ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Toán rời rạc

Phần thứ hai

LÝ THUYẾT ĐỒ THỊ

Graph Theory

Nội dung phần 2: Lý thuyết đồ thị

Nội dung chi tiết

4.1. Cây và các tính chất cơ bản của cây

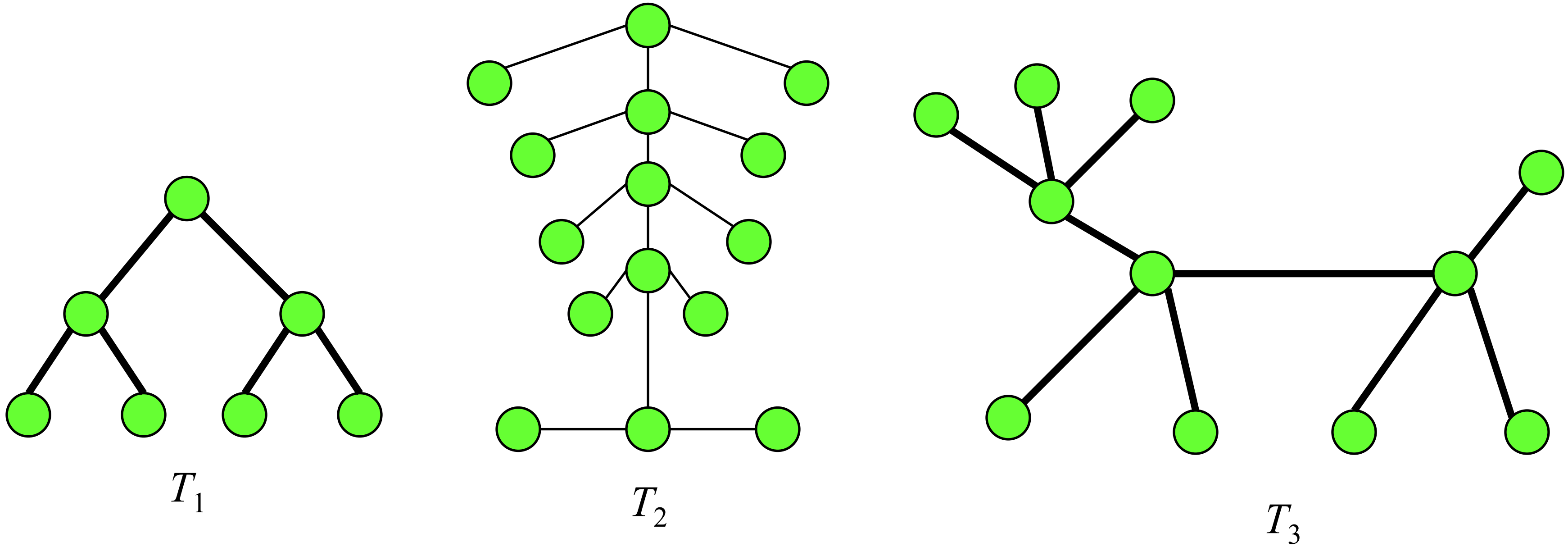
4.2. Cây khung của đồ thị

4.3. Bài toán cây khung nhỏ nhất

Cây và rừng (Tree and Forest)

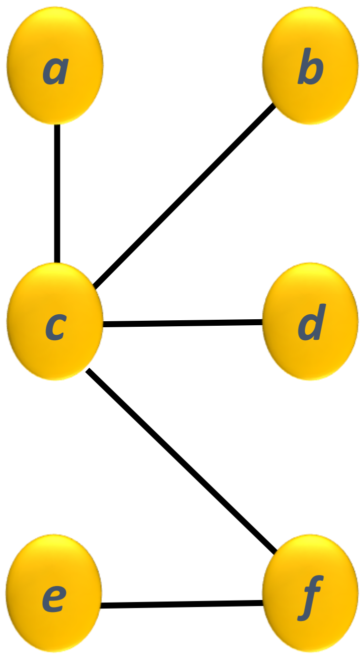
- ***Cây*** là đồ thị vô hướng liên thông không có chu trình.
- ***Rừng*** là đồ thị không có chu trình.

→ **Rừng** là đồ thị mà mỗi thành phần liên thông của nó là một cây.



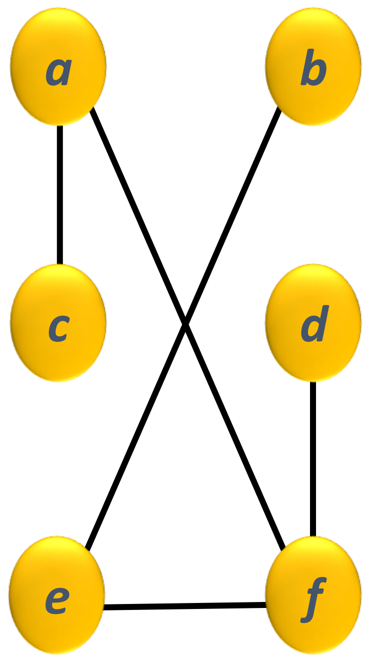
Rừng F gồm 3 cây T_1, $T_{2,}$, T_3

Ví dụ



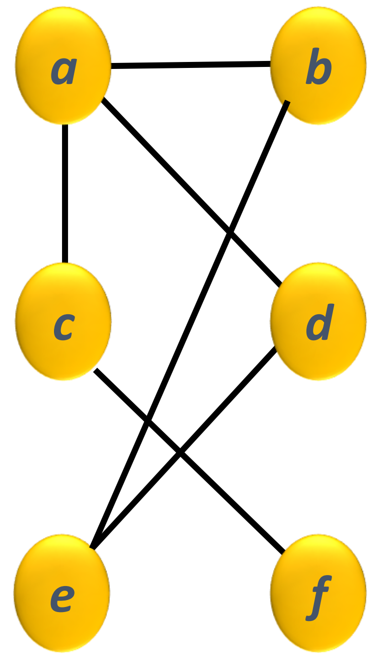
G_1

Cây



G_2

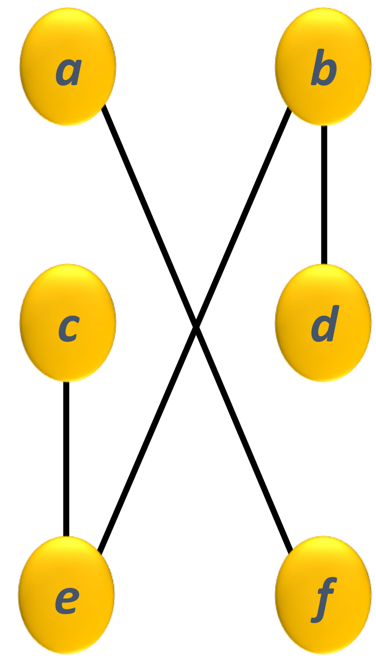
Cây



G_3

Chu trình {a,b,e,d,a}

Không là cây



G_4

Rừng

Không liên thông → không là cây

Gồm 2 cây: {a,f} và {c,e,b,d}

Các tính chất cơ bản của cây

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

(1) *T là liên thông và không chứa chu trình;*

(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) *Hai đỉnh bất kỳ của T được nối với nhau bởi đúng một đường đi đơn;*

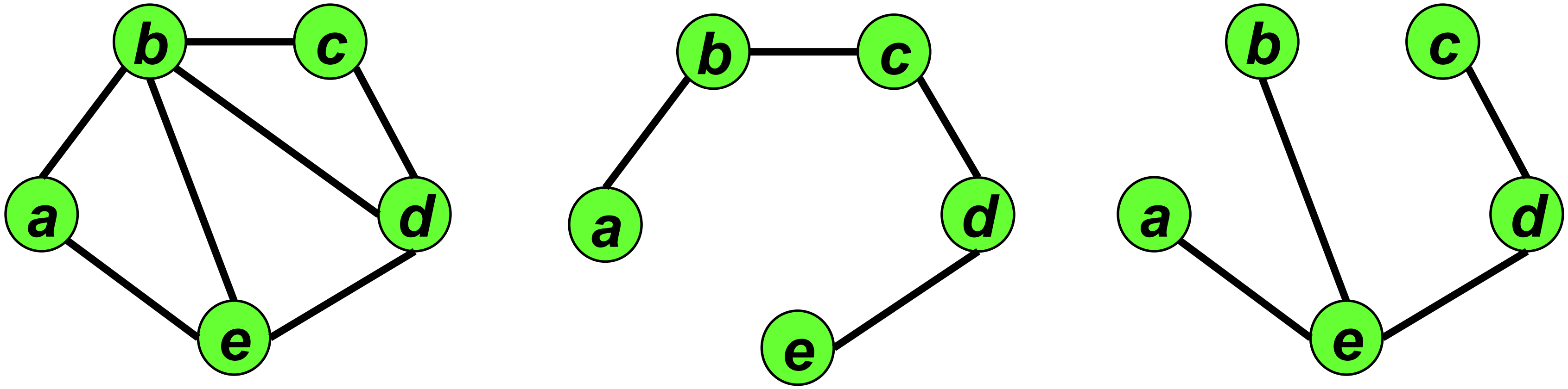
(6) *T không chứa chu trình nhưng hễ cứ thêm vào nó một cạnh ta thu được đúng một chu trình.*

Nội dung chi tiết

4.2. Cây khung của đồ thị

- Định nghĩa. *Giả sử G=(V,E) là đồ thị vô hướng liên thông. Cây T=(V,F) với F⊆ E được gọi là cây khung của đồ thị G.*

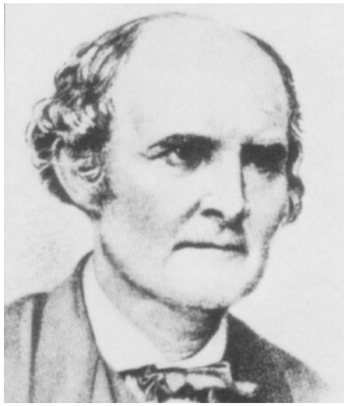
vô hướng liên thông không có chu trình



G *Cây khung T_1* *Cây khung T_2*

Đồ thị G và 2 cây khung T_1 và T_2 của nó

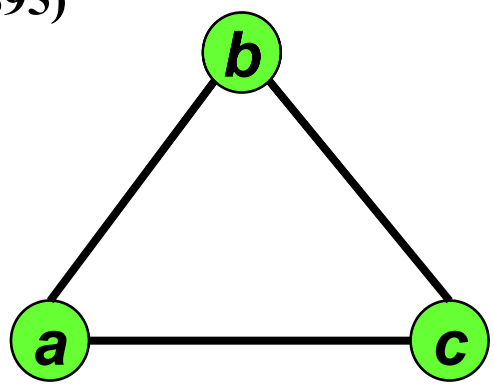
Số lượng cây khung của đồ thị



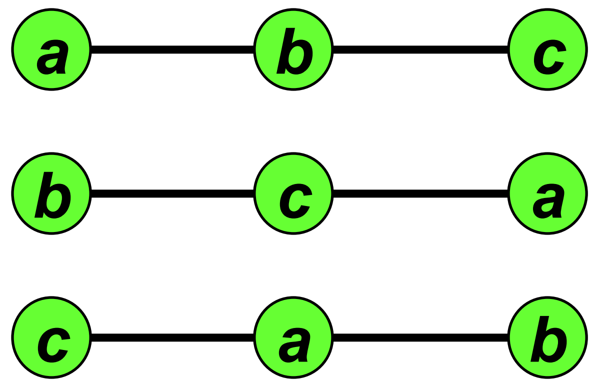
Arthur Cayley
(1821 – 1895)

Định lý sau đây cho biết số lượng cây khung của đồ thị đầy đủ K_n:

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

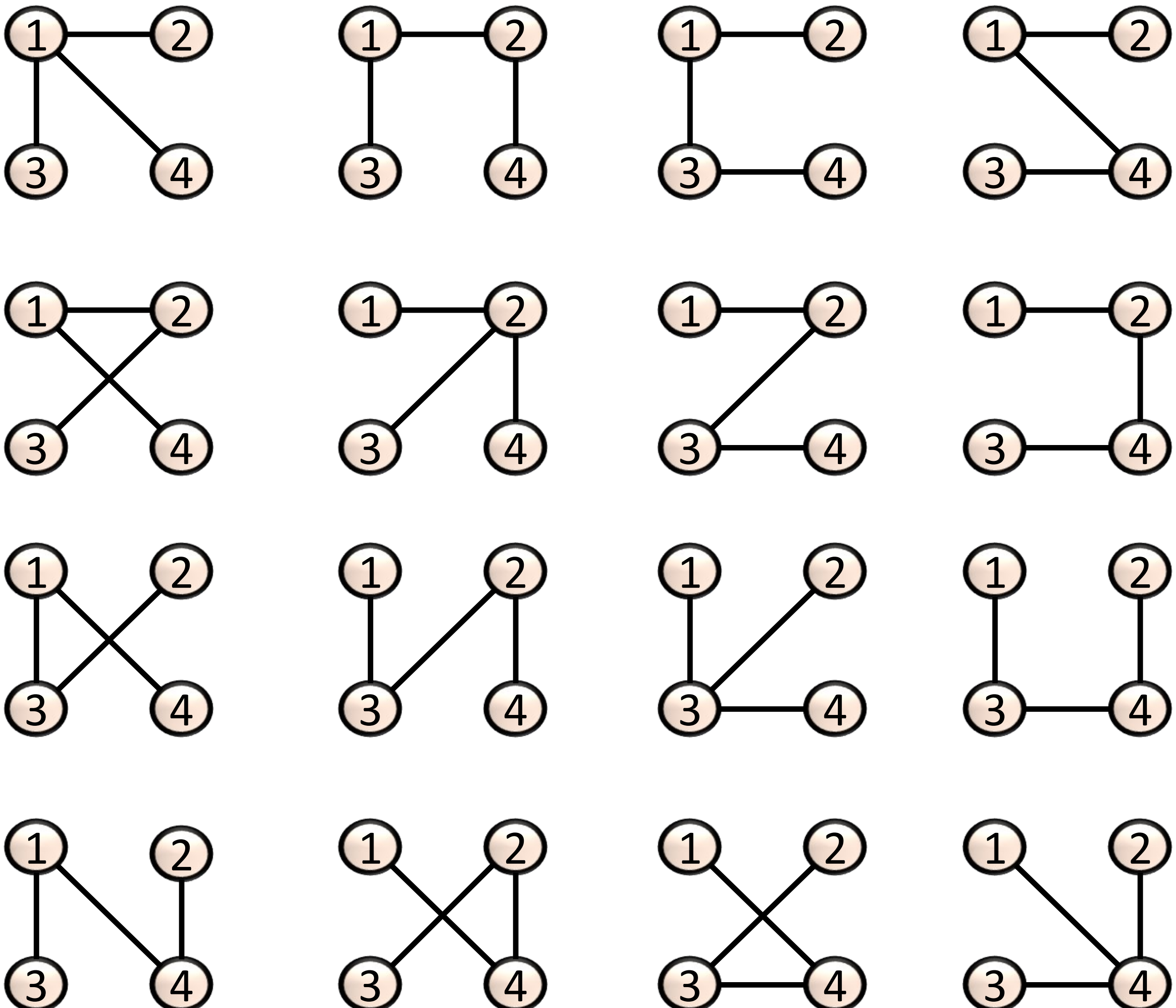


K_3



Ba cây khung của K_3

16 cây khung của K_4



Nội dung chi tiết

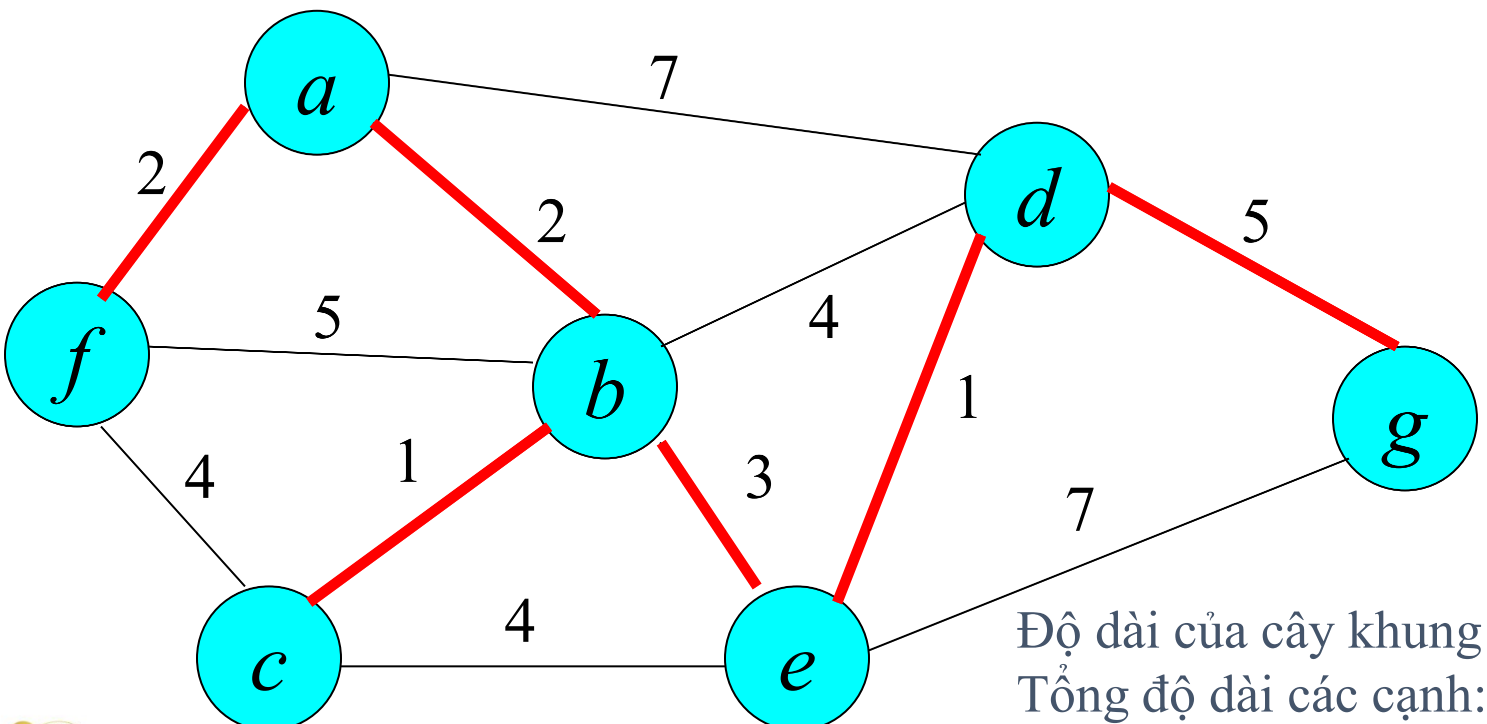
4.1. Cây và các tính chất cơ bản của cây

4.2. Cây khung của đồ thị

4.3. Bài toán cây khung nhỏ nhất

4.3. Bài toán cây khung nhỏ nhất

Bài toán: Cho đồ thị vô hướng liên thông $G=(V,E)$ với trọng số $c(e)$, $e \in E$. Độ dài của cây khung là tổng trọng số trên các cạnh của nó. Cần tìm cây khung có độ dài nhỏ nhất.



Độ dài của cây khung là
Tổng độ dài các cạnh: 14

4.3. Bài toán cây khung nhỏ nhất

- Có thể phát biểu dưới dạng bài toán tối ưu tổ hợp:

  Tìm cực tiểu

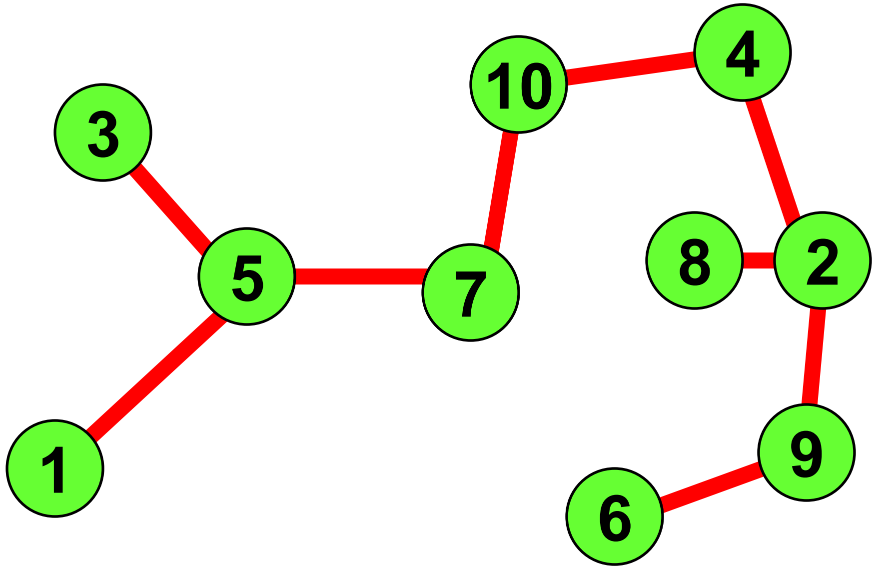
$$c(H) = \sum_{e \in T} c(e) \rightarrow \min,$$

  với điều kiện $H=(V,T)$ là cây khung của G.

Do số lượng cây khung của G là rất lớn (xem định lý Cayley), nên không thể giải nhờ duyệt toàn bộ

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?

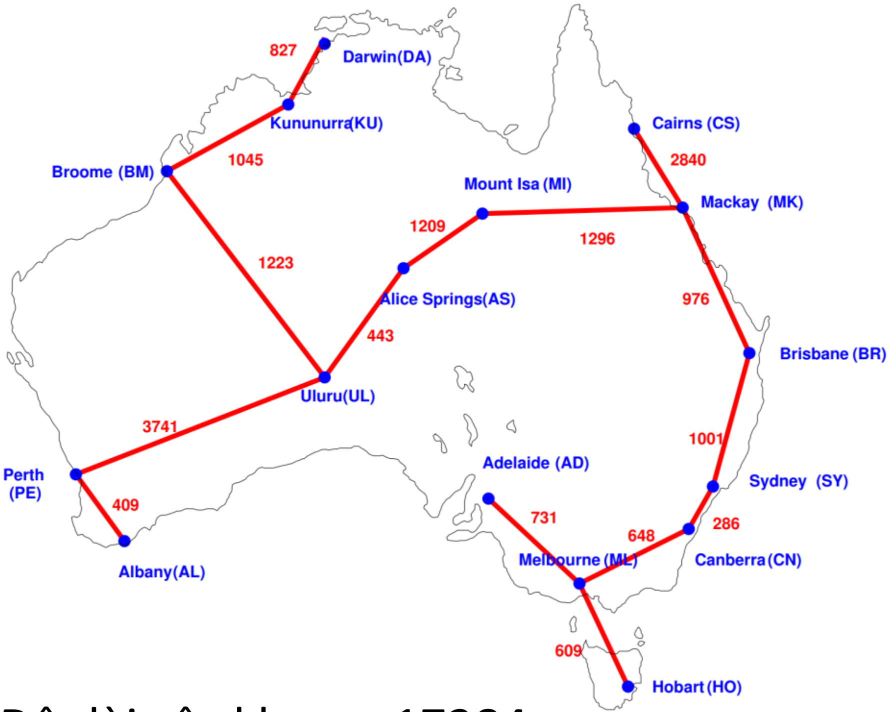


Giả thiết là: Chỉ có cách kết nối duy nhất là đặt kênh nối trực tiếp giữa hai nút.

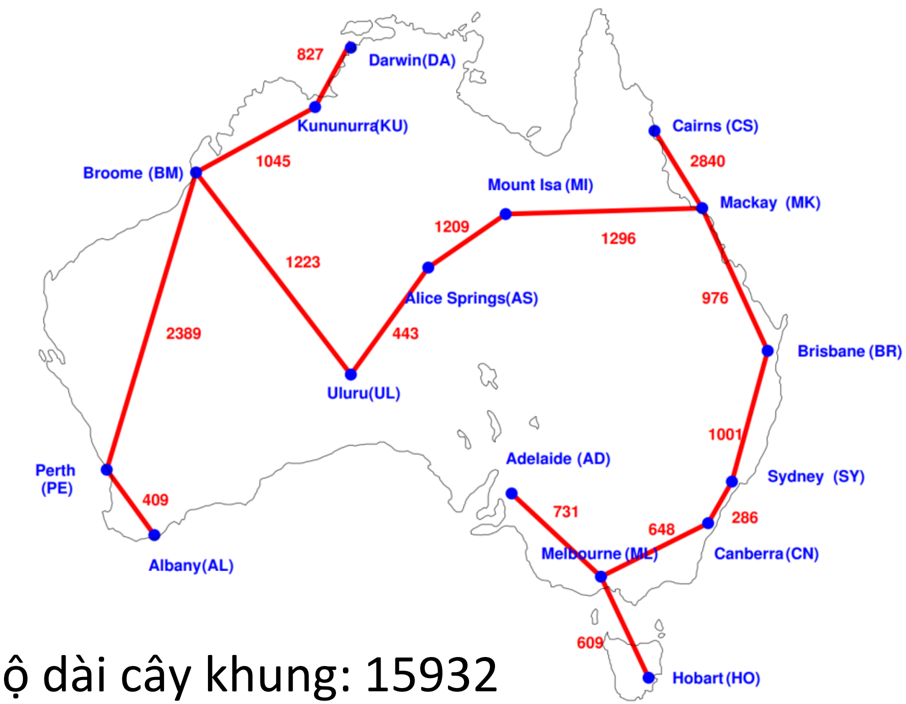
Bài toán xây dựng hệ thống đường sắt

- 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

Bài toán xây dựng hệ thống đường sắt



Độ dài cây khung: 17284



Độ dài cây khung: 15932

Sơ đồ chung của thuật toán

Generic-MST(G, c)
    $A = \varnothing$
    //Bất biến: A là tập con các cạnh của CKNN nào đó
    while A chưa là cây khung do
        tìm cạnh (u, v) là an toàn đối với A
        $A = A \cup \{(u, v)\}$
        // A vẫn là tập con các cạnh của CKNN nào đó
    return A

Cạnh rẻ nhất
và bổ sung nó vào A
không tạo nên chu
trình

Tìm cạnh an toàn bằng cách nào?

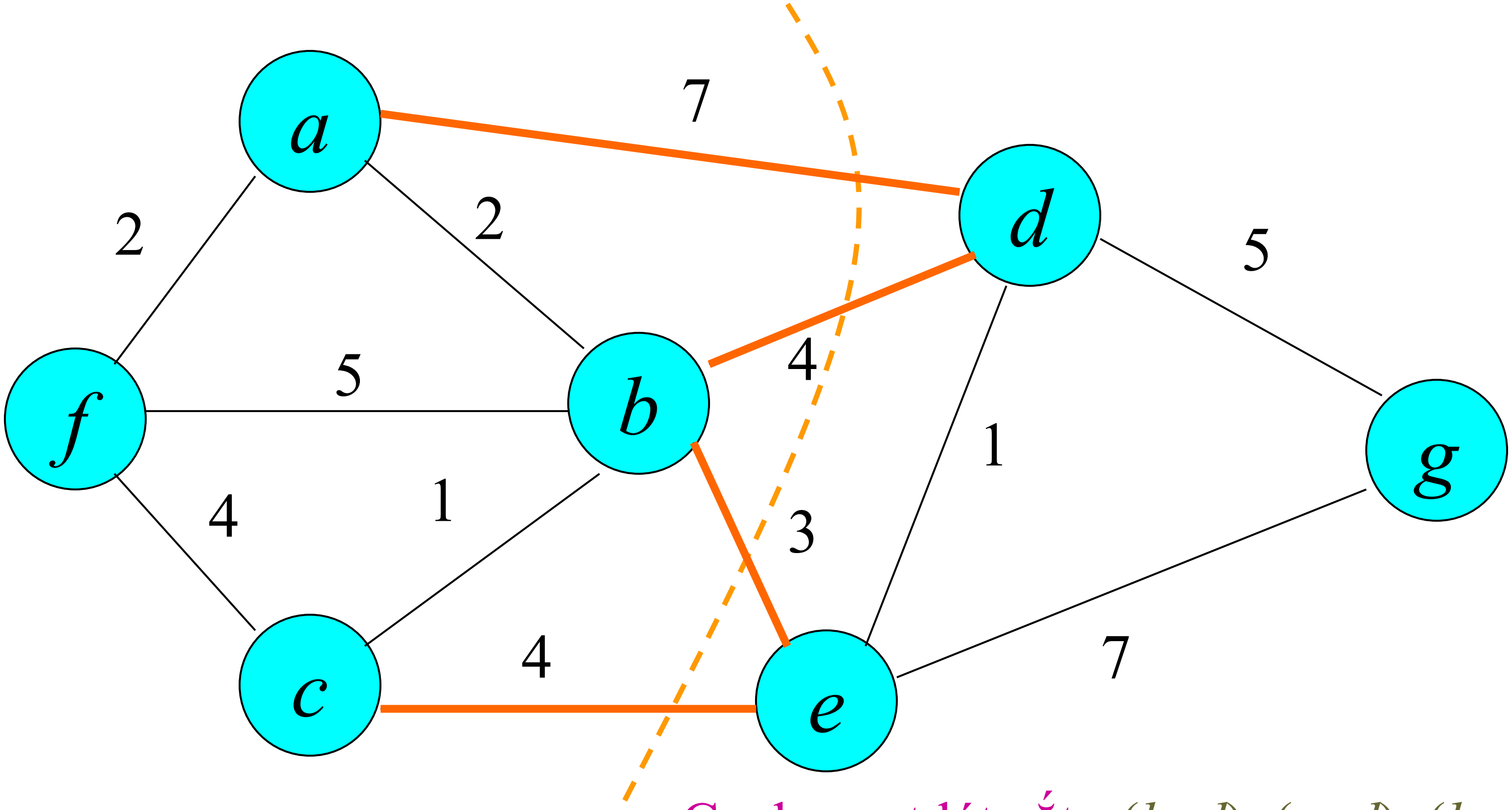
Lát cắt

- Ta gọi *lát cắt* $(S, V-S)$ là một cách phân hoạch tập đỉnh V ra thành hai tập S và $V-S$. Ta nói cạnh e là cạnh *vượt lát cắt* $(S, V-S)$ nếu một đầu mút của nó là thuộc S còn đầu mút còn lại thuộc $V-S$.

Ví dụ

Lát cắt của $G = (V, E)$ là phân hoạch V thành $(S, V - S)$.

Ví dụ. Lát cắt $(S, V\text{-}S)$: $S = \{a, b, c, f\}$, $V - S = \{e, d, g\}$



Cạnh vượt lát cắt : (b, d), (a, d), (b, e), (c, e)

Các cạnh còn lại không vượt lát cắt.

Lát cắt

- Ta gọi *lát cắt* $(S, V-S)$ là một cách phân hoạch tập đỉnh V ra thành hai tập S và $V-S$. Ta nói cạnh e là cạnh *vượt lát cắt* $(S, V-S)$ nếu một đầu mút của nó là thuộc S còn đầu mút còn lại thuộc $V-S$.
- Giả sử A là một tập con các cạnh của đồ thị. Lát cắt $(S, V-S)$ được gọi là *tương thích* với A nếu như không có cạnh nào thuộc A là cạnh vượt lát cắt.

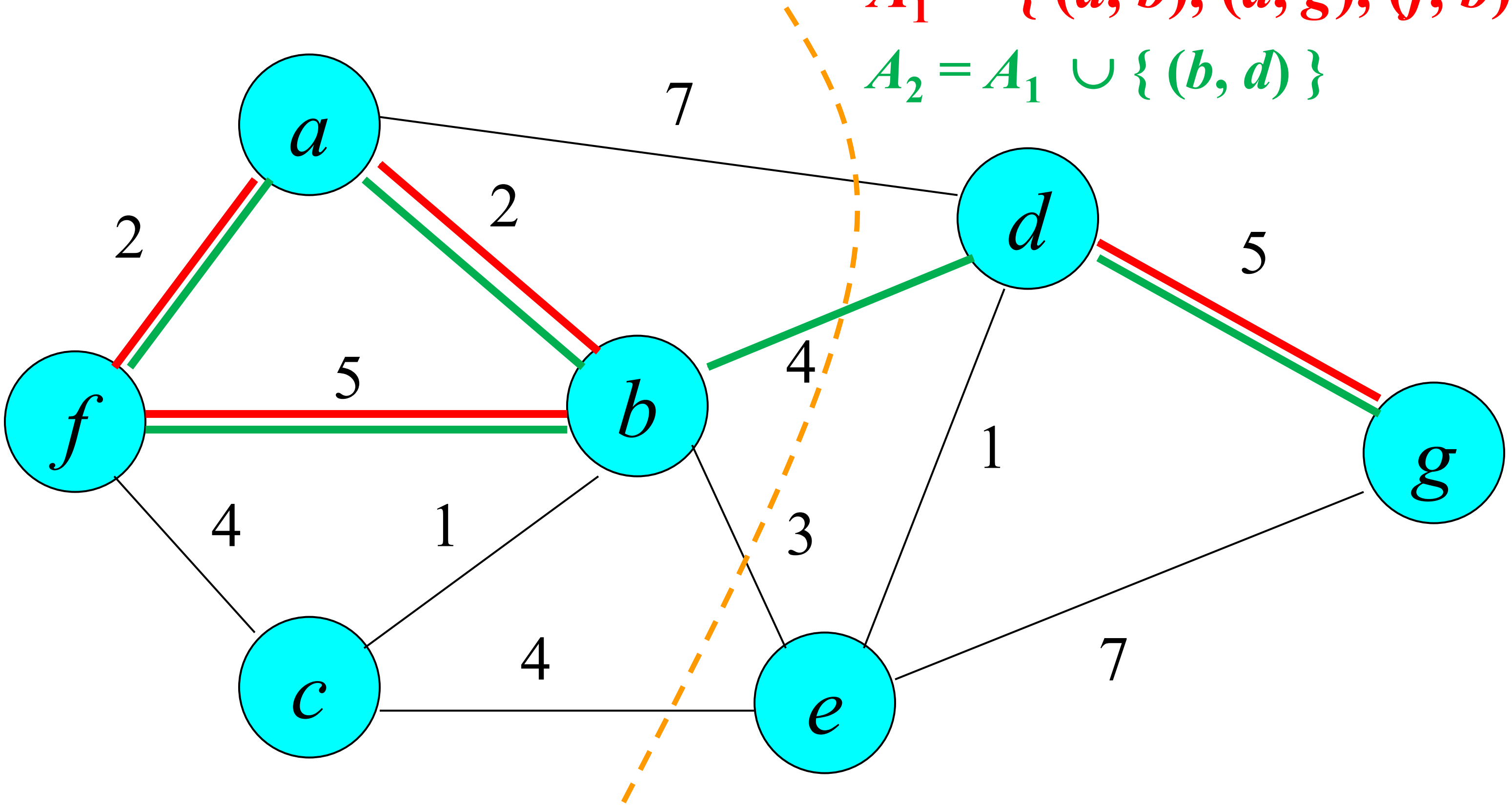
Ví dụ: lát cắt tương thích với tập cạnh

Ví dụ. Lát cắt (S, V-S): $S = \{a, b, c, f\}$

Tập cạnh:

$A_1 = \{ (a, b), (d, g), (f, b), (a, f) \}$

$A_2 = A_1 \cup \{ (b, d) \}$



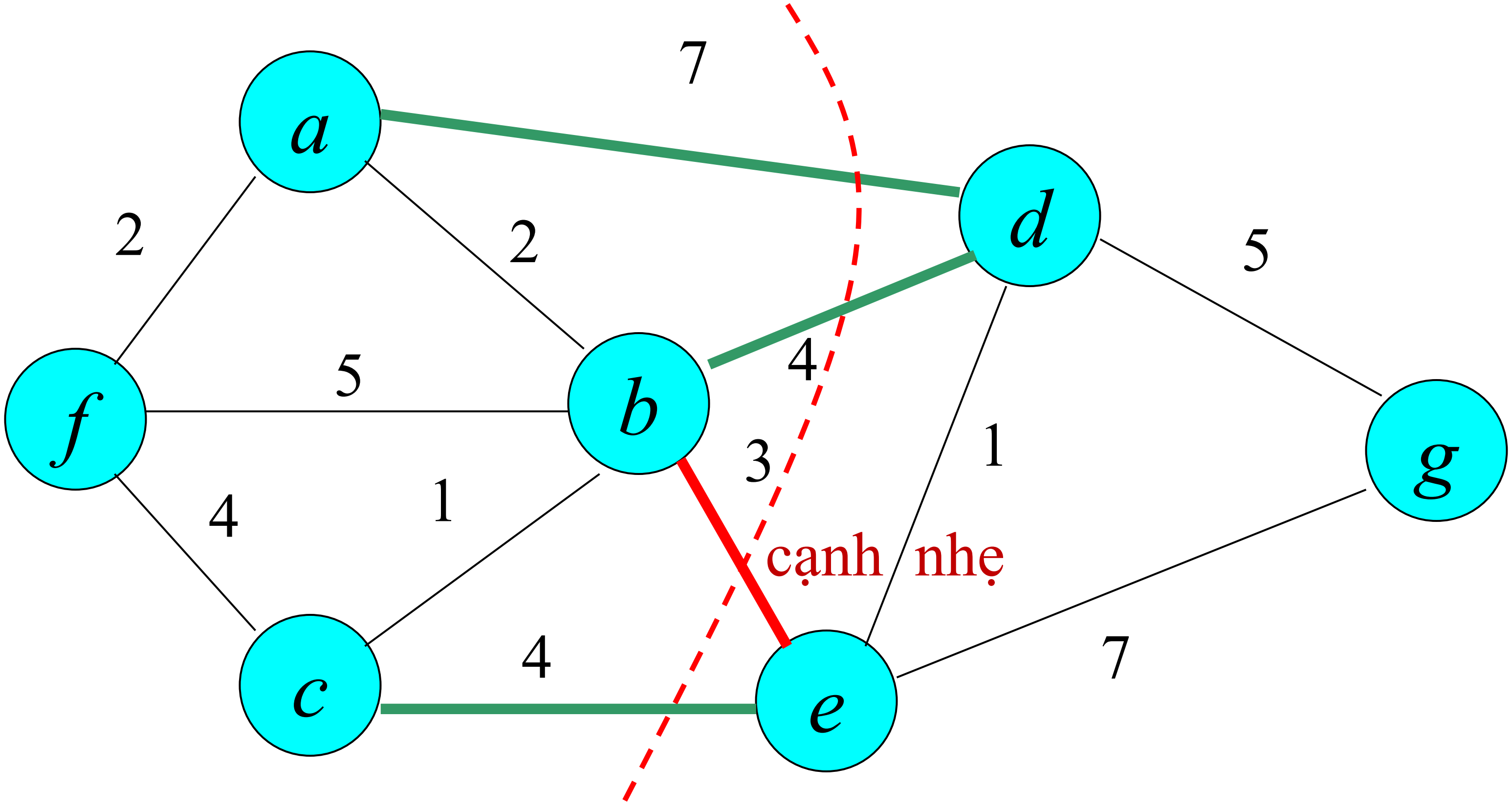
Lát cắt (S, $V - S$) là tương thích với A_1 vì

không tương thích với A_2 vì

Cạnh nhẹ

Cạnh nhẹ là cạnh có trọng số nhỏ nhất trong số các cạnh vượt lát cắt.

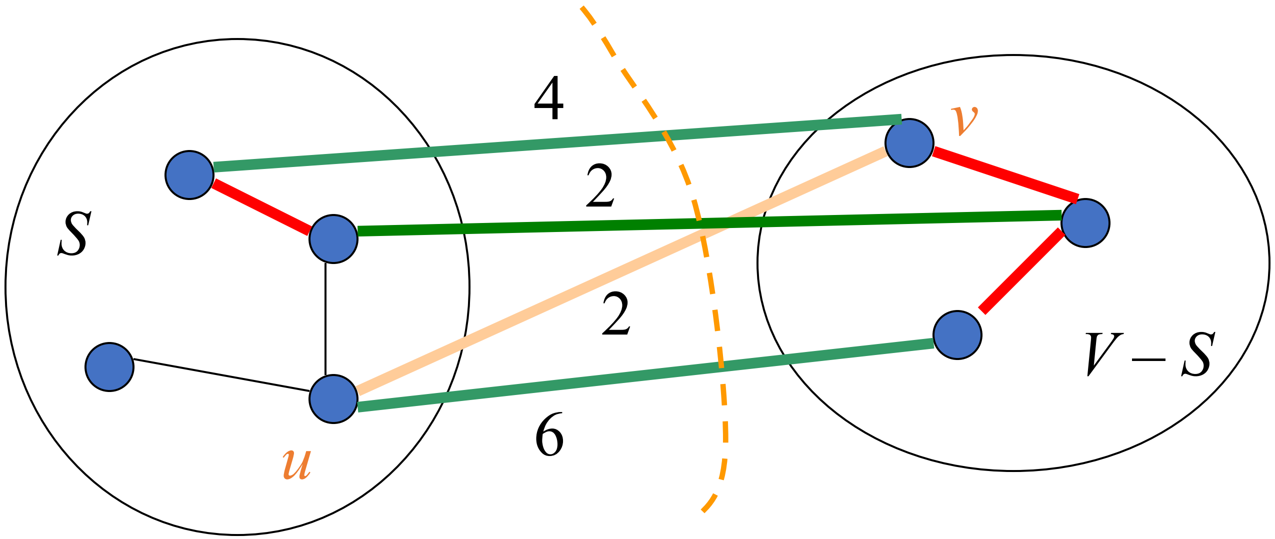
Ví dụ. Lát cắt (S, V-S): $S = \{a, b, c, f\}$



Cạnh (b, e) có trọng số 3, "nhẹ hơn" các cạnh vượt lát cắt còn lại (a, d), (b, d), và (c, e).

Cạnh nhẹ là an toàn

Định lý. Giả sử $(S, V - S)$ là lát cắt của $G=(V, E)$ tương thích với tập con A của E, và A là tập con của tập cạnh của CKNN của G.
Gọi (u, v) là cạnh nhẹ vượt lát cắt $(S, V - S)$. Khi đó (u, v) là ***an toàn*** đối với A; nghĩa là, $A \cup \{(u, v)\}$ cũng vẫn là tập con của tập cạnh của CKNN.



A gồm các cạnh đỏ.

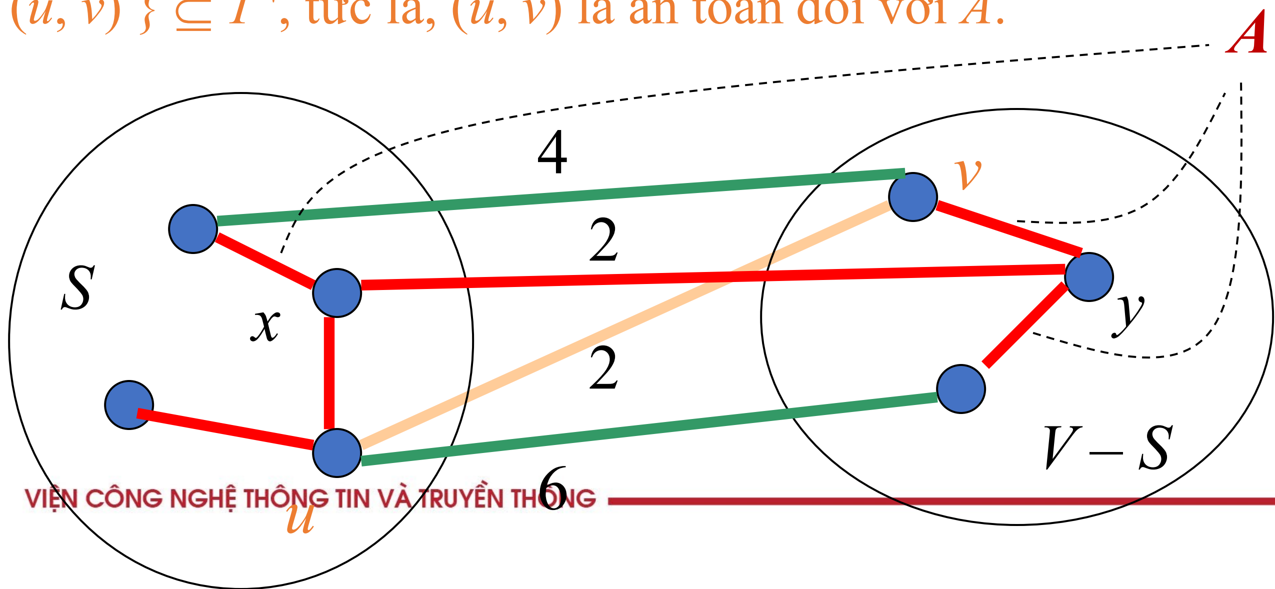
Tại sao cạnh nhẹ (u, v) là an toàn với A

Cần chứng minh. (u, v) là ***an toàn*** đối với A; nghĩa là: $A \cup \{(u, v)\}$ cũng vẫn là tập con của tập cạnh của CKNN.

Giả sử T là CKNN (gồm các cạnh đỏ) chứa A.

Giả sử cạnh nhẹ $(u, v) \notin T$. Ta có

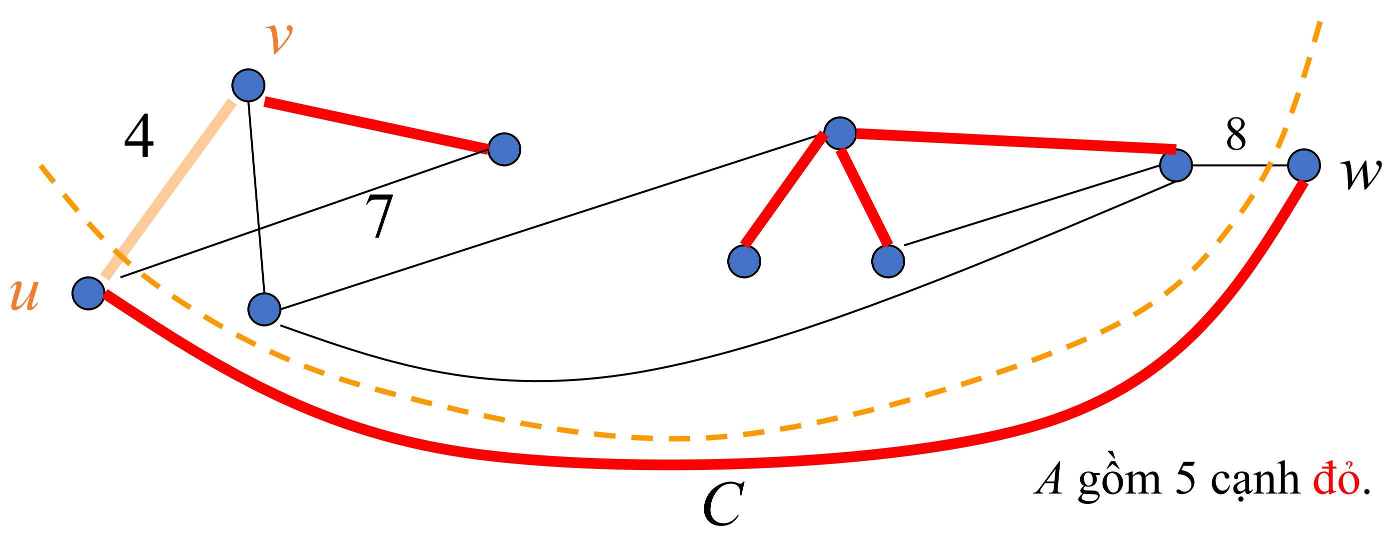
- $T \cup \{ (u, v) \}$ chứa chu trình.
- Tìm được cạnh $(x, y) \in T$ vượt lát cắt $(S, V - S)$.
- Cây khung $T' = T - \{ (x, y) \} \cup \{ (u, v) \}$ có độ dài $\le$ độ dài của cây khung T. Suy ra T' cũng là CKNN.
- $A \cup \{ (u, v) \} \subseteq T'$, tức là, (u, v) là an toàn đối với A.



Hệ quả Giả sử A là tập con của E và cũng là tập con của tập cạnh của CKNN nào đó của $G=(V,E)$, và C là một thành phần liên thông trong rừng $F = (V, A)$. Nếu (u, v) là cạnh nhẹ nối C với một thành phần liên thông khác trong F, thì (u, v) là an toàn đối với A.

Chứng minh

Cạnh (u, v) là cạnh nhẹ vượt lát cắt $(C, V - C)$ tương thích với A.
Theo định lý trên, cạnh (u, v) là an toàn đối với A.



A gồm 5 cạnh đỏ.

Tìm cạnh an toàn

Giả sử *T* là tập con của tập cạnh của một CKNN nào đó.

Thuật toán Kruskal

- *T* là **rừng.**
- Cạnh an toàn được bổ sung vào *T* có *trọng số nhỏ nhất* trong số các cạnh nối các cặp thành phần liên thông của nó.

Thuật toán Prim

- *T* là **cây**.
- Cạnh an toàn là cạnh có trọng số nhỏ nhất nối đỉnh trong *T* với một đỉnh *không ở trong T*.

Thuật toán Kruskal

Generic-MST(G, c)

$T = \varnothing$

//Bất biến: T là tập con các cạnh của CKNN nào đó

while T chưa là cây khung **do**

tìm cạnh (u, v) là an toàn đối với T

$T = T \cup \{(u, v)\}$

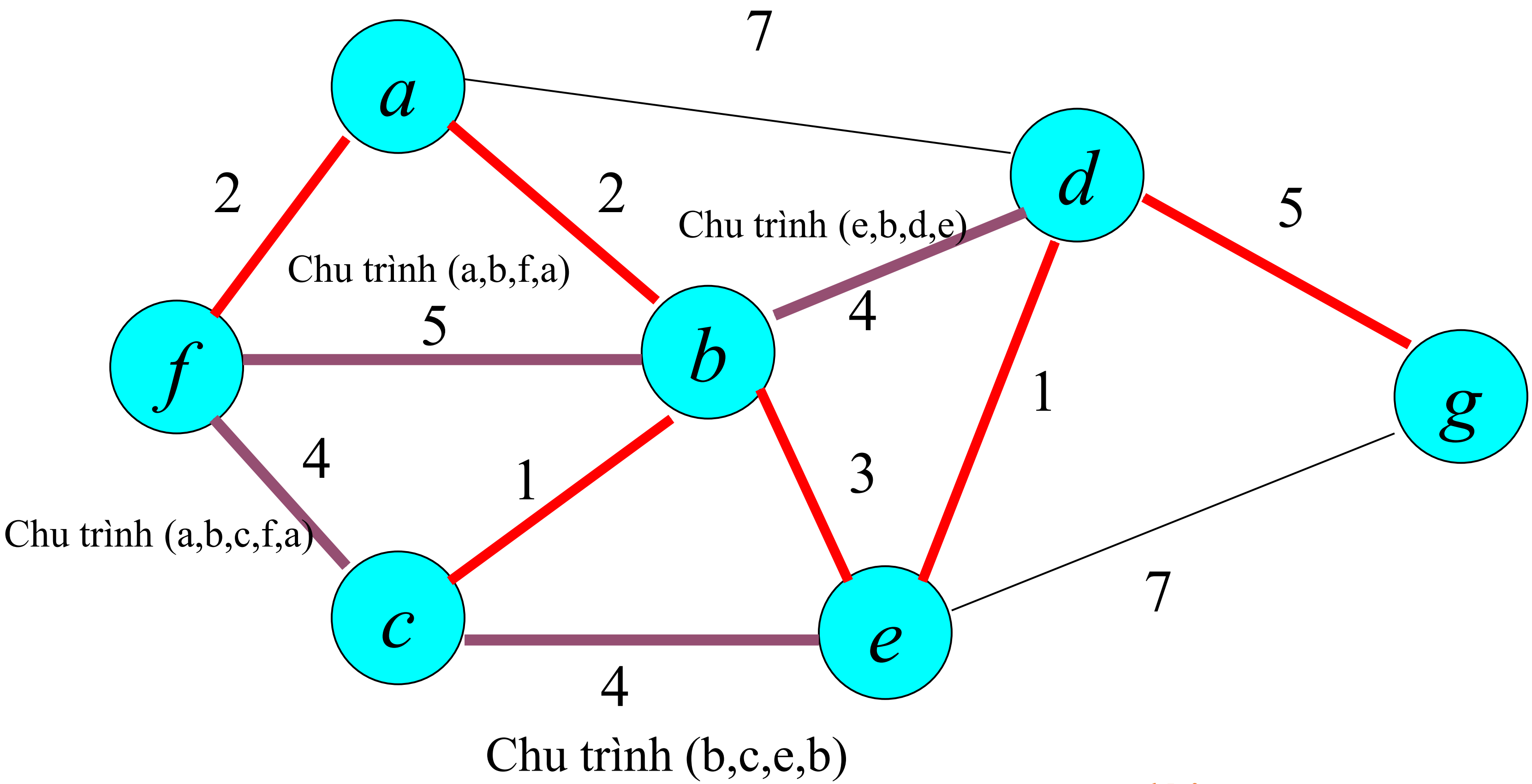
// T vẫn là tập con các cạnh của CKNN nào đó

return T

Thuật toán Kruskal

- ☆ T là **rừng (bắt đầu từ rừng rỗng).**
- ☆ Cạnh an toàn được bổ sung vào T có *trọng số nhỏ nhất* trong số các cạnh nối các cặp thành phần liên thông của nó.

Thuật toán Kruskal – Ví dụ



Độ dài của CKNN: 14

Mô tả thuật toán Kruskal

```
void Kruskal ( )
{
    Sắp xếp m cạnh của đồ thị e1, . . . , em theo thứ tự tăng dần của độ dài;
    T = ∅;      // T: tập cạnh của CKNN
    for (i = 1; i <=m; i++)
        if (T ∪ {ei} không chứa chu trình) T = T ∪ {ei};
}
```

Thời gian tính

void Kruskal ( )

{

Sắp xếp m cạnh của đồ thị $e_1, \ldots, e_m$ theo thứ tự tăng dần của độ dài;

$T = \varnothing$; // T: tập cạnh của CKNN

for (i = 1; i <=m; i++)

if ($T \cup \{e_i\}$ không chứa chu trình) $T = T \cup \{e_i\}$;

}

- Bước 1. Sắp xếp dãy độ dài cạnh.
  - Có thể sử dụng sắp xếp vun đống, hoặc sắp xếp trộn mất thời gian $O(m \log m)$
- Bước lặp: Mỗi lần lặp cần xác định xem $T \cup \{ e_i \}$ có chứa chu trình hay không?
  - Có thể sử dụng DFS để kiểm tra với thời gian $O(m+n)$.
  - Thời gian $O(m(m+n))$

Tổng cộng: $O(m \log m + m(m+n))$

với n, m lần lượt là số đỉnh và cạnh của đồ thị

Thuật toán PRIM

Generic-MST(G, c)

$T = \varnothing$

//Bất biến: *T* là tập con các cạnh của CKNN nào đó

while *T* chưa là cây khung do

tìm cạnh (u, v) là an toàn đối với *T*

$T = T \cup \{(u, v)\}$

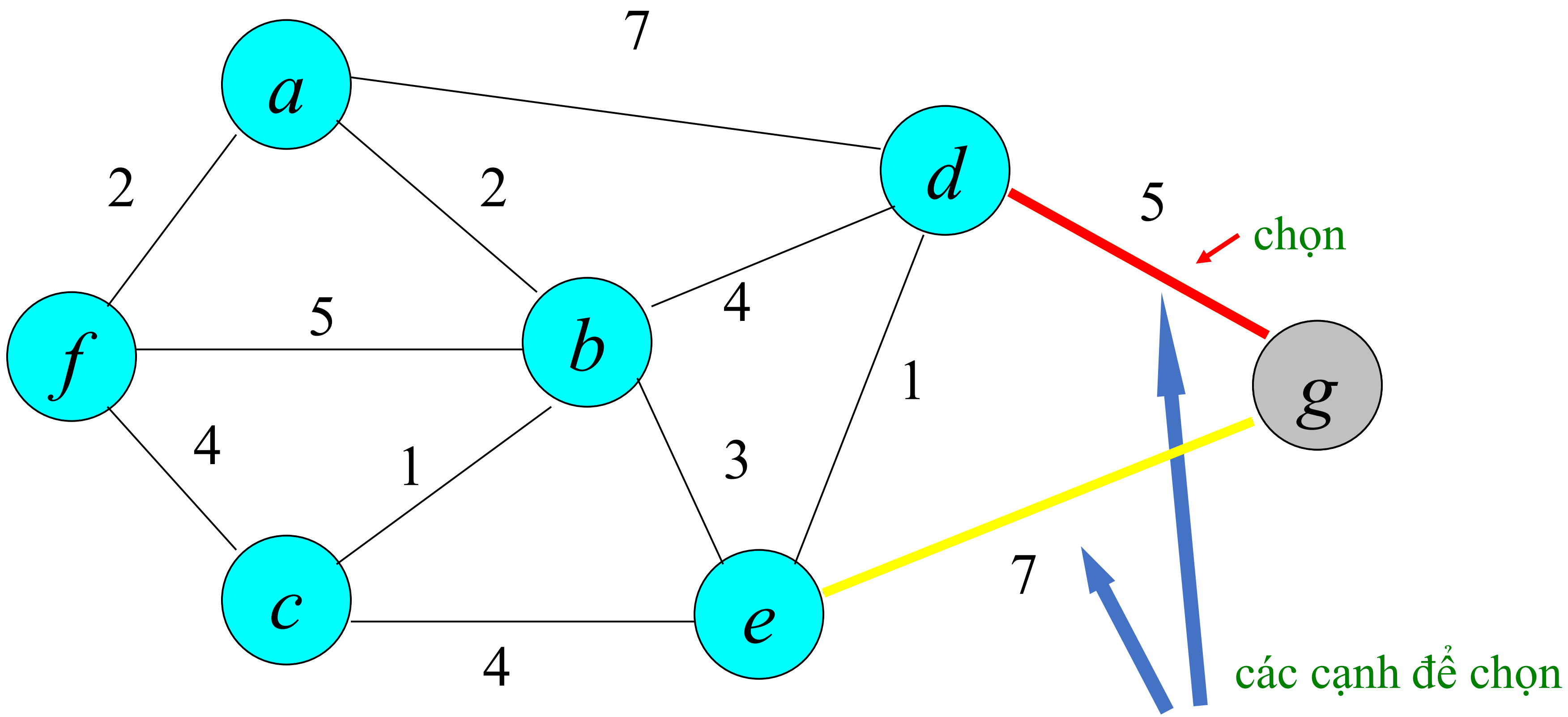
// *T* vẫn là tập con các cạnh của CKNN nào đó

return *T*

Thuật toán PRIM

- *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)
- Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

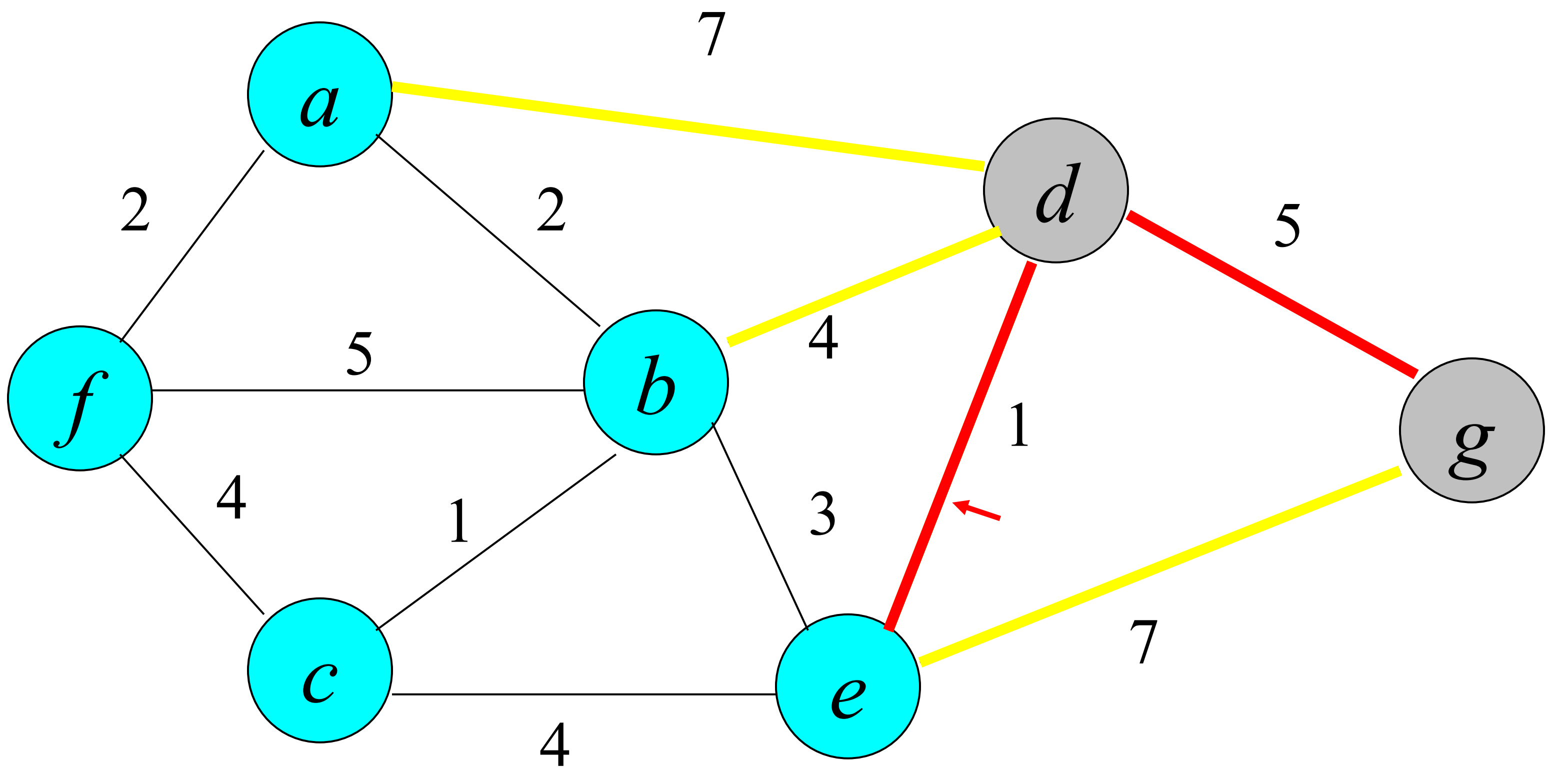
Thuật toán PRIM – Ví dụ



★ *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)

★ Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

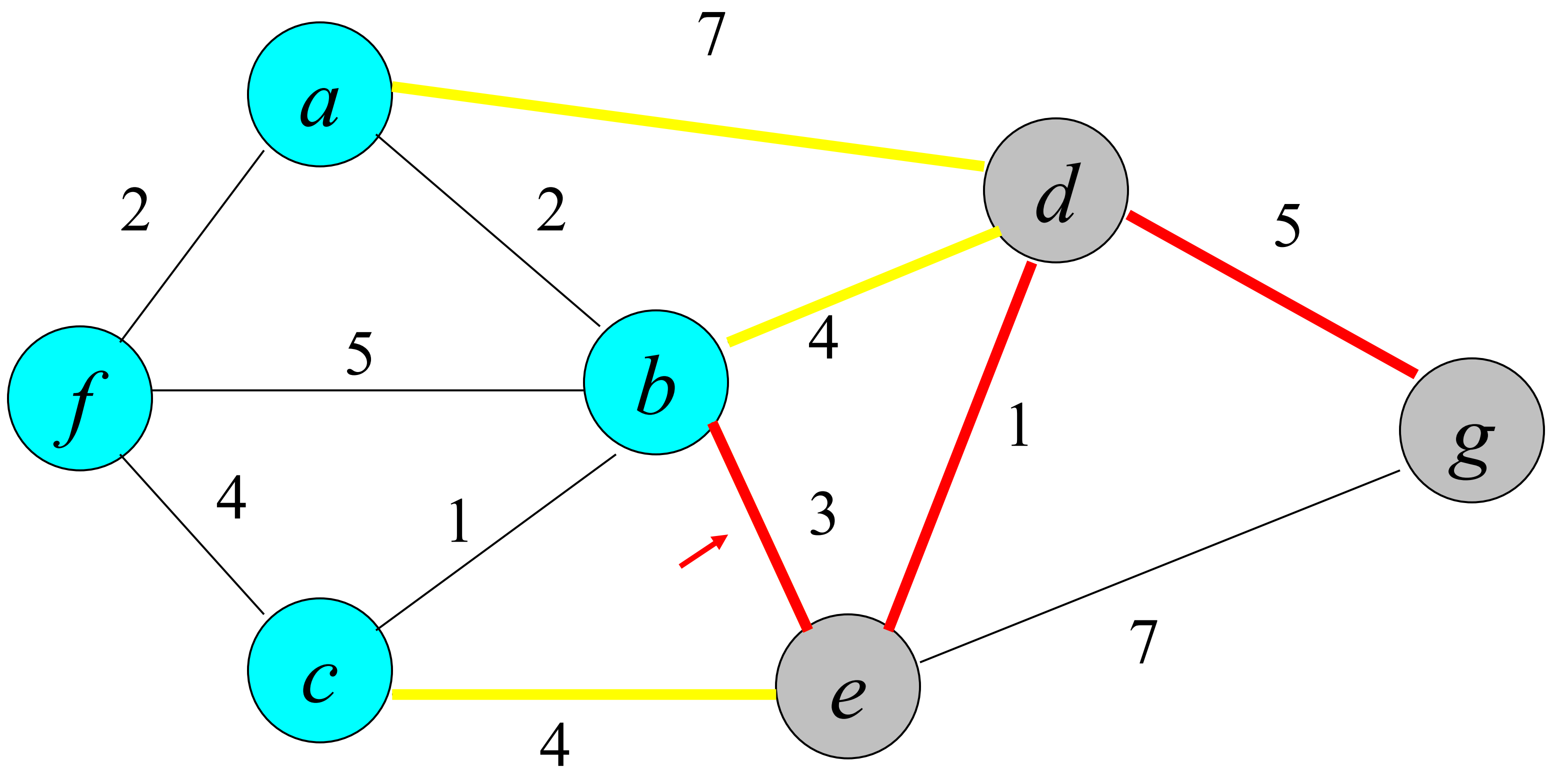
Thuật toán PRIM – Ví dụ



★ *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)

★ Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

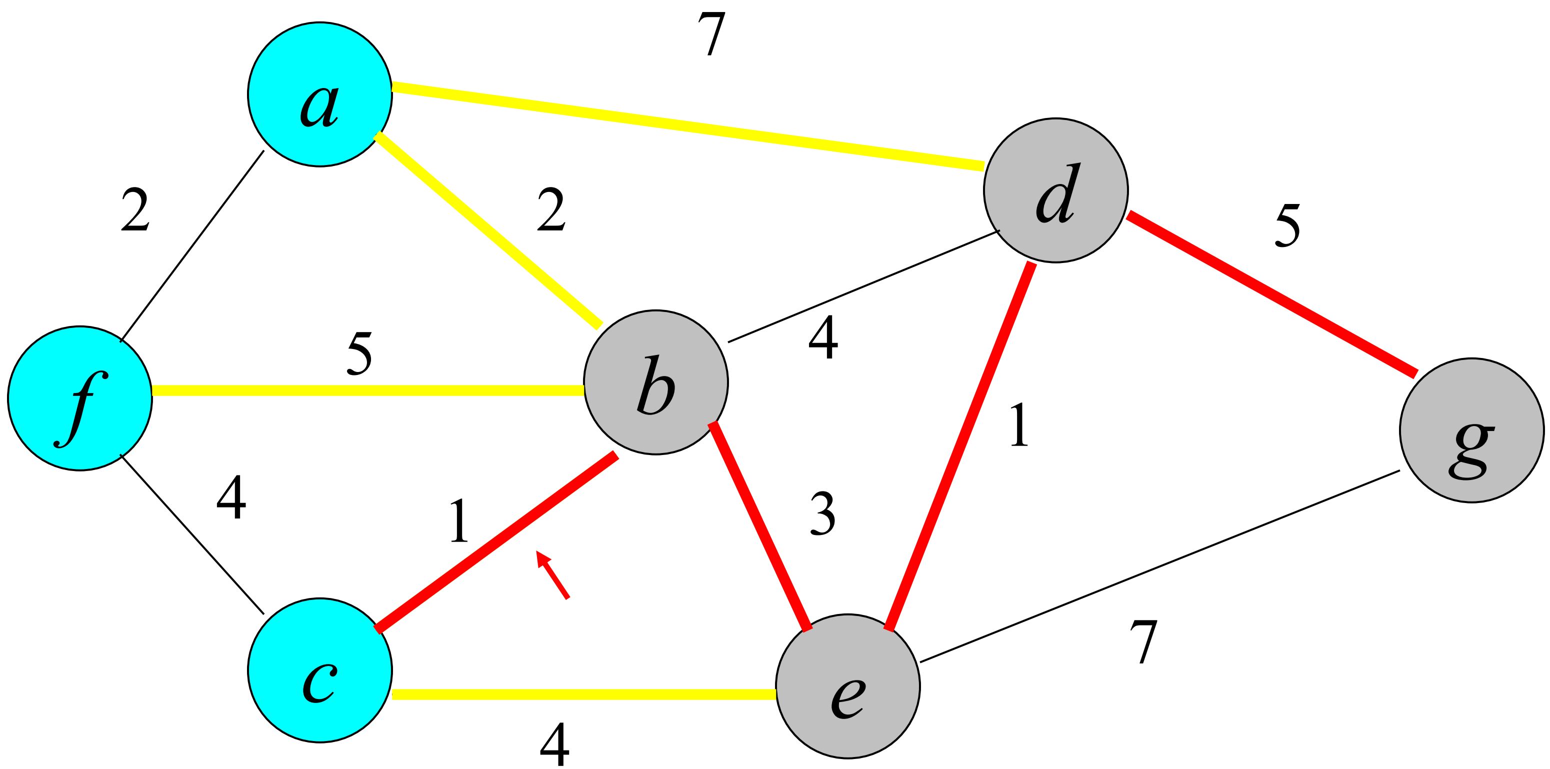
Thuật toán PRIM – Ví dụ



★ *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)

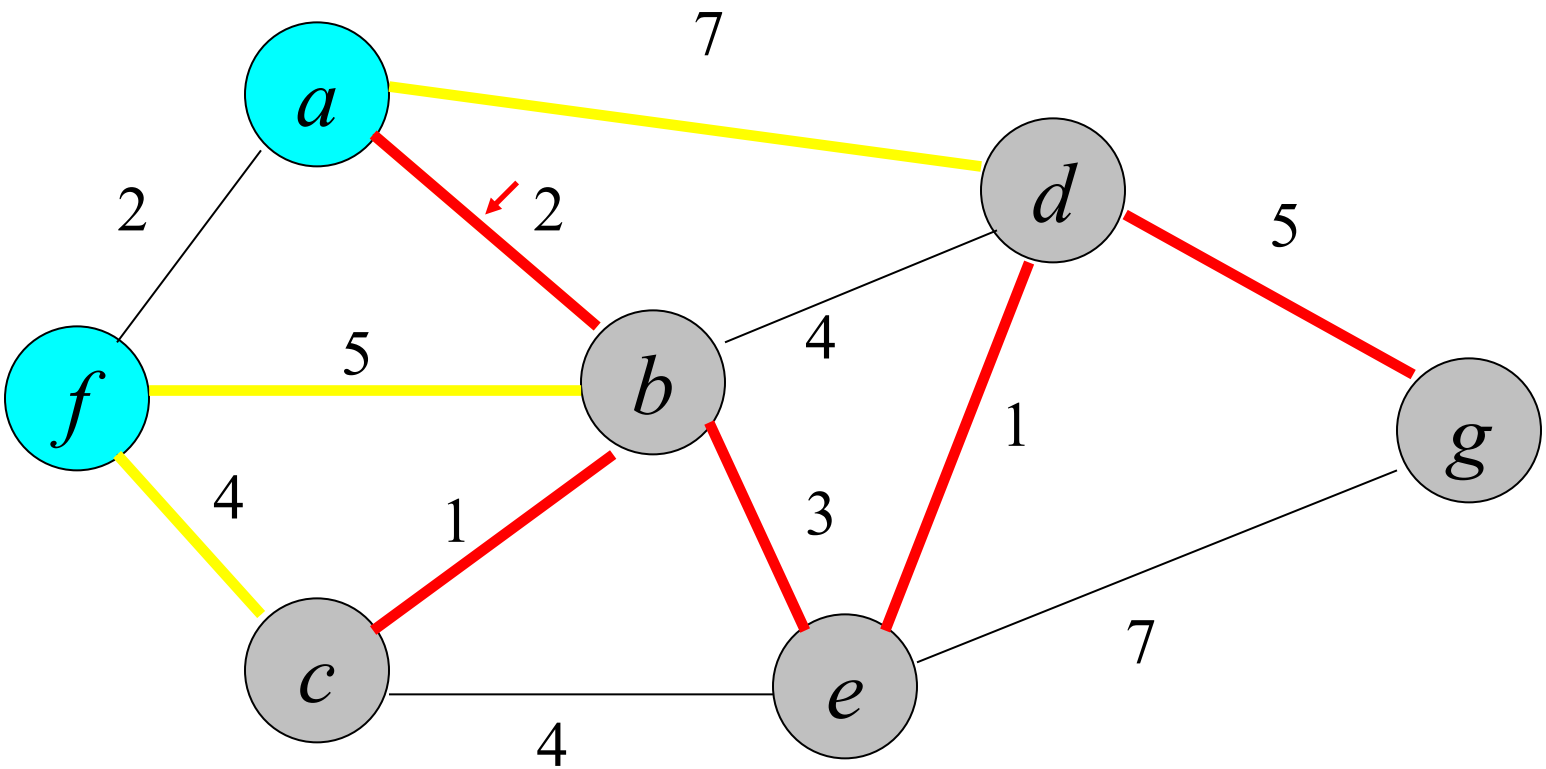
★ Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

Thuật toán PRIM – Ví dụ



★ *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)
★ Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

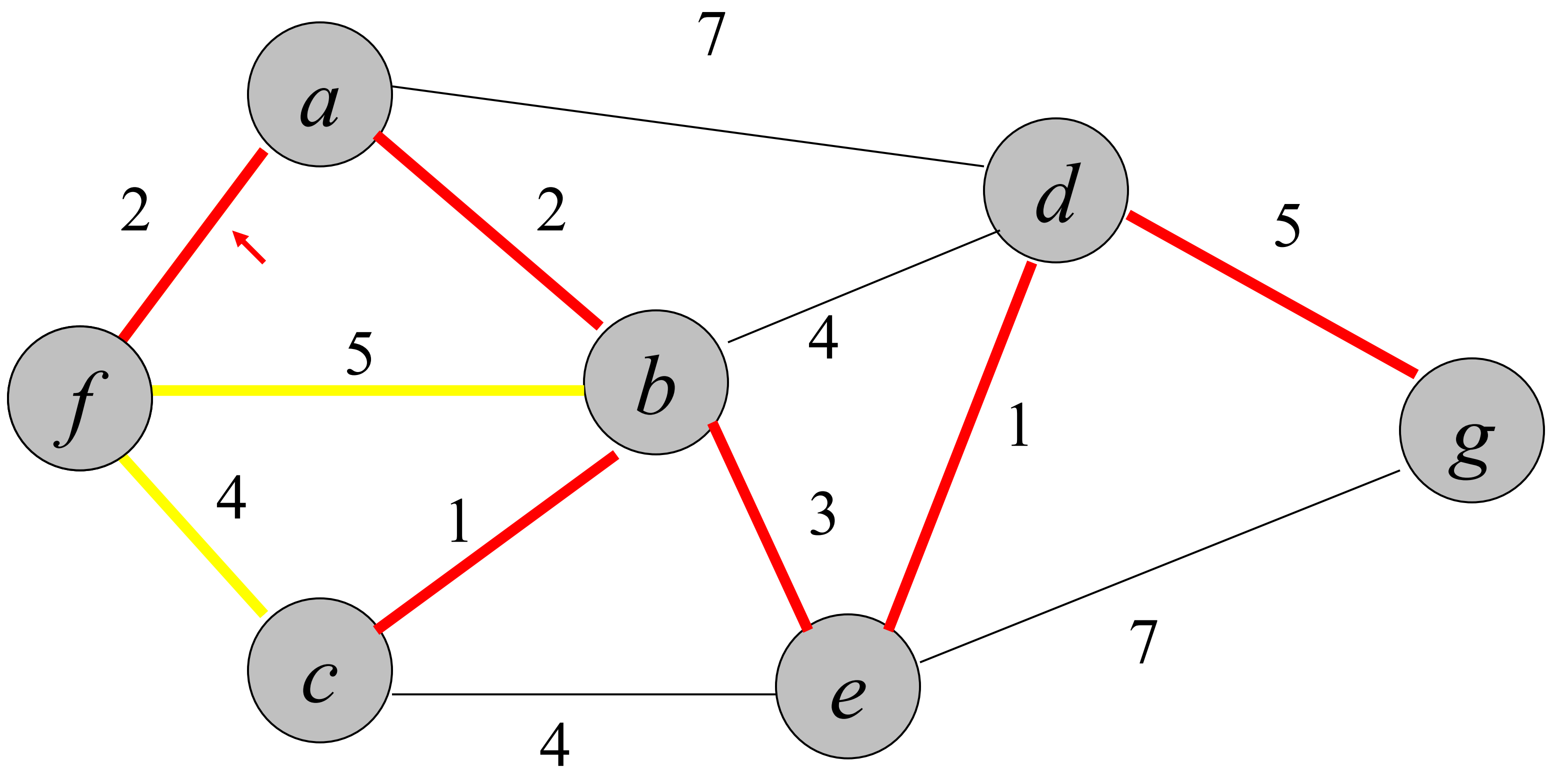
Thuật toán PRIM – Ví dụ



☆ *T* là **cây** (bắt đầu từ cây chỉ có 1 đỉnh)

☆ Cạnh an toàn được bổ sung vào *T* là cạnh nhẹ nhất trong số các cạnh nối đỉnh trong *T* với một đỉnh *không ở trong* *T*.

Thuật toán PRIM – Ví dụ



CKNN gồm các cạnh: (*g*,*d*), (*d*,*e*), (*e*,*b*), (*b*,*c*), (*b*,*a*), (*a*,*f*)

Độ dài của CKNN: 14

$5+1+3+1+2+2 = 14$

Mô tả thuật toán PRIM

```
void  Prim(G, C) //G: đồ thị; C: ma trận trọng số biểu diễn trọng số các cạnh trên đồ thị
{
    Chọn đỉnh tuỳ ý r ∈V;
    Khởi tạo cây T=(V(T), E(T)) với V(T)= {r}và E(T)=∅;
    while (T có  < n đỉnh )
    {
        Gọi (u, v) là cạnh nhẹ nhất với u ∈ V(T) và  v∈V(G) – V(T)
        E(T) ← E(T) ∪ { (u, v) };
        V(T) ← V(T) ∪ { v }
    }
}
```

Cây T chỉ có 1 đỉnh duy nhất r

Trong số các cạnh nối 1 đỉnh thuộc T với 1 đỉnh không thuộc T, *tìm cạnh có trọng số nhỏ nhất*

bổ sung đỉnh v và cạnh (u,v) vào cây T

Tính đúng đắn suy từ hệ quả đã chứng minh:

Giả sử A là tập con của E và cũng là tập con của tập cạnh của CKNN của G, và C là một thành phần liên thông trong rừng $F = (V, A)$. Nếu (u, v) là cạnh nhẹ nối C với một tplt khác trong F, thì (u, v) là an toàn đối với A.

Cài đặt thuật toán PRIM

- Giả sử đồ thị cho bởi ma trận trọng số $C=\{c[i,j],\ i, j = 1, 2,..., n\}$.
- 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 $v \in V \setminus V(T)$ có dạng $[d[v], near[v]]$:

- $d[v]$ dùng để ghi nhận khoảng cách từ đỉnh v đến tập đỉnh $V(T)$:

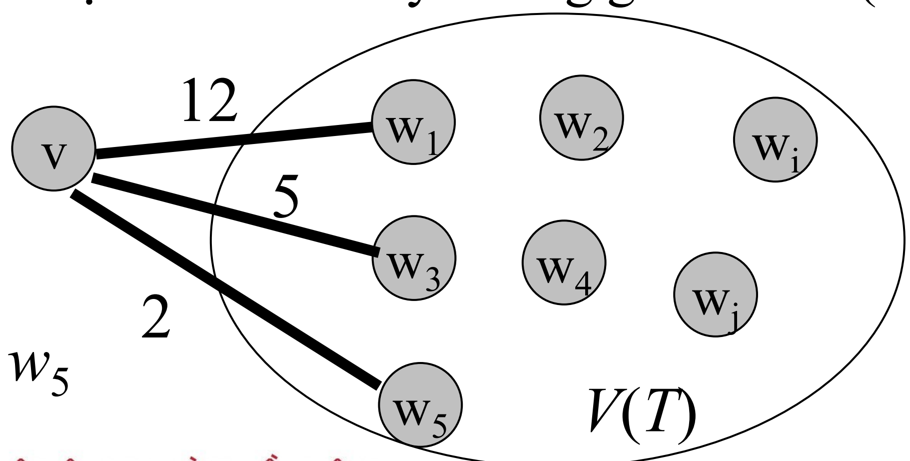
$$\boxed{d[v] := \min\{\ c[v, w] :\ w \in V(T)\ \}\ (= c[v, z])}$$

Cạnh có trọng số nhỏ nhất trong số các cạnh nối v với 1 đỉnh trong tập $V(T)$

- $near[v] := z$ ghi nhận đỉnh của cây khung gần v nhất (tức là $c[v, z] = d[v]$)

$d(v) = 2$

$near[v] = w_5$



```
void Prim(G, C) //G: đồ thị; C: ma trận trọng số biểu diễn trọng số các cạnh trên đồ thị
{
    Chọn đỉnh tuỳ ý r ∈V;
    Khởi tạo cây T=(V(T), E(T)) với V(T)= {r}và E(T)=∅;
    while (T có < n đỉnh )
    {
        Gọi (u, v) là cạnh nhẹ nhất với u ∈ V(T) và v∈V(G) – V(T)
        E(T) ← E(T) ∪ { (u, v) };
        V(T) ← V(T) ∪ { v }
    }
}
```

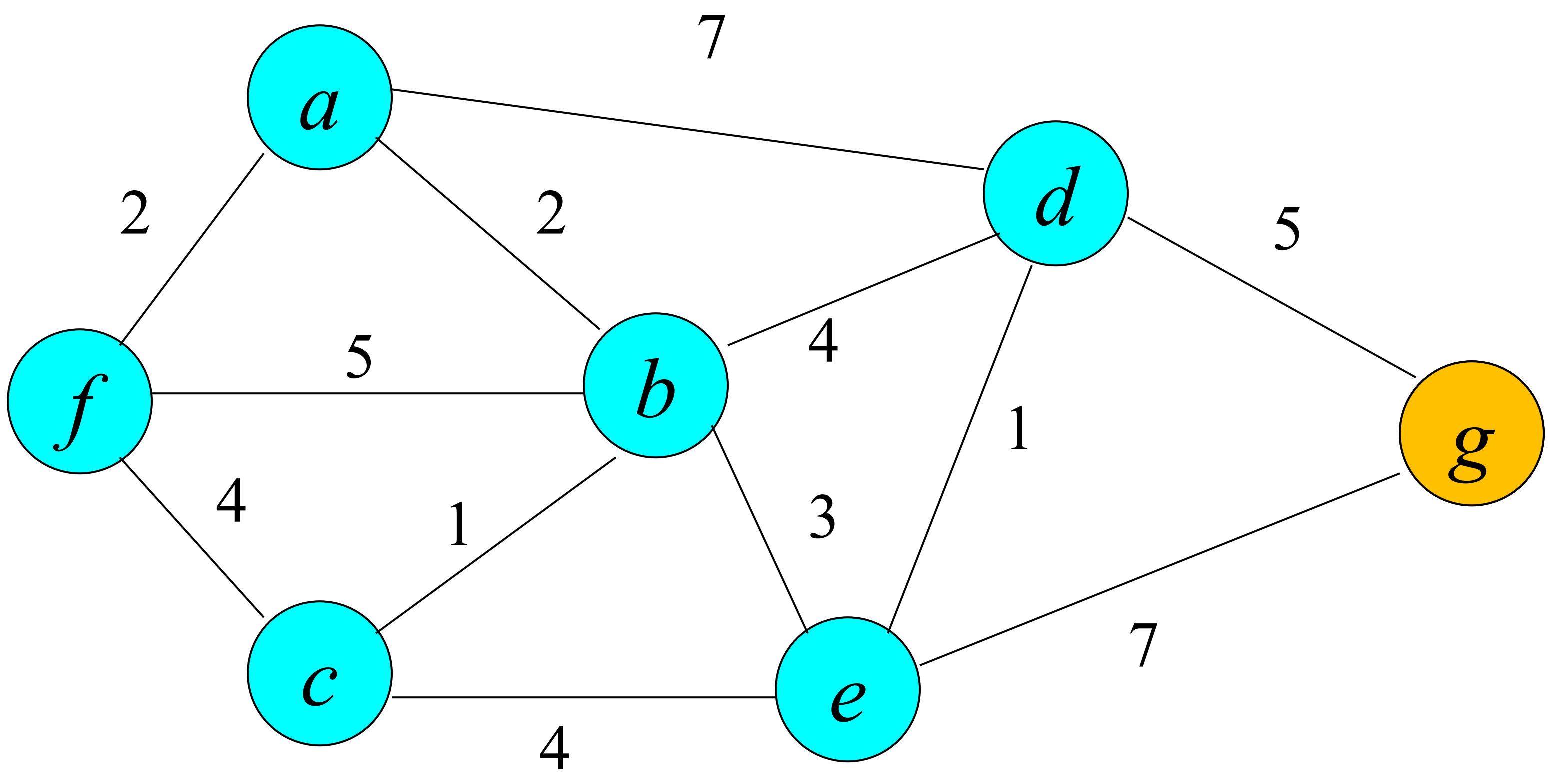
```
void Prim ( ) {
    // Bước khởi tạo:
    V(T) = { r }; E(T) = ∅ ;
    d[r] = 0; near[r] = r;
    for v ∈ V \ V(T)
    {
        d[v] = c[r,v]; near[v] = r;
    }
    // Bước lặp:
    for (k=2; k <= n; k++)
    {
        Tìm v ∈ V \ V(T) thoả mãn: d[v] = min { d[i] : i ∈ V \ V(T) };
        V(T) = V(T) ∪ { v }; E(T) = E(T) ∪ { ( v, near[v] ) } ;
        for v'∈ V \ V(T)
            if (d[v'] > c[v,v'])
            {
                d[v'] = c[v,v'] ; near[v'] = v;
            }
    }
    T là cây khung nhỏ nhất của đồ thị ;
}
```

Chuẩn bị dữ liệu cho việc tìm cạnh an toàn
d[v]: trọng số cạnh nhẹ nhất nối v *(chưa thuộc cây khung T)* với một đỉnh trong cây khung T

Vì cây khung T có sự thay đổi: đỉnh v vừa được bổ sung vào cây khung T
→ cập nhật lại nhãn của các đỉnh chưa được bổ sung vào cây T nếu cần thiết

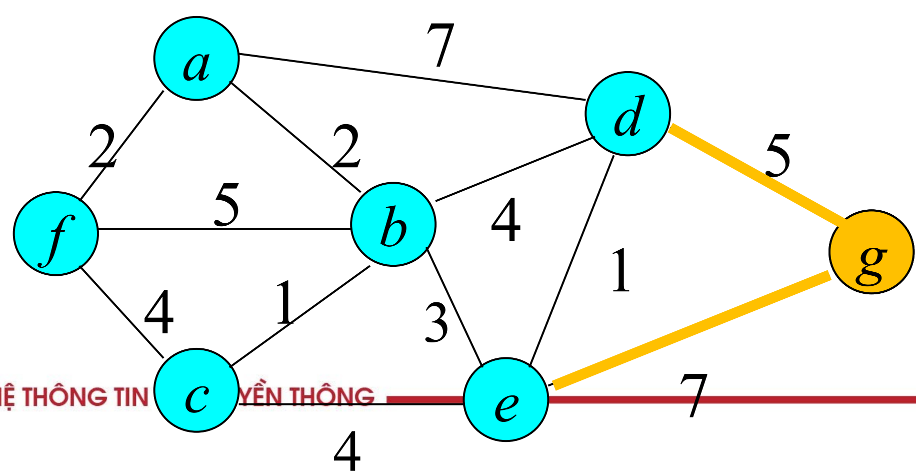
Thời gian tính: $O(|V|^2)$

Thuật toán PRIM: Ví dụ



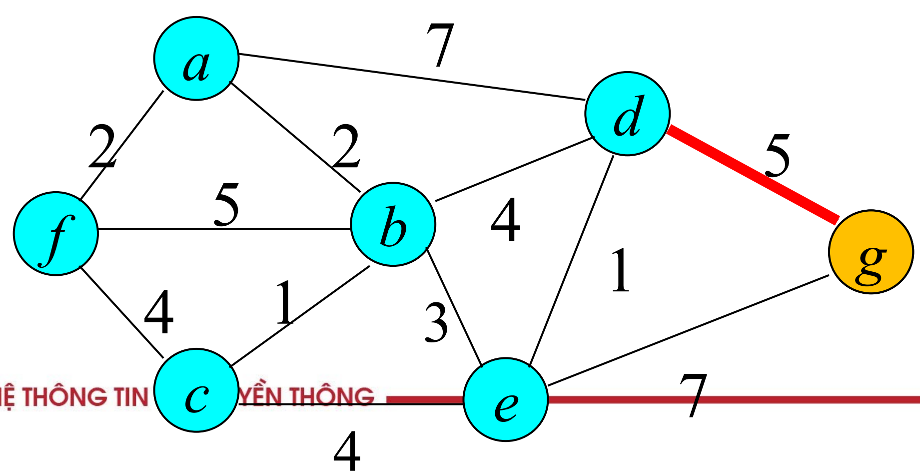
Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	*V*(*T*)
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $\mathrm{d}[v] = \min \{ \mathrm{d}[i] : i \in V \setminus V(T) \}$;



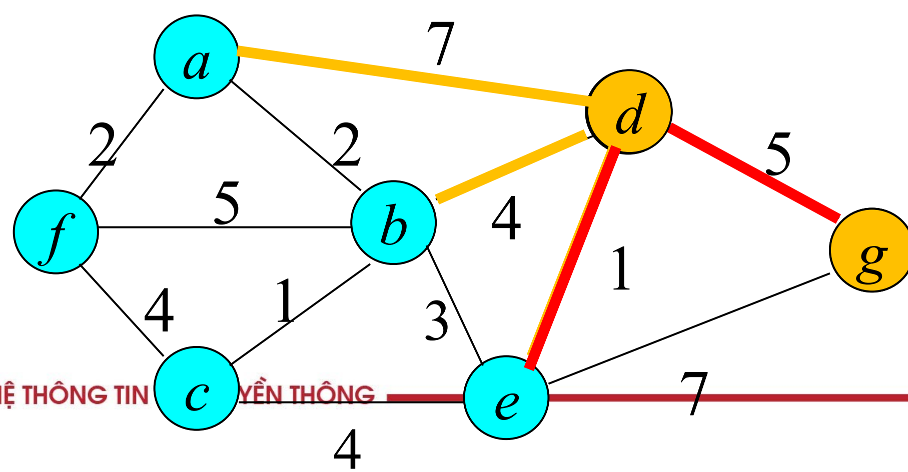
Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	*V*(*T*)
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $d[v] = \min \{ d[i] : i \in V \setminus V(T) \}$;



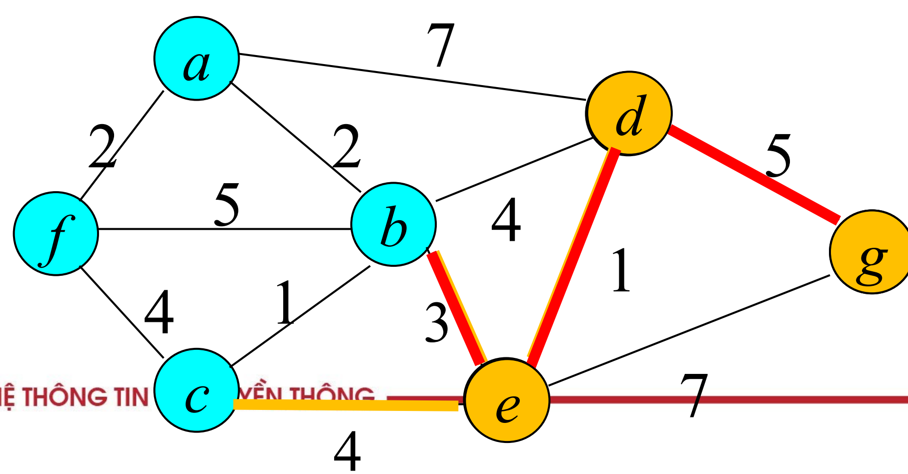
Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	$V(T)$
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g
[7, d]	[4, d]	[∞, g]	-	[1, d]	[∞, g]	-	g, d

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $\mathrm{d}[v] = \min \{ \mathrm{d}[i] : i \in V \setminus V(T) \}$;



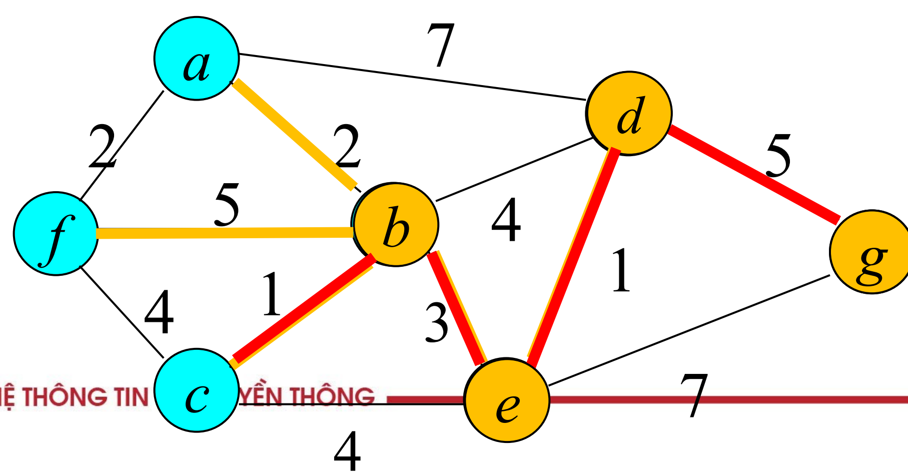
Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	*V*(*T*)
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g
[7, d]	[4, d]	[∞, g]	-	[1, d]	[∞, g]	-	g, d
[7, d]	[3, e]	[4, e]	-	-	[∞, g]	-	g, d, e

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $\mathrm{d}[v] = \min \{ \mathrm{d}[i] : i \in V \setminus V(T) \}$;



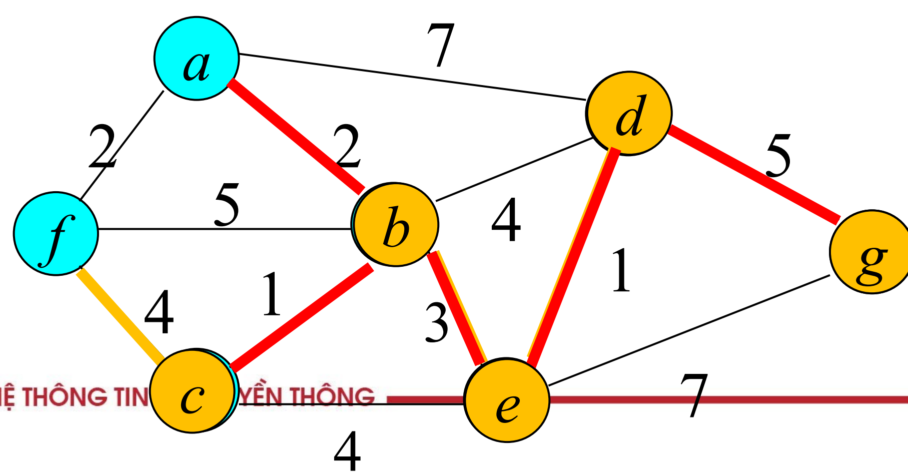
Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	$V(T)$
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g
[7, d]	[4, d]	[∞, g]	-	[1, d]	[∞, g]	-	g, d
[7, d]	[3, e]	[4, e]	-	-	[∞, g]	-	g, d, e
[2, b]	-	[1, b]	-	-	[5, b]	-	g, d, e, b

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $d[v] = \min \{ d[i] : i \in V \setminus V(T) \}$;



Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	*V*(*T*)
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g
[7, d]	[4, d]	[∞, g]	-	[1, d]	[∞, g]	-	g, d
[7, d]	[3, e]	[4, e]	-	-	[∞, g]	-	g, d, e
[2, b]	-	[1, b]	-	-	[5, b]	-	g, d, e, b
[2, b]	-	-	-	-	[4, c]	-	g, d, e, b, c

Tìm $v \in V \setminus V(T)$ *thoả mãn*: $d[v] = \min \{ d[i] : i \in V \setminus V(T) \}$;



Đỉnh a	Đỉnh b	Đỉnh c	Đỉnh d	Đỉnh e	Đỉnh f	Đỉnh g	$V(T)$
[∞, g]	[∞, g]	[∞, g]	[5, g]	[7, g]	[∞, g]	[0, g]	g
[7, d]	[4, d]	[∞, g]	-	[1, d]	[∞, g]	-	g, d
[7, d]	[3, e]	[4, e]	-	-	[∞, g]	-	g, d, e
[2, b]	-	[1, b]	-	-	[5, b]	-	g, d, e, b
[2, b]	-	-	-	-	[4, c]	-	g, d, e, b, c
-	-	-	-	-	[2, a]	-	g, d, e, b, c, a
							g, d, e, b, c, a, f

Tập cạnh của CKNN:

Độ dài của CKNN : 14