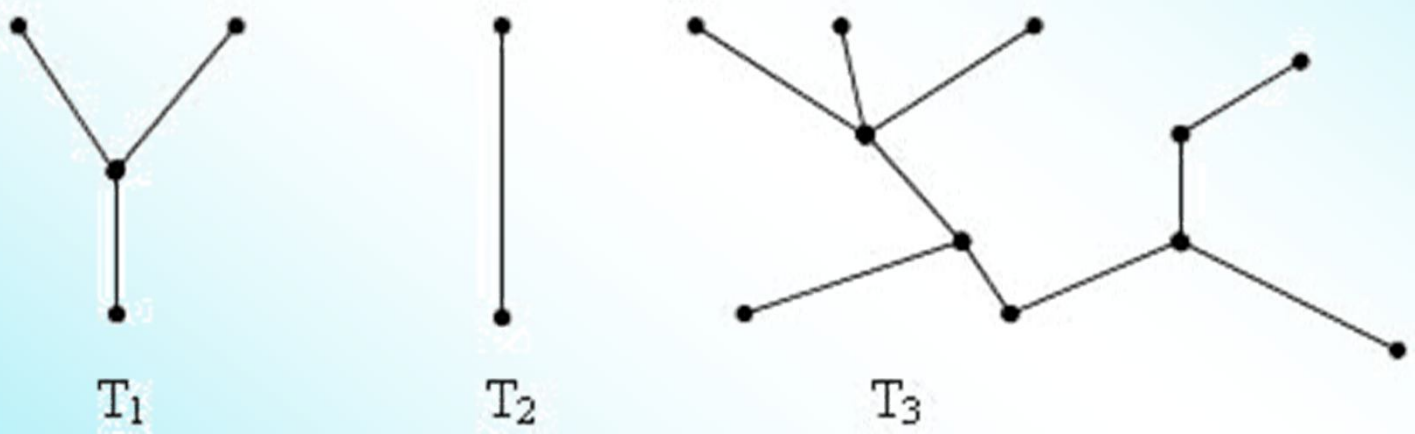
Cây và Cây khung của đồ thị

Cây

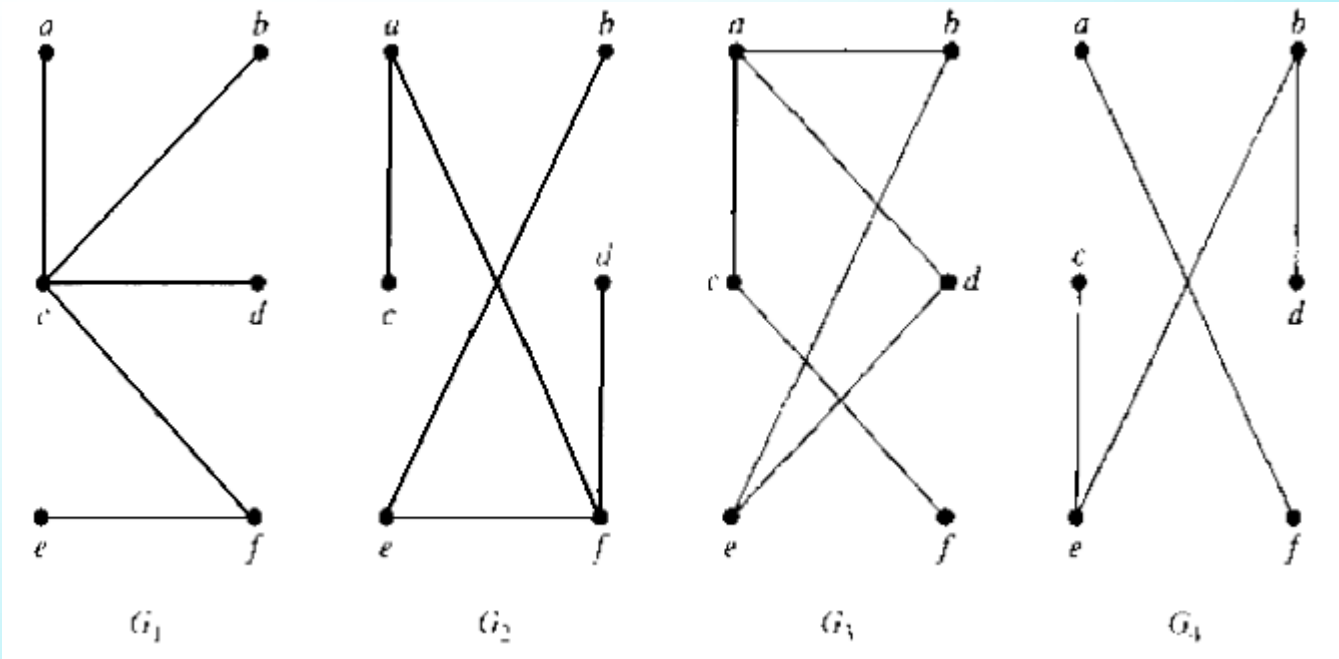
- **Định nghĩa:** **Cây** là một đơn đồ thị **vô hướng**, **liên thông** và **không chứa chu trình**.
- **Ví dụ:** Trong các đồ thị sau, đồ thị nào là cây?



- Cả 3 đồ thị trên đều là cây.

Cây (tt)

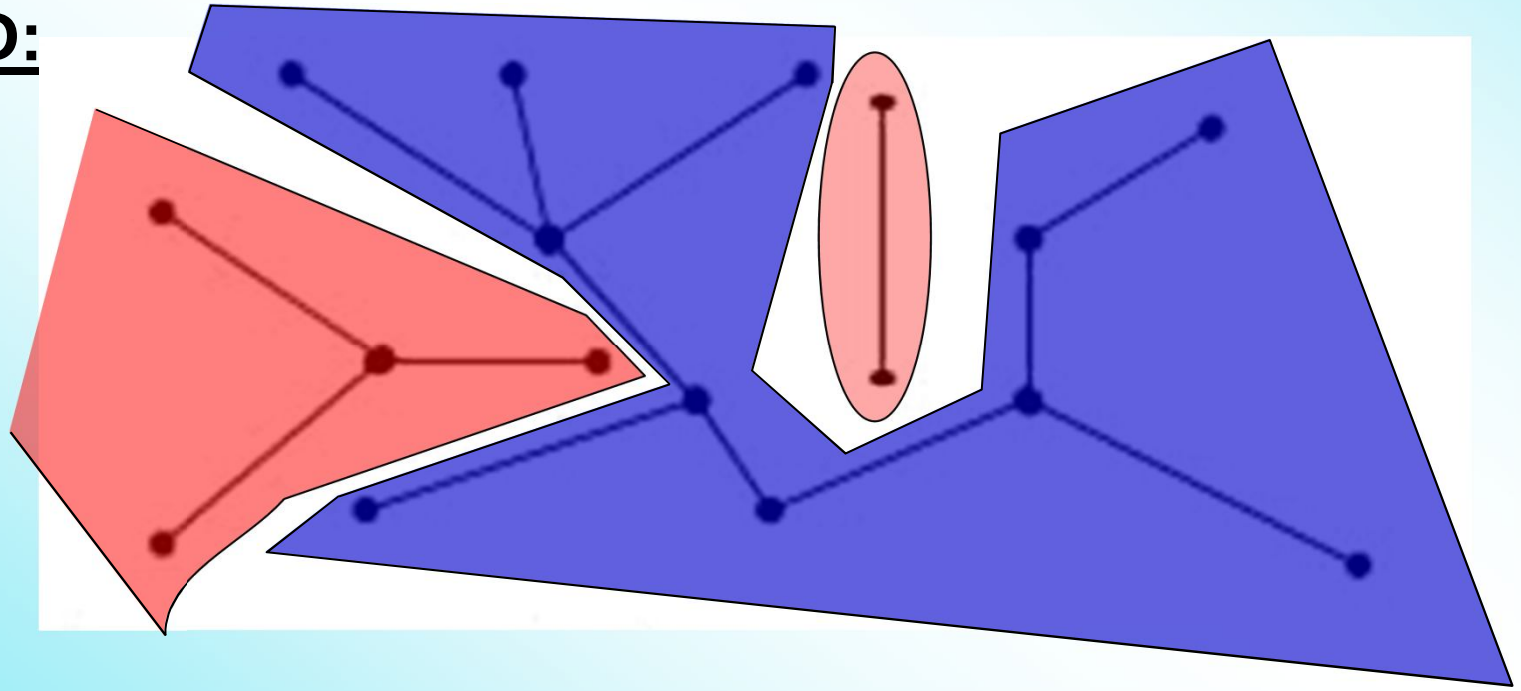
- **VD:** Trong các đồ thị sau, đồ thị nào là cây?



- G1, G2 là cây. G3, G4 không là cây do có chứa chu trình

Cây (tt)

- **Định nghĩa:** Nếu G là một đồ thị vô hướng và không chứa chu trình thì G được gọi là một **rừng**. Khi đó mỗi thành phần liên thông của G sẽ là một cây.
- **VD:**



- Đồ thị trên là rừng có 3 cây

Tính chất của cây

- **Định lý:** Cho T là một đồ thị vô hướng. Khi đó, các điều 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 T đều là cạnh cắt (cầu).
  5. Hai đỉnh bất kỳ của T được nối với nhau bằng đúng 1 đường đi đơn.
  6. T không chứa chu trình nhưng nếu thêm 1 cạnh bất kỳ vào T thì ta sẽ được thêm đúng 1 chu trình.

Tính chất của cây (tt)

- **Chứng minh định lý:**
  - (1) $\Rightarrow$ (2): T là cây $\Rightarrow$ T không chứa chu trình và có n-1 cạnh
    - Hiển nhiên T không chứa chu trình (do T là cây)
    - Ta chỉ cần chứng minh T có n-1 cạnh.
    - Xét T_n là cây có n đỉnh. Ta sẽ chứng minh quy nạp theo n
      - n = 2, Cây có 2 đỉnh thì có 1 cạnh. Đúng.
      - Giả sử mọi cây có k đỉnh thì sẽ có k-1 cạnh
      - Xét T_{k+1} là cây có k + 1 đỉnh. Dễ thấy rằng trong cây T_{k+1} luôn tồn tại ít nhất 1 đỉnh treo.
      - Loại đỉnh treo này (cùng với cạnh nối) ra khỏi T_{k+1} ta được đồ thị T' có k đỉnh. Dễ thấy T' vẫn liên thông và không có chu trình (do T_{k+1} không có chu trình)
      - Suy ra T' là cây. Theo giả thiết quy nạp, T' có k đỉnh thì sẽ có k-1 cạnh. Vậy cây T_{k+1} có k cạnh. (đpcm)

Tính chất của cây (tt)

- **Chứng minh định lý (tt):**
  - (2) $\Rightarrow$ (3): T không chứa chu trình và có n-1 cạnh $\Rightarrow$ T liên thông và có n-1 cạnh
    - Hiển nhiên T có n-1 cạnh (theo giả thiết)
    - Ta chỉ cần chứng minh T liên thông.
    - Giả sử T có k thành phần liên thông với số đỉnh lần lượt là $n_1,\ldots, n_k$.
    - Khi đó mỗi thành phần liên thông của T sẽ là một cây và sẽ có số cạnh lần lượt là n_1-1, n_2-1,…, n_k-1.
    - Suy ra, số cạnh của T sẽ là n_1-1 + n_2-1 +…+ n_k-1 = n – k.
    - Theo giả thiết, số cạnh của cây là n-1. Từ đó suy ra k = 1 hay T chỉ có 1 thành phần liên thông. Suy ra T liên thông (đpcm).

Tính chất của cây (tt)

- **Chứng minh định lý (tt):**
  - (3) $\Rightarrow$ (4): T liên thông và có n-1 cạnh $\Rightarrow$ T liên thông và mỗi cạnh của T đều là cạnh cắt (cầu)
    - Hiển nhiên T liên thông (theo giả thiết)
    - Ta chỉ cần chứng minh mỗi cạnh của T đều là cạnh cắt (cầu).
    - Xét (u,v) là cạnh bất kỳ của T. Nếu bỏ (u,v) ra khỏi T, ta sẽ được đồ thị T' có n đỉnh và n-2 cạnh.
    - Ta đã chứng minh được đồ thị có n đỉnh và n-2 cạnh thì không thể liên thông.
    - Vậy nếu bỏ cạnh (u,v) ra thì sẽ làm mất tính liên thông của đồ thị. Suy ra (u,v) là cạnh cắt (cầu). (đpcm).

Tính chất của cây (tt)

- **Chứng minh định lý (tt):**
  - (4) $\Rightarrow$ (5): T liên thông và mỗi cạnh của T đều là cạnh cắt (cầu) $\Rightarrow$ Giữa hai đỉnh bất kỳ của T luôn tồn tại đúng 1 đường đi đơn
    - Xét u, v là hai đỉnh bất kỳ trong T.
    - Do T liên thông nên luôn tồn tại đường đi giữa u và v. Ta sẽ chứng minh đường đi này là duy nhất.
    - Giả sử có hai đường đi đơn khác nhau giữa u và v. Khi đó hai đường đi này sẽ tạo thành một chu trình.
    - Suy ra, các cạnh trên chu trình này sẽ không thể là cạnh cắt được (???) – Mâu thuẫn.
    - Vậy giữa u và v chỉ có thể tồn tại đúng 1 đường đi đơn. (đpcm)

Tính chất của cây (tt)

- **Chứng minh định lý (tt):**
  - (5) $\Rightarrow$ (6): Giữa hai đỉnh bất kỳ của T luôn tồn tại đúng 1 đường đi đơn $\Rightarrow$ T không chứa chu trình, nhưng nếu thêm vào 1 cạnh bất kỳ thì sẽ phát sinh đúng 1 chu trình
    - T không thể có chu trình, vì nếu có chu trình thì giữa hai đỉnh trên chu trình này sẽ có 2 đường đi đơn khác nhau – mâu thuẫn với GT.
    - Giả sử ta thêm vào T cạnh (u,v) bất kỳ (trước đó không có cạnh này trong T).
    - Khi đó cạnh này sẽ tạo với đường đi duy nhất giữa u và v trong T tạo thành 1 chu trình duy nhất. (Vì nếu tạo thành 2 chu trình thì chứng tỏ trước đó có 2 đường đi khác nhau giữa u và v – mâu thuẫn với giả thiết)

Tính chất của cây (tt)

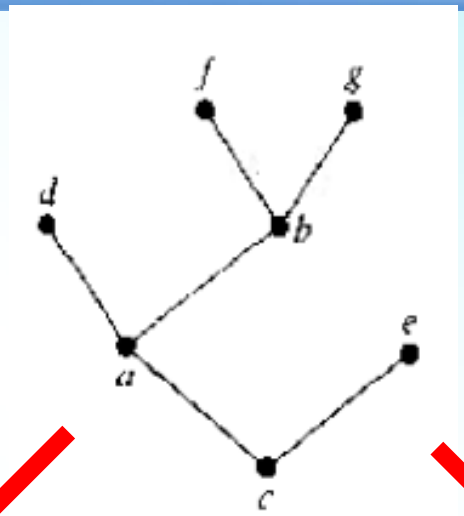
- **Chứng minh định lý (tt):**
  - (6) $\Rightarrow$ (1): T không chứa chu trình, nhưng nếu thêm vào 1 cạnh bất kỳ thì sẽ phát sinh đúng 1 chu trình $\Rightarrow$ T là cây
    - Hiển nhiên T không chứa chu trình (theo giả thiết).
    - Giả sử T không liên thông. Khi đó T sẽ có nhiều hơn 1 thành phần liên thông
    - Suy ra, nếu thêm vào một cạnh bất kỳ giữa hai đỉnh thuộc 2 thành phần liên thông khác nhau sẽ không tạo thêm chu trình nào – mâu thuẫn với giả thiết.
    - Vậy, T phải liên thông. Suy ra T là cây. (đpcm)

Cây có gốc

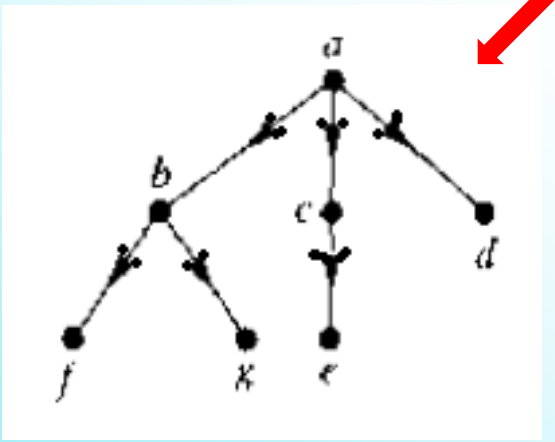
- Trong một số cây, một đỉnh đặc biệt được chọn làm gốc
- Đường đi từ gốc đến các đỉnh được định hướng từ gốc đến đỉnh đó
- Suy ra một cây cùng với gốc sẽ sinh ra đồ thị có hướng, được gọi là cây có gốc.
- Trong cây có gốc:
  - Mỗi đỉnh chỉ có một cha duy nhất – là đỉnh mà trực tiếp đi đến nó trên đường đi từ gốc
  - Mỗi đỉnh có thể không có, có 1 hoặc nhiều đỉnh con
  - Các đỉnh có con được gọi là đỉnh trong, các đỉnh không có con được gọi là đỉnh ngoài (nút lá)

Cây có gốc (tt)

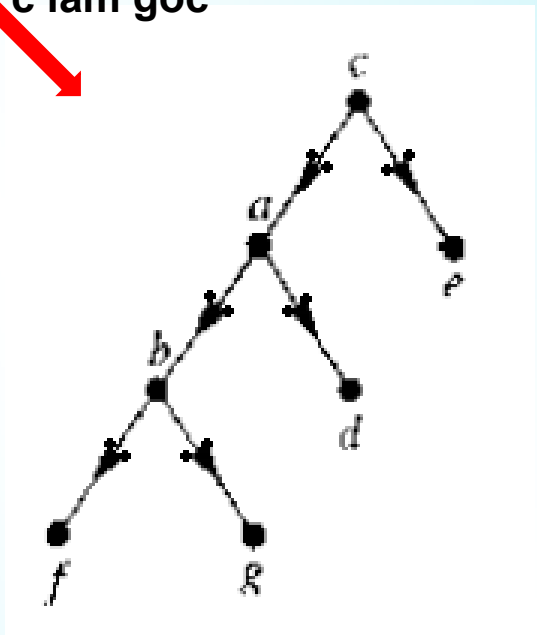
- **VD:**



Chọn đỉnh a làm gốc

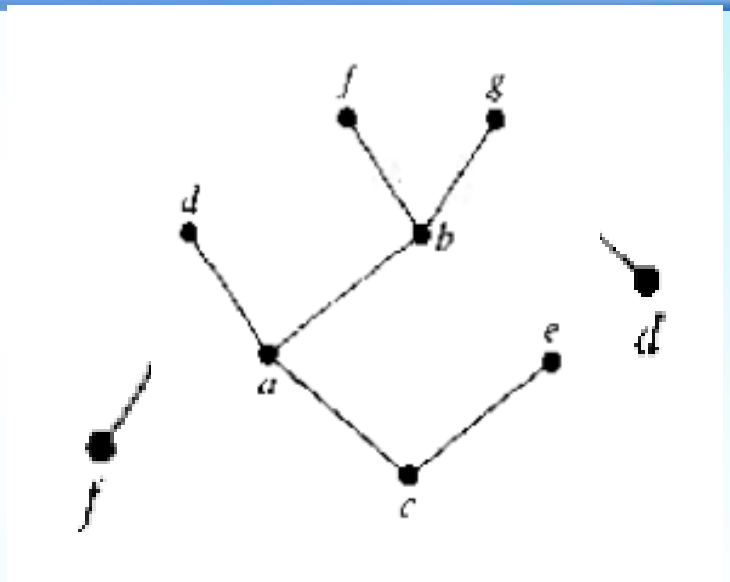


Chọn đỉnh c làm gốc



Cây có gốc (tt)

- **VD:**



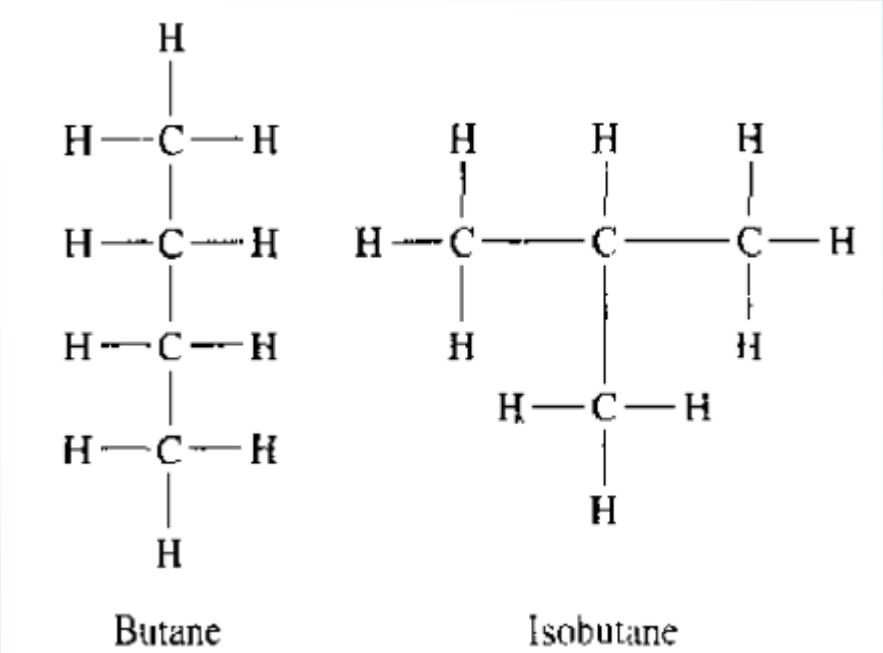
- Đỉnh a là đỉnh gốc
- Các đỉnh con của đỉnh a: b, c và d.
- Đỉnh cha của đỉnh f: đỉnh b (duy nhất)
- Các đỉnh trong: a, b, và c.
- Các đỉnh ngoài (lá): f, k, e và d.

Cây có gốc (tt)

- **Định nghĩa:**
  - Cây có gốc được gọi là cây m-phân nếu tất cả các đỉnh trong của nó đều có không quá m đỉnh con.
  - Cây được gọi là m-phân đầy đủ nếu tất cả các đỉnh trong của nó đều có đúng m đỉnh con
  - Với m = 2, ta có cây nhị phân.
- **Định nghĩa:** Cây có gốc được sắp (hay có thứ tự) là cây có gốc trong đó các con của mỗi đỉnh luôn được sắp theo thứ tự nào đó (thường là lớn dần từ trái sang phải)

Các mô hình dạng cây

- Các Hydrocarbon no:



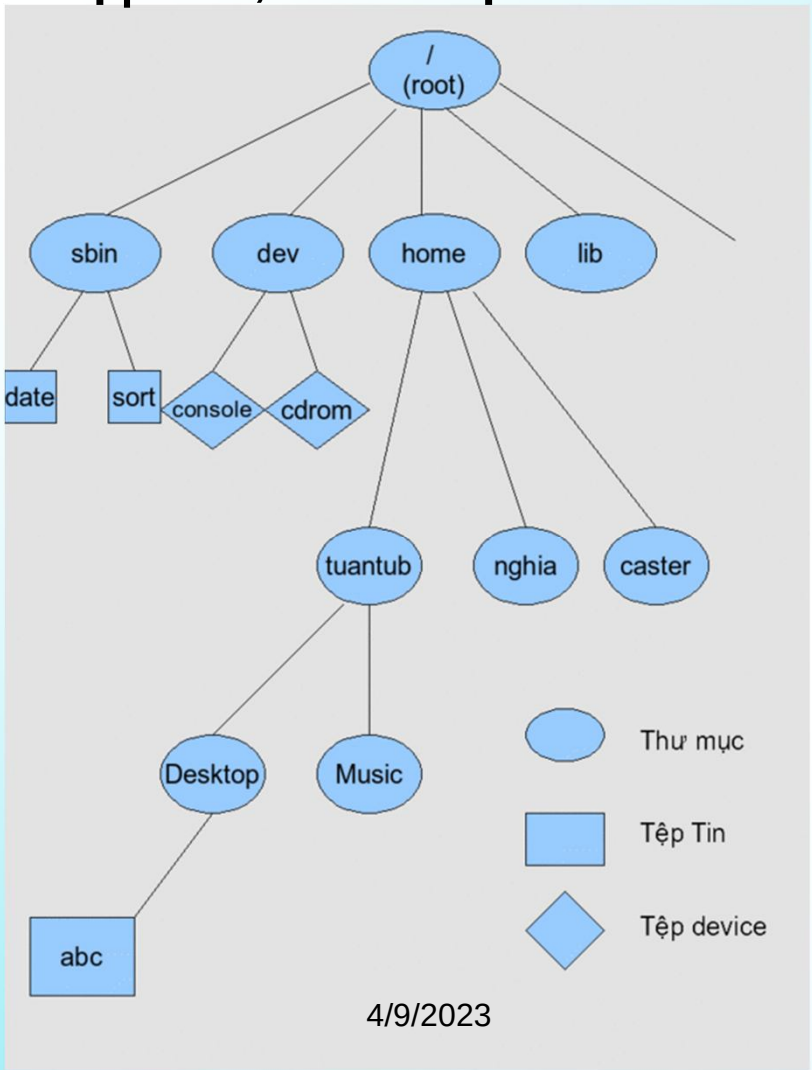
Hai đồng phân của Butane

Các mô hình dạng cây (tt)

- Biểu diễn các tổ chức:

Các mô hình dạng cây (tt)

- Hệ thống các tập tin, thư mục:

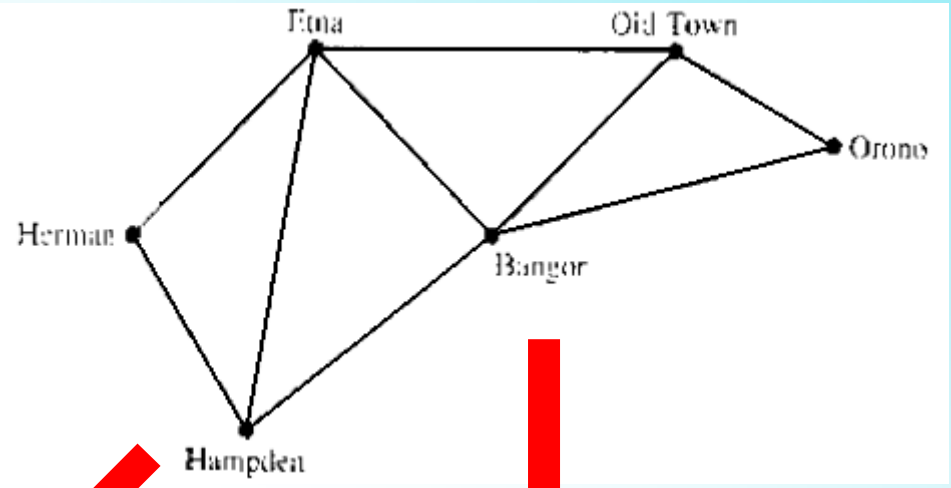


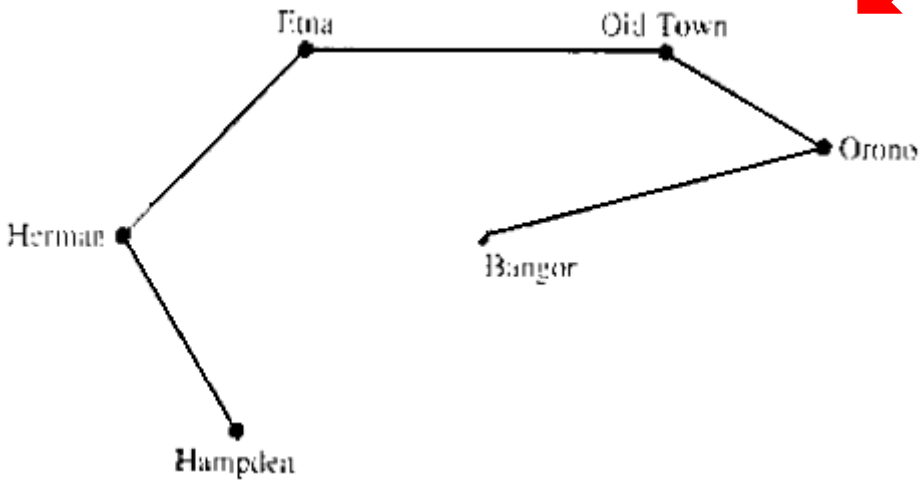
Các ứng dụng của cây

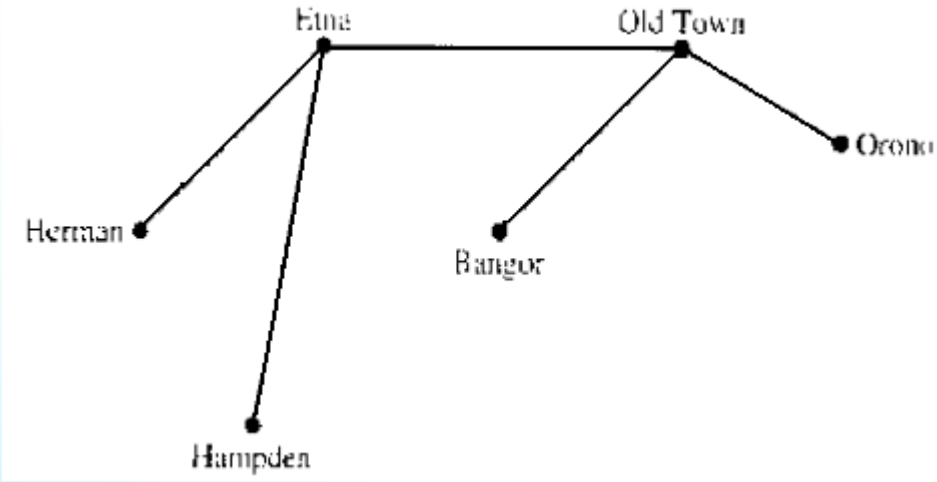
- Cây nhị phân tìm kiếm (đã học trong môn CTDL)
- Cây quyết định.
  - Là cây có gốc
  - Mỗi đỉnh ứng với một quyết định
  - Mỗi cây con tại đỉnh này sẽ ứng với các kết quả có thể của quyết định đó
- Mã tiền tố Huffman. (đề tài nghiên cứu)

Bài toán mở đầu

- Hệ thống đường giao thông ở Maine như hình bên.
- Tuyết đang phủ toàn bộ các con đường.
- Cần khôi phục lại hệ thống bằng cách cào tuyết một số con đường.
- Không nhất thiết phải cào tuyết hết mọi con đường.

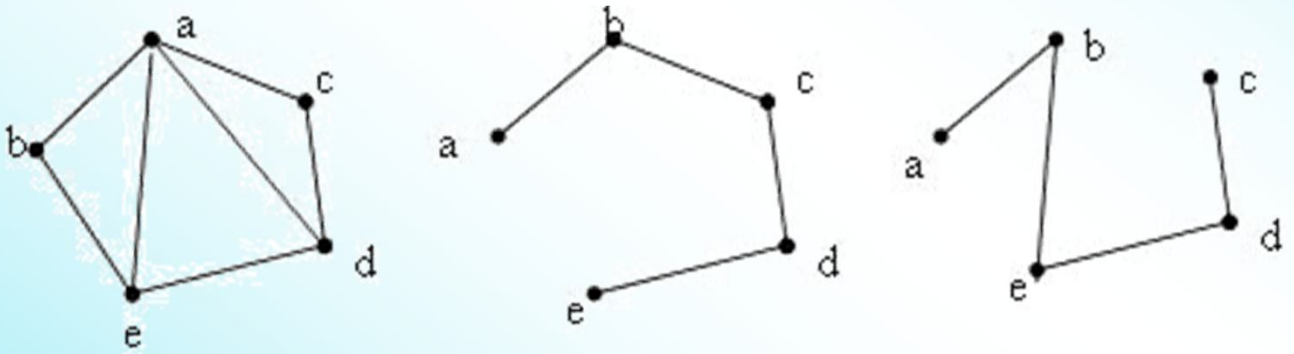






Cây khung

- **Định nghĩa:** Cho G là đơn đồ thị. Một cây T được gọi là cây khung của G nếu và chỉ nếu:
  - T là đồ thị con của G
  - T chứa tất cả các đỉnh của G
- **VD:**



Đồ thị và các cây khung của nó

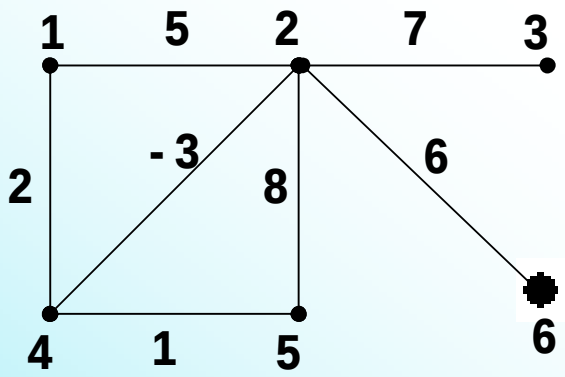
Cây khung (tt)

- **Định lý:** Một đơn đồ thị liên thông nếu và chỉ nếu nó có cây khung.
- **Chứng minh:**
  - Nếu G có chứa cây khung thì do tính chất của cây khung là liên thông và cây khung chứa tất cả các đỉnh của G. Suy ra các đỉnh của G luôn được nối với nhau hay G liên thông.
  - Xét G liên thông. Giả sử trong G còn tồn tại chu trình, xóa bớt một cạnh trong chu trình này, khi đó đồ thị vẫn còn liên thông. Nếu vẫn còn chu trình thì lặp lại bước trên. Cứ thế cho đến khi không còn chu trình nữa. Khi đó ta sẽ được cây khung

Đồ thị có trọng số

- Đồ thị có trọng số: là đồ thị mà mỗi cạnh của nó được gán với một con số thực chỉ chi phí phải tốn khi đi qua cạnh đó.
- Ký hiệu: c(u,v) là trọng số của cạnh (u,v)
- Trọng số có thể âm, có thể dương tùy theo ứng dụng.

VD:



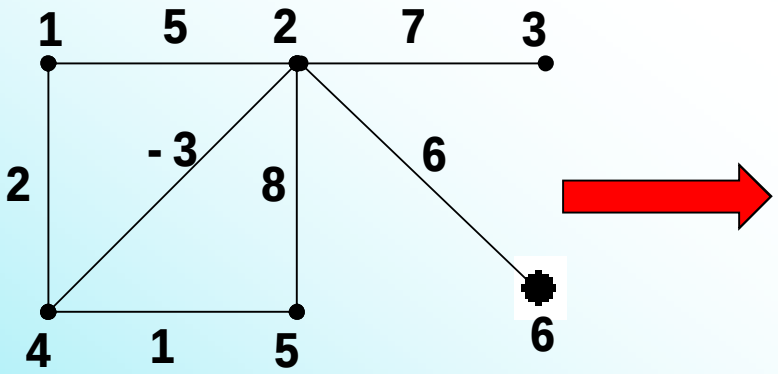
Đồ thị có trọng số (tt)

- Đồ thị có trọng số có thể được biểu diễn bằng ma trận kề trọng số.
- Cụ thể, Cho đồ thị G = <V, E>, với V = {v_1, v_2, …, v_n}. Ma trận kề trọng số biểu diễn G là một ma trận vuông A, kích thước nxn, được xác định như sau:

$$A_{ij} = \begin{cases} c(v_i, v_j), & (v_i, v_j) \in E \\ \infty, & (v_i, v_j) \notin E \end{cases}$$

Đồ thị có trọng số (tt)

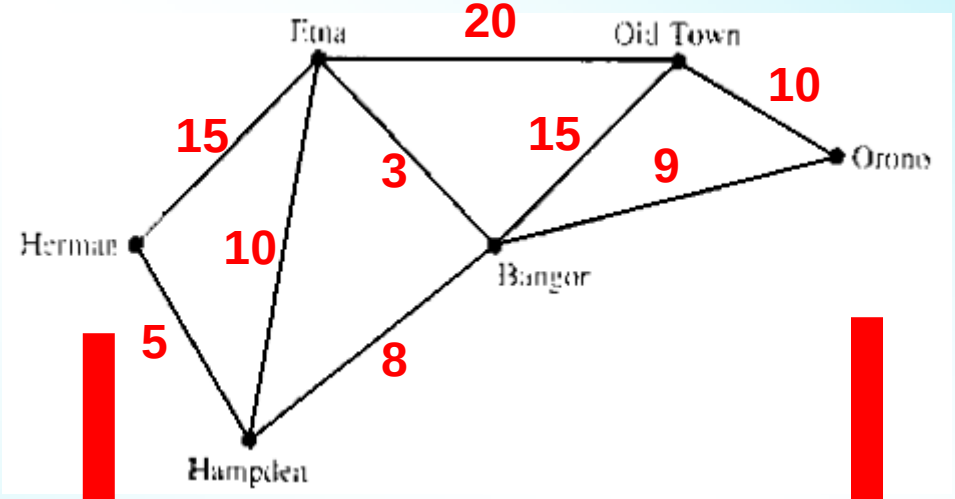
- **VD:**

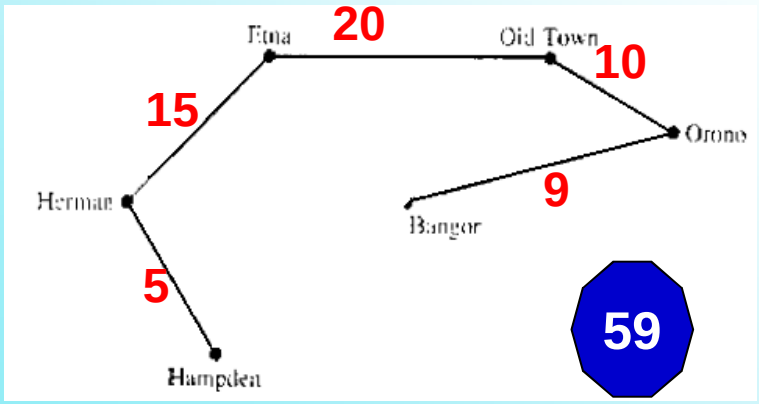


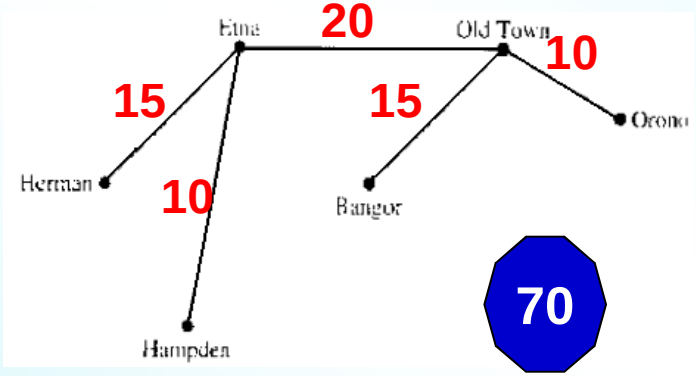
$$A = \begin{bmatrix} 0 & 5 & \infty & 2 & \infty & \infty \\ 5 & 0 & 7 & -3 & 8 & 6 \\ \infty & 7 & 0 & \infty & \infty & \infty \\ 2 & -3 & \infty & 0 & 1 & \infty \\ \infty & 8 & \infty & 1 & 0 & \infty \\ \infty & 6 & \infty & \infty & \infty & 0 \end{bmatrix}$$

Bài toán cây khung nhỏ nhất

- Tìm các con đường để cào tuyết sao cho chi phí là nhỏ nhất







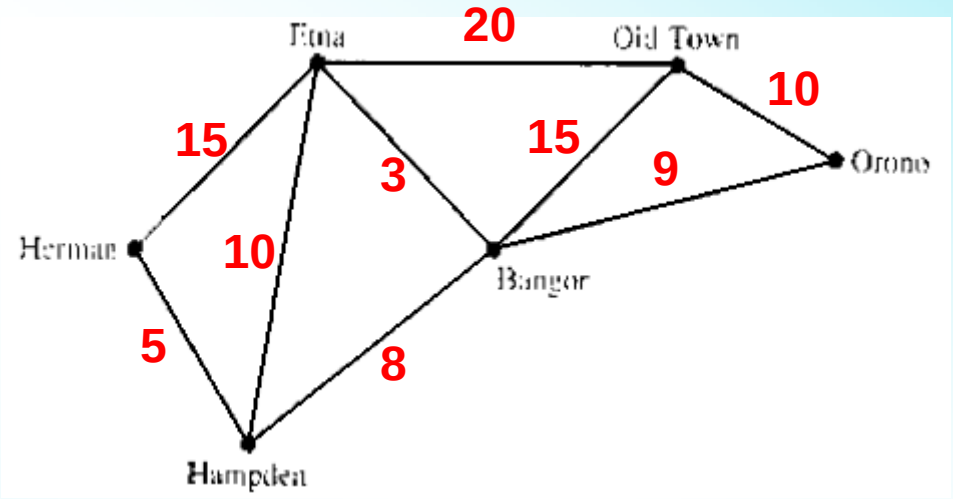
Bài toán cây khung nhỏ nhất (tt)

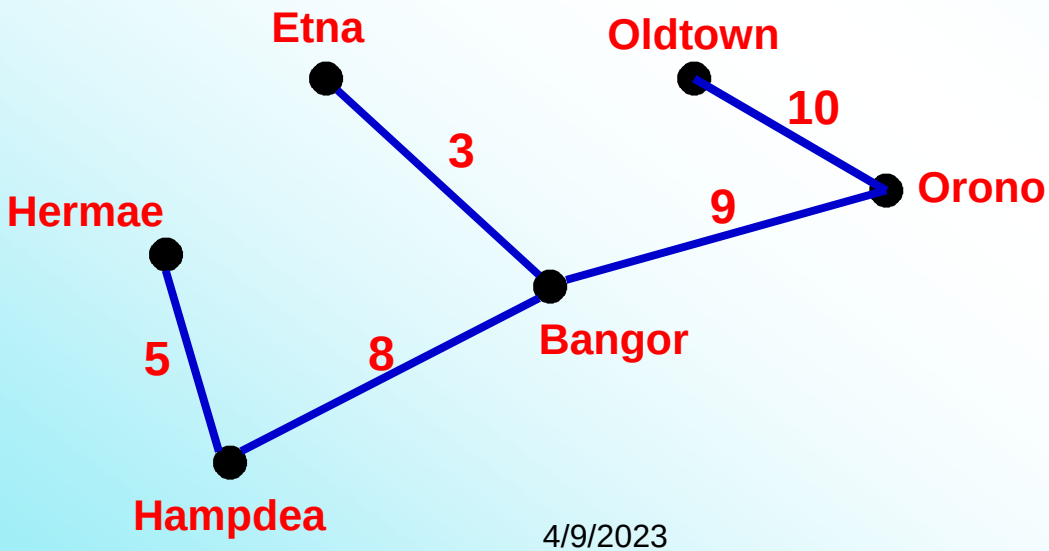
- **Định nghĩa.** Cho đồ thị có trọng số G. Cây khung nhỏ nhất của G (nếu tồn tại) là cây khung có tổng trọng số nhỏ nhất trong số các cây khung của G.
- **Các thuật toán tìm cây khung nhỏ nhất:**
  - Thuật toán Prim
  - Thuật toán Kruskal

Thuật toán Prim

- Ý tưởng:
  - Xuất phát từ 1 đỉnh bất kỳ. Đưa đỉnh này vào cây khung T.
  - Tại mỗi bước, luôn chọn cạnh có trọng số nhỏ nhất trong số các cạnh liên thuộc với một đỉnh trong T (đỉnh còn lại nằm ngoài T)
  - Đưa cạnh mới chọn và đỉnh đầu của nó vào cây T
  - Lặp lại quá trình trên cho đến khi đưa đủ n-1 cạnh vào T

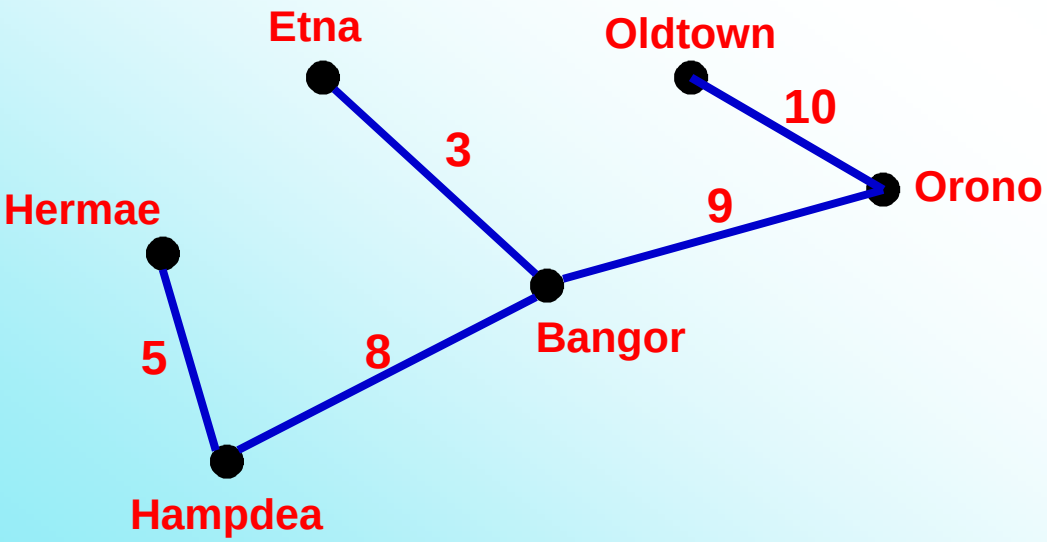
Thuật toán Prim (tt)





Thuật toán Prim (tt)

- Để biểu diễn lời giải, ta sẽ sử dụng 2 mảng:
  - Mảng d: d[v] dùng để lưu độ dài cạnh ngắn nhất nối với v trong số các cạnh chưa xét.
  - Mảng near: near[v] dùng để lưu đỉnh còn lại (ngoài v) của cạnh ngắn nhất nói ở trên.



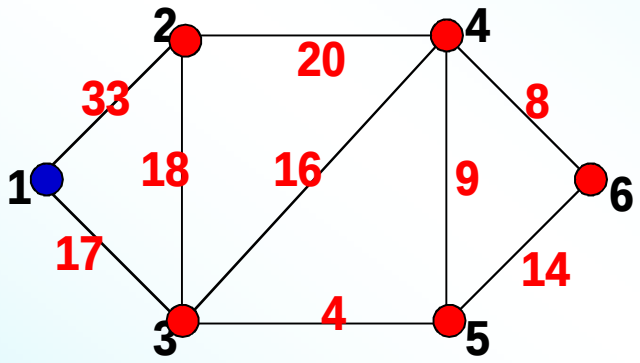
v	d[v]	near[v]
Etna	0	0
Bangor	3	Etna
Hampdea	8	Bangor
Hermae	5	**Hampdea**
Orono	9	Bangor
Oldtown	10	Orono

Thuật toán Prim (tt)

```
(* Khởi tạo *)
Chọn s là một đỉnh nào đó của đồ thị
V_H := {s}; (* Tập những đỉnh đã đưa vào cây *)
T := ∅;     (* Tập cạnh của cây *)
d[s] = 0;   near[s] = s;
For  v∈V\V_H  do
    Begin
        d[v] := a[s,v];
        near[v] := s;
    End;
```

```
(* Bước lặp *)
Stop := False;
While (not Stop) do
    Begin
        Tìm  u∈V\V_H thỏa mãn d[u] = min{d[v]: v∈V\V_H};
        V_H := V_H ∪ {u};
        T := T ∪ { (u, near[u]) };
        If  |V_H| = n then
            Begin
              H := (V_H, T) là cây khung của đồ thị.
              Stop := True;
            End;
        Else
            For v ∈V\V_H do
              If d[v] > a[u,v] then
                 Begin
                    d[v] := c[u,v];
                    near[v] := u;
                 End;
    End;
```

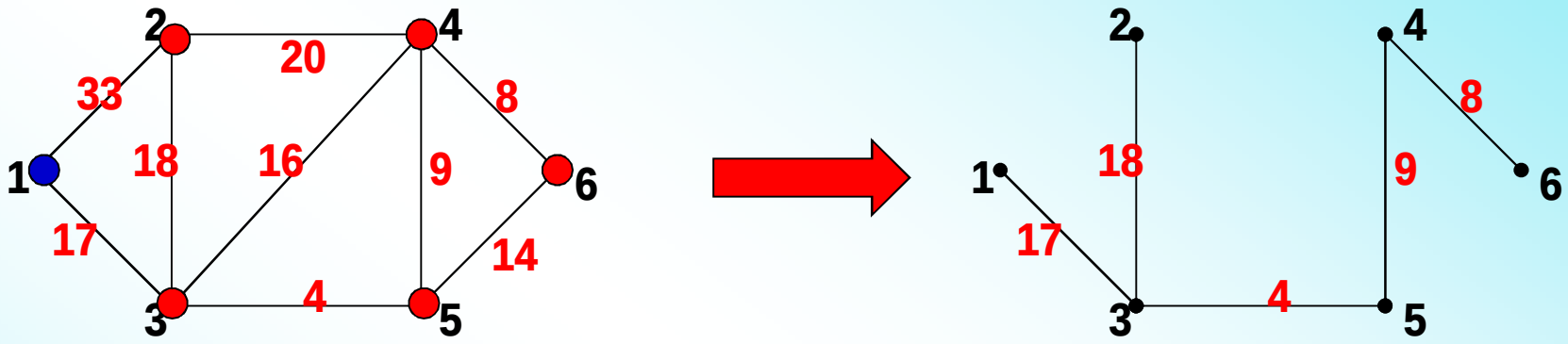
Thuật toán Prim (tt)



?

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	V_H	T
Khởi tạo	[0,1]	[33,1]	[17,1]*	[∞,1]	[∞,1]	[∞,1]	1	∅
1	-	[18,3]	-	[16,3]	[4,3]*	[∞,1]	1,3	(3,1)
2	-	[18,3]	-	[9.5]*	-	[14,5]	1,3,5	(3,1),(5,3)
3	-	[18,3]	-	-	-	[8,4]*	1,3,5,4	(3,1),(5,3),(4.5)
4	-	[18,3]*	-	-	-	-	1,2,3,4,6	(3,1),(5,3),(4.5),(6,4)
5	-	-	-	-	-	-	1,2,3,4,6,2	(3,1),(5,3),(4.5),(6,4),(2,3)

Thuật toán Prim (tt)

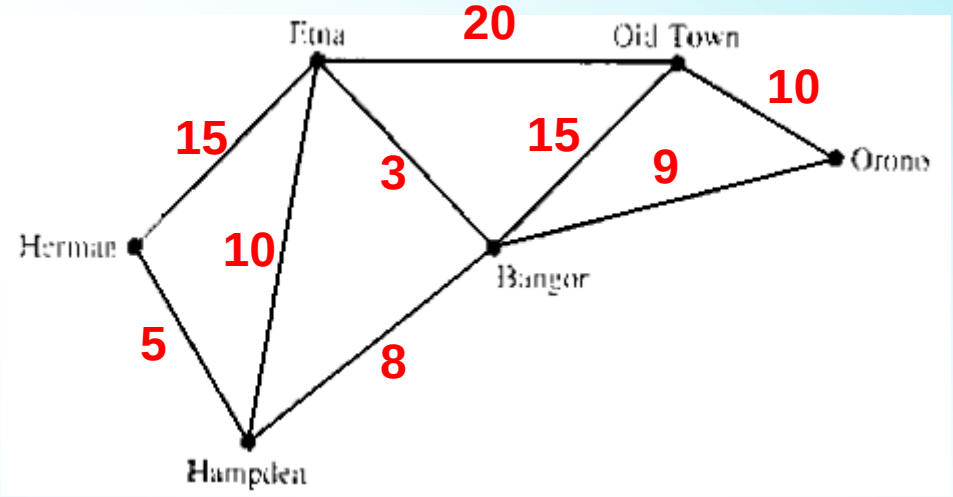


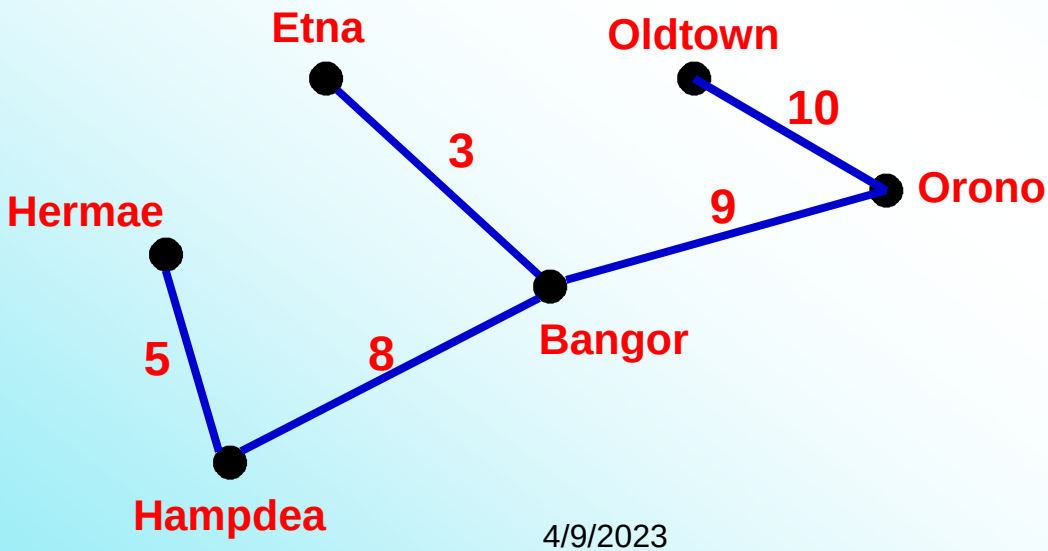
Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	V_H	T
Khởi tạo	[0,1]	[33,1]	[17,1]*	[∞ ,1]	[∞ ,1]	[∞ ,1]	1	∅
1	-	[18,3]	-	[16,3]	[4,3]*	[∞ ,1]	1,3	(3,1)
2	-	[18,3]	-	[9,5]*	-	[14,5]	1,3,5	(3,1), (5,3)
3	-	[18,3]	-	-	-	[8,4]*	1,3,5,4	(3,1), (5,3), (4,5)
4	-	[18,3]*	-	-	-	-	1,3,5,4,6	(3,1), (5,3), (4,5), (6,4)
5	-	-	-	-	-	-	1,3,5,4,6,2	(3,1), 5,3), (4,5), 6,4), (2,3)

Thuật toán Kruskal

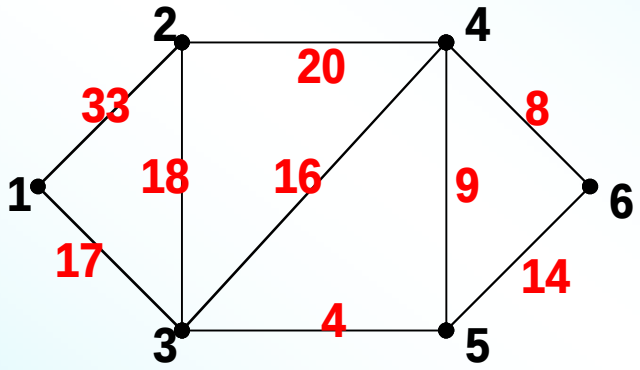
- **Ý tưởng:**
  - Lần lượt xét các cạnh theo thứ tự trọng số tăng dần
  - Ứng với mỗi cạnh đang xét, ta thử đưa nó vào cây khung T:
    - Nếu không tạo thành chu trình với các cạnh đã chọn thì chấp nhận cạnh mới này và đưa vào cây.
    - Nếu tạo thành chu trình với các cạnh đã chọn thì bỏ qua và xét cạnh kế tiếp.
  - Cứ tiếp tục như vậy cho đến khi tìm đủ n-1 cạnh để đưa vào cây T

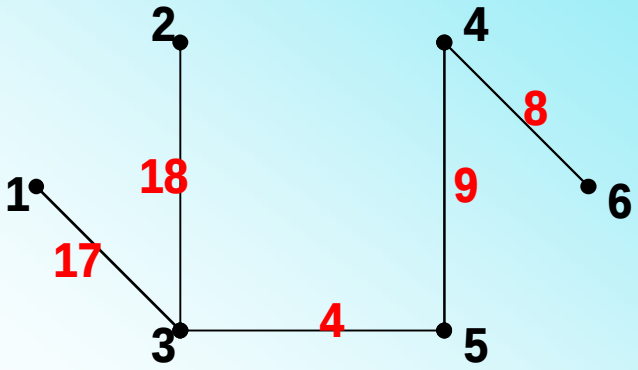
Thuật toán Kruskal (tt)





Thuật toán Kruskal (tt)





Trọng số	Cạnh	
4	(3,5)	← Chọn
8	(4,6)	← Chọn
9	(4,5)	← Chọn
14	(5,6)	← Không chọn vì tạo chu trình: 4 5 6 4
16	(3,4)	← Không chọn vì tạo chu trình: 3 4 5 3
17	(1,3)	← Chọn
18	(2,3)	← Chọn. Dừng vì đã đủ cạnh.
20	(2,4)	
33	(1,2)	