 cạnh ta thu được đúng 1 chu trình.

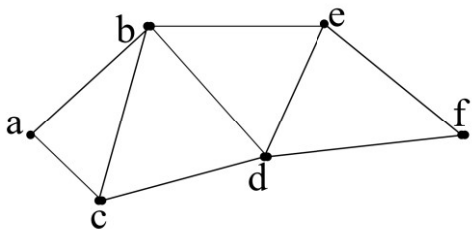
Một số ứng dụng của cây

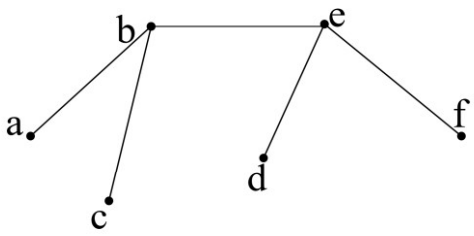
- Cây tìm kiếm nhị phân:
  - Tìm 1 phần tử trong ds.
- Cây quyết định:
  - Mỗi *đỉnh trong* ứng với 1 quyết định
  - Mỗi cây con tại các đỉnh này ứng với 1 kết quả của quyết định.
  - Những lời giải tương ứng với các đường đi tới các lá của cây.

Cây khung

- Giới thiệu bài toán:

  Hệ thống giao thông ở thành phố A





Cây khung (tt)

- Định nghĩa:
  - Cho G là một đơn đồ thị. Một cây được gọi là **cây khung của G** nếu nó là một *đồ thị con* của G và *chứa tất cả các đỉnh* của G.
- Định lý (sự tồn tại cây khung)
  - Mọi đồ thị liên thông đều chứa ít nhất một cây khung

Cây khung (tt)

- Tìm cây khung
  - **B1**: Chọn tuỳ ý 1 đỉnh của G đặt vào H
  - **B2**: Nếu mọi đỉnh của G đều nằm trong H thì dừng
  - **B3**: Nếu không, tìm 1 đỉnh $\in$ G nhưng $\notin$ H, sao cho 1 cạnh e nào đó của G nối nó với 1 đỉnh của H. Thêm đỉnh và cạnh này vào H. Quay về B2

Cây khung (tt)

- Giải thuật tìm cây khung
  - Tìm cây khung bằng DFS
  - Tìm cây khung bằng BFS

Cây khung (tt)

- Tìm cây khung của đồ thị dưới đây bằng giải thuật DFS và BFS?

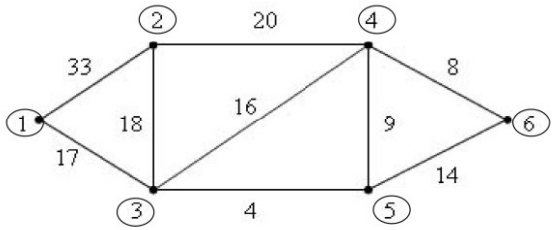
Cây khung nhỏ nhất

- ĐN: Cho đồ thị G vô hướng, liên thông và có trọng số không âm. Cây khung nhỏ nhất của đồ thị G là cây khung có tổng trọng số trên các cạnh của nó nhỏ nhất.
- Giải thuật:
  - Giải thuật **Kruskal**
  - Giải thuật **Prim**

Giải thuật **Kruskal**

- **B1**. Sắp xếp các cạnh có trọng số *tăng dần*
- **B2**. Lần lượt chọn cạnh có *trọng số nhỏ nhất* trong số các cạnh còn lại để đưa vào cây nếu sau khi đưa vào *không tạo thành chu trình*.
- **B3**. Chọn đủ n-1 cạnh thì dừng.

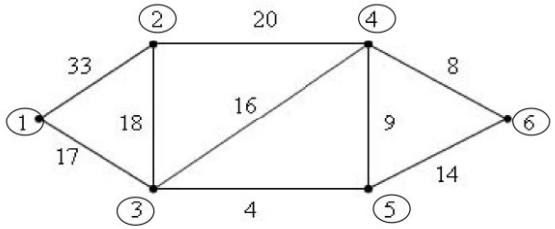
Giải thuật **Kruskal** (tt)



Giải thuật **Prim**

- **B1**. Chọn đỉnh đầu tiên *tuỳ ý* đưa vào cây (nếu không yêu cầu thì có thể chọn đỉnh 1).
- **B2**. Lần lượt chọn n-1 đỉnh còn lại (tương ứng n-1 cạnh) vào cây khung bằng cách: mỗi lần *chọn 1 cạnh có trọng số nhỏ nhất* mà một đầu của cạnh đã thuộc cây, đầu kia chưa thuộc cây.

Giải thuật **Prim**



Giải thuật **Prim** (tt)

```
//tìm cạnh có trọng số nhỏ nhất trong các cạnh có 1 đầu đã nạp và 1 đầu
    chưa nạp vào cây khung nhỏ nhất đang hình thành
void  timcanh (int *x, int *y)
{min=32000;
 for (i=1; i<=n; i++)
    if (chuaxet[i]= =1)              //đỉnh i đã đưa vào cây khung
       for (j=1; j<=n; j++)
          if (chuaxet[j]= =0)                 //đỉnh chưa xét
             if (a[i][j]>0 and a[i][j]<min)
                   { *x=i;
                     *y=j;
                     min=a[i][j];
                   }
}
```

Giải thuật **Prim** (tt)

- Trong quá trình cài đặt, để chọn đỉnh và cạnh cần bổ sung vào cây khung, ta có thể dùng *phương pháp gán nhãn cho đỉnh*:
- Mỗi đỉnh ***s*** được gán cho 1 nhãn là *(d[s], near[s])* với ý nghĩa:
  - *near[s]* là đỉnh đã nạp vào cây khung mà *near[s]* **kề** và **gần s nhất** . near[s] có thể gọi là đỉnh trước của s
- *d[s]* là độ dài cạnh *(near[s], s)*

Giải thuật **Prim** (tt)

- **B1**. Khởi tạo và gán nhãn ban đầu.
  - Lấy 1 đỉnh ***s*** tuỳ ý đưa vào cây khung. Khi đó tập đỉnh của cây khung có duy nhất 1 đỉnh là VH = {***s***}. Tập cạnh T là rỗng.
  - Với mỗi đỉnh v kề s thì gán nhãn (d[v], s), với d[v] là trọng số cạnh (s,v).
  - Các đỉnh v không kề s thì gán nhãn (∞, s)

Giải thuật **Prim** (tt)

- **B2**. Vòng lặp cho đến khi chọn đủ n-1 cạnh (hoặc xét hết các đỉnh)
- Vòng lặp thực hiện các thao tác sau:
  - Đưa đỉnh và cạnh và cây: Chọn đỉnh u ngoài tập Đ, có nhãn d[u] nhỏ nhất, nạp đỉnh u đó vào tập Đ. Vậy Đ=Đ+{u}. nạp cạnh (s,u) vào tập cạnh T.
  - Nếu số cạnh (hoặc đỉnh) thoả → kết thúc vòng lặp
  - Sửa nhãn: sau khi đỉnh u nạp vào cây khung, nó sẽ tạo cho 1 số đỉnh kề với nó gần cây khung hơn, nên phải sửa lại nhãn. Cụ thể: xét các đỉnh v $\notin$ Đ, nếu d[v]>c[u][v] thì d[v]=c[u]v].

Câu hỏi???