Chương 4

CÂY (TREE)

TRẦN QUỐC VIỆT

Tài liệu tham khảo

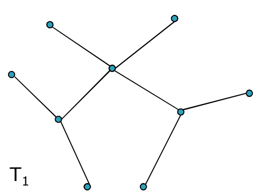
- Nguyễn Cam –Chu Đức Khánh, *Lý thuyết đồ thị* – NXB Trẻ Tp. HCM, 1998.
- Kenneth H. Rosen: *Discrete Mathematics and its Applications*, 7 Edition, McGraw Hill, 2010.

2

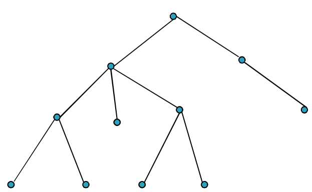
1. Định nghĩa và các tính chất cơ bản

❑ **Định nghĩa:** Cây (Tree), còn gọi là cây tự do (free tree) là đột đồ thị vô hướng liên thông và không có chu trình

Ví dụ: T_1 và T_2 sau đây là 2 cây



T_1



T_2

Định nghĩa và các tính chất cơ bản

❑ **Định lý 1**: Giữa 2 đỉnh bất kỳ trong cây T luôn có một và chỉ một đường đi trong T nối chúng

C/m: Xét 2 đỉnh x, y bất kỳ trong T ($x \neq y$), T liên thông nên có đường đi nối x và y

Giả sử có ít nhất 2 đường đi p_1, p_2 ($p_1 \neq p_2$) giữa x và y

p_1: $x=v_0, v_1, .., v_i, .., v_{k1}, \ldots, v_j, v_{j+1}, \ldots, v_{j+m}=y$

p_2: $x=v'_0, v'_1, \ldots, v'_i, \ldots, v'_{k2}, \ldots, v'_j, v'_{j+1} \ldots, v'_{j+m}=y$

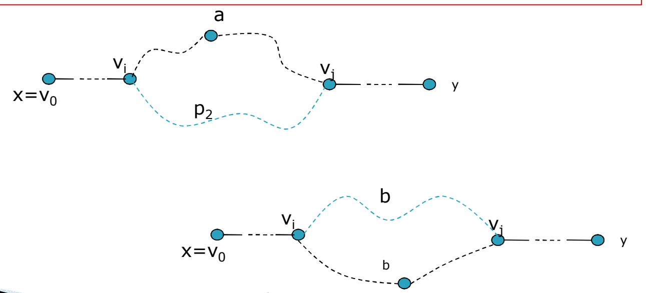
Với i: chỉ số lớn nhất thỏa: $v_k = v'_k$, $\forall k$, $0 \le k \le i$

Và j: là chỉ số nhỏ nhất thỏa: $v_r = v'r$, $\forall r$, $j \le r \le j+m$

4

Định nghĩa và các tính chất cơ bản

- Do $p_1 \neq p_2$ nên phải có ít nhất một đỉnh *a* trên p_1 và không nằm trong p_2 hoặc phải có ít nhất một đỉnh b trong p_2 và không nằm trong p_1.



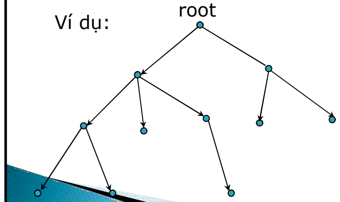
Trong T có chu trình

5

Định nghĩa và các tính chất cơ bản (tt)

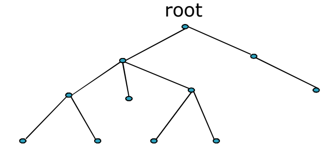
- **Định nghĩa: Cây có gốc (rooted tree)** là một cây có hướng, trên đó đã chọn một đỉnh là gốc (root) của cây và các cạnh định hướng sao cho với mọi đỉnh luôn có một đường đi từ gốc đến đỉnh đó

Ví dụ:



Cây có gốc

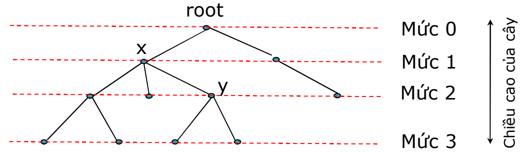
Một cây tự do có thể chọn một đỉnh bất kỳ làm gốc để trở thành cây có gốc



Định nghĩa và các tính chất cơ bản (tt)

Xét xây có gốc T

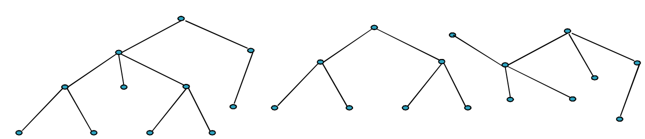
- Mức của đỉnh: Khoảng cách từ gốc đến nút đó
- Chiều cao của cây: Mức lớn nhất của một đỉnh bất kỳ



- Nếu (xy) là cạnh của T: ta gọi x đỉnh cha (parent) của y, y là đỉnh con (child) của x
- Lá (leaves): Những đỉnh không có cây con.
- Đỉnh trong (internal vertices): là những đỉnh có ít nhất 1 cây con

Định nghĩa và các tính chất cơ bản (tt)

Định nghĩa: Tập hợp các cây đôi một không có đỉnh chung gọi là một rừng (Forest)



một rừng (forest): gồm nhiều cây không có đỉnh chung

Mọi đỉnh x trong một cây mà là gốc của một cây con, Khi xóa đỉnh x khỏi cây ta được một rừng

Định nghĩa và các tính chất cơ bản (tt)

Định lý 2: Nếu một cây có n đỉnh thì sẽ có m=n-1 cạnh

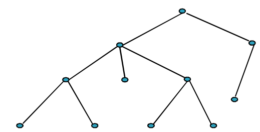
C/m: Ta chọn một nút làm gốc để được cây có gốc

- Với cây chỉ có 1 đỉnh (n=1), số cạnh là 0, nghĩa là: m=n-1
- Giả sử cây có k đỉnh thì có k-1 cạnh là đúng,
- Xét cây có k+1 đỉnh và xét một đỉnh lá v bất kỳ, nếu loại bỏ v cùng với cạnh nối đến v (chỉ có duy nhất 1 cạnh nối đến v), đồ thị T' còn lại cũng là một cây có k đỉnh ⇒ T' có k-1 cạnh ⇒ T có k cạnh
- Theo nguyên lý quy nạp, "một cây có n đỉnh thì sẽ có m=n-1 cạnh" đúng với mọi n (n>=1)

9

Định nghĩa và các tính chất cơ bản (tt)

Ví dụ:



Cây có 11 đỉnh ⇒ có 11-1=10 cạnh

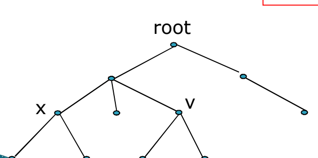
10

Định nghĩa và các tính chất cơ bản (tt)

Định nghĩa (độ lệch tâm):Xét xây có gốc T

- Độ lệch tâm (eccentricity) của đỉnh v: là Khoảng cách lớn nhất từ v đến đỉnh bất kỳ trong T. Kí hiệu E(v)

$$\boxed{E(v) = \max_{u \in T} \delta(u,v)}$$



E(x)=?

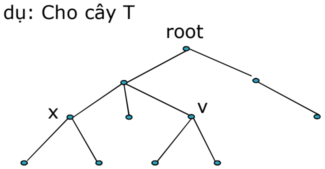
E(v)=?

Định nghĩa và các tính chất cơ bản (tt)

Xét xây có gốc T

- Đỉnh có độ lệch tâm nhỏ nhất gọi là tâm (center) của T
- Độ lệch tâm của tâm gọi là bán kính (radius) của T

Ví dụ: Cho cây T

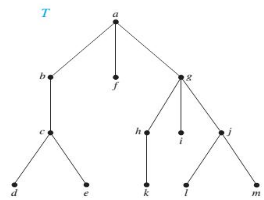


Xác định tâm của T?

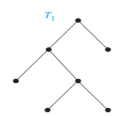
Xác định bán kính của T?

Định nghĩa và các tính chất cơ bản (tt)

- Cho cây có gốc T:
  - Nếu số con tối đa của một đỉnh trong T là m và có ít nhất một đỉnh đúng m con thì T gọi là cây m-phân (**m-ary tree**)
  - Nếu mọi đỉnh trong của T đều có đúng m cây con thì T gọi là cây m-phân đủ (complete m-ary tree)
  - Cây m-phân với m=2 gọi là cây nhị phân



Cây tam phân



Cây nhị phân đủ

Định nghĩa và các tính chất cơ bản (tt)

- <u>Định lý 3 (Định lý Daisy Chain):</u> T là một đồ thị có n đỉnh, các mệnh đề sau là tương đương

(i) T là cây

(ii) T không có chu trình và có n-1 cạnh

(iii) T Liên thông và nếu hủy bất kỳ một cạnh nào trong T thì T sẽ mất tính liên thông

(iv) Giữ 2 đỉnh bất kỳ trong T luôn tồn tại duy nhất một đường đi nối chúng

(v) T không có chu trình, nếu thêm 1 cạnh bất kỳ nối 2 đỉnh trong T thì T sẽ có chu trình

(vi) T liên thông và có n-1 cạnh

<u>C/m định lý 3</u>

Bài tập!

Định nghĩa và các tính chất cơ bản (tt)

- ***<u>Định lý 4</u>***: Một cây tự do có nhiều nhất 2 tâm
- ***<u>Định lý 5</u>***: Một cây m-phân đầy đủ có i đỉnh trong thì có mi+1 đỉnh
- ***<u>Hệ luận: T là một cây m-phân đầy đủ</u>***

(i) T có i đỉnh trong ⇒ T có l = (m-1)i+1 lá

(ii) T có l lá ⇒ T có I = (l-1)/(m-1) đỉnh trong và n = (ml-1)/(m-1) đỉnh

(i) T có n đỉnh ⇒ T có i = (n-1)/m đỉnh trong và l = [(m-1)n+1)]/m nút lá

Định nghĩa và các tính chất cơ bản (tt)

- ***Định lý 5***:

(i) Một cây m-phân có chiều cao h thì có nhiều nhất là m^h lá

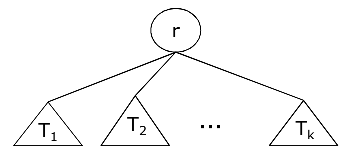
(ii) Một cây m-phân có l lá thì có chiều cao h $\geq[\log_m l]$

(iii) Một cây m-phân đầy đủ, cân bằng có l lá thì có chiều cao h=$[\log_m l]$

17

2. Các phương pháp duyệt cây

Xét cây có gốc T, gọi T_{r1}, T_{r2},...,T_{rk} lần lượt là các cây con của nút r theo thứ tự từ trái qua phải



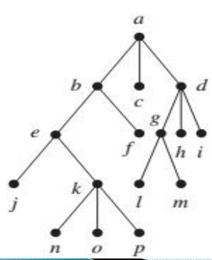
18

2. Các phương pháp duyệt cây

2.1. Duyệt cây theo thứ tự trước (preoder)

- Thăm gốc r của T
- Đệ quy: Duyệt từng cây con lần lượt từ T_1 đến T_{rk} theo thứ tự trước

Ví dụ:



Kết quả duyệt theo Preorder?

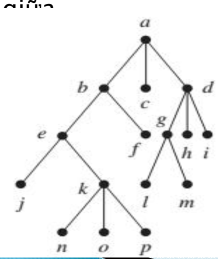
19

2. Các phương pháp duyệt cây

2.2. Duyệt cây theo thứ tự giữa (inoder)

- Duyệt T_{r1} theo thứ tự giữa
- Thăm r
- Duyệt T_{r2}theo thứ tự giữa,..., duyệt T_{rk} theo thứ tự giữa

Ví dụ:



Kết quả duyệt theo Inorder?

20

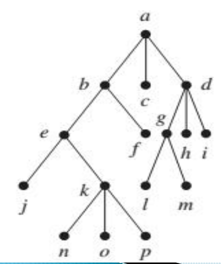
2. Các phương pháp duyệt cây

2.3. Duyệt cây theo thứ tự sau (postorder)

– Duyệt T_{r1} theo postorder, duyệt T_{r2} theo postorder,…, duyệt T_{rk} theo postorder

– Thăm r

Ví dụ:

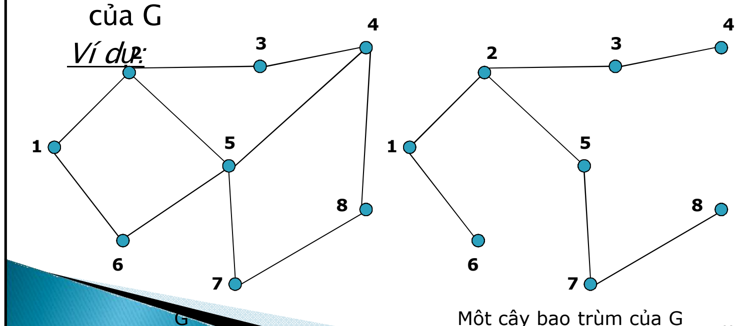


Kết quả duyệt theo postorder?

Cây bao trùm(spanning tree)

- Cho đồ thị vô hướng G. Cây T gọi là một cây bao trùm của G nếu T≤G và T chứa mọi đỉnh của G

Ví dụ:

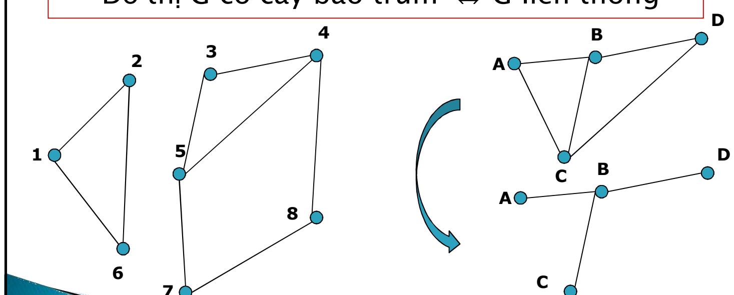


Một cây bao trùm của G

Cây bao trùm(spanning tree)

- Định lý:

  Đồ thị G có cây bao trùm ⇔ G liên thông



G không liên thông ⇒ G không có cây bao trùm

H liên thông ⇒ H có cây bao trùm

Tìm cây bao trùm

Hai phương pháp:

- ❑Tìm theo chiều sâu (DFS: Depth First Search)
- ❑Tìm theo chiều rộng (BFS: Breadth First Search)

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

Bước 1: Chọn một đỉnh bất kỳ v của G làm điểm xuất phát (gốc) của T.

Bước 2:Xác định một đường đi sơ cấp xuất phát từ v qua các đỉnh ∈ G nhưng ∉ T, cho đến khi không thể đi tiếp. Gọi k là đỉnh kết thúc đường đi. Đặt đường đi này vào T rồi quay trở về đặt đỉnh liền trước k làm điểm xuất phát. Lập lại thủ tục này cho đến khi mọi đỉnh của G đều nằm trong T.

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

DFS(G: Liên thông với các đỉnh $v_1,v_2,\ldots,v_n$)

{ T=Cây chỉ gồm 1 đỉnh v_1

visit(v_1)

}

Visit(v: đỉnh của G)

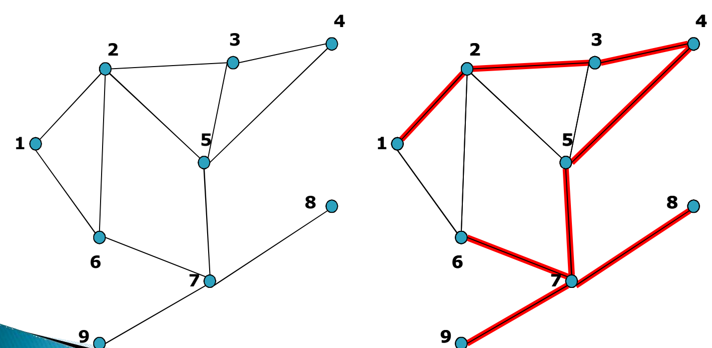
{ **For** (mỗi đỉnh w kề với v nhưng chưa có trong T)

– Thêm đỉnh w và cạnh (v,w) vào T

visit(w);

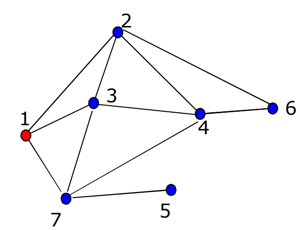
}

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)



Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

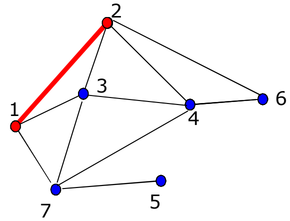
	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0



v=1; $T=<V_T,E_T>$ với $V_T=\{1\}$, $E_T=\varnothing$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

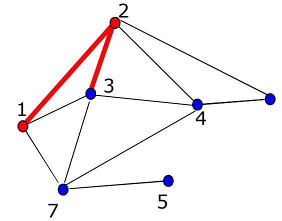
	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0



V=1,w=2, $a_{12}\neq 0 \wedge 2\notin V_T, V_T=\{1,2\}, E_T=\{(1,2)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

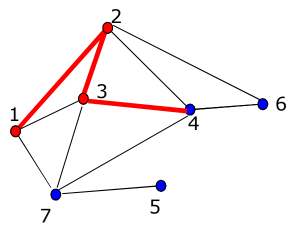
	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0



V=2,w=3, $a_{23}\neq 0 \wedge 3\notin V_T, V_T=\{1,2,3\}, E_T=\{(1,2),\{2,3)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

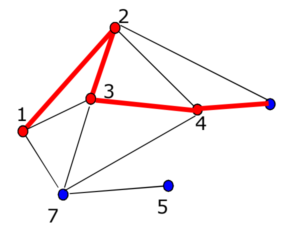
	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0



V=3,w=4, $a_{34}\neq 0 \wedge 4\notin V_T, V_T=\{1,2,3,4\}, E_T=\{(1,2),(2,3),(3,4)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0

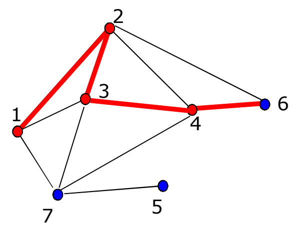


V=4

w=6, $a_{46}\neq 0 \wedge 6\notin V_T, V_T=\{1,2,3,4,6\}, E_T=\{(1,2),(2,3),(3,4),(4,6)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0

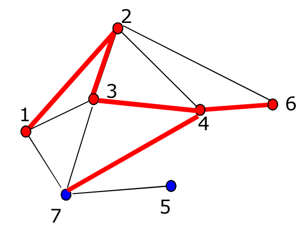


V=4

W=7, $a_{47} \neq 0 \wedge 7 \notin V_T, V_T=\{1,2,3,4,6,7\}, E_T=\{(1,2),(2,3),(3,4),(4,6),(4,7)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0

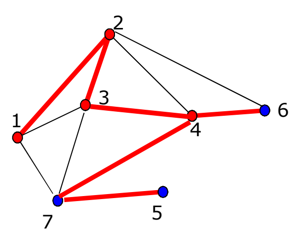


V=4

W=7, $a_{47} \neq 0 \wedge 7 \notin V_T, V_T=\{1,2,3,4,6,7\}, E_T=\{(1,2),(2,3),(3,4),(4,6),(4,7)\}$

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0



V=7

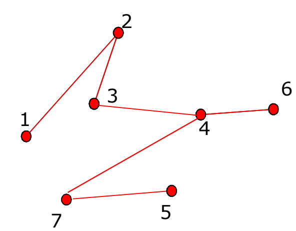
W=5, $a_{75} \neq 0 \wedge 5 \notin V_T, V_T=\{1,2,3,4,6,7,5\}, E_T=\{(1,2),(2,3),(3,4),(4,6),(4,7),(7,5)\}$

Mọi cạnh của G đều có trong T, dừng

Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	1
2	1	0	1	1	0	1	0
3	1	1	0	1	0	0	1
4	0	1	1	0	0	1	1
5	0	0	0	0	0	0	1
6	0	1	0	1	0	0	0
7	1	0	1	1	1	0	0

Cây bao trùm tìm được



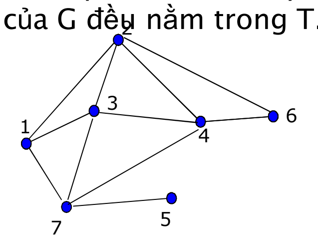
Tìm cây bao trùm theo phương pháp – DFS (Depth First Search)

Thuật toán tìm cây bao trùm theo DFS (không đệ quy)

Bài tập

Tìm cây bao trùm theo phương pháp – BFS (Breadth First Search)

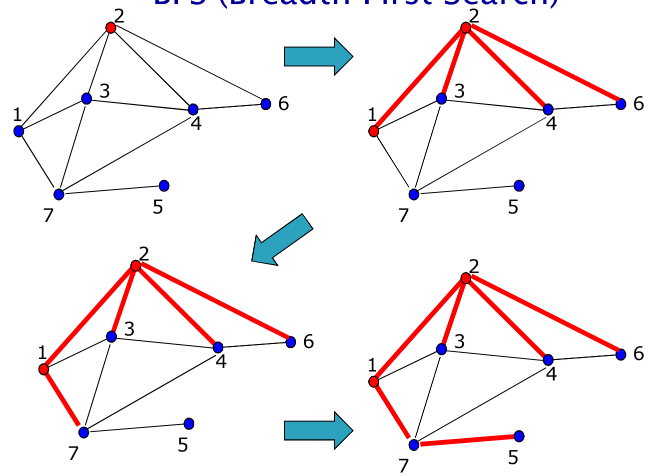
1) Chọn 1 đỉnh bất kỳ của G làm đỉnh xuất phát (gốc) của T.
2) Đặt mọi cạnh nối gốc với 1 đỉnh ∉ T vào T. Lần lượt xét từng đỉnh con trực tiếp của gốc. Xem đỉnh này là gốc mới, lặp lại thủ tục này cho đến khi mọi đỉnh của G đều nằm trong T.



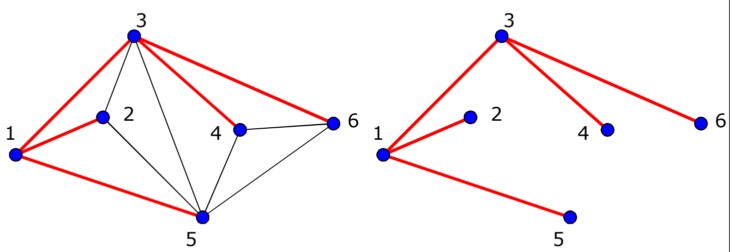
Tìm cây bao trùm theo phương pháp – BFS (Breadth First Search)

BFS(G: liên thông với tập đỉnh $v_1, v_2, \ldots, v_n$)
{ T:= Cây chỉ gồm 1 đỉnh v_1;
Q={v}// Queue: các đỉnh chưa xử lý
while (Q≠∅)
{ v = Q.Remove();
for (mỗi đỉnh w kề với v)
if w ∉ Q and w∉T
{ Q.Add(w)
Thêm cạnh (v,w) vào T
}
}
}

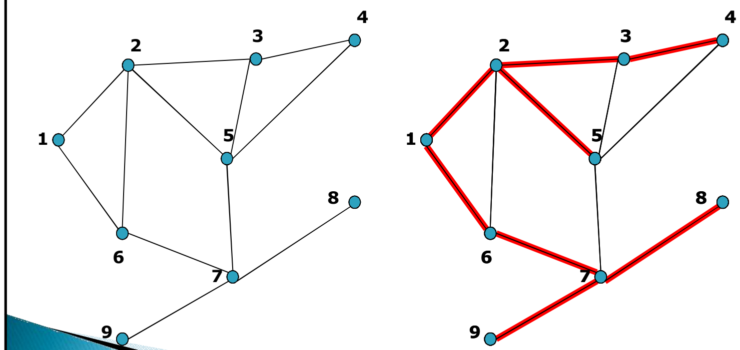
Tìm cây bao trùm theo phương pháp – BFS (Breadth First Search)



Tìm cây bao trùm theo phương pháp – BFS (Breadth First Search)

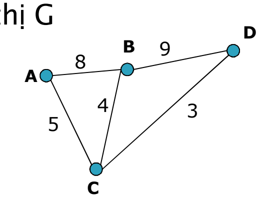


Tìm cây bao trùm theo phương pháp – BFS (Breadth First Search)



Cây bao trùm nhỏ nhất

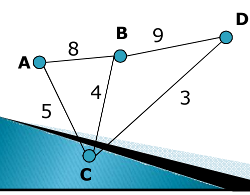
- **<u>Đồ thị có trọng số</u>**: Là đồ thị trong đó mỗi cạnh (cung) được gán thêm một số thực gọi là trọng số (weight) của cạnh (cung)
  <u>Kí hiệu</u>:
  c(e): Trọng số của cạnh e
  c(G): Trọng số của đồ thị G



Cây bao trùm nhỏ nhất

- **<u>Ma trận kề của đồ thị có trọng số</u>**: Cho G=<V,E> có trọng số, ma trận kề trọng số của G là ma trận A có kích cỡ |V|×|V| trong đó mỗi phần tử a_{ij} có giá trị như sau:

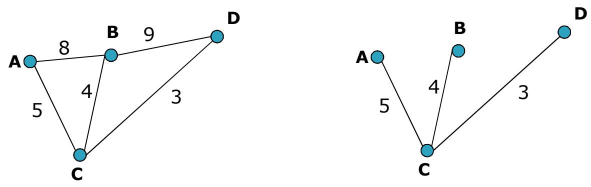
$$a_{ij} = \begin{cases} \text{Trọng số của cạnh/cung } (v_i,v_j) \text{ nếu } (v_i,v_j)\in E \\ \infty \quad \text{Nếu } (v_i,v_j)\notin E \end{cases}$$



$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{pmatrix} \infty & 8 & 5 & \infty \\ 8 & \infty & 4 & 9 \\ 5 & 4 & \infty & 3 \\ \infty & 9 & 3 & \infty \end{pmatrix} \end{array}$$

Cây bao trùm nhỏ nhất

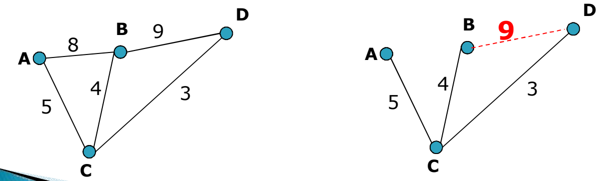
- **Bài toán**: Cho G là đồ thị liên thông, có trọng số. Hãy tìm một cây bao trùm của G có trọng số nhỏ nhất



45

Cây bao trùm nhỏ nhất

- **Định lý**: Cho T là một cây bao trùm của đồ thị có trọng số G liên thông. T là một cây bao trùm tối thiểu nếu và chỉ nếu mỗi cạnh $e \notin T$ đều có trọng số lớn nhất trên chu trình tạo bởi e và các cạnh của T



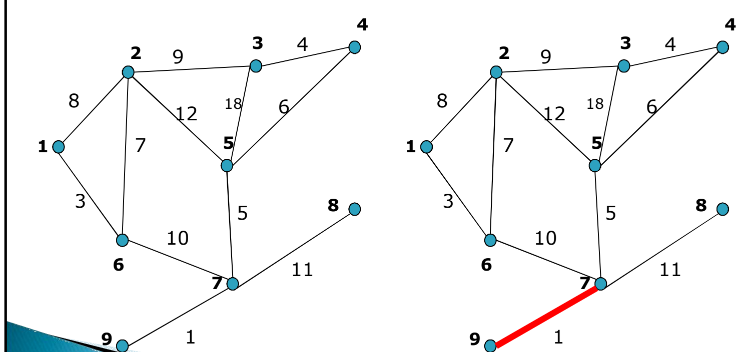
46

Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL

Kruskal(G: đồ thi liên thông có trọng số)
begin T := ∅;
E=E_G;// E_G là tập cạnh của G
While $|E_T|<n-1$ and (E≠∅)
begin Chọn e là cạnh độ dài nhỏ nhất trong E
E := E\{e}
If (T∪{e} không chứa chu trình) then
T := T∪{e} // Kết nạp cạnh e vào cây khung nhỏ nhất
end
If ($|E_T|<(n-1)$) then *Đồ thị không liên thông.*
else return T;
end

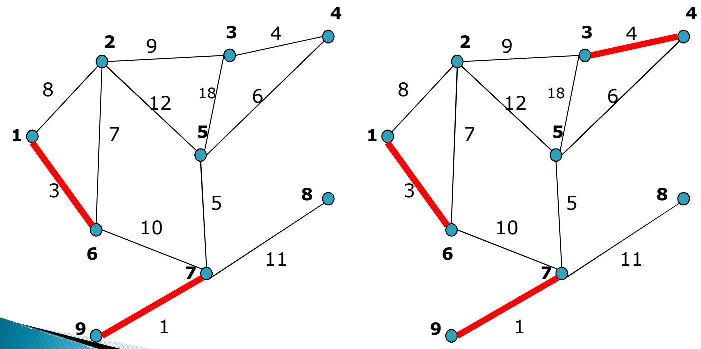
47

Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL

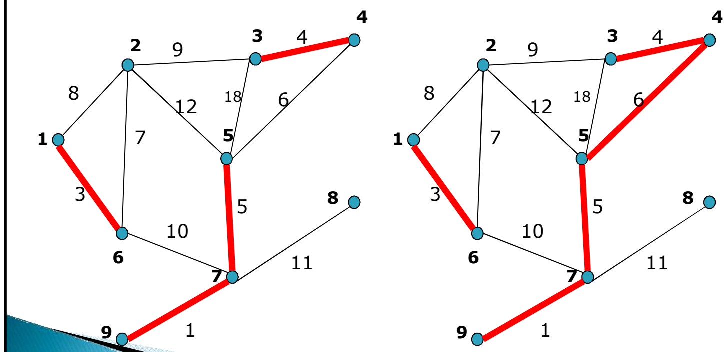


48

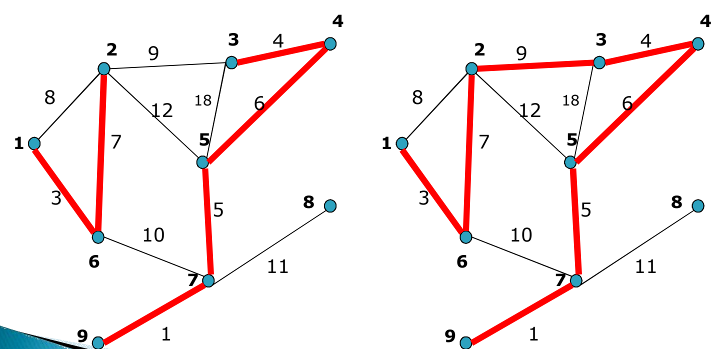
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL



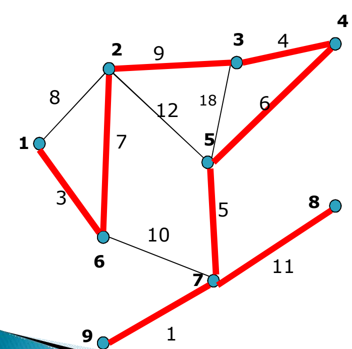
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL



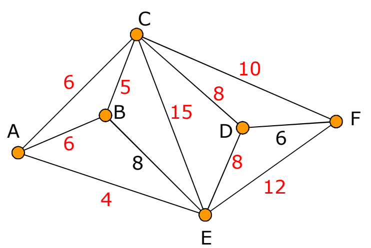
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL



Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL



Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán KRUSKAL



G

Sử dụng thuật toán Kruskal tìm một cây bao trùm nhỏ nhất của G?

Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM

<u>Prim(G: liên thông, có trọng số, v_0: đỉnh bắt đầu)</u>

begin $E_T := \varnothing$; $V_T=\{v_0\}$;// V_T là tập đỉnh của T, E_T: tập cạnh của T

while ($|E_T|<n-1$)

begin

Tính $\Gamma(V_T) = \{(i,j)\in E/\ \ i\in V_T \wedge j \in V - V_T\}$

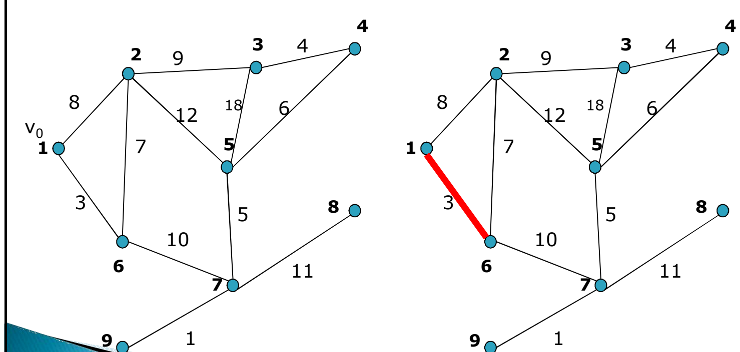
Chọn cạnh e=(u,v) trong $\Gamma(V_T)$ có trọng số bé nhất, bổ sung e vào T (Nghĩa là: $E_T=E_T \cup \{e\}$; $V_T = V_T \cup \{v\}$)

end

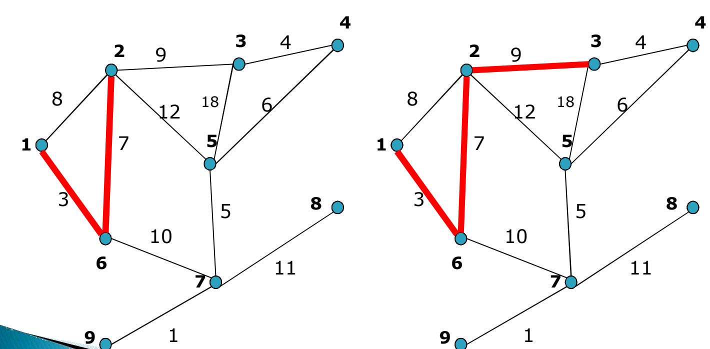
return T=<V_T, E_T>;

end

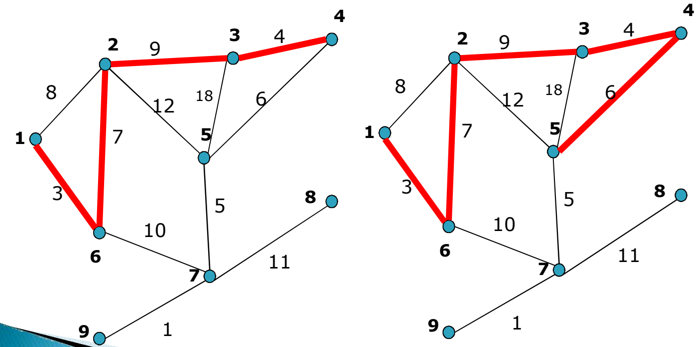
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM



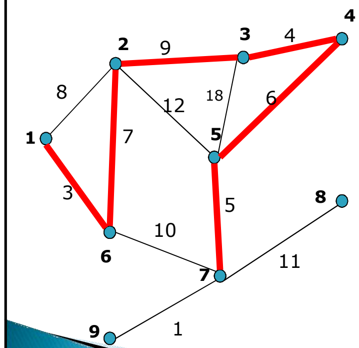
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM



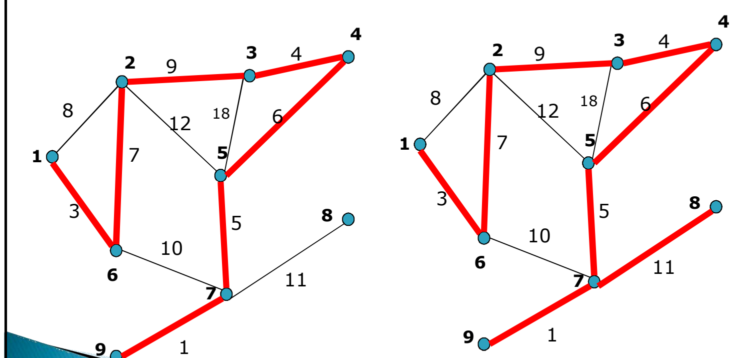
Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM



Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM



Thuật toán tìm cây bao trùm nhỏ nhất: Thuật toán PRIM



Cây mã HUFFMAN

- Mã tiền tố (prefix code)
- Thuật thoát Huffman

Mã tiền tố (prefix code)

- Cho X là một tập hữu hạn các ký hiệu:
  Ví dụ: X={a,b,c,d,e,f}
- M là một bản tin gồm ký hiệu lấy từ X theo xác suất biết trước.
  Ví dụ: M gồm 10^5 kí hiệu với tần suất xuất hiện như bảng sau:

Ký hiệu	a	b	c	d	e	f
Tần suất (%) xuất hiện	45	10	12	3	20	10

- **Cần mã hóa các kí hiệu trong X bằng chuỗi bit sao cho chiều dài bản tin là ngắn nhất?**

Mã tiền tố (tt)

- Cách 1, dùng tối thiểu 3 bit/1 ký hiệu ($2^3 \geq 6$)

Ký hiệu	a	b	c	d	e	f
Mã	000	001	010	011	100	101

Tổng chiều dài của M là:

3×10^5 = 300 000 bit

Mã tiền tố (tt)

- Cách 2:

Ký hiệu	a	b	c	d	e	f
Mã	1	0010	011	010	000	1011

Tổng chiều dài của M là:

$(1\times45\%+4\times10\%+3\times12\%+3\times3\%+3\times20\%+4\times10\%)\times10^5$ =230 000 bit $\leq$ 300 000 bit

Mã tiền tố (tt)

- ***Nhận xét***:Với cách mã hóa không cho phép bất kỳ mã của một ký hiệu nào là tiền tố (prefix) của mã một ký hiệu khác thì có thể giải mã được. Cách mã này gọi là mã tiền tố (prefix code)
- Ví dụ: Xét cách mã hóa sau:

Ký hiệu	a	b	c	d	e	f
Mã	0	01	00	11	1	10

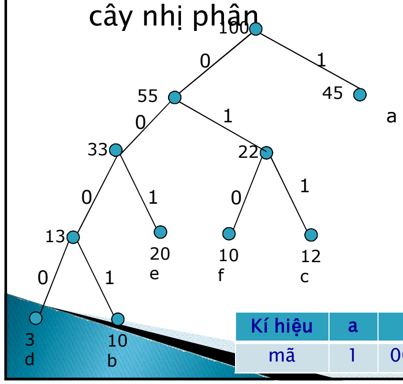
Trong đó: chuỗi mã hóa của a là tiền tố của của chuỗi mã hóa của b và c

- Xét bản tin M gồm 3 kí hiệu M=aac. Bản mã: C=0000

⇨ Không thể giải mã

Mã tiền tố

- Một mã tiền tố có thể biểu diễn bằng một cây nhị phân



Nút lá: một ký hiệu trong X
Nhánh/cạnh:
0: đến cây con trái
1: đến cây con phải
Chuỗi bit đường đi từ gốc đến lá là mã của ký hiệu tương ứng

Kí hiệu	a	b	c	d	e	f
mã	1	0001	011	0000	001	010

Giải thuật Huffman

Procedure Huffman(X: các kí hiệu a_i với tần suất n_i, i=1,2,…,n)
Begin F:=rừng gồm n cây có gốc, mỗi T_i cây chỉ chứa 1 đỉnh a_i được gán trọng số $w(T_i)$
While <*F không là cây*>
Begin
- Tìm 2 cây trong F (T_i và T_j) sao cho gốc của chúng (y và z) có trọng số nhỏ nhất.
- Nối y và z với đỉnh mới u để thành cây mới T có gốc u (nghĩa là T_i là cây con trái, T_j cây con phải của cây mới)
- Gán nhãn cạnh mới để T_i là 0 và cạnh mới đến T_j là 1
- w(u)=w(y)+w(z)

end
End

66

Giải thuật Huffman

- Định lý: Khi giải thuật kết thúc, cây mã nhận được là tối ưu

Ví dụ: M gồm 10^5 kí hiệu với tần suất xuất hiện như bảng sau:

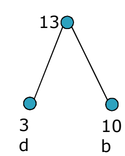
Ký hiệu	a	b	c	d	e	f
Tần suất (%) xuất hiện	45	10	12	3	20	10

67

Giải thuật Huffman

Ví dụ: M gồm 10^5 kí hiệu với tần suất xuất hiện như bảng sau:

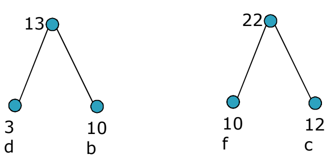
Ký hiệu	a	b	c	d	e	f
Tần suất (%) xuất hiện	4 5	10	12	3	20	10



68

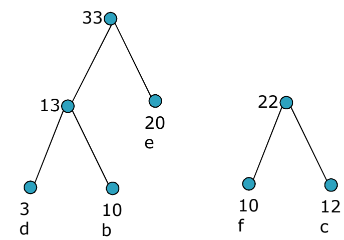
Giải thuật Huffman

Ký hiệu	a	b	c	d	e	f
Tần suất (%) xuất hiện	4 5	10	12	3	20	10



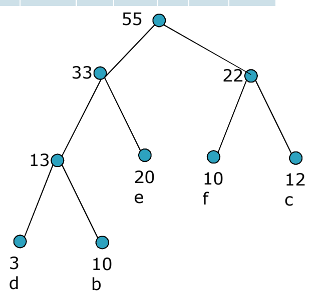
Giải thuật Huffman

Ký hiệu	a	b	c	d	e	f
Tần suất (%) xuất hiện	4 5	10	12	3	20	10

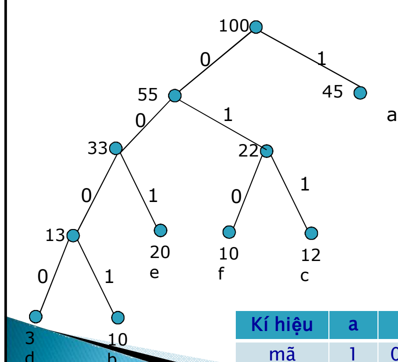


Giải thuật Huffman

Ký hiệu	a	B	c	d	e	f
Tần suất (%) xuất hiện	45	1 0	12	3	20	10



Giải thuật Huffman



Kí hiệu	a	b	c	d	e	f
mã	1	0001	011	0000	001	010