Cây bao trùm nhỏ nhất (Đang sửa V0.1)

Trần Vĩnh Đức

HUST

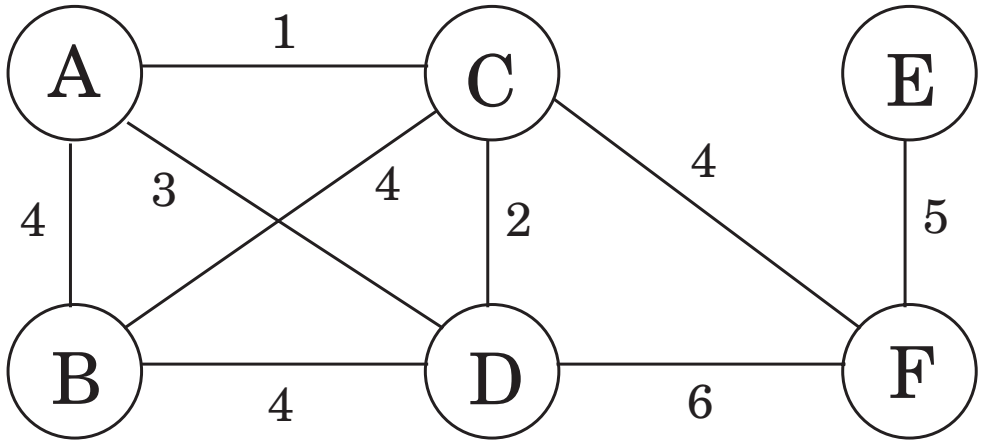
Ngày 7 tháng 9 năm 2017

Tài liệu tham khảo

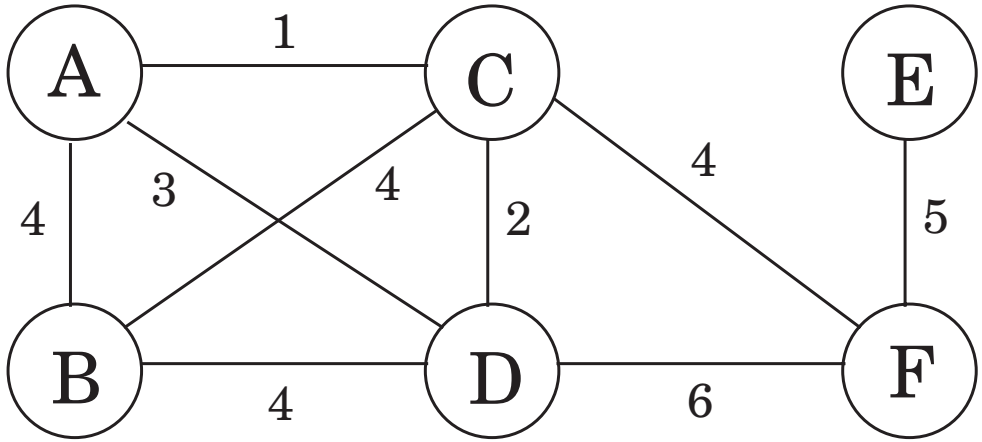
- S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, July 18, 2016.
- Chú ý: Nhiều hình vẽ trong tài liệu được lấy tùy tiện mà chưa xin phép.

Nội dung

Bài toán



- Bạn cần xây dựng mạng máy tính bằng cách kết nối từng cặp máy.
- Cần chọn một số kết nối để mạng liên thông;
- nhưng không phải tất cả các cặp: Mỗi kết nối tốn một chi phí (tiền bảo trì).
- Mạng với chi phí nhỏ nhất là gì?



Tính chất

Xóa một cạnh trên chu trình không làm mất tính liên thông của đồ thị.

Vậy, mạng với chi phí nhỏ nhất phải là một cây.

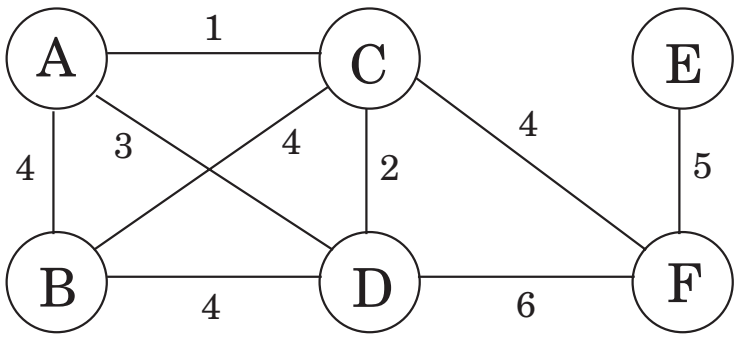
Bài toán Cây bao trùm nhỏ nhất (Minimal Spaning Tree)

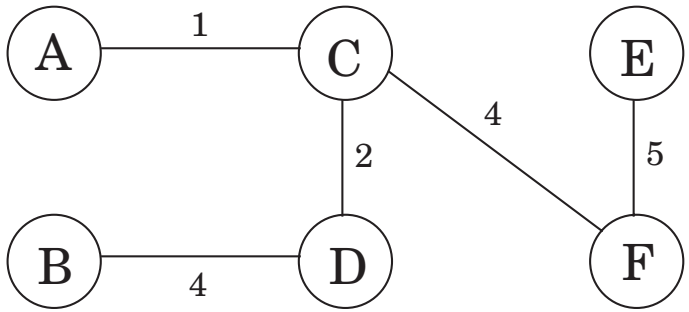
- *Input:* Đồ thị vô hướng $G = (V, E)$; mỗi cạnh có trọng số w_e.
- *Output:* Một cây $T = (V, E')$ với $E' \subseteq E$, với tổng trọng số

$$\text{weight}(T) = \sum_{e \in E'} w_e$$

là nhỏ nhất.

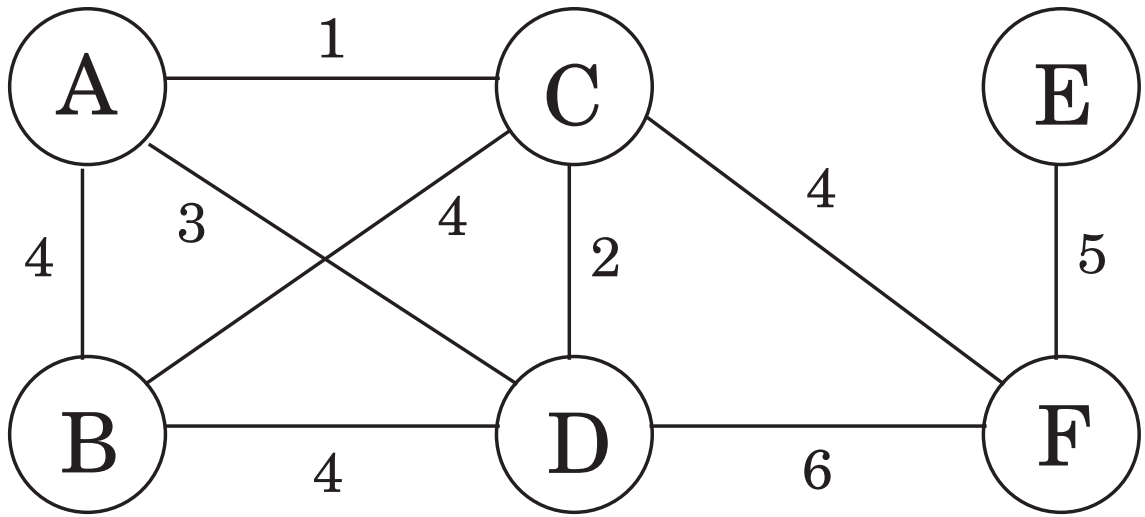
Tìm cây bao trùm





Đây có phải lời giải tối ưu không?

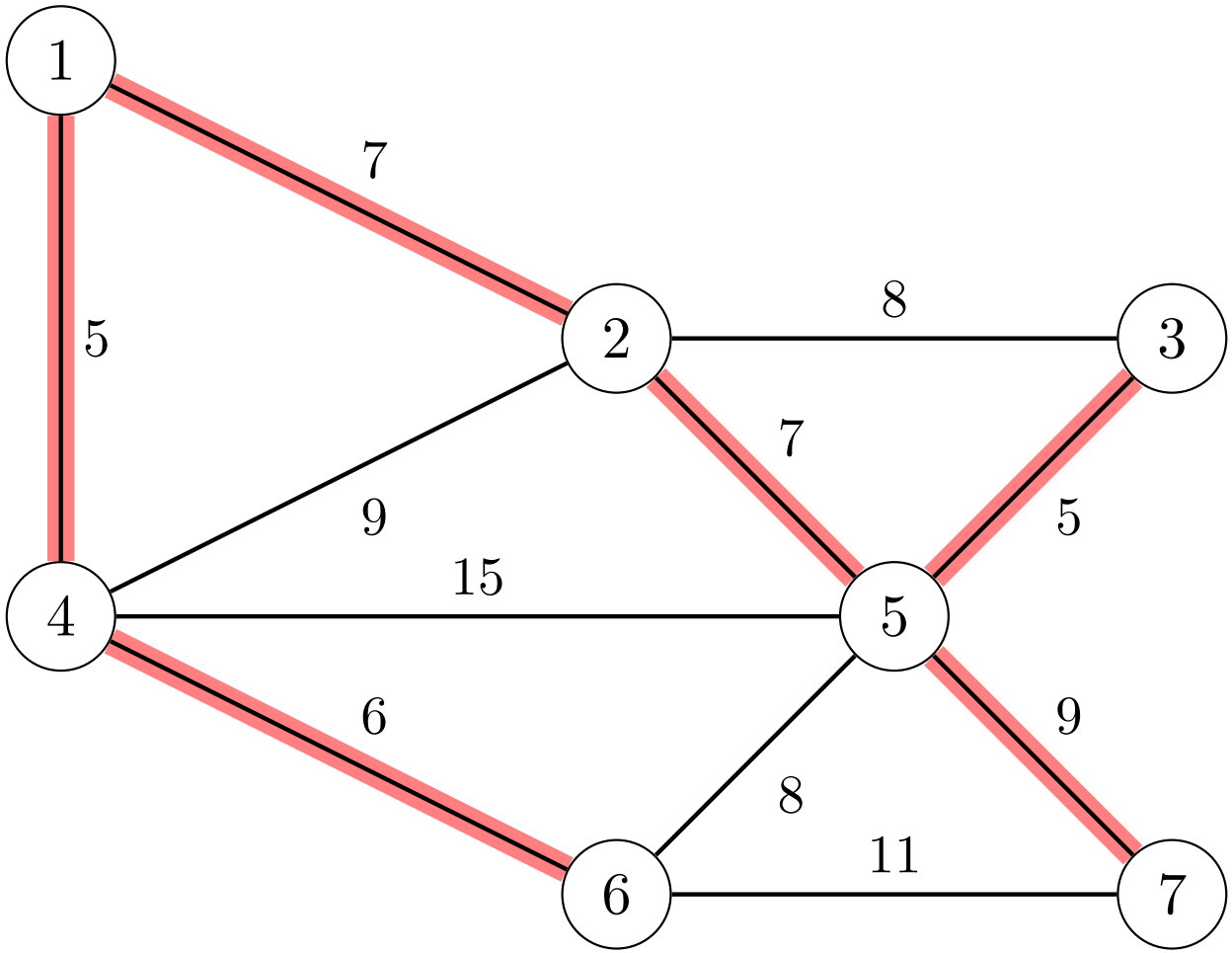
Thuật toán Kruskal



Bắt đầu với đồ thị rỗng và chọn cạnh từ E theo quy tắc sau.

Lặp lại việc thêm cạnh nhỏ nhất mà không tạo thành chu trình.

Ví dụ: Thuật toán Kruskal

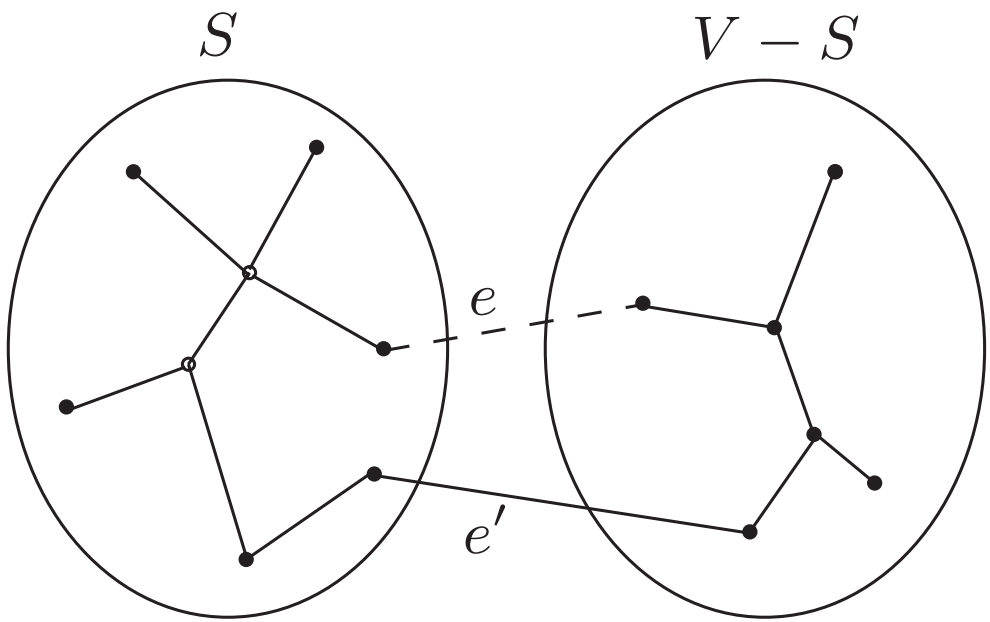


Hình: Nguồn: tikz examples

Nhát cắt

Định nghĩa

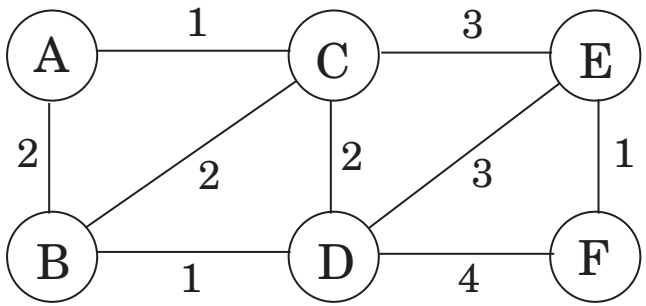
Xét đồ thị $G = (V, E)$. Một nhát cắt là một cách chia tập đỉnh thành hai nhóm: S và $V - S$.

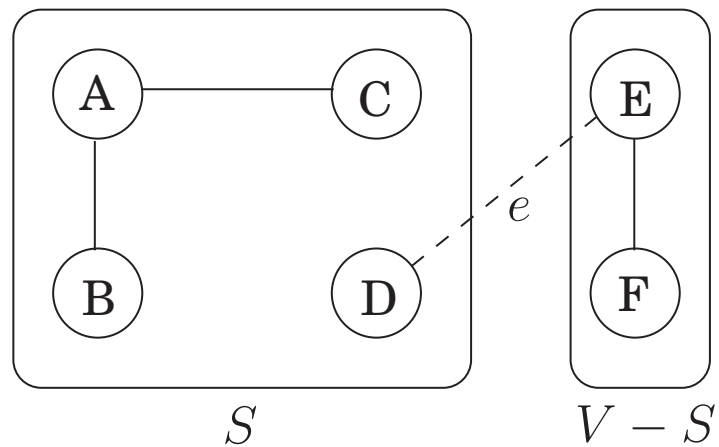


Hình: Nhát cắt và các cạnh nối giữa hai phân hoạch.

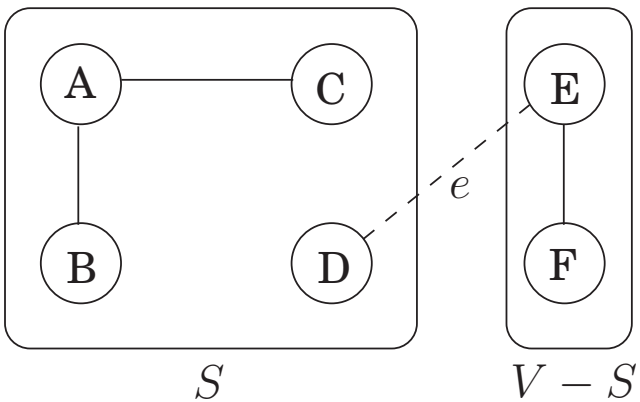
Tính chất Cắt

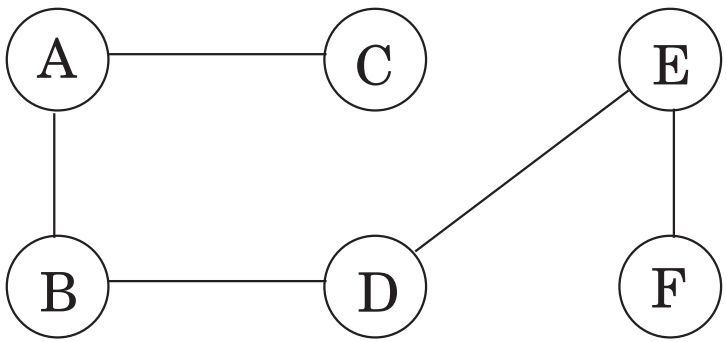
Giả sử các cạnh X là một phần của một MST nào đó của $G = (V, E)$. Chọn một tập đỉnh bất kỳ S sao cho không có cạnh nào của X nối giữa S và $V - S$, và xét e là cạnh có trọng số nhỏ nhất nối hai phân hoạch này. Khi đó, $X \cup \{e\}$ là một phần của một MST nào đó.





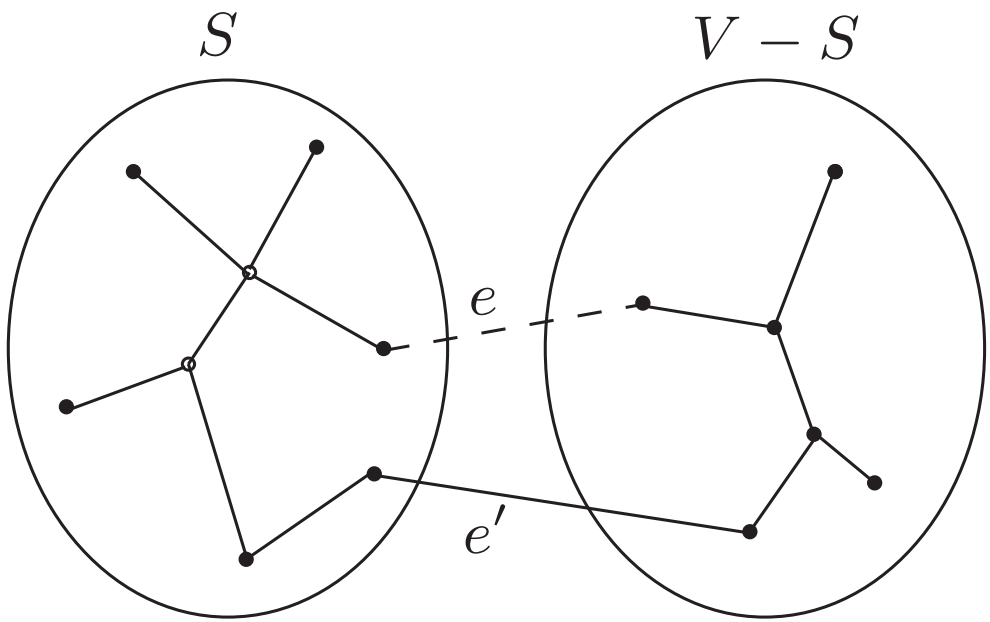
Ví dụ





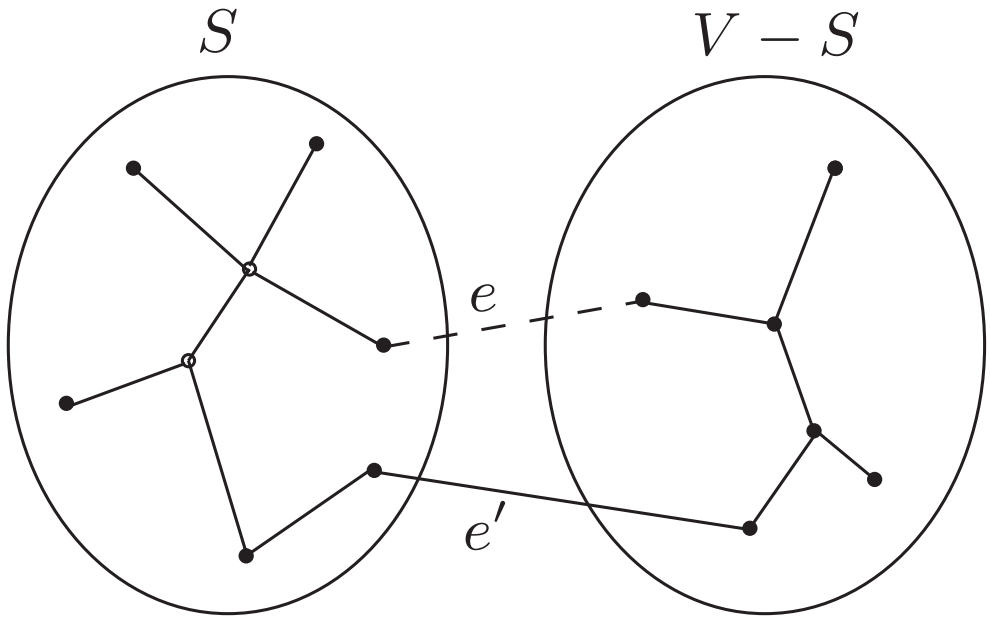
Nhát cắt S và $V - S$ và một cây bao trùm nhỏ nhất.

Chứng minh Tính chất Cắt



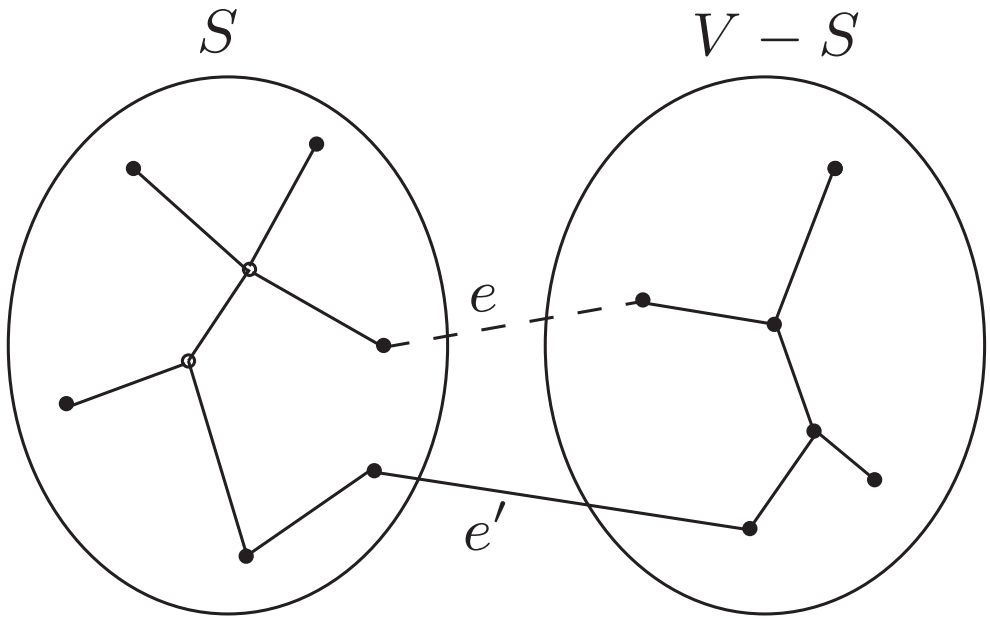
Xét X là một phần của MST T; nếu cạnh e cũng là một phần của T thì Tính chất Cắt đúng.

Chứng minh Tính chất Cắt (2)



- Giả sử e không thuộc MST T. Xét $T \cup \{e\}$.
- Việc thêm cạnh e vào T sẽ tạo ra một chu trình.
- Chu trình này chứa ít nhất một cạnh e' khác đi qua nhát cắt.

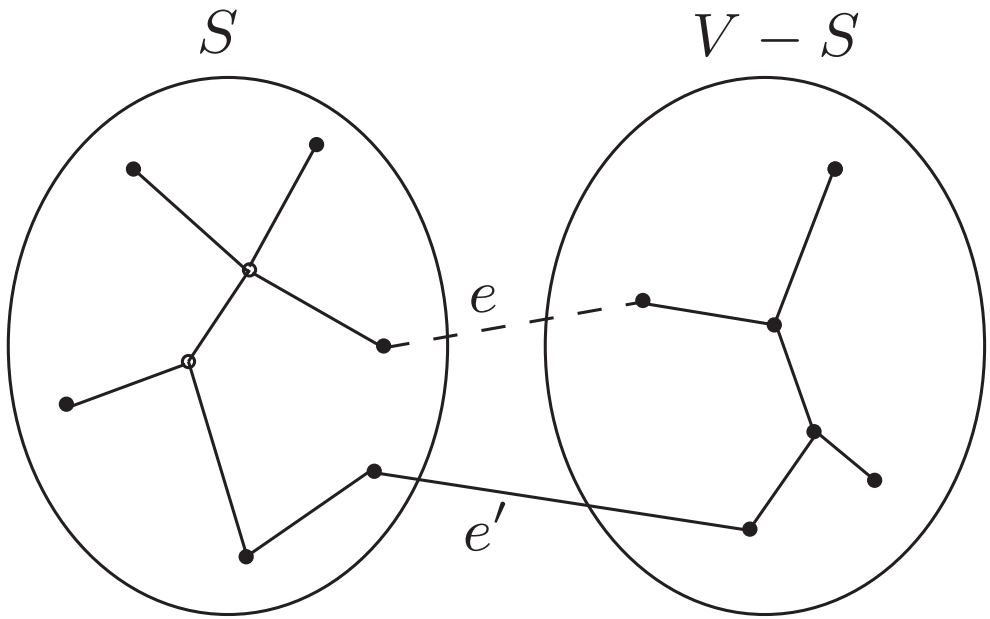
Chứng minh Tính chất Cắt (3)



- Xét đồ thị $T' = T \cup \{e\} - \{e'\}$.
- T' là một cây. Tại sao?

$G = (V, E)$ là một cây nếu và chỉ nếu G liên thông và $|E| = |V| - 1$;

Chứng minh Tính chất Cắt (3)

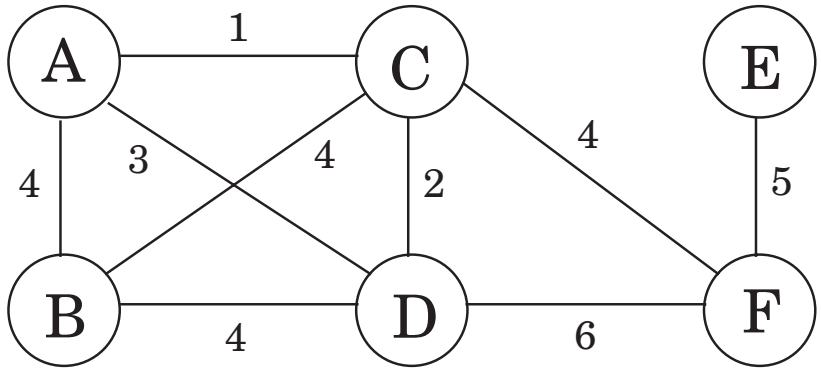


- Xét đồ thị $T' = T \cup \{e\} - \{e'\}$.
- T' là một cây.
- Cây T' cũng là cây bao trùm nhỏ nhất vì:

$$\text{weight}(T') = \text{weight}(T) + w(e) - w(e') \quad \text{và} \quad w(e) \leq w(e').$$

Nội dung

Tính đúng đắn của Thuật toán Kruskal?



Bắt đầu với đồ thị rỗng và chọn cạnh từ E theo quy tắc sau.

Lặp lại việc thêm cạnh nhỏ nhất mà không tạo thành chu trình.

Cài đặt thuật toán Kruskal

Sử dụng cấu trúc dữ liệu disjoint sets: mỗi tập là một thành phần liên thông.

Disjoint sets có ba phép toán:

- `makeset`(x): tạo ra một tập chỉ chứa phần tử x.
- `find`(x): x thuộc tập nào?
- `union`(x, y): hợp hai tập chứa x và y.

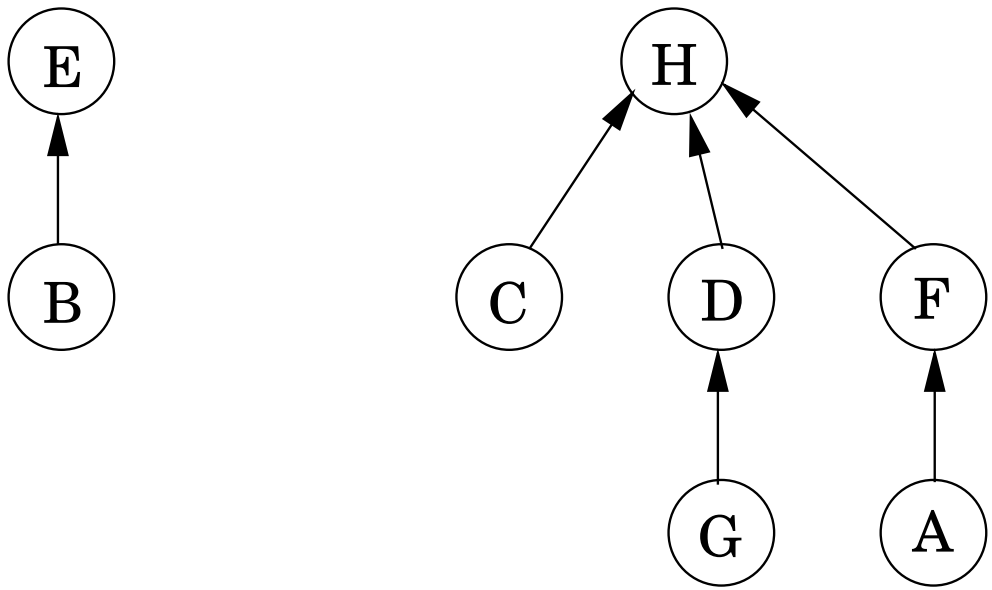
```
procedure kruskal(G, w)
Input: đồ thị liên thông vô hướng G = (V, E);
       với trọng số cạnh w_e
Output: MST định nghĩa bởi tập cạnh X.

for all u ∈ V:
    makeset(u)

X = ∅
Sắp xếp các cạnh e theo trọng số
for all {u, v} ∈ E, thứ tự tăng theo trọng số:
    if find(u) ≠ find(v):
        thêm cạnh {u, v} vào X
        union(u, v)
```

Cấu trúc dữ liệu Disjoint sets

- Lưu trữ tập dùng cây có hướng.
- Các nút là các phần tử của tập.
- Mỗi nút x có một con trỏ tới nút cha $\pi(x)$ của nó.
- Ngoài ra mỗi nút có một *rank* để lưu trữ độ cao của cây con từ nút này.
- Phần tử ở gốc là *đại diện*, hoặc là *tên*, của tập.
- Cha của gốc là chính nó.

Ví dụ

Cây có hướng biểu diễn hai tập $\{B, E\}$ và $\{A, C, D, F, G, H\}$



Cài đặt Disjoint sets

```
procedure markset(x)
```
$\pi(x) = x$
$\text{rank}(x) = 0$

```
function find(x)
while x ≠ π(x):   x = π(x)
return x
```

```
procedure union(x, y)
r_x = find(x)
r_y = find(x)
if r_x = r_y: return
if rank(r_x) > rank(r_y):    π(r_y) = r_x
else:
    π(r_x) = r_y
    if rank(r_x) = rank(r_y):    rank(r_y) = rank(r_x) + 1
```

Bài tập

Hãy vẽ cây biểu diễn disjoint sets sau các phép toán sau:

- `makeset(`A`)`, `makeset(`B`)`,..., `makeset(`G`)`
- `union(`A, D`)`, `union(`B, E`)`, `union(`C, F`)`
- `union(`C, G`)`, `union(`E, A`)`
- `union(`B, G`)`

Tính chất của `union-find`

Tính chất
Với mọi x, ta luôn có $\mathrm{rank}(x) < \mathrm{rank}(\pi(x))$.

Tính chất
Mọi nút gốc r với $\mathrm{rank}\ k$ có ít nhất 2^k nút trong cây của nó.

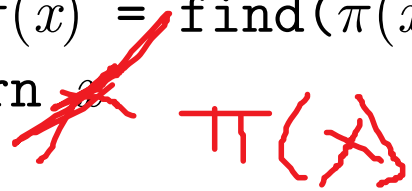
Tính chất
Nếu có n phần tử, có nhiều nhất $n/2^k$ nút có $\mathrm{rank}\ k$.

Cải tiến: Path Compression

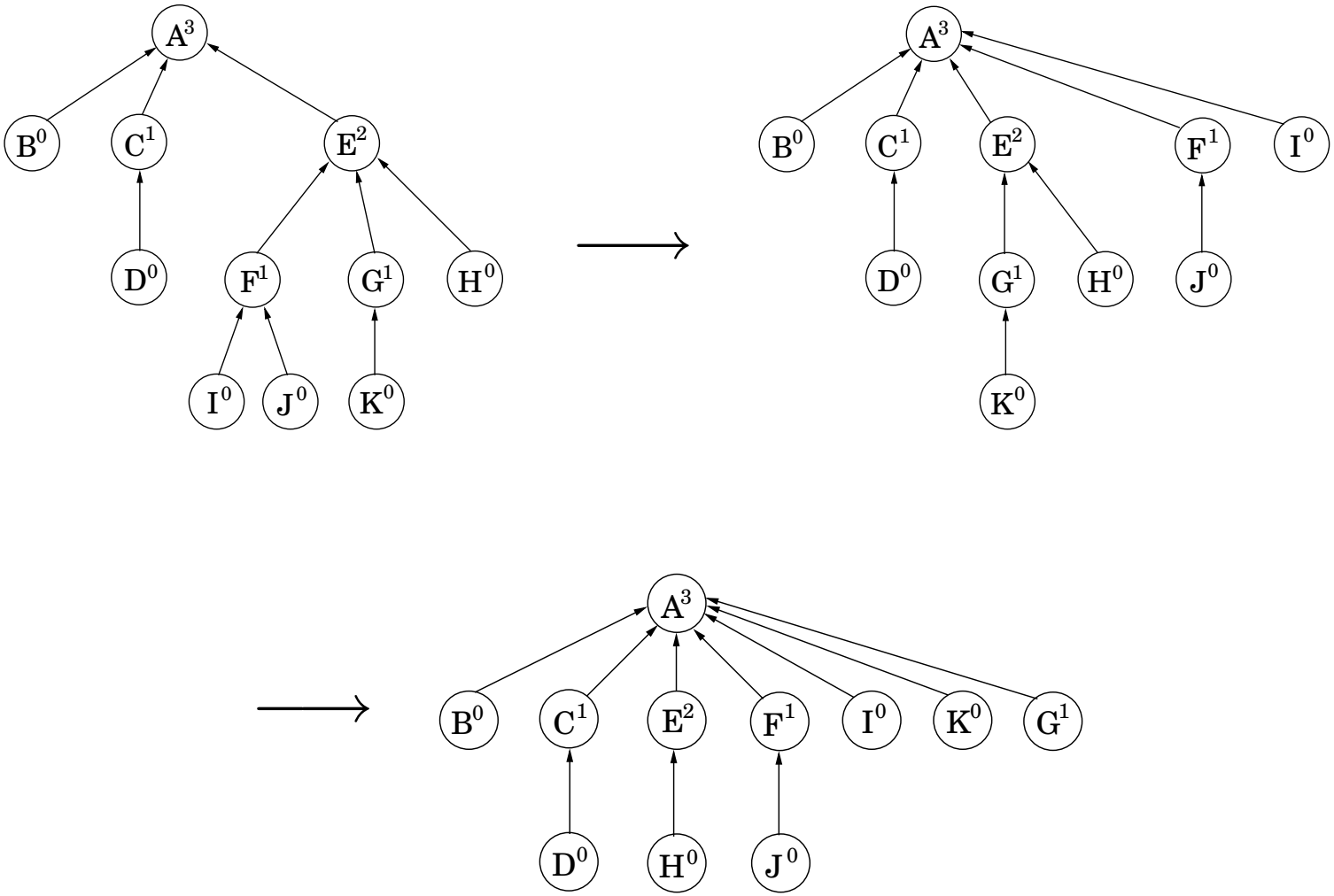
Gán luôn nút cha của x là gốc

```
function find(x)
if x ≠ π(x):
    π(x) = find(π(x))
return x
```

(Gán luôn cha của x là gốc)



Ví dụ: `find`(I) rồi `find`(K)



Nội dung

Thuật toán tổng quát dựa trên tính chất cắt

$X = \{\ \}$

Lặp lại các bước sau cho đến khi $|X| = |V| - 1$:

- Chọn một tập $S \subset V$ thỏa mãn: X không chứa cạnh nối giữa nhát cắt S và $V - S$.
- Xét $e \in E$ là cạnh có trọng số nhỏ nhất nối giữa nhát cắt S và $V - S$
- $X = X \cup \{e\}$

Thuật toán Prim

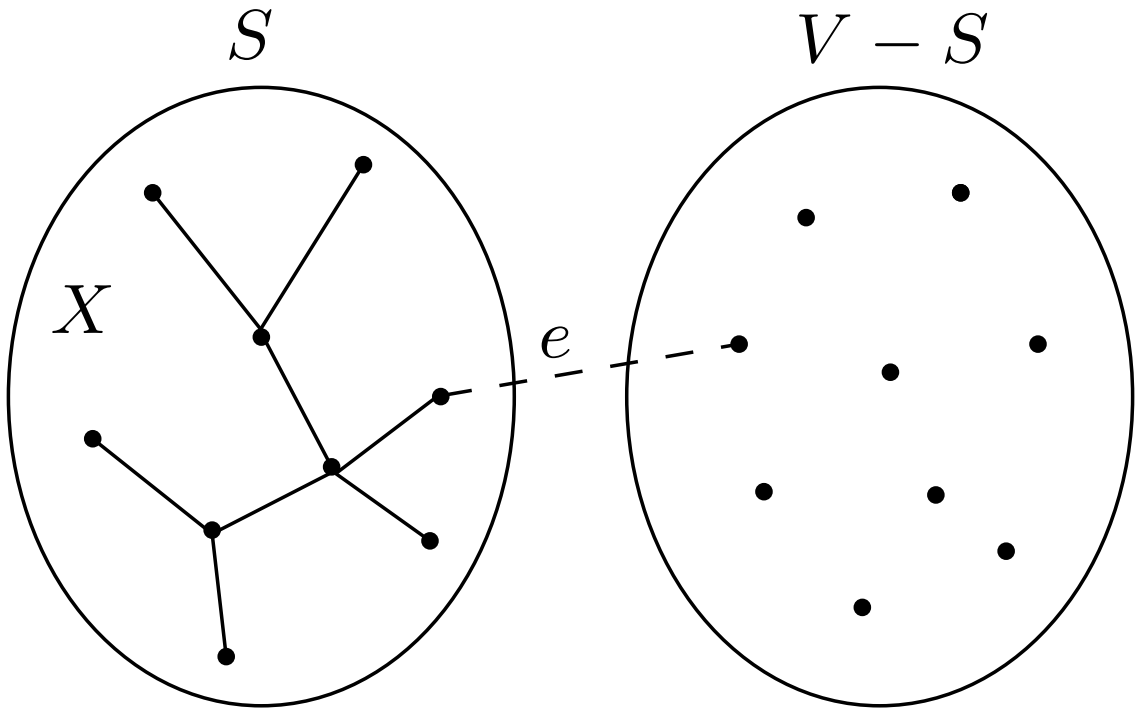
$X = \{\ \}$

Lặp lại các bước sau cho đến khi $|X| = |V| - 1$:

- Chọn một tập $S \subset V$ thỏa mãn: X không chứa cạnh nối giữa nhát cắt S và $V - S$.
- Xét $e \in E$ là cạnh có trọng số nhỏ nhất nối giữa nhát cắt S và $V - S$
- $X = X \cup \{e\}$

Thuật toán Prim

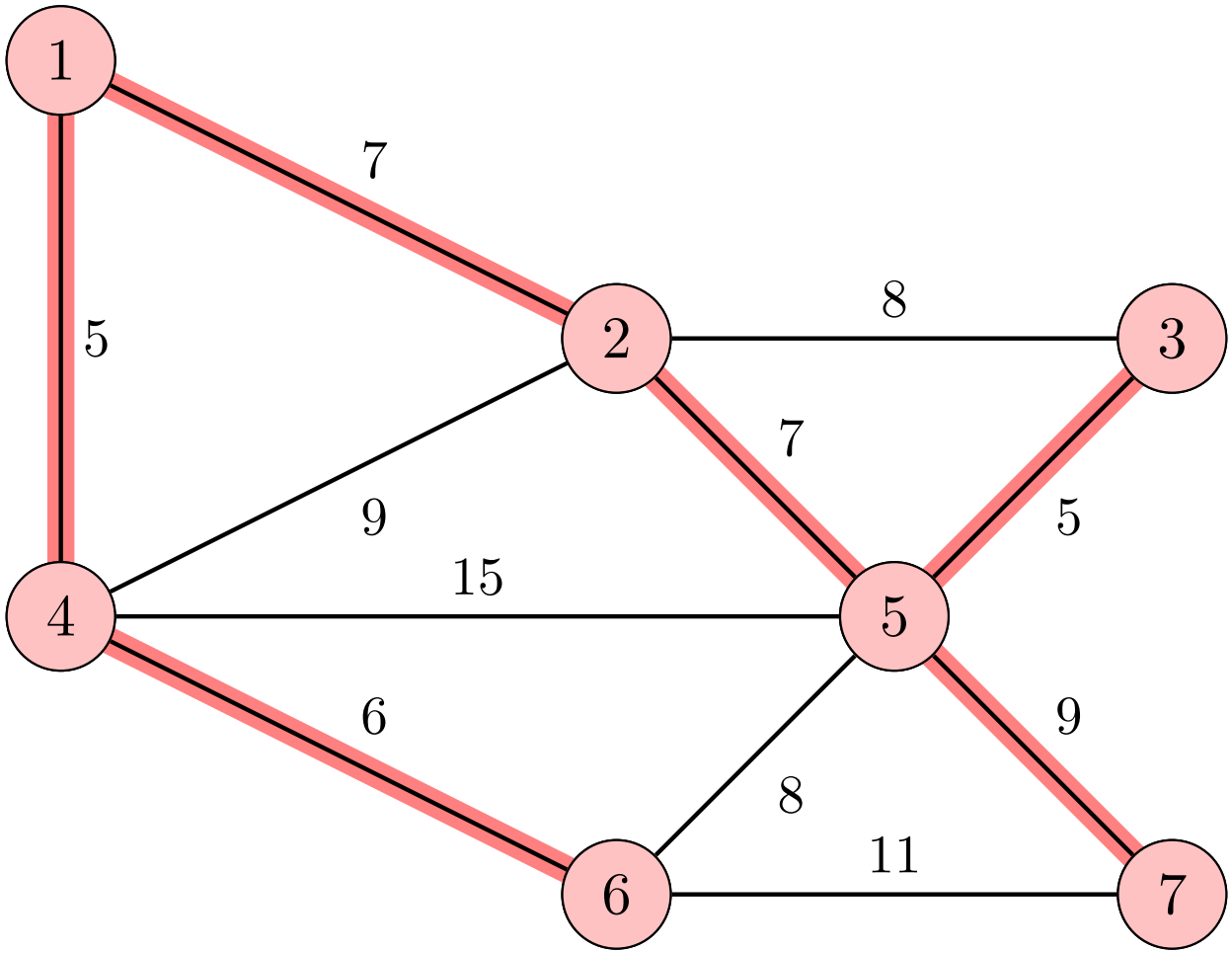
- Tập cạnh X luôn tạo ra một cây con, và
- Tập S được chọn là tập đỉnh của cây con này.



Thuật toán Prim

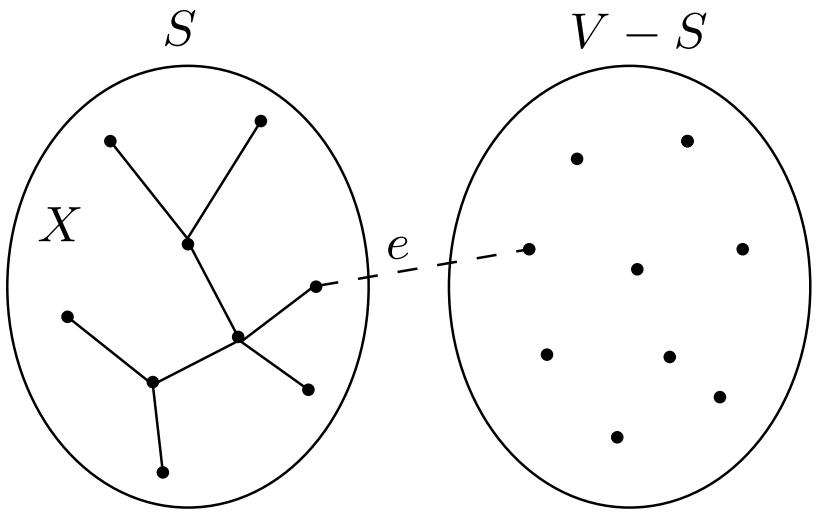
- Tập cạnh X luôn tạo ra một cây con, và
- Tập S được chọn là tập đỉnh của cây con này.

Ví dụ: Thuật toán Prim



Hình: Nguồn: tikz examples

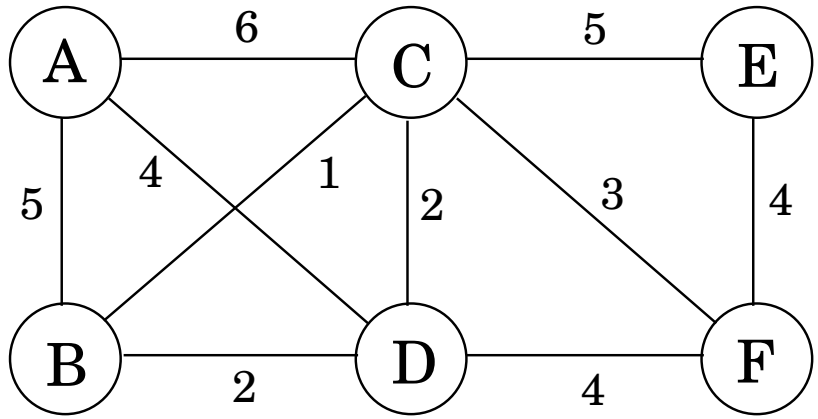
Thuật toán Prim vs Dijkstra



- Mỗi bước lặp, cây con X sẽ được thêm một cạnh.
- Đây là cạnh có trọng số nhỏ nhất nối giữa một đỉnh trong S và một đỉnh ngoài S.
- Có nghĩa rằng, thêm vào S đỉnh $v \notin S$ có `cost` nhỏ nhất:

$$\texttt{cost}(v) = \min_{u \in S} \; w(u, v)$$

Ví dụ



Tập S	A	B	C	D	E	F
$\{\}$	$0/\text{nil}$	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		$5/A$	$6/A$	$4/A$	∞/nil	∞/nil
A, D		$2/D$	$2/D$		∞/nil	$4/D$
A, D, B			$1/B$		∞/nil	$4/D$
A, D, B, C					$5/C$	$3/C$
A, D, B, C, F					$4/F$	

```
procedure prim(G, w)
Input: đồ thị vô hướng G = (V, E) với cạnh có trọng số
w_e.
Output: MST định nghĩa bởi mảng prev.
for all u ∈ V:
    cost(u) = ∞
    prev(u) = nil          (đỉnh trước u khi xây dựng cây)
Chọn tùy ý một đỉnh ban đầu u_0
cost(u_0) = 0

H = makequeue(V)      (dùng các giá trị cost làm khóa)
while H khác rỗng:
    v = deletemin(H)
    for all edges {v, z} ∈ E:
        if cost(z) > w(v, z):
            cost(z) = w(v, z)
            prev(z) = v                (đỉnh trước z là v)
            decreasekey(H, z)
```