

Phần 2

LÝ THUYẾT ĐỒ THỊ *Graph Theory*

GV: Nguyễn Huy Đức

Bộ môn: Khoa học Máy tính – ĐH Thủy lợi



0903 402 655



ducnghuy@gmail.com

Nội dung

- Chương 1: Các khái niệm cơ bản
- Chương 2: Biểu diễn đồ thị
- Chương 3: Các thuật toán tìm kiếm trên đồ thị
- Chương 4: Đồ thị Euler và đồ thị Hamilton
- **Chương 5: Cây và cây khung của đồ thị**
- Chương 6: Bài toán đường đi ngắn nhất

Chương 5

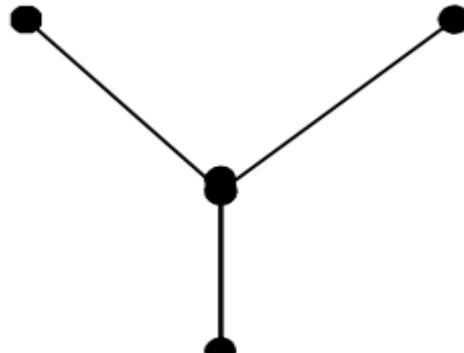
CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

Chương 5: CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

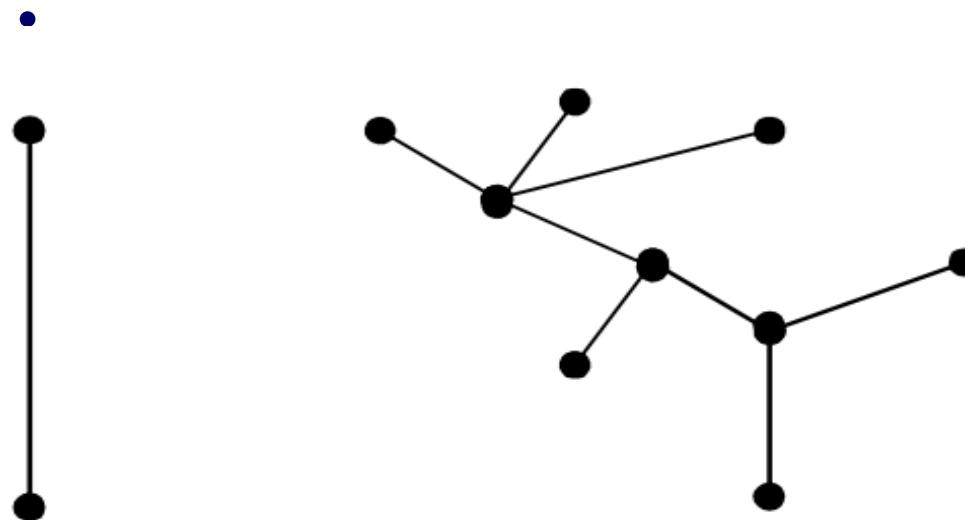
- ❖ Nội dung của chương này đề cập đến một loại đồ thị đơn giản nhất, đó là cây.
- ❖ Cây được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau của tin học như tổ chức các thư mục, lưu trữ dữ liệu, biểu diễn tính toán, biểu diễn quyết định và tổ chức truyền tin.
- ❖ Những nội dung được trình bày bao gồm:
 - ✓ Cây và các tính chất cơ bản của cây.
 - ✓ Cây khung của đồ thị & thuật toán xây dựng cây khung.
 - ✓ Bài toán tìm cây khung nhỏ nhất & các thuật toán tìm cây khung nhỏ nhất.

5.1 Cây và các tính chất của cây

- **Định nghĩa 1:** Cây là đồ thị vô hướng, liên thông, không có chu trình. Đồ thị vô hướng không có chu trình gọi là rừng (hợp của nhiều cây).
- Như vậy, rừng là đồ thị mà mỗi thành phần liên thông của nó là một cây.
- **Ví dụ 1:** Rừng gồm 3 cây T1, T2, T3



T1

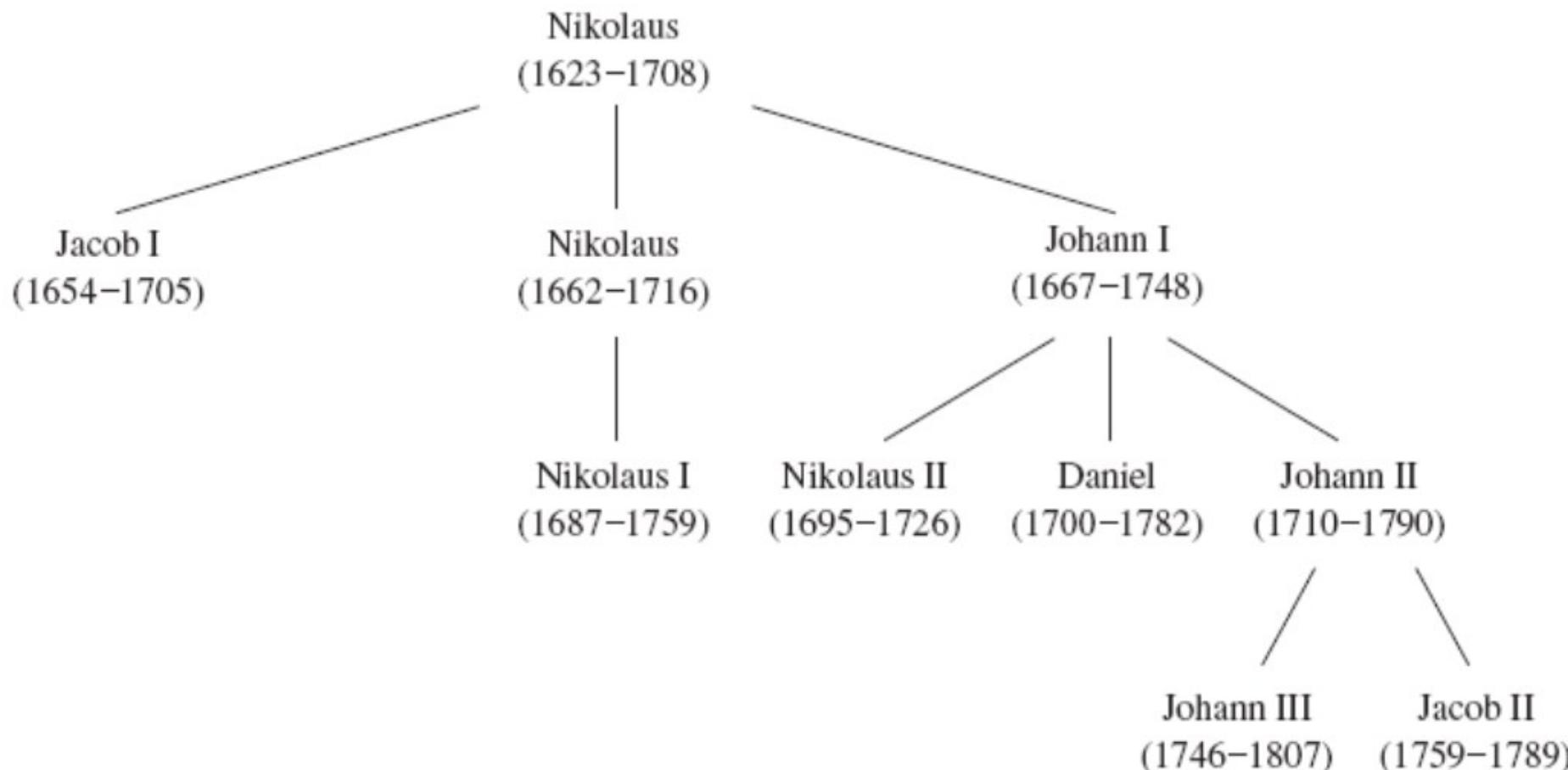


T2

T3

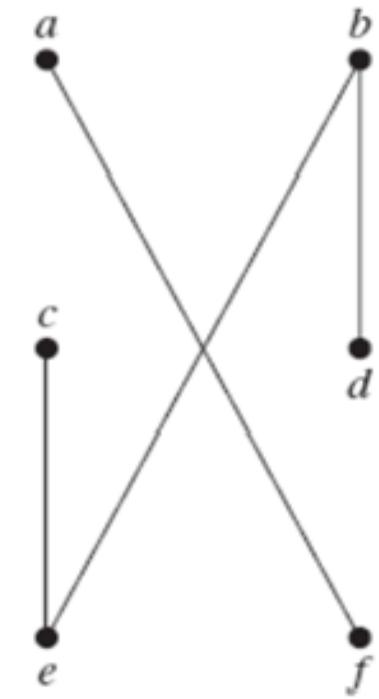
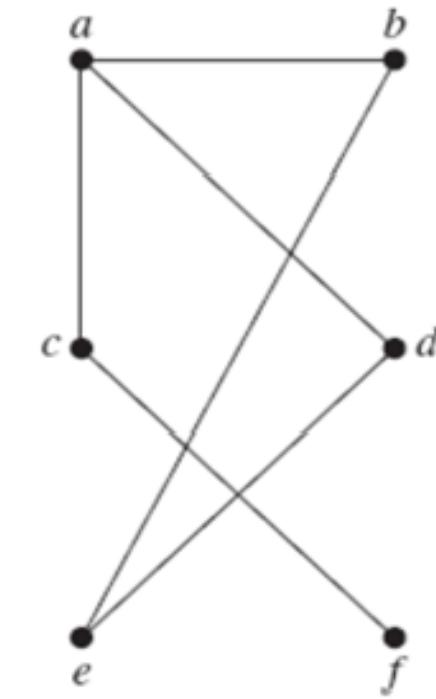
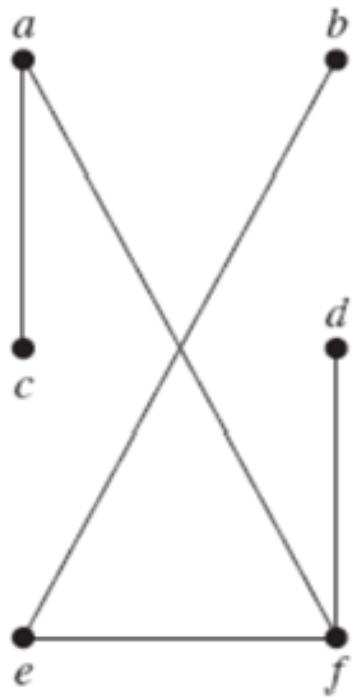
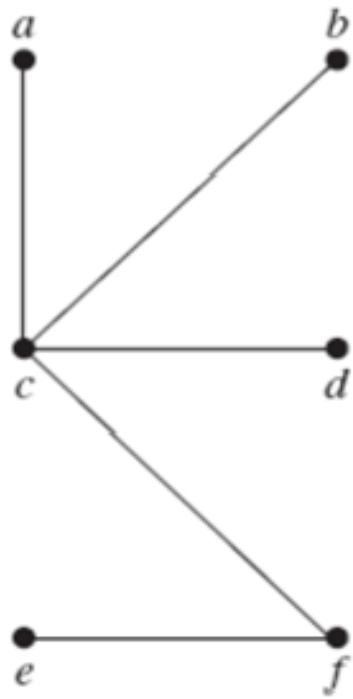
5.1 Cây và các tính chất của cây

- Ví dụ: biểu đồ phả hệ là cây



5.1 Cây và các tính chất của cây

BT: Đồ thị nào sau đây là cây?



G_1

G_2

G_3

G_4

5.1 Cây và các tính chất của cây

Cây được coi là dạng đồ thị đơn giản nhất của đồ thị. Định lý sau đây cho ta một số tính chất của cây.

- **Định lý 1:** Giả sử $T = (V, E)$ là đồ thị vô hướng với n đỉnh. Khi đó các mệnh đề sau là tương đương:

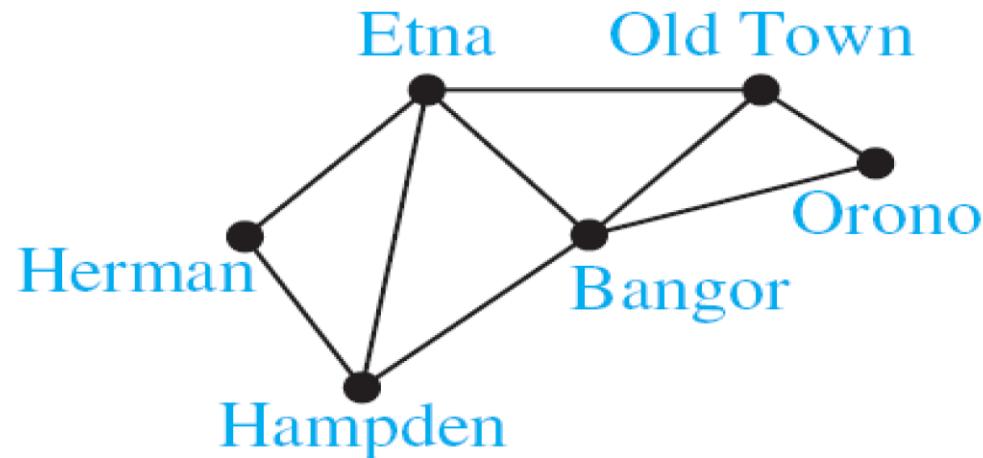
- (1) T là cây.
- (2) T không chứa chu trình đơn và có $n - 1$ cạnh
- (3) T liên thông và mỗi cạnh của nó đều là cầu
- (4) Giữa hai đỉnh bất kỳ của T đều tồn tại đúng một đường đi đơn
- (5) T không chứa chu trình đơn nhưng nếu thêm vào **một cạnh** ta thu được một chu trình đơn.
- (6) T liên thông và có $n - 1$ cạnh

Chứng minh định lý chi tiết trong giáo trình.

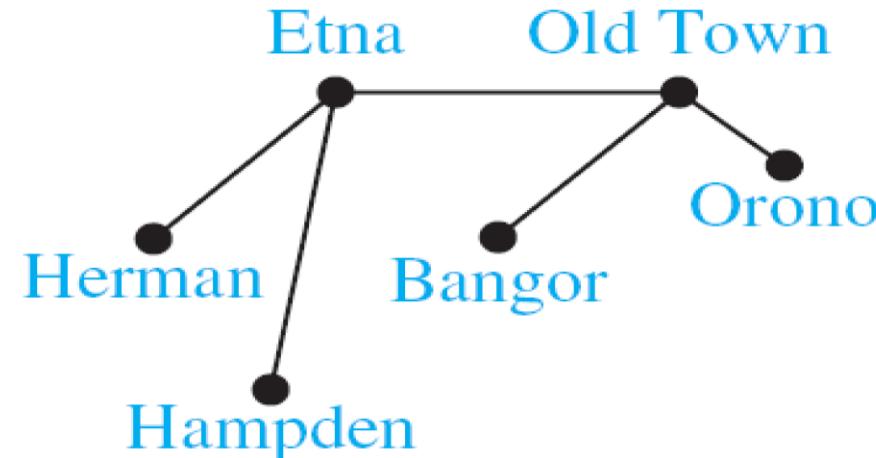
5.2 CÂY KHUNG CỦA ĐỒ THỊ

Bài toán giao thông ở Maine:

- Chính quyền địa phương muốn cào tuyết một số ít nhất các con đường sao cho luôn luôn có đường thông suốt nối hai thành phố bất kì



(a)



(b)

Chỉ cần cào tuyết theo hình b)

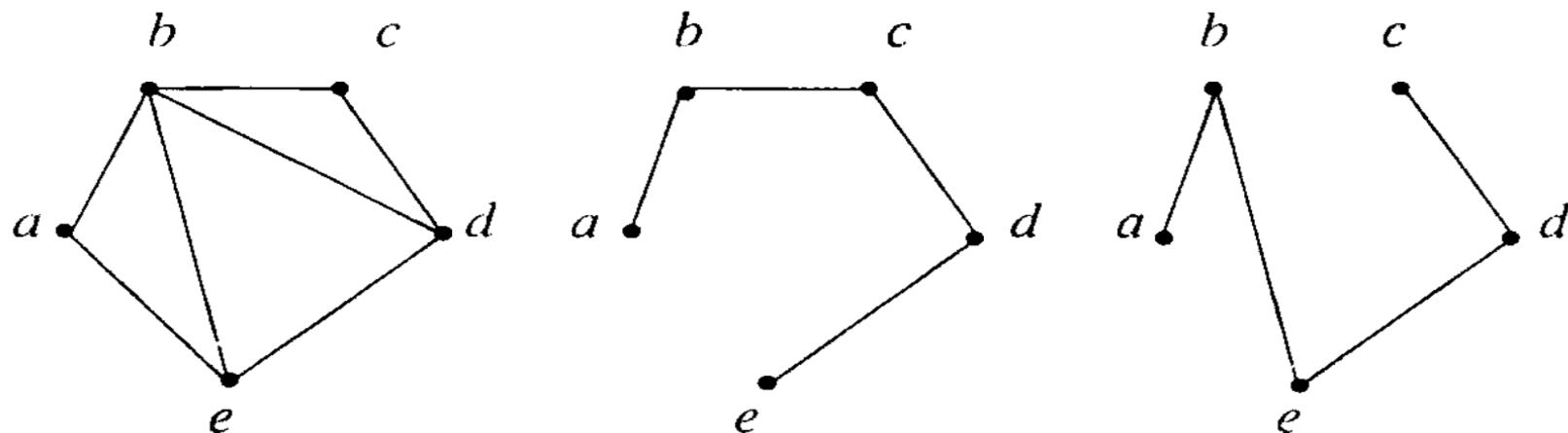
5.2.1 Cây khung của đồ thị

- **Định nghĩa 2:**

Giả sử $G = (V, E)$ là đồ thị vô hướng liên thông. Cây $T = (V, F)$ với $F \subset E$ gọi là cây khung của đồ thị G .

Tức là nếu như loại bỏ một số cạnh của G để được một cây thì cây đó gọi là cây khung (hay cây bao trùm của đồ thị).

- Dễ thấy rằng với một đồ thị vô hướng liên thông có thể có nhiều cây khung
- **Ví dụ:** Đồ thị và một số cây khung của nó



5.2.1 Cây khung của đồ thị

- **Nhận xét:**

- ✓ Điều kiện cần và đủ để một đồ thị vô hướng có cây khung là đồ thị đó phải liên thông.
- ✓ Đồ thị con T của G là một cây khung của G nếu T thoả mãn hai điều kiện:
 - T là một cây;
 - Tập đỉnh của T bằng tập đỉnh của G

- **Định lý 2 (Cayley):** Số cây khung của đồ thị đầy đủ K_n là n^{n-2} .

- *Hai bài toán cơ bản về cây:*

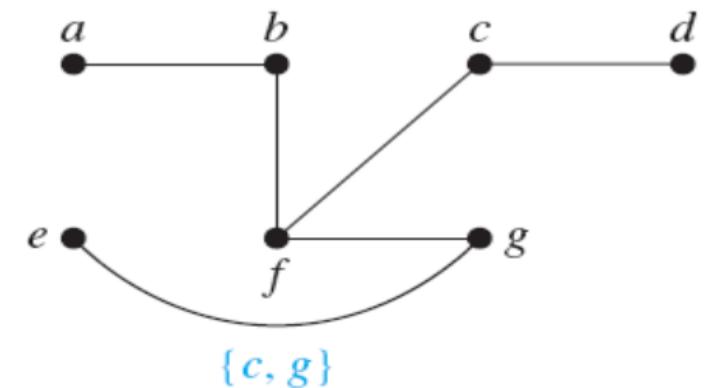
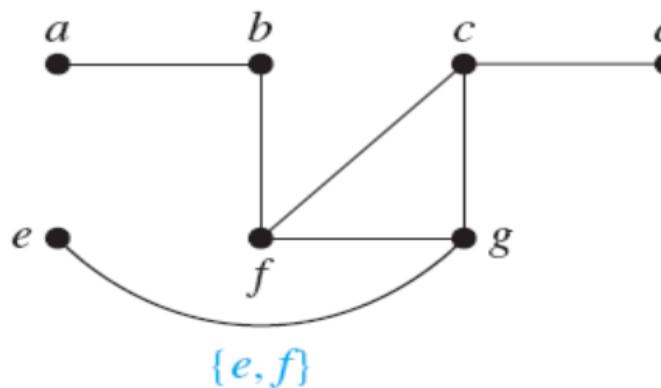
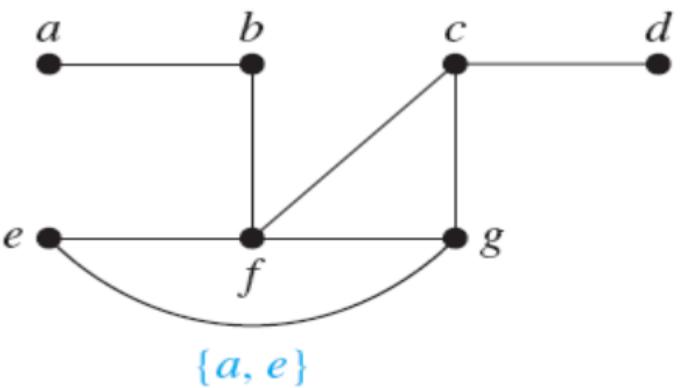
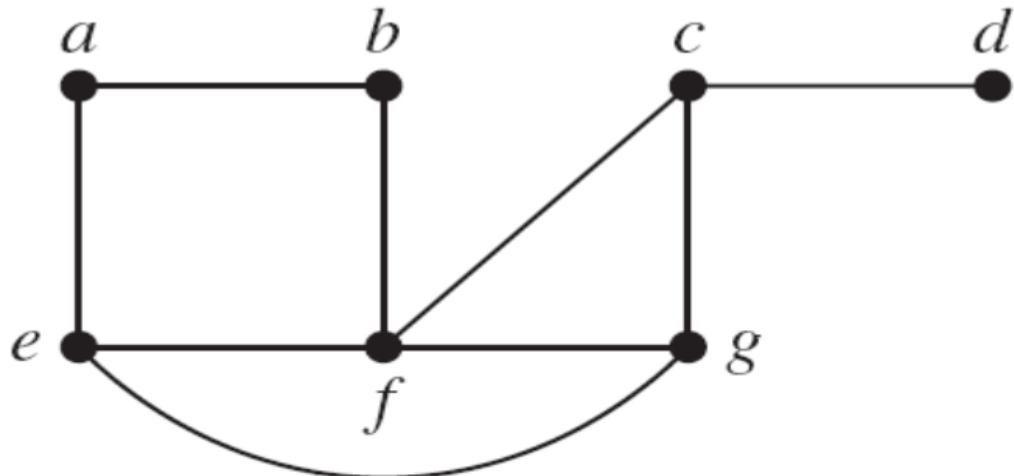
Bài toán 1. Cho đồ thị vô hướng $G = \langle V, E \rangle$. Hãy xây dựng một cây khung của đồ thị.

Bài toán 2. Cho đồ thị vô hướng $G = \langle V, E \rangle$ có trọng số. Hãy xây dựng cây khung có độ dài nhỏ nhất.

5.2.2 Tìm cây khung của đồ thị

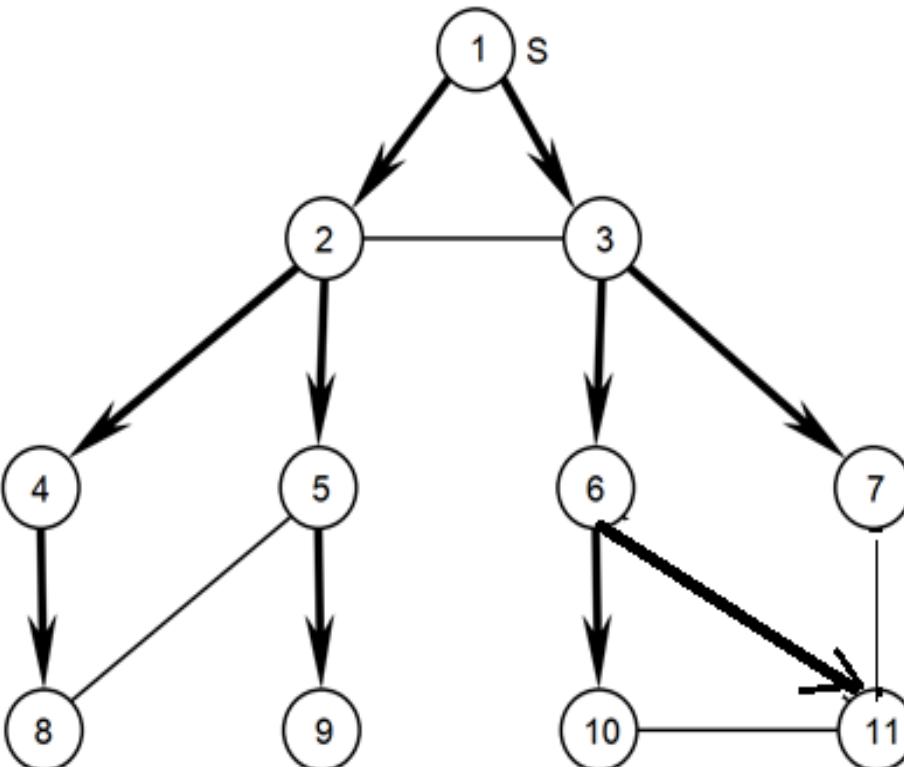
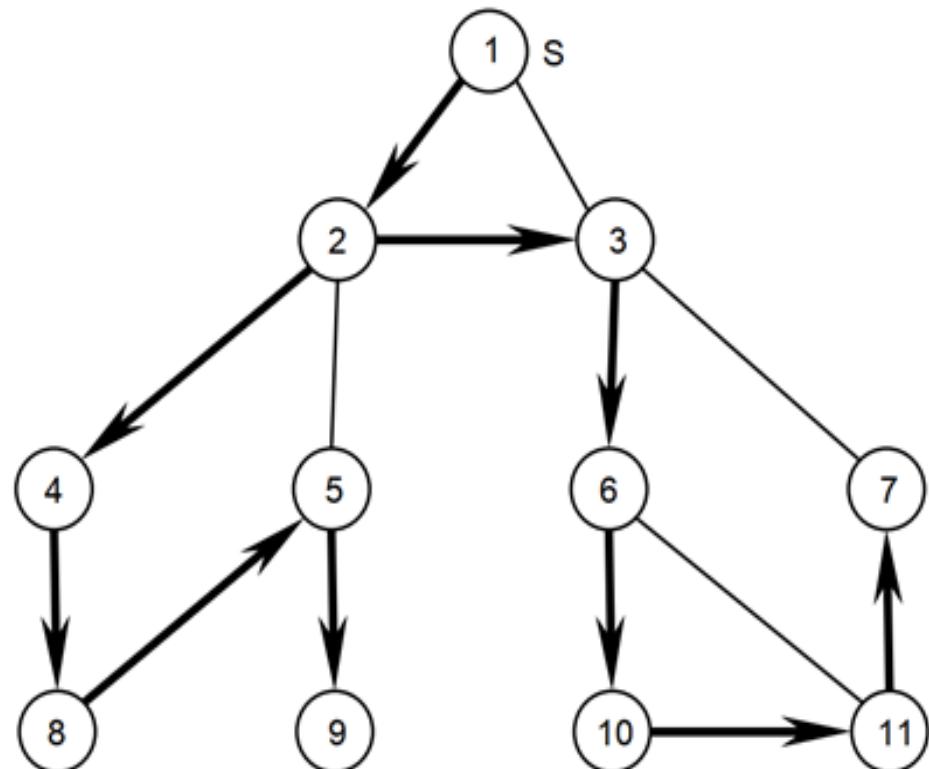
- Xét đồ thị vô hướng liên thông $G = (V, E)$ có n đỉnh, có nhiều thuật toán xây dựng cây khung của G :
 - ✓ *Xóa đi các cạnh tạo ra chu trình*
 - ✓ *Bằng phương pháp tìm kiếm theo chiều sâu*
 - ✓ *Bằng phương pháp tìm kiếm theo chiều rộng*

Xóa cạnh tạo ra chu trình



Xây dựng cây khung bằng các thuật toán tìm kiếm trên đồ thị

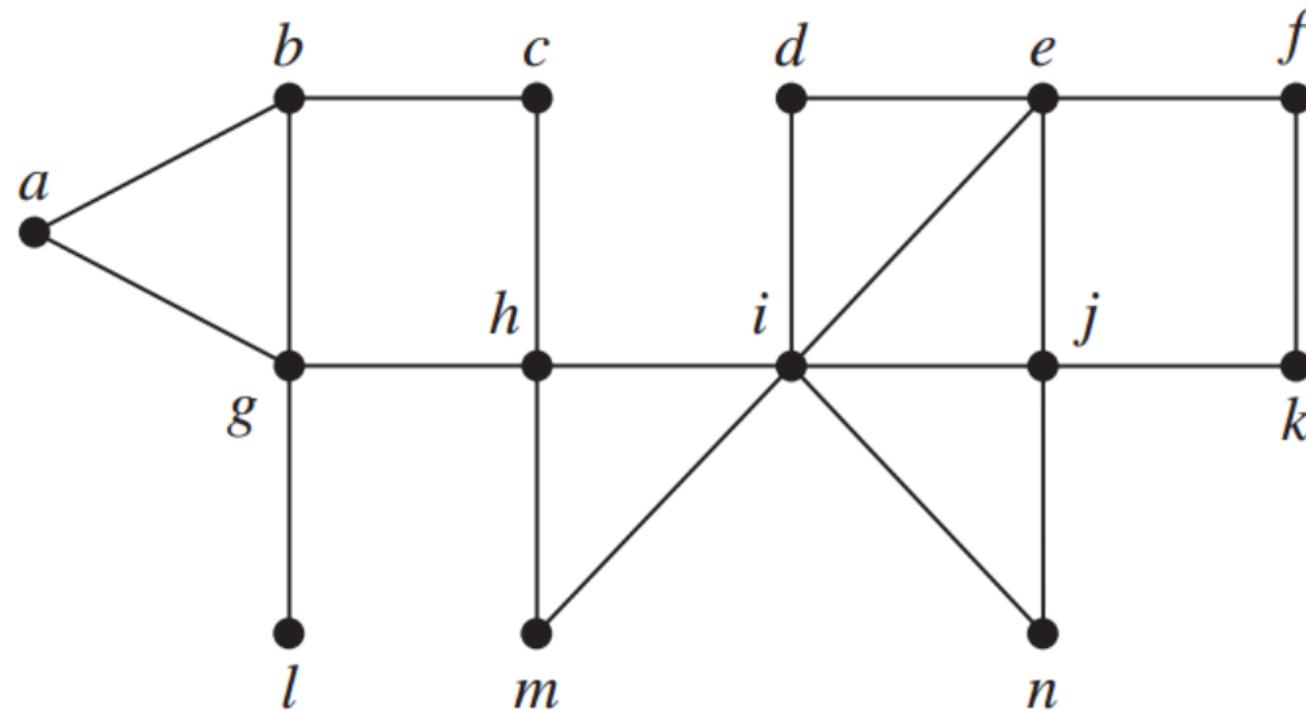
Để tìm một cây khung trên đồ thị vô hướng liên thông ta có thể sử dụng thuật toán duyệt các đỉnh (theo DFS hoặc BFS). Xuất phát từ đỉnh u nào đó, mỗi khi ta đến thăm được đỉnh v (`chuaxet[v] = 0`) từ đỉnh u thì cạnh (u,v) được kết nạp vào cây khung.



Cây khung DFS và cây khung BFS (Mũi tên chỉ chiều đi thăm các đỉnh)

Tìm cây khung theo DFS và BFS

- Ví dụ: tìm cây khung của đồ thị bắt đầu từ đỉnh a theo DFS và BFS.



Tìm cây khung theo DFS và BFS

```
procedure STREE_DFS(v);
(* Tìm kiếm theo chiều sâu áp dụng vào tìm tập cạnh của cây khung T
   của đồ thị vô hướng liên thông G cho bởi danh sách kề.
   Các biến Chuaxet, Ke, T là toàn cục *)  
begin
    . .
    Chuaxet[v] := false ;
    for u ∈ Ke(v) do
        if Chuaxet[u] then
            begin
                T:= T ∪ (v,u);
                STREE_DFS(u);
            end;
    end;
end;
```

Tìm cây khung theo DFS và BFS

```
procedure STRE_BFS(r);
(* Tìm kiếm theo chiều rộng áp dụng tìm tập cạnh của cây khung T
của đồ thị vô hướng liên thông G cho bởi danh sách Ke *)
begin
    QUEUE := ∅ ;
    QUEUE ← r ;
    Chuaxet[r]:=false;
    while QUEUE ≠ ∅ do
        begin
            v ← QUEUE;
            for u ∈ Ke(v) do
                if Chuaxet[u] then
                    begin
                        QUEUE ← u; Chuaxet[u]:=false;
                        T:= T ∪ (v,u);
                    end;
            end;
    end;
```

Tìm cây khung theo DFS và BFS

- Khi đó hàm chính như sau:

```
(*      Main Program    *)
BEGIN
    (* Initialiation *)
    for  u ∈ V  do  Chuaxet[u]:=true;
    T := Ø;          (*  T là tập cạnh của cây khung  *)
    STREE_DFS(root); (*  root  là đỉnh nào đó của đồ thị  *)
END.
```

Có thể thay thành STREE_BFS(root)

Tìm cây khung theo DFS và BFS

Ví dụ: Tìm cây khung của đồ thị cho bởi ma trận kè sau bằng DFS

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

Tìm cây khung theo DFS và BFS

Quá trình thực hiện thuật toán:

Bước	Tree-DFS(u) =?	$T = ?$
1	1	$T = \emptyset$
2	1, 2	$T = T \cup (1, 2)$
3	1, 2, 3	$T = T \cup (2, 3)$
4	1, 2, 3, 4	$T = T \cup (3, 4)$
5	1, 2, 3, 4, 5	$T = T \cup (3, 5)$
6	1, 2, 3, 4, 5, 6	$T = T \cup (5, 6)$
7	1, 2, 3, 4, 5, 6, 7	$T = T \cup (6, 7)$
8	1, 2, 3, 4, 5, 6, 7, 8	$T = T \cup (7, 8)$
9	1, 2, 3, 4, 5, 6, 7, 8, 9	$T = T \cup (8, 9)$
10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	$T = T \cup (9, 10)$
11	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	$T = T \cup (10, 11)$
12	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	$T = T \cup (11, 12)$
13	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	$T = T \cup (12, 13)$

Kết luận $T = \{(1,2), (2,3), (3,4), (3,5), (5,6), (6,7), (7,8), (8,9), (9,10), (10,11), (11,12), (12,13)\}$

5.3 CÂY KHUNG NHỎ NHẤT CỦA ĐỒ THỊ

Minimum Spanning Tree (MST)

5.3.1 Bài toán cây khung nhỏ nhất

■ Bài toán:

Cho $G = \langle V, E \rangle$ là đồ thị vô hướng liên thông với tập đỉnh $V = \{1, 2, \dots, n\}$ và tập cạnh E gồm m cạnh. Mỗi cạnh e của đồ thị được gán với một số không âm $c(e)$ được gọi là độ dài cạnh. Giả sử $H = \langle V, T \rangle$ là một cây khung của đồ thị G . Ta gọi độ dài $c(H)$ của cây khung H là tổng độ dài các cạnh:

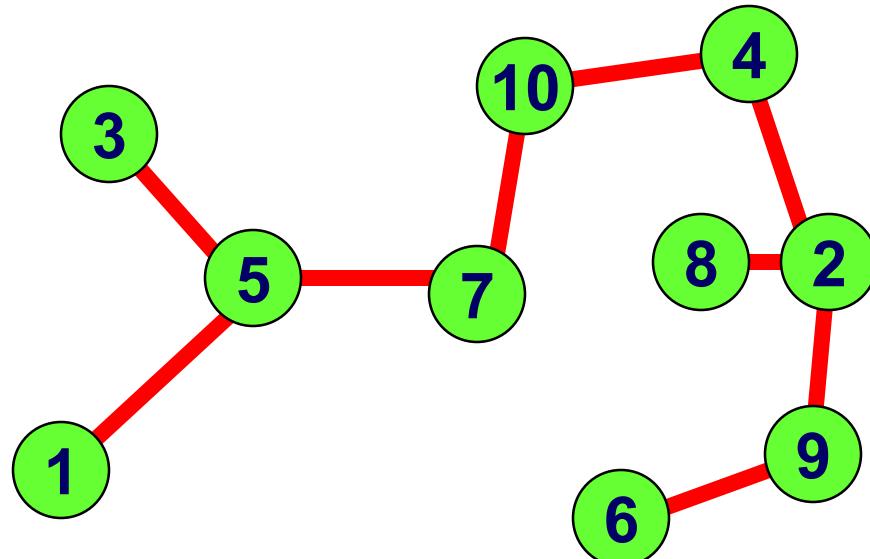
$$c(H) = \sum_{e \in T} c(e).$$

Bài toán được đặt ra là, trong số các cây khung của đồ thị hãy tìm cây khung có độ dài nhỏ nhất của đồ thị.

☞ Để minh họa cho những ứng dụng của bài toán này, chúng ta có thể tham khảo hai mô hình thực tế sau của bài toán.

Ứng dụng thực tế: Mạng truyền thông

- Công ty truyền thông AT&T cần xây dựng mạng truyền thông kết nối n khách hàng. Chi phí thực hiện kênh nối i và j là c_{ij} . Hỏi chi phí nhỏ nhất để thực hiện việc kết nối tất cả các khách hàng là bao nhiêu?



Giả thiết là: Chỉ có cách kết nối duy nhất là đặt kênh nối trực tiếp giữa hai nút.

Bài toán xây dựng hệ thống đường sắt

- Giả sử ta muốn xây dựng một hệ thống đường sắt nối n thành phố sao cho hành khách có thể đi lại giữa hai thành phố bất kỳ đồng thời tổng chi phí xây dựng phải là nhỏ nhất.
- Rõ ràng là đồ thị mà đỉnh là các thành phố còn các cạnh là các tuyến đường sắt nối các thành phố tương ứng với phương án xây dựng tối ưu phải là cây.
- Vì vậy, bài toán đặt ra dẫn về bài toán tìm cây khung nhỏ nhất trên đồ thị đầy đủ n đỉnh, mỗi đỉnh tương ứng với một thành phố, với độ dài trên các cạnh chính là chi phí xây dựng đường ray nối hai thành phố tương ứng.
- Chú ý: *Bài toán này giả thiết là không được xây dựng tuyến đường sắt có các nhà ga phân tuyến nằm ngoài các thành phố.*

5.4.2 Thuật toán tìm cây khung nhỏ nhất

- Để giải bài toán cây khung nhỏ nhất, tất nhiên có thể liệt kê tất cả các cây khung của đồ thị và chọn trong số chúng cây khung nhỏ nhất.
- Phương pháp liệt kê như vậy, trong trường hợp đồ thị đầy đủ, sẽ đòi hỏi thời gian cỡ n^{n-2} , rõ ràng không thể thực hiện được ngay cả với những đồ thị có hàng chục đỉnh.
- Hai thuật toán hiệu quả:*

Thuật toán Kruskal và Thuật toán Prim

Thuật toán Kruskal



Ý tưởng thuật toán (*Joseph Kruskal – 1956*)

Cho đồ thị vô hướng $G = (V, E)$ có n đỉnh.

Thuật toán xây dựng tập cạnh T của cây khung nhỏ nhất $H=(V,T)$ theo từng bước,

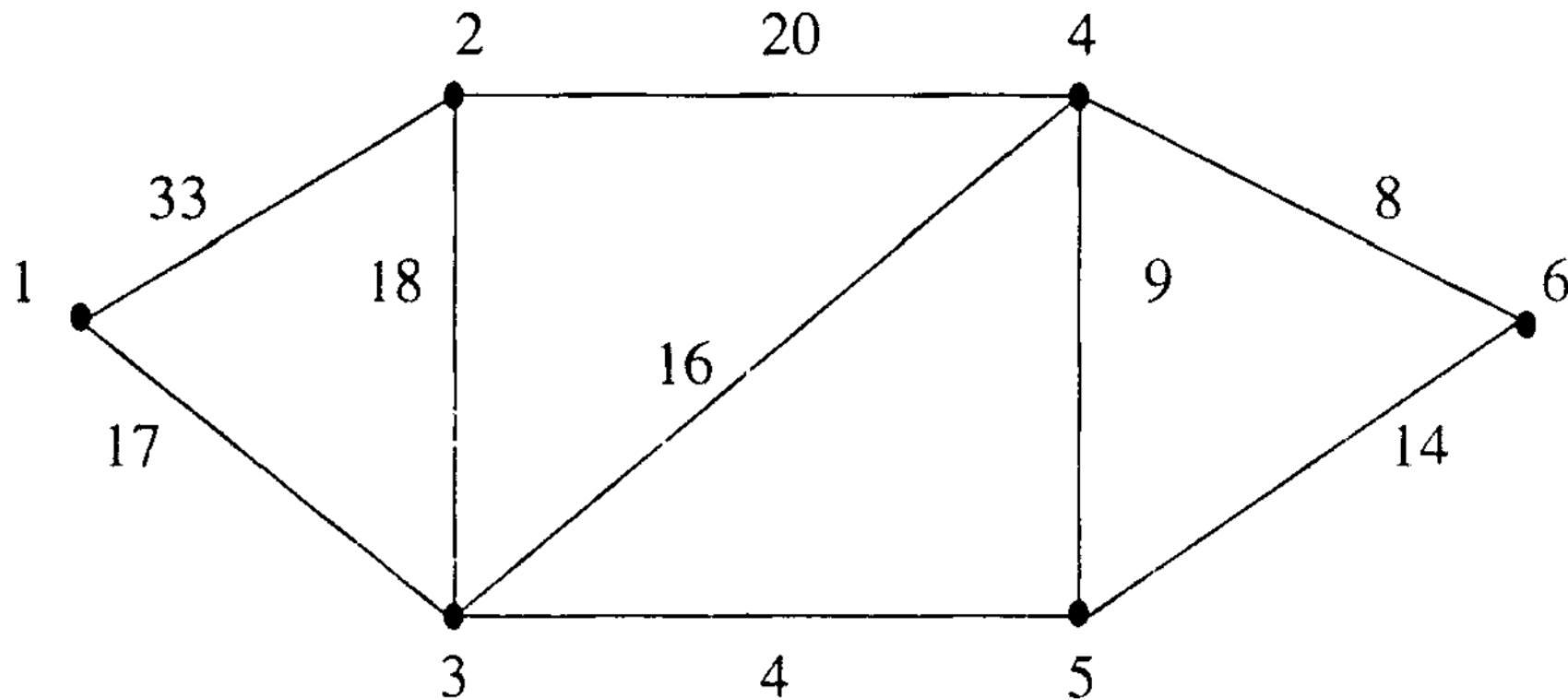
- Trước hết sắp các cạnh của đồ thị theo thứ tự không giảm của độ dài.
- Bắt đầu từ tập cạnh $T = \emptyset$. Ở mỗi bước, lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài lớn, để tìm ra cạnh mà việc bổ sung nó vào tập T không tạo thành chu trình trong tập này.
- Thuật toán sẽ kết thúc khi thu được tập T có $(n-1)$ cạnh.

Thuật toán Kruskal

```
procedure Kruskal;  
begin  
    T := Ø;  
    while |T| < (n-1) and ( E ≠ Ø ) do  
        begin  
            Chọn e là cạnh có độ dài nhỏ nhất trong E;  
            E := E \ {e};  
            if ( T ∪ {e} không chứa chu trình ) then T := T ∪ {e};  
        end;  
        if ( |T| < n-1 ) then Đồ thị không liên thông;  
    end;
```

Ví dụ thuật toán Kruskal

Ví dụ 1:



Ví dụ thuật toán Kruskal

Bước khởi tạo. Đặt $T := \emptyset$. Sắp xếp các cạnh của đồ thị theo thứ tự không giảm của độ dài ta có dãy:

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

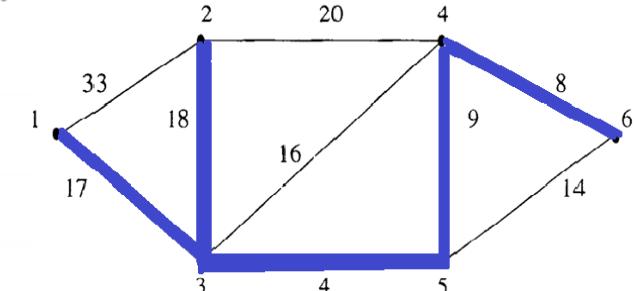
dãy độ dài tương ứng của chúng

4, 8, 9, 14, 16, 17, 18, 20, 23.

ở ba lần lặp đầu tiên ta lần lượt bổ sung vào tập T các cạnh (3,5), (4,6), (4,5). Rõ ràng nếu thêm cạnh (5,6) vào T thì nó sẽ tạo thành với 2 cạnh (4,5), (4,6) đã có trong T chu trình. Tình huống tương tự cũng xảy ra đối với cạnh (3,4) là cạnh tiếp theo trong dãy. Tiếp theo ta bổ sung cạnh (1,3), (2,3) vào T và thu được tập T gồm 5 cạnh:

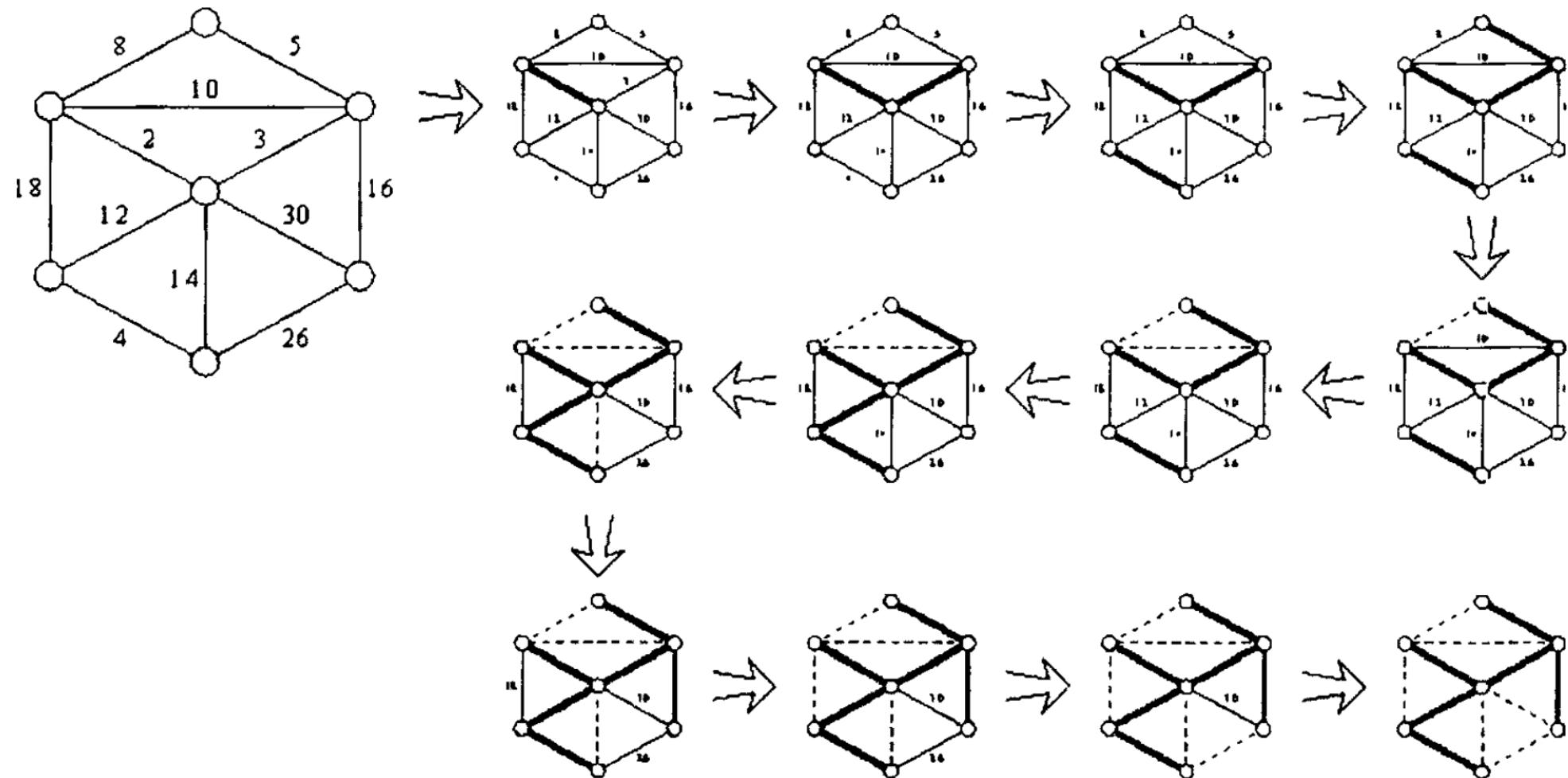
$$T = \{(3,5), (4,6), (4,5), (1,3), (2,3)\}$$

chính là tập cạnh của cây khung nhỏ nhất cần tìm.



Ví dụ thuật toán Kruskal

Ví dụ 2: Quá trình tìm cây khung nhỏ nhất. Cạnh được chọn – đậm, cạnh bỏ qua – nét đứt.



Thuật toán Prim



Nhận xét: Thuật toán Kruskal làm việc kém hiệu quả đối với những đồ thị dày (số cạnh $m \approx n(n-1)/2$). Trong trường hợp đó thuật toán Prim tỏ ra hiệu quả hơn. Ý tưởng chính của Thuật toán Prim được gọi là phương pháp lân cận gần nhất

Mô tả thuật toán Prim (*Robert Clay Prim – 1957*):

- ✓ Bắt đầu từ một đỉnh tùy ý s , đầu tiên ta nối s với đỉnh lân cận gần nó nhất, chẳng hạn là đỉnh y . Nghĩa là trong số các cạnh nối với đỉnh s , cạnh (s, y) có độ dài nhỏ nhất.
- ✓ Tiếp theo, trong số các cạnh nối với đỉnh s hoặc đỉnh y ta tìm cạnh có độ dài nhỏ nhất, cạnh này dẫn đến đỉnh thứ ba z , và ta thu được cây bộ phận gồm ba đỉnh và hai cạnh.
- ✓ Quá trình này sẽ được tiếp tục cho đến khi ta thu được cây gồm n đỉnh và $n-1$ cạnh, đó là cây khung nhỏ nhất cần tìm.

Thuật toán Prim

Thuật toán PRIM (s):

Begin:

Bước 1 (Khởi tạo):

$V_H = \{s\}$; // Tập đỉnh cây khung thiết lập ban đầu là s

$V = V \setminus \{s\}$; // Tập đỉnh V được bớt đi s

$T = \emptyset$; // Tập cạnh cây khung thiết lập ban đầu là \emptyset

$d(H) = 0$; // Độ dài cây khung được thiết lập là 0

Bước 2 (Lặp):

while ($V \neq \emptyset$) do {

$e = \langle u, v \rangle$: cạnh có độ dài nhỏ nhất thỏa mãn $u \in V$, $v \in V_H$;

$d(H) = d(H) + d(e)$; // Thiết lập đồ dài cây khung nhỏ nhất

$T = T \cup \{e\}$; // Kết nạp e vào cây khung

$V = V \setminus \{u\}$; // Tập đỉnh V bớt đi đỉnh u

$V_H = V_H \cup \{u\}$; // Tập đỉnh V_H thêm vào đỉnh u

endwhile;

Bước 3 (Trả lại kết quả):

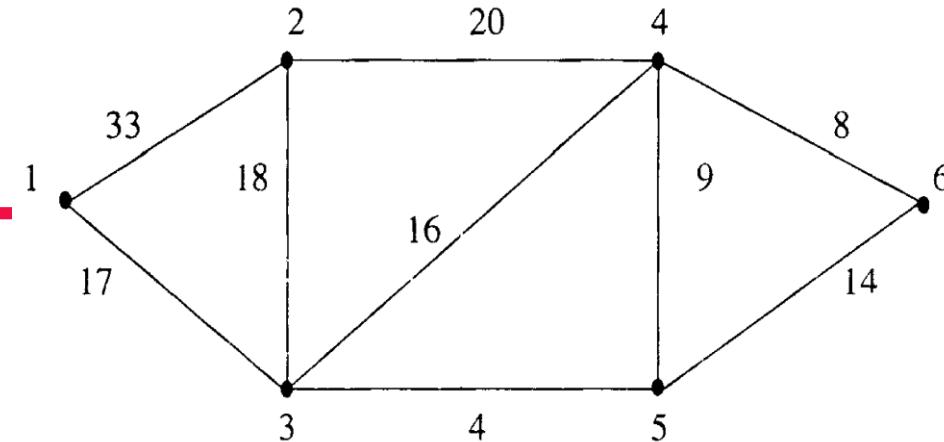
if ($|T| < n - 1$) then \langle Đồ thị không liên thông \rangle ;

else Return(T, d(H));

End.

Thuật toán Prim

Ví dụ: Tìm CKNN của đồ thị bên



Bước	Cạnh được chọn	Tập đỉnh ngoài cây $V - \{u\}$	Tập đỉnh của cây $V_H \cup \{u\}$	Tập cạnh của cây T và độ dài cây $D(T)$
0		2, 3, 4, 5, 6	1	$T = \emptyset$ $D(T)=0$
1	(1,3)	2, 4, 5, 6	1, 3	$T \cup (1,3)$ $D(T)=0+17=17$
2	(3,5)	2, 4, 6	1, 3, 5	$T \cup (3,5)$ $D(T)=17+4=21$
3	(5,4)	2, 6	1, 3, 5, 4	$T \cup (4,5)$ $D(T)=21+9=30$
4	(4,6)	2	1, 3, 5, 4, 6	$T \cup (4,6)$ $D(T)=30+8=38$
5	(3,2)	\emptyset	1, 3, 5, 4, 6, 2	$T \cup (3,2)$ $D(T)=38+18=56$

Thuật toán Prim

Cài đặt thuật toán:

- Đồ thị cho bởi ma trận trọng số $C = \{c[ij], i, j = 1, 2, \dots, n\}$.
Hai đỉnh i và j không có cạnh nối thì đặt $c[ij] = +\infty$.
- Ở mỗi bước, để nhanh chóng chọn đỉnh và cạnh cần bổ sung vào cây khung, các đỉnh của đồ thị sẽ được gán cho các nhãn. Nhãn của một đỉnh v gồm hai phần và có dạng $[d[v], near[v]]$, trong đó $d[v]$ là độ dài của cạnh ngắn nhất trong số các cạnh nối đỉnh v với các đỉnh của cây khung đang xây dựng.

$$d[v] := \min \{ c[v, w] : w \in V_H \} = c[v, z],$$

$near[v] = z$, lưu lại đỉnh gần v nhất.

Thuật toán Prim

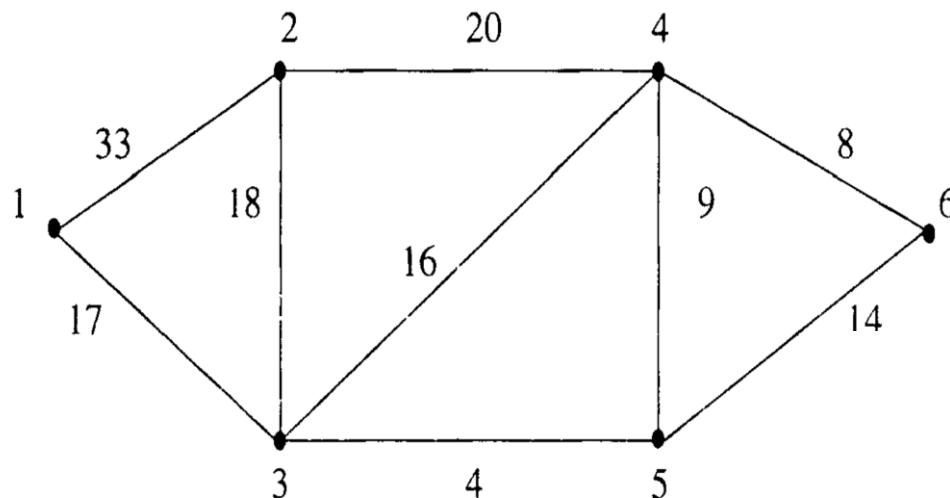
procedure Prim;

{ // *Bước khởi tạo*

1. Chọn s là một đỉnh nào đó của đồ thị;
2. $V_H := \{ s \}$; $T := \emptyset$; $d[s] := 0$; $\text{near}[s] := s$;
3. for $v \in V \setminus V_H$ do { $d[v] := c[s, v]$; $\text{near}[v] := s$; }
4. $\text{Stop} := 0$;
5. while not Stop do {*// Bước lặp*
 6. Tìm $u \in V \setminus V_H$ thỏa mãn: $d[u] = \min \{ d[v] : v \in V \setminus V_H \}$;
 7. $V_H := V_H \cup \{ u \}$; $T := T \cup \{ (u, \text{near}[u]) \}$;
 8. if $|V_H| = n$ then { $H = (V_H, T)$ là cây khung nhỏ nhất;
 $\text{Stop} := 1$;
 10. else
 11. for $v \in V \setminus V_H$ do
 12. if $d[v] > c[u, v]$ then { $d[v] := c[u, v]$; $\text{near}[v] := u$; } }

Ví dụ thuật toán Prim

Ví dụ 1: Tìm cây khung nhỏ nhất của đồ thị sau bằng TT Prim


$$C = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 33 & 17 & \infty & \infty & \infty \\ 2 & 33 & 0 & 18 & 20 & \infty & \infty \\ 3 & 17 & 18 & 0 & 16 & 4 & \infty \\ 4 & \infty & 20 & 16 & 0 & 9 & 8 \\ 5 & \infty & \infty & 4 & 9 & 0 & 14 \\ 6 & \infty & \infty & \infty & 8 & 14 & 0 \end{bmatrix}$$

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

Bước	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	\emptyset

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

Bước	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	\emptyset
1	-	[18, 3]	-	[16, 3]	[4, 3]*	[∞ , 1]	1, 3	(3,1)

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

Bước	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	\emptyset
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)

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

Bước	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	\emptyset
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)

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

Bước	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	\emptyset
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)

Ví dụ thuật toán Prim

Bảng ghi nhãn của các đỉnh trong các bước lặp, đỉnh đánh dấu * là đỉnh được chọn để bổ sung vào cây khung.

Sửa nhãn:

```
if d[v] > c[u,v] {d[v] := c[u,v] ; near[v] := u;}
```

	1	2	3	4	5	6
1	0	33	17	∞	∞	∞
2	33	0	18	20	∞	∞
3	17	18	0	16	4	∞
4	∞	20	16	0	9	8
5	∞	∞	4	9	0	14
6	∞	∞	∞	8	14	0

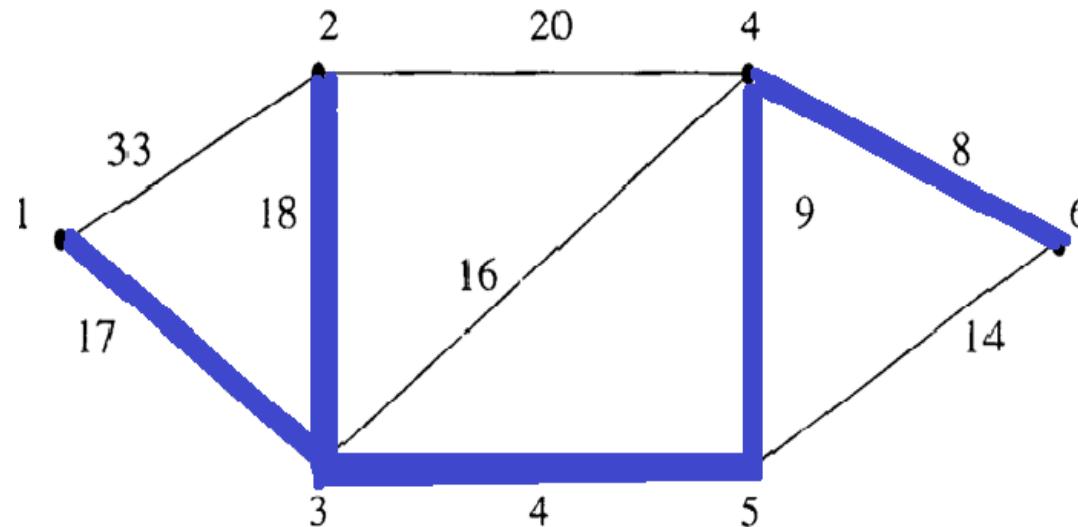
Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	V_H	T
Khởi tạo	[0, 1]	[33, 1]	[17, 1]*	[∞ , 1]	[∞ , 1]	[∞ , 1]	1	\emptyset
1	-	[18, 3]	-	[16, 3]	[4, 3]*	[∞ , 1]	1, 3	(3,1)
2	-	[18, 3]	-	[9,5]*	-	[14, 5]	1, 3, 5	(3,1), (5,3)
3	-	[18,3]	-	-	-	[8,4]*	1,3,5,4	(3,1), (5,3), (4,5)
4	-	[18,3]*	-	-	-	-	1,3,5,4,6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

Ví dụ thuật toán Prim

KẾT QUẢ:

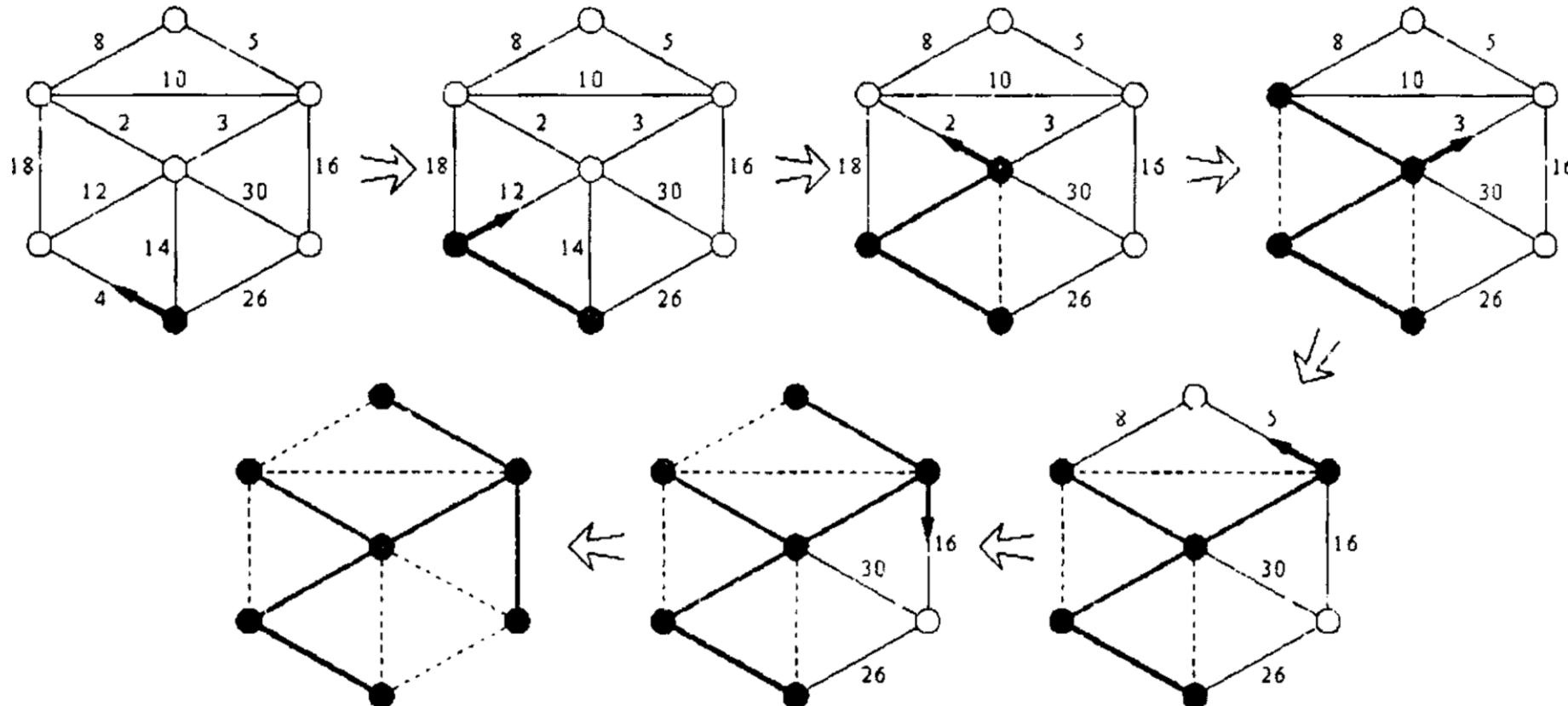
Tập cạnh của CKNN: { (3,1), (5,3), (4,5), (6,4), (2,3) }

Độ dài của CKNN : $17 + 4 + 9 + 8 + 18 = 56$



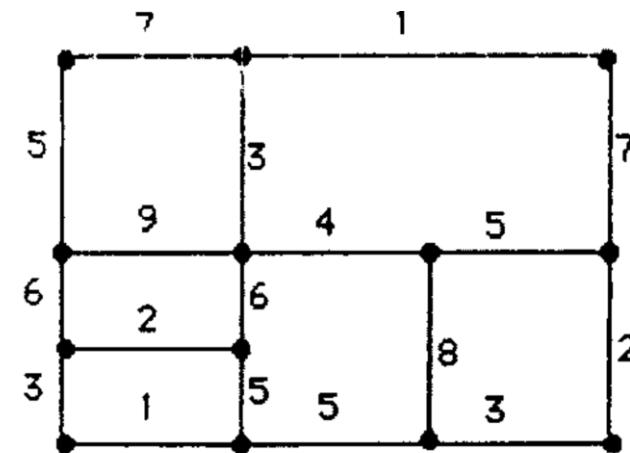
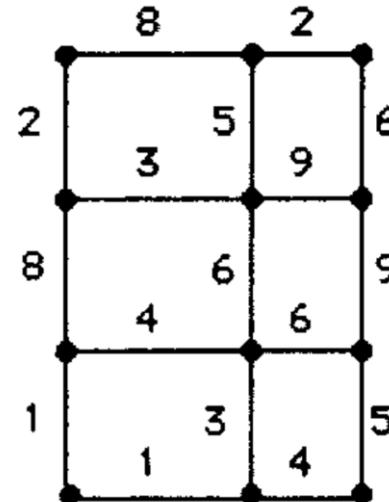
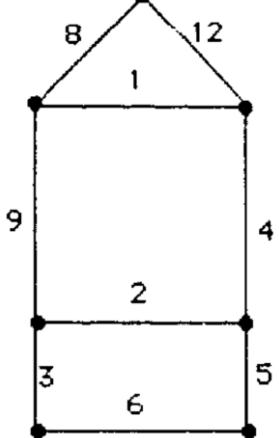
Ví dụ thuật toán Prim

Ví dụ 2: Quá trình tìm cây khung nhỏ nhất của đồ thị, các cạnh đậm là các cạnh được chọn vào cây khung. Mũi tên chỉ hướng chọn đỉnh tiếp theo, cạnh đứt nét không cần phải xét trong các bước tiếp theo.



Bài tập

1. Tìm cây khung nhỏ nhất của đồ thị



2. Xét đồ thị gồm 6 đỉnh A, B, C, D, E, F cho bởi ma trận trọng số.

*Tìm cây khung nhỏ nhất bằng 2 thuật toán
Prim và Kruscal*

A	0	33	17	85	85	85
B	33	0	18	20	85	85
C	17	18	0	16	4	85
D	85	20	16	0	9	8
E	85	85	4	9	0	14
F	85	85	85	8	14	0

Người đề xuất bài toán MST

Otakar Borůvka

- Nhà khoa học Séc (Czech)
- Người đề xuất bài toán
- Bài báo được xuất bản ở Séc từ năm 1926.
- Ứng dụng vào việc phát triển hệ thống mạng điện ở Bohemia.



Nhận xét

Bài toán cây khung lớn nhất ?

Dễ dàng nhận thấy rằng, trong các thuật toán trên không đòi hỏi về dấu của độ dài cạnh. Vì vậy có thể áp dụng chúng đối với đồ thị có các cạnh với độ dài có dấu tùy ý.

Vì vậy, giả sử ta phải tìm cây khung lớn nhất (tức là có độ dài $c(H)$ lớn nhất) thì chỉ cần đổi dấu tất cả các độ đo và áp dụng một trong hai thuật toán vừa mô tả.

