Thuật toán tham lam

Trần Vĩnh Đức

HUST

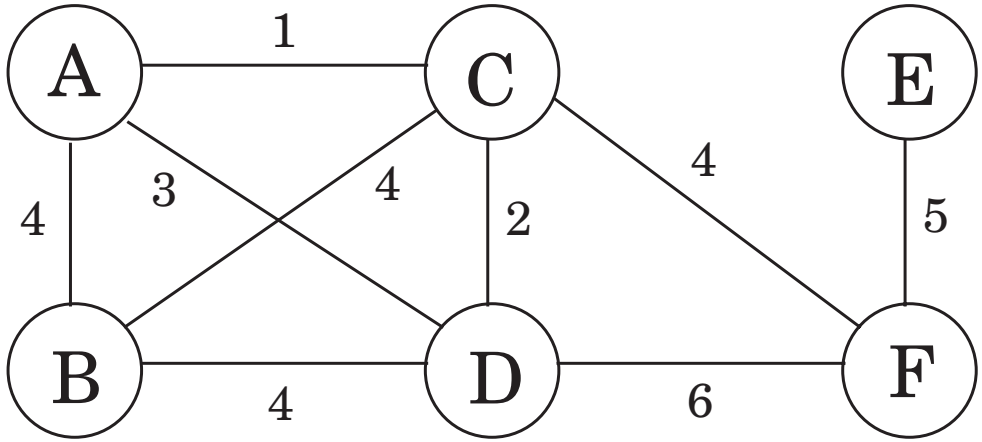
Ngày 14 tháng 9 năm 2020

Tài liệu tham khảo

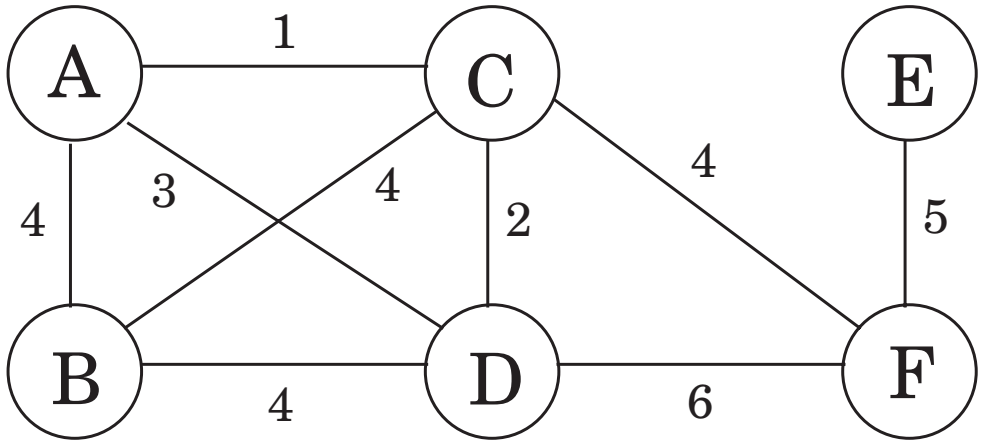
- S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, July 18, 2006.

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.

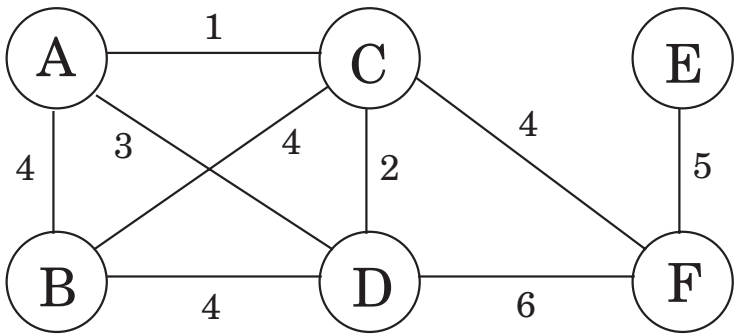
Bài toán Cây bao trùm nhỏ nhất (Minimal Spaning Tree)

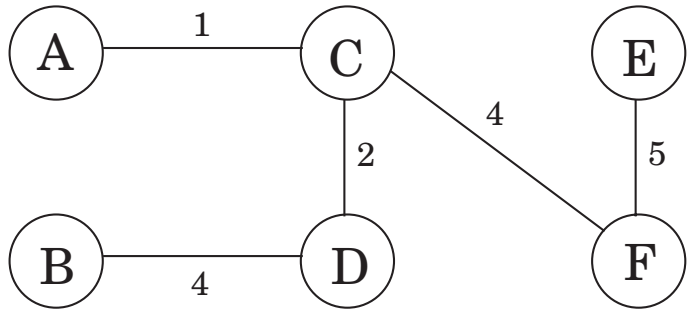
- *Input:* Đồ thị vô hướng $G = (V, E)$; mỗi cạnh có trọng số w_e.
- *Output:* Một cây $T = (V, E')$ với $E' \subseteq E$, với tổng trọng số

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

là nhỏ nhất.

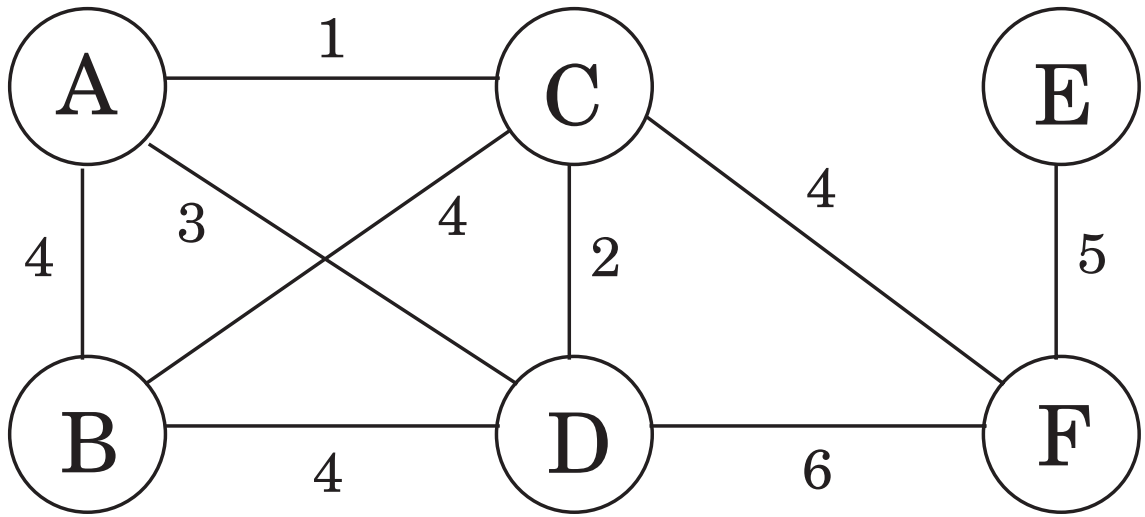
Tìm cây bao trùm





Đây có phải lời giải tối ưu không?

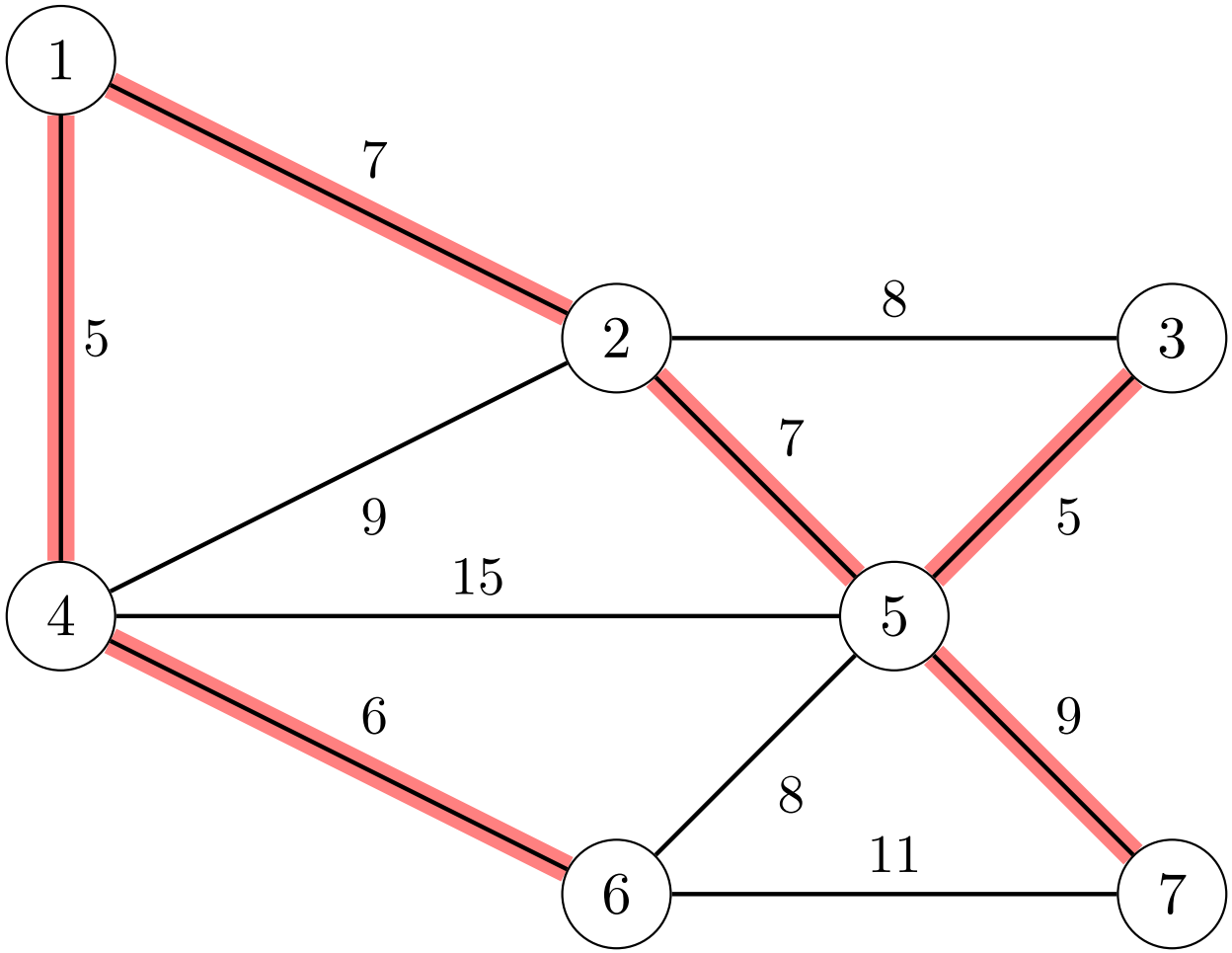
Thuật toán Kruskal



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

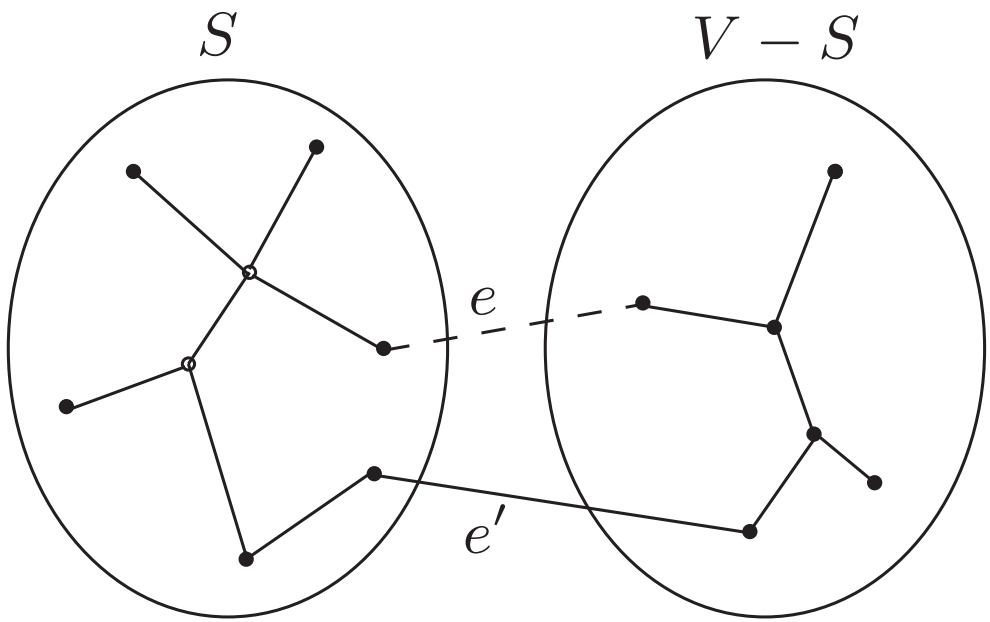


Hình: Nguồn: tikz examples

Nhát cắt

Định nghĩa

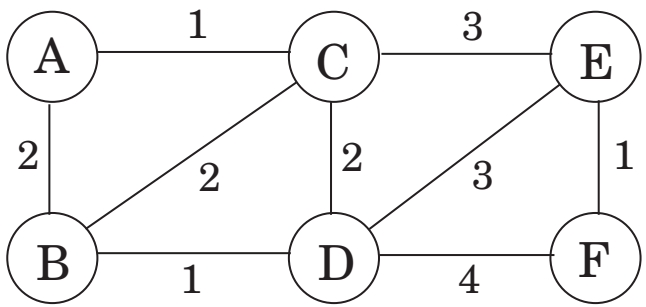
Xét đồ thị $G = (V, E)$. Một nhát cắt là một cách chia tập đỉnh thành hai nhóm: S và $V - S$.

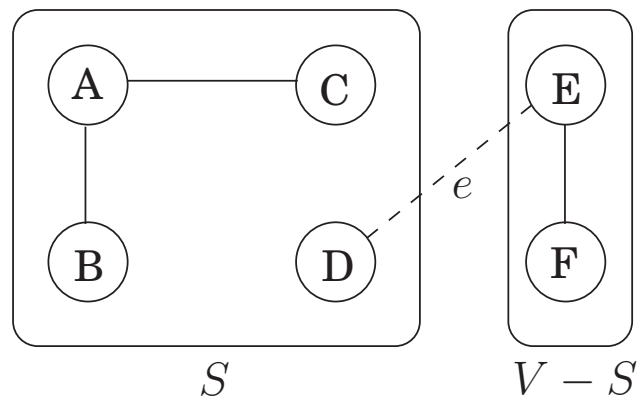


Hình: Nhát cắt và các cạnh nối giữa hai phân hoạch.

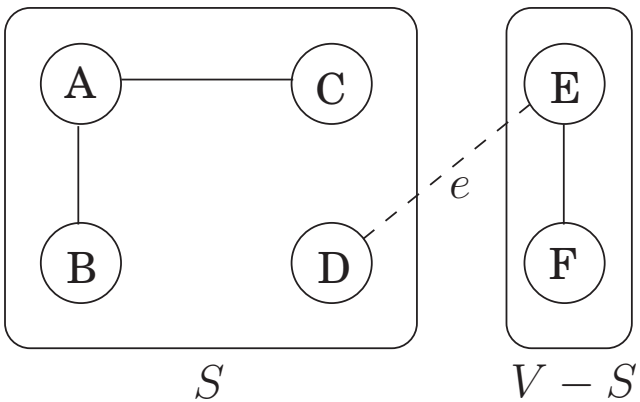
Tính chất Cắt

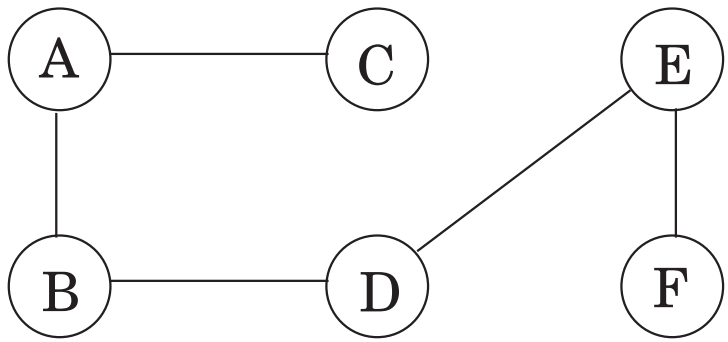
- Giả sử các cạnh X là một phần của một MST nào đó của $G = (V, E)$.
- Chọn một tập đỉnh bất kỳ S sao cho không có cạnh nào của X nối giữa S và $V - S$; và xét e là cạnh có trọng số nhỏ nhất nối hai phân hoạch này.
- Khi đó, $X \cup \{e\}$ là một phần của một MST nào đó.





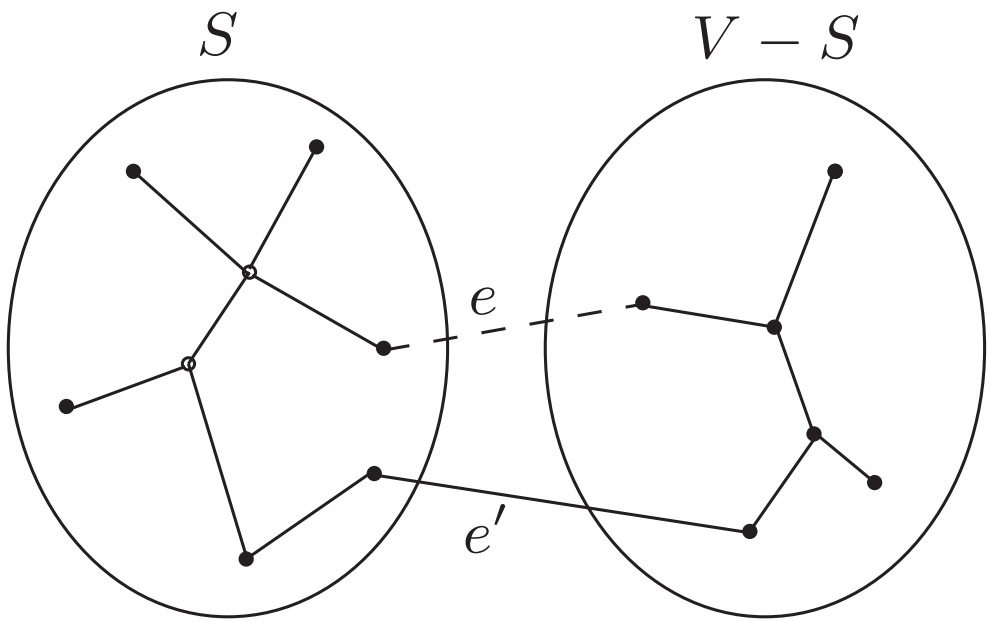
Ví dụ





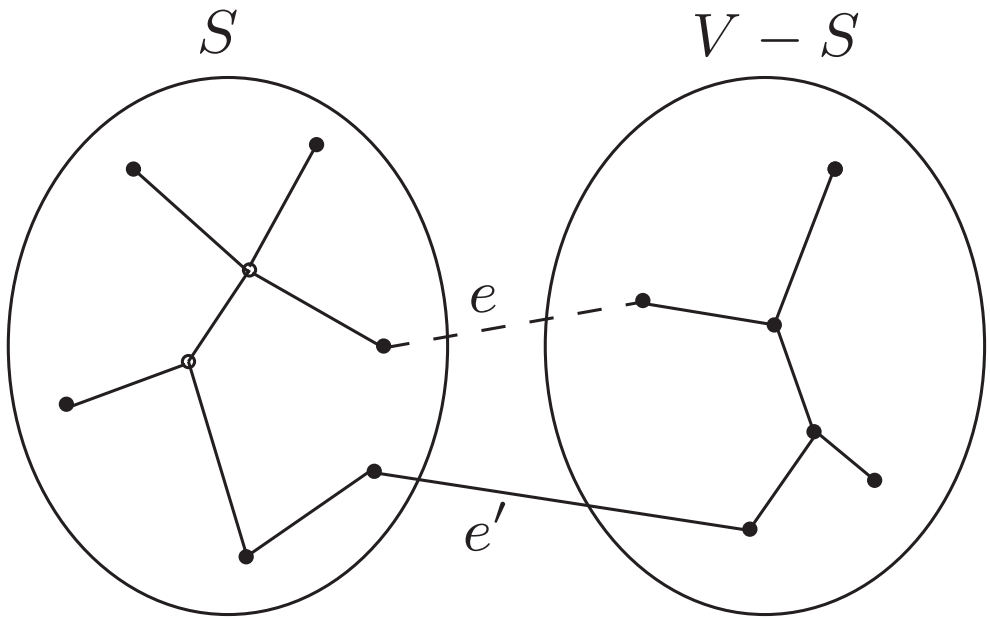
Nhát cắt S và $V - S$ và một cây bao trùm nhỏ nhất.

Chứng minh Tính chất Cắt



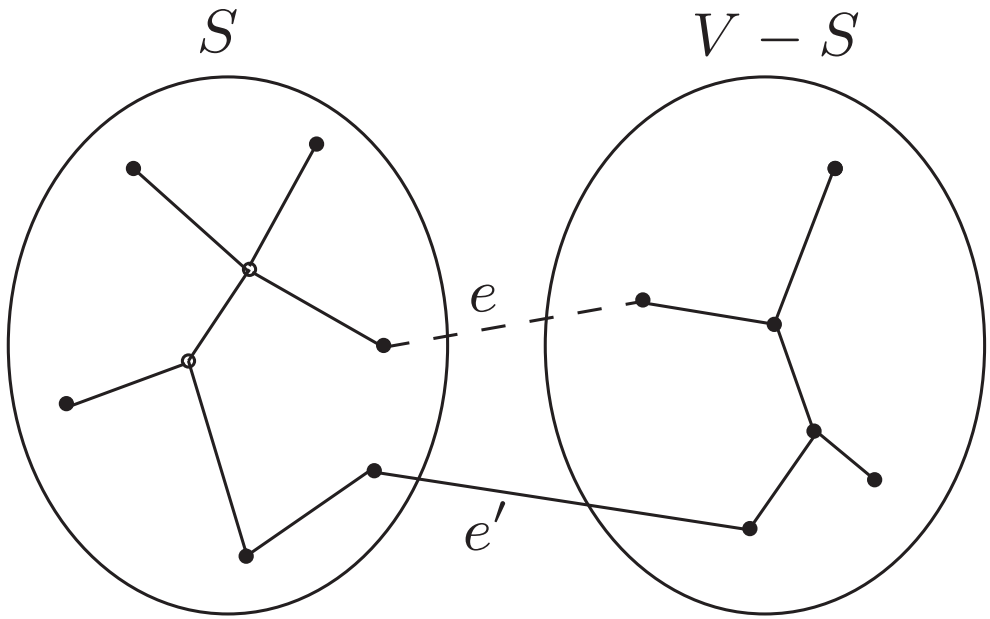
Xét X là một phần của MST T; nếu cạnh e cũng là một phần của T thì Tính chất Cắt đúng.

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.

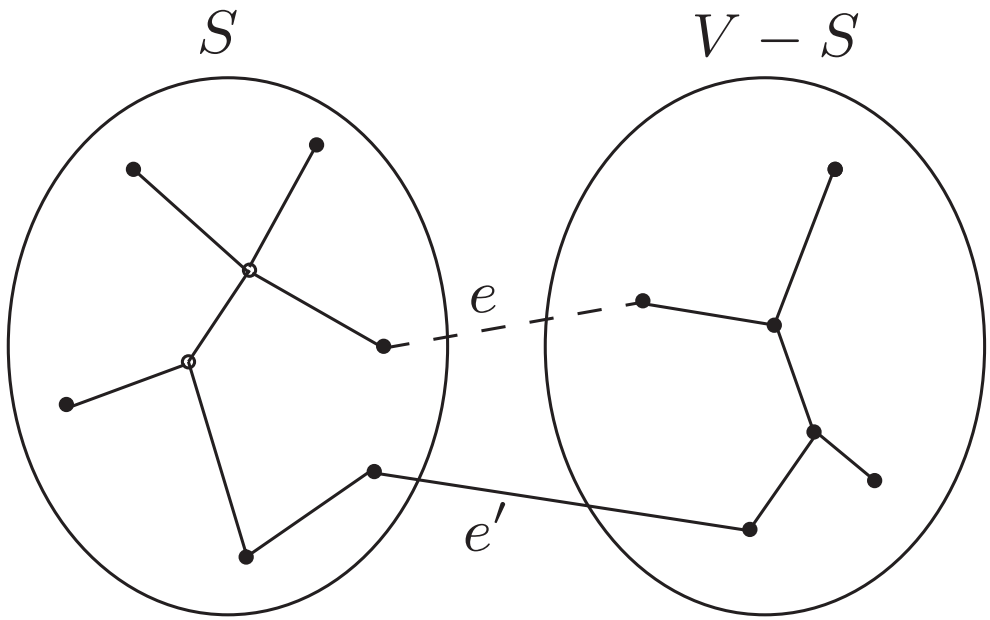
Chứng minh Tính chất Cắt (3)



- 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 $|E| = |V| - 1$;

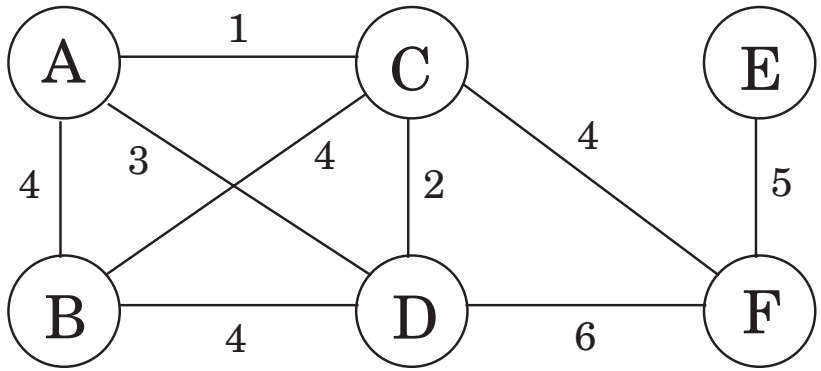
Chứng minh Tính chất Cắt (3)



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

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

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.

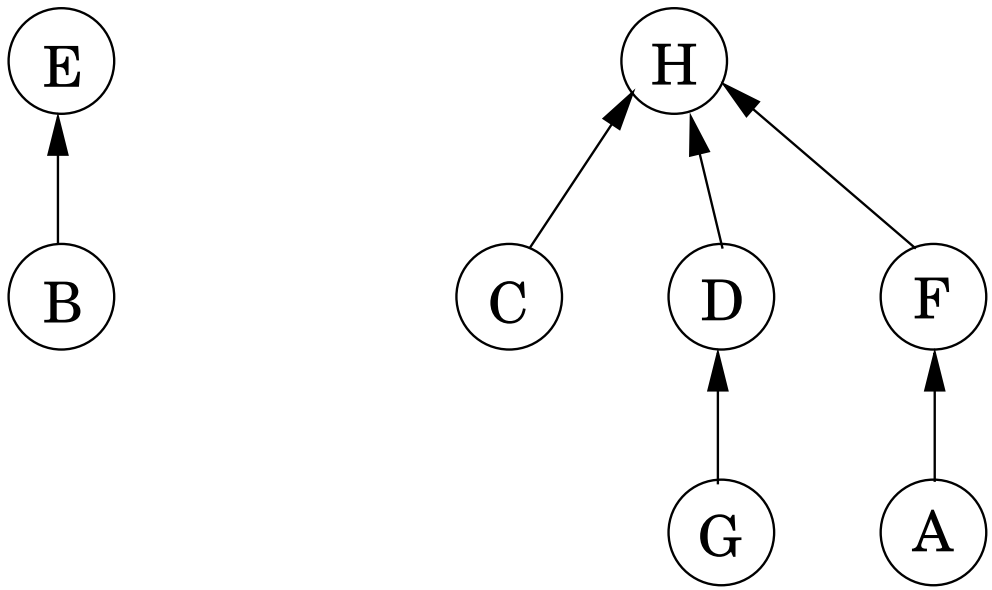
```
procedure kruskal(G, w)
Input: đồ thị liên thông vô hướng G = (V, E);
       với trọng số cạnh w_e
Output: MST định nghĩa bởi tập cạnh X.

for all u ∈ V:
    makeset(u)

X = ∅
Sắp xếp các cạnh e theo trọng số
for all {u, v} ∈ E, lấy không giảm 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` $x \neq \pi(x)$`:` $x = \pi(x)$
`return` x

Ở đây, $\pi(x)$ chỉ đến cha của x.

```
procedure union(x, y)
r_x = find(x)
r_y = find(y)
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

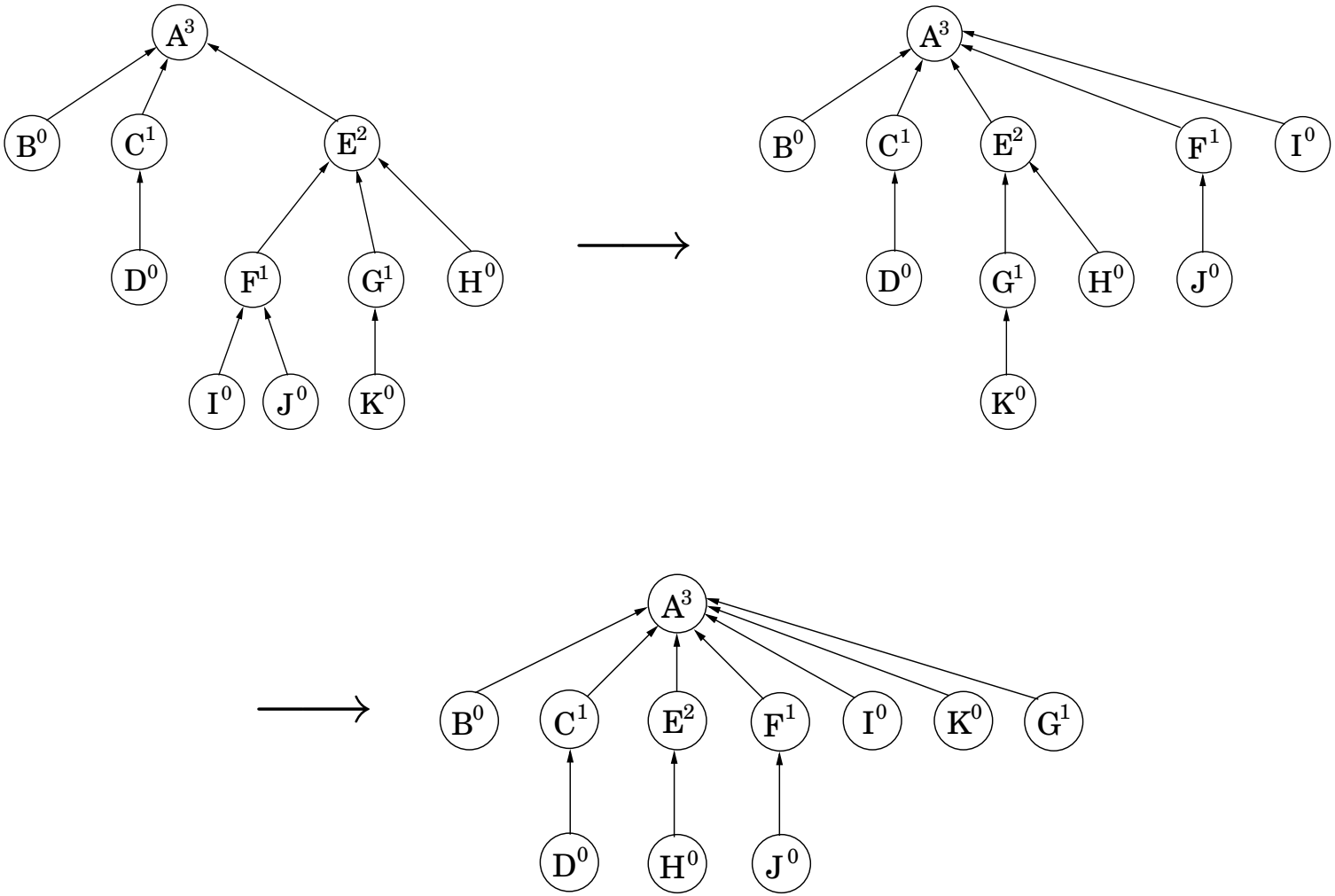
1. Với mọi x, ta luôn có $\mathrm{rank}(x) < \mathrm{rank}(\pi(x))$.
2. Mọi nút gốc r với $\mathrm{rank}\ k$ có ít nhất 2^k nút trong cây của nó.
3. Nếu có n phần tử, có nhiều nhất $n/2^k$ nút có $\mathrm{rank}\ k$.

Cải tiến: Path Compression

Gán nút gốc là cha của mọi nút x

```
function find(x)
if x ≠ π(x):
    π(x) = find(π(x))          (Gán nút gốc là cha của x)
return π(x)
```

Ví dụ: `find`(I) rồi `find`(K)



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

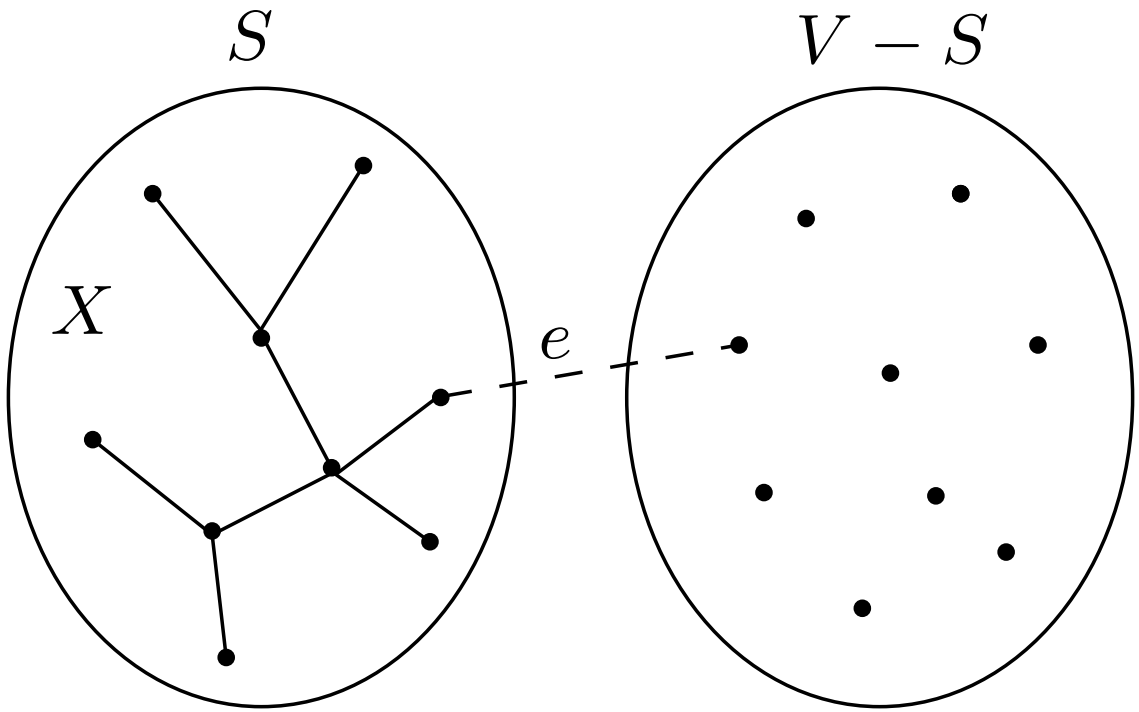
$X = \{\ \}$

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

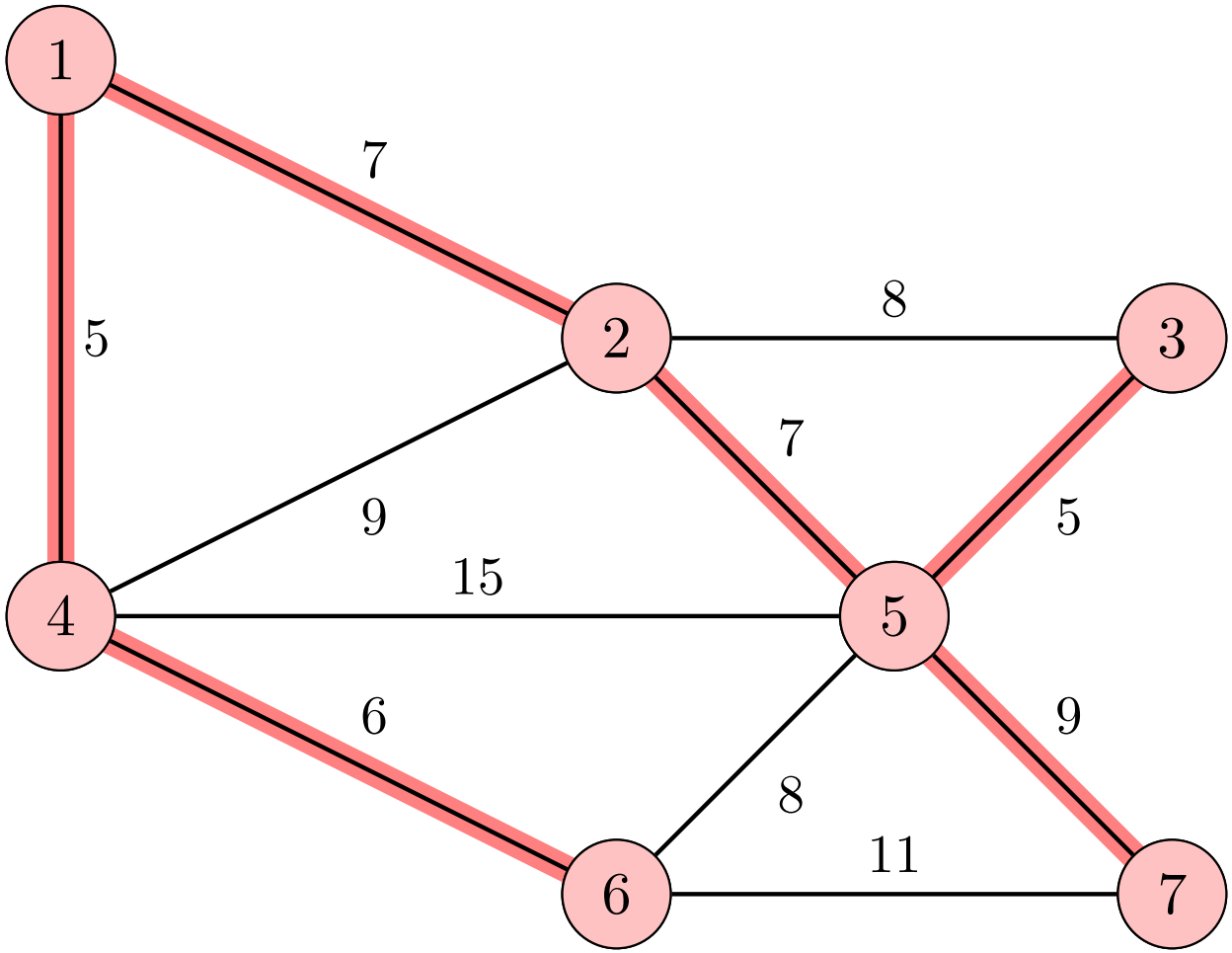
- Tập cạnh X luôn tạo ra một cây con, và
- Tập S được chọn là tập đỉnh của cây con này.



Thuật toán Prim

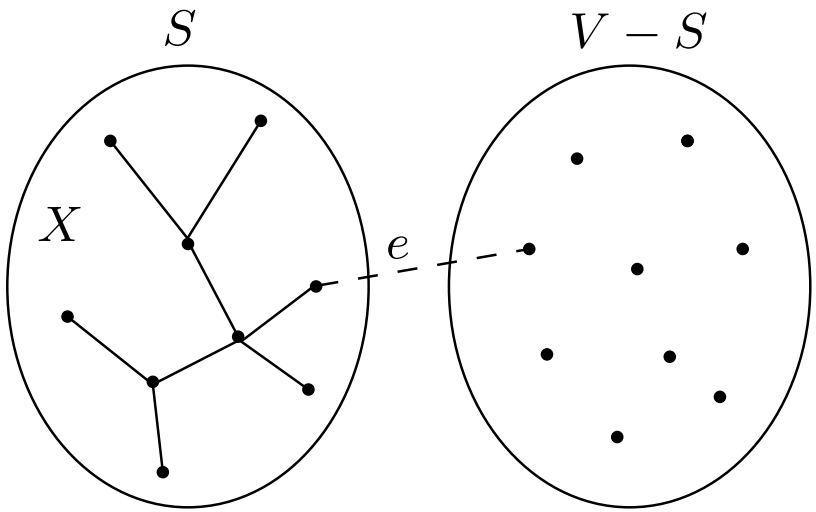
- Tập cạnh X luôn tạo ra một cây con, và
- Tập S được chọn là tập đỉnh của cây con này.

Ví dụ: Thuật toán Prim



Hình: Nguồn: tikz examples

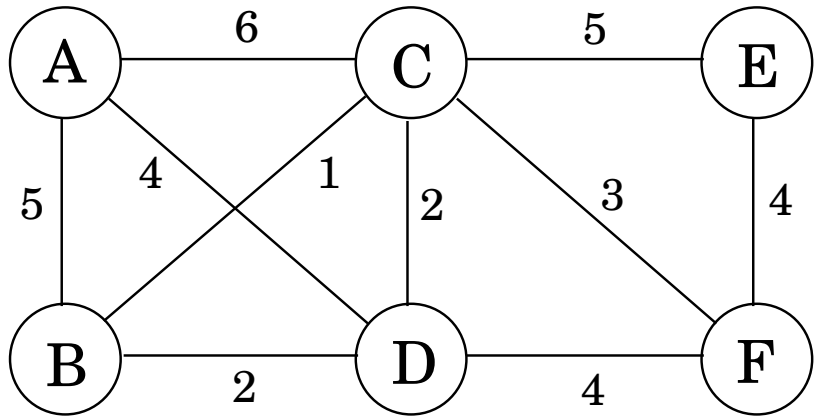
Thuật toán Prim vs Dijkstra



- 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)
```

Nội dung

Sơ đồ nén MP3

1. ***Số hóa bằng cách lấy mẫu theo khoảng.*** Tạo ra một dãy số thực

$$s_1, s_2, \ldots, s_T$$

Ví dụ, với tốc độ $44,100$ mẫu trên giây, bản giao hưởng 50 phút có

$$T = 50 \times 60 \times 44,100 \approx 130 \text{ triệu.}$$

2. ***Lượng hóa.*** Xấp xỉ s_i bởi giá trị gần nhất thuộc tập hữu hạn Γ.
3. ***Mã hóa.*** Xâu $s_1 s_2 \ldots s_T$ trên bảng chữ Γ được mã hóa ở dạng nhị phân. (Dùng mã Huffman)

Mã hóa

Ký hiệu	Số lần xuất hiện
A	70 triệu
B	3 triệu
C	20 triệu
D	37 triệu

- Bảng chữ $\Gamma = \{A, B, C, D\}$ và $T = 130$ triệu.
- Nếu mã hóa dùng 2 bit cho mỗi ký hiệu, ví dụ
$$A \to 00, \qquad B \to 01, \qquad C \to 10, \qquad D \to 11$$
ta cần 260 megabits.
- Liệu ta có thể dùng ***mã độ dài thay đổi*** để giảm kích thước bản mã?

Mã độ dài thay đổi

- Dùng các dãy bit độ dài khác nhau để mã hóa các chữ cái.
- Chữ cái xuất hiện thường xuyên hơn sẽ được mã bằng dãy bit ngắn hơn.
- Vấn đề: Làm thế nào xác định được mỗi chữ bắt đầu và kết thúc ở đâu trong dãy bit.?

Ví dụ

Cách mã hóa

$$A \to 0, \qquad C \to 01, \qquad D \to 11, \qquad B \to 001$$

gây ra nhập nhằng khi giải mã

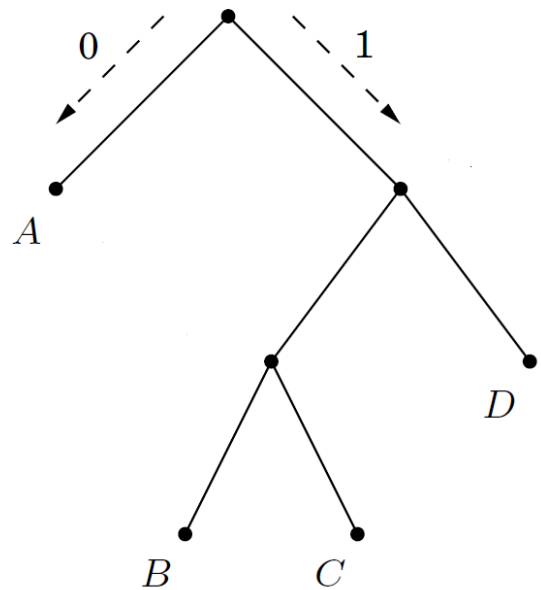
$$001 \to AC$$
$$001 \to B$$

Mã tiền tố

Định nghĩa

Mã tiền tố là tập xâu thỏa mãn không có xâu nào là khúc đầu của xâu khác.

Symbol	Codeword
A	0
B	100
C	101
D	11



Hãy giải mã dãy bit 10100100?

Kích thước bản mã

Ký hiệu	Số lần xuất hiện	Từ mã
A	70 triệu	0
B	3 triệu	100
C	20 triệu	101
D	37 triệu	11

- Kích thước bản mã

$$= (1 \times 70 + 3 \times 3 + 3 \times 20 + 2 \times 37) \text{ megabits}$$
$$= 213 \text{ megabits}$$

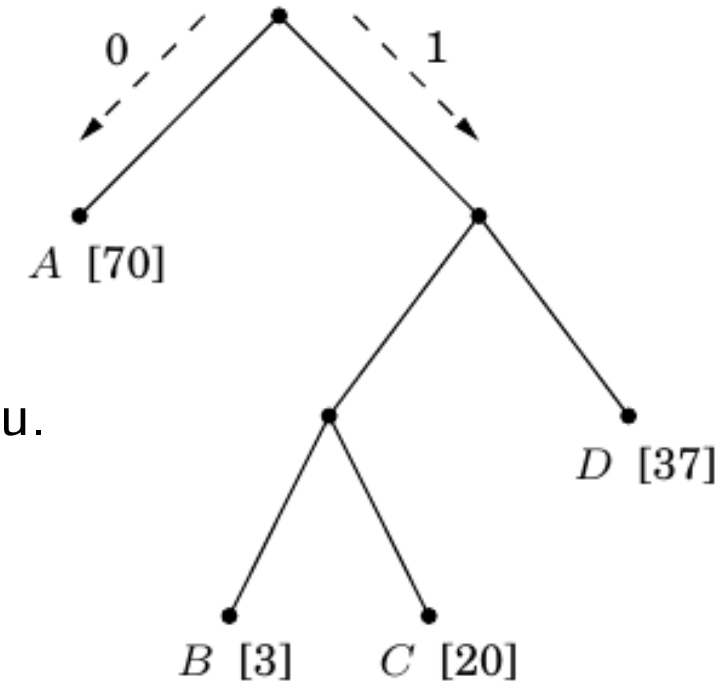
- Cải thiện 17% so với 260 megabits khi dùng mã độ dài cố định.

Bài toán

- Cho n ký hiệu có tần suất

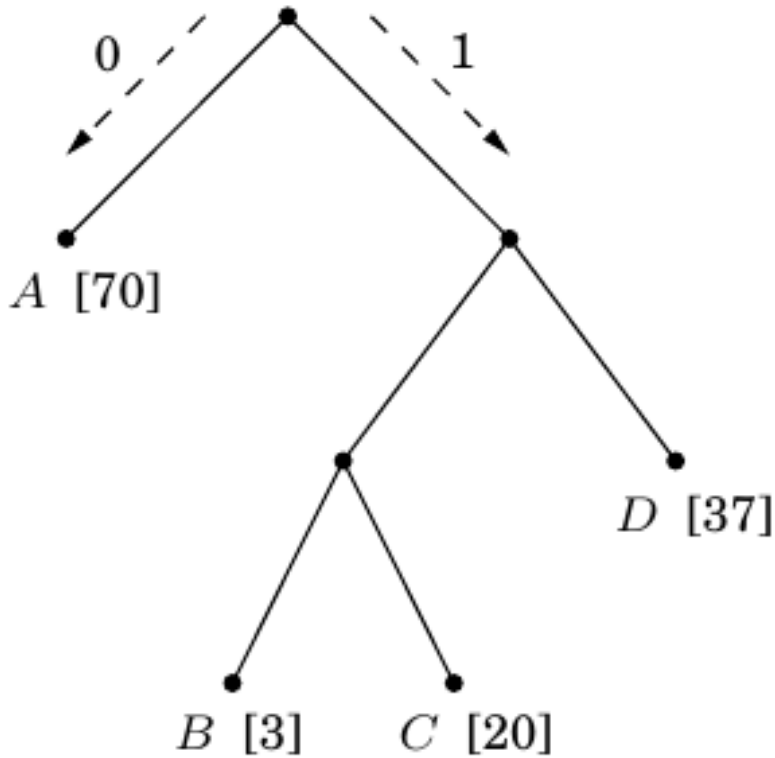
$$f_1, f_2, \ldots, f_n.$$

- Hãy tìm cây ở đó mỗi lá ứng với một ký hiệu và có chi phí cực tiểu.



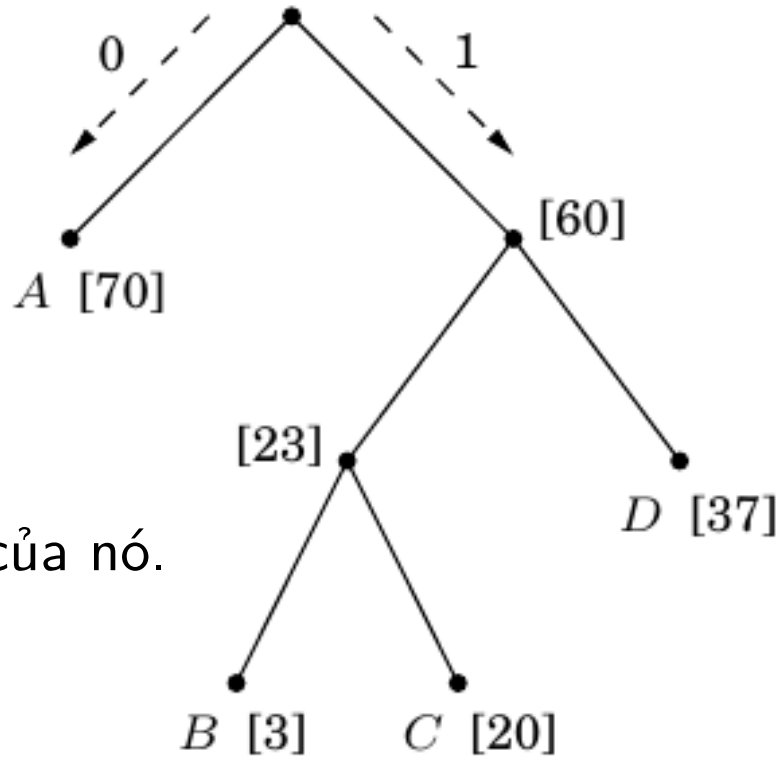
$$\text{Chi phí của cây } = \sum_{i=1}^{n} f_i \cdot (\text{độ sâu ký hiệu thứ } i \text{ trong cây})$$

Hãy tính chi phí của cây sau.



$$\text{Chi phí của cây } = \sum_{i=1}^{n} f_i \cdot (\text{độ sâu ký hiệu thứ } i \text{ trong cây})$$

- Tần suất nút ***lá*** là f_i.
- Tần suất ***nút trong*** là tổng tần suất của các ***lá*** con cháu của nó.



Mệnh đề

Chi phí của cây là tổng tần suất của mọi nút ngoại trừ nút gốc.

Tối ưu hàm chi phí

$$\text{Chi phí của cây } = \sum_{i=1}^{n} f_i \cdot (\text{độ sâu của ký hiệu thứ } i \text{ trong cây})$$

Nhận xét

Hai ký hiệu với tần suất nhỏ nhất sẽ phải ở đáy của cây tối ưu.

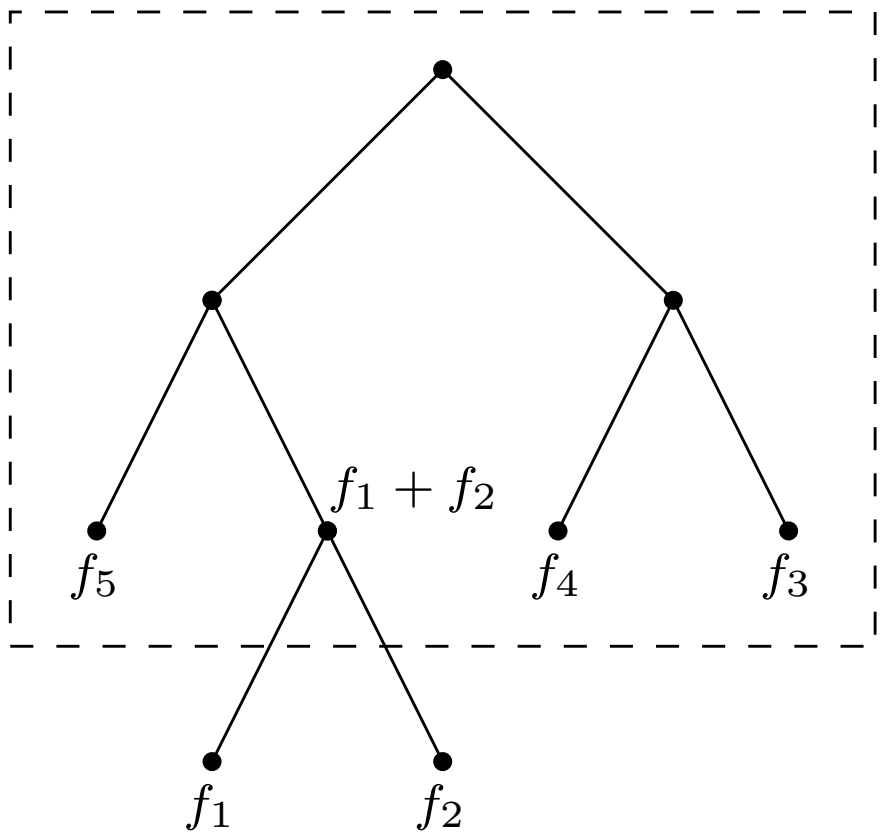
Xây dựng cây một cách tham lam

- Tìm hai ký hiệu có tần suất nhỏ nhất, gọi là i và j, và tạo nút cha của chúng với tần suất $f_i + f_j$.
  Để đơn giản ký hiệu, ta giả sử chúng là f_1 và f_2.
- Mọi cây trong đó f_1 và f_2 là nút lá anh em có chi phí $f_1 + f_2$ cộng với chi phí cho cây gồm $n - 1$ nút lá của các tần suất:

$$(f_1 + f_2),\ f_3,\ f_4, \ldots, f_n.$$

- Ta đưa về bài toán kích thước nhỏ hơn. Ta loại bỏ f_1 và f_2 khỏi dãy tần suất và thêm $(f_1 + f_2)$ vào, và ***lặp lại***.

Xây dựng cây một cách tham lam



Hình: Loại f_1, f_2 và thêm $f_1 + f_2$ vào dãy tần suất.

procedure Huffman(f)
Input: mảng $f[1 \cdots n]$ của các tần suất
Output: Một cây mã hóa với n lá

Xét H là hàng đợi ưu tiên của các số nguyên, thứ tự bởi f
for $i = 1$ to n: insert(H, i)
for $k = n + 1$ to $2n - 1$:
    i = deletemin(H), j = deletemin(H)
    Tạo một nút đánh số k với các con là i, j
    $f[k] = f[i] + f[j]$
    insert(H, k)

Nội dung

Biến Boolean

- Đối tượng cơ sở trong công thức Horn là *biến Boolean*, nhận giá trị `true` hoặc `false`.
- Một literal là một biến x hoặc nghịch đảo của nó $\overline{x}$ ("NOT x").

Ví dụ

Biến x, y và z ký hiệu các khả năng sau đây:

$$\begin{aligned} x &\equiv \text{vụ giết người xảy ra trong bếp} \\ y &\equiv \text{người quản gia vô tội} \\ z &\equiv \text{ngài đại tá đã ngủ lúc 8 giờ tối} \end{aligned}$$

Công thức Horn

Trong công thức Horn, tri thức về các biến được biển diễn bởi hai loại mệnh đề:

1. ***Kéo theo***, vế trái là một AND của một số các literal dương và vế phải là một literal dương.
   Các khẳng định này có dạng "nếu mệnh đề vế trái mà đúng, thì vế phải cũng phải đúng". Ví dụ,

   $$(x \wedge w) \Rightarrow u$$

   Một trường hợp riêng là dạng đơn $\Rightarrow x$ với ý nghĩa rằng "x là `true`".
2. ***Toàn phủ định***, bao gồm một OR của một số literal âm.
   Ví dụ

   $$(\overline{u} \vee \overline{v} \vee \overline{y})$$

   ("không thể tất cả cùng vô tội").

Bài toán nhất quán

- Cho một tập các mệnh đề kéo theo hoặc toàn phủ định;
- Hãy xác định liệu chúng có nhất quán. Tức là, có cách gán `true`/`false` cho các biến để thỏa mãn mọi mệnh đề. Đây cũng gọi là *phép gán thỏa được*.

Câu hỏi

Hãy xác định liệu các mệnh đề sau có nhất quán:

$$(w \wedge y \wedge z) \Rightarrow x, \qquad (x \wedge z) \Rightarrow w, \quad x \Rightarrow y,$$
$$\Rightarrow x, \qquad (x \wedge y) \Rightarrow w, \quad (\overline{w} \vee \overline{x} \vee \overline{y}), \quad (\overline{z}).$$

Thuật toán tham lam

```
 Input: Một công thức Horn
Output: Một phép gán thỏa được, nếu tồn tại

Đặt mọi biến bằng false
while còn phép kéo theo chưa thỏa mãn:
    Đặt biến ở vế phải của phép kéo theo này bằng
true

if mọi mệnh đề toàn phủ định được thỏa mãn:
    return phép gán
else:
    return ``công thức không thỏa mãn''
```

Bài tập

Hãy chạy thuật toán với công thức Horn gồm các mệnh đề sau:

$$(w \wedge y \wedge z) \Rightarrow x,$$
$$(x \wedge z) \Rightarrow w,$$
$$x \Rightarrow y,$$
$$\Rightarrow x,$$
$$(x \wedge y) \Rightarrow w,$$
$$(\overline{w} \vee \overline{x} \vee \overline{y}),$$
$$(\overline{z}).$$

Tính đúng đắn của thuật toán

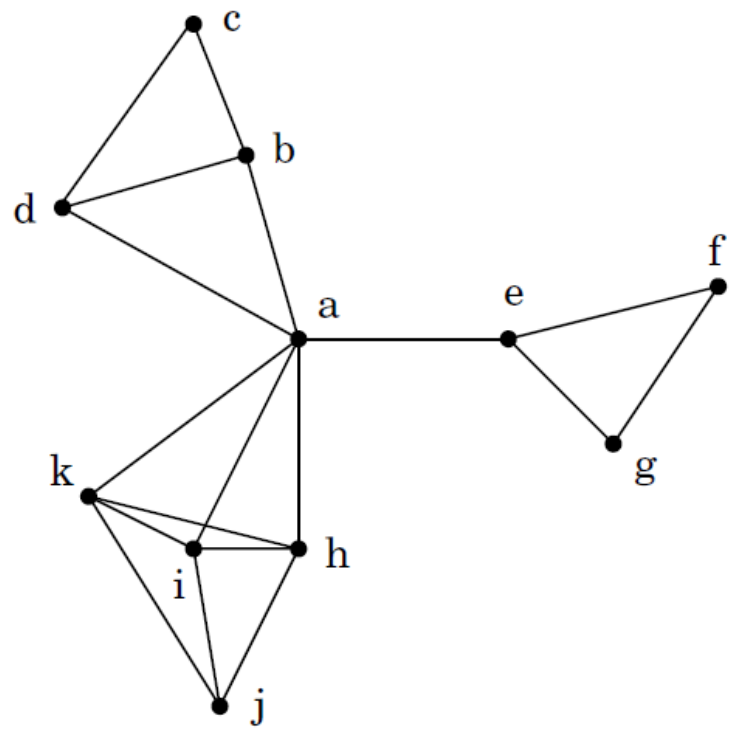
- Nếu `return` một phép gán, thì phép gán này phải thỏa mãn cả các mệnh đề ***kéo theo*** và các mệnh đề ***toàn phủ định***.
- Ngược lại, ta có bất biến trong vòng lặp `while`:

  Nếu một biến được đặt bằng `true`*, vậy thì nó phải bằng* `true` *trong mọi* ***phép gán thỏa được***.

  Nếu có một phép gán được tìm thấy sau vòng lặp `while` mà không thỏa mãn mệnh đề ***toàn phủ định***, vậy không có phép gán nào thỏa được.

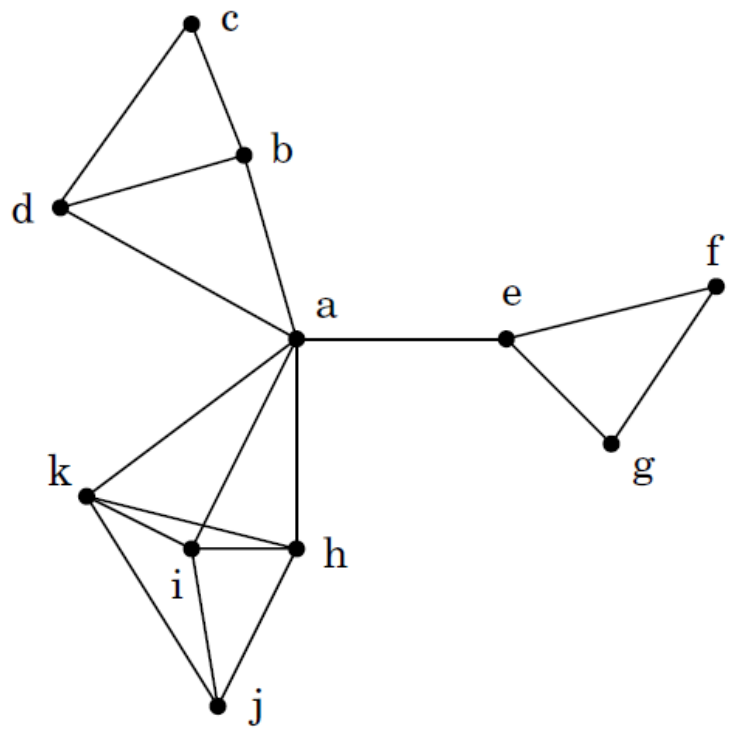
Nội dung

Xây trường học trong các thị trấn



- Mỗi trường phải ở trong một thị trấn và không ai phải đi hơn 30 dặm để tới trường.
- Số trường tối thiểu cần xây là bao nhiêu?

Xây trường học trong các thị trấn



- Mỗi thị trấn x, đặt S_x là tập thị trấn cách x trong vòng 30 dặm. Một trường tại x sẽ "phủ" các thị trấn này.
- Cần lấy bao nhiêu tập S_x để phủ mọi thị trấn?

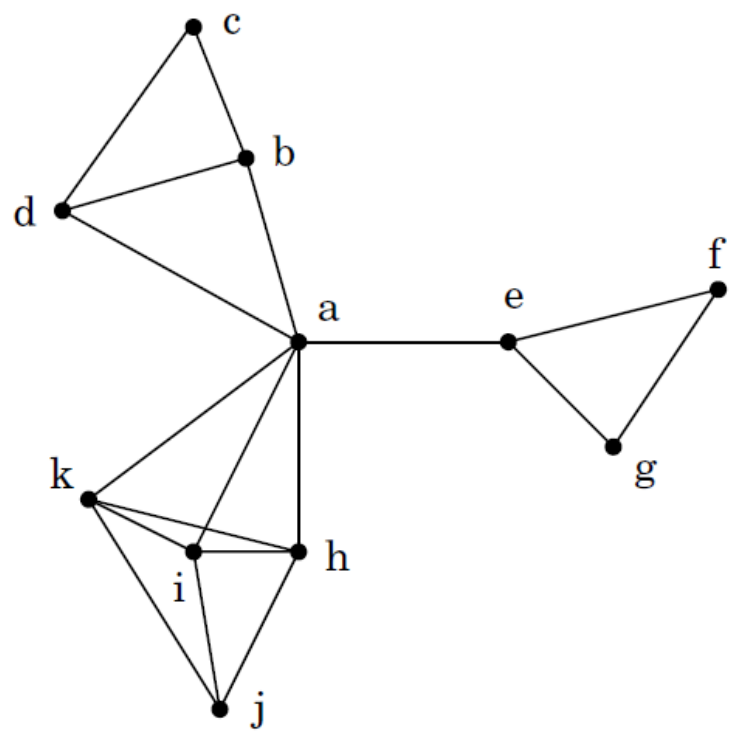
Bài toán phủ tập

- Input: Một tập B; và các tập $S_1, S_2, \ldots, S_m \subseteq B$
- Output: Một cách chọn các S_i sao cho hợp của chúng là B
- Chi phí: Số tập được chọn

Thuật toán tham lam

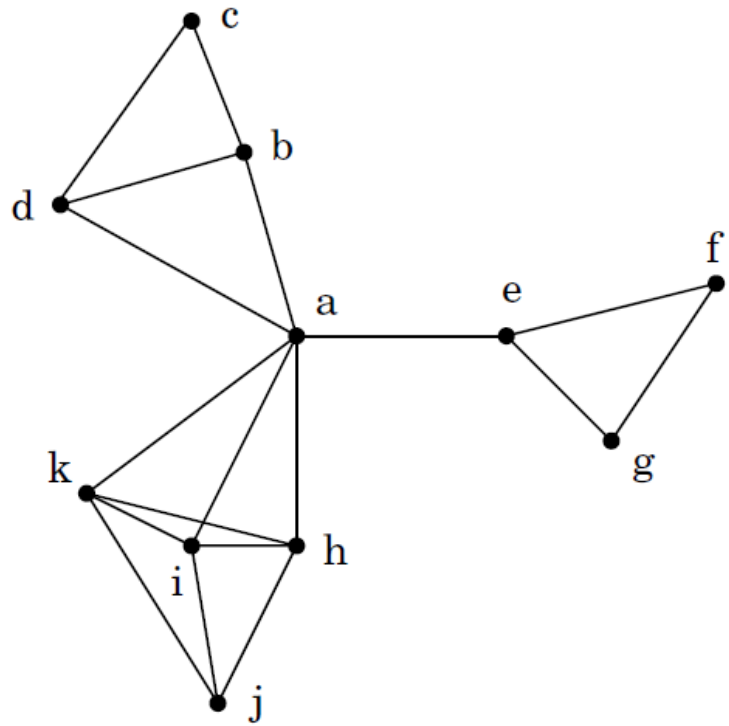
Lặp lại cho đến khi mọi phần tử của B được phủ:

Chọn tập S_i với nhiều phần tử chưa được phủ nhất.



Bài tập

Hãy tìm phủ tối ưu cho tập thị trấn như sau:



Khẳng định

Giả sử B chứa n phần tử và phủ tối ưu gồm k tập. Thuật toán tham lam sẽ dùng nhiều nhất $k \ln n$ tập.

Chứng minh

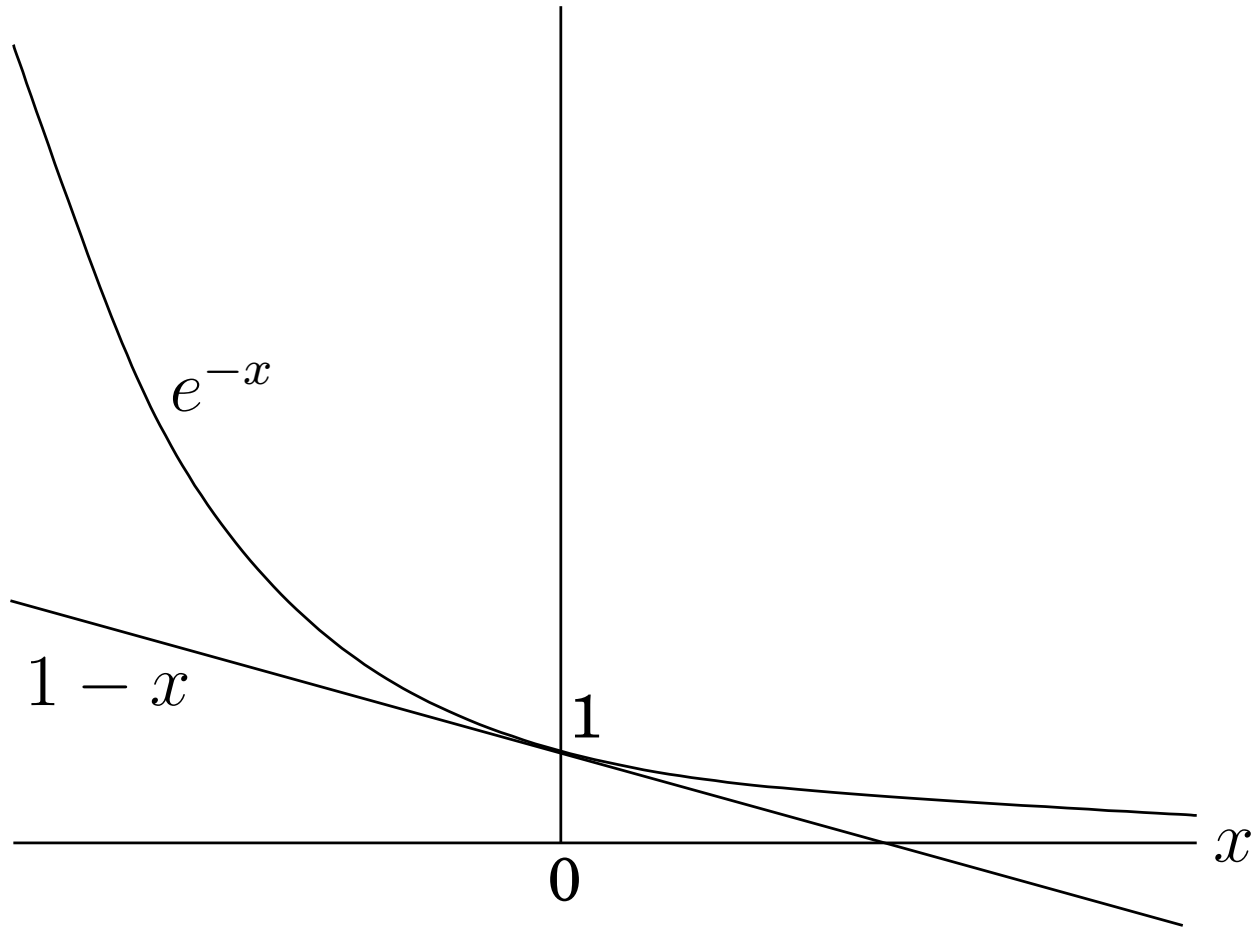
- Xét n_t là số phần tử vẫn chưa được phủ sau t lần lặp của thuật toán tham lam. Ta có $n_0 = n$.
- Vì các phần tử này bị phủ bởi k tập tối ưu, nên phải có một tập với ít nhất n_t/k phần tử trong số k tập này.
- Vậy chiến lược tham lam đảm bảo

$$n_{t+1} \leq n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right) < n_{t-1}\left(1 - \frac{1}{k}\right)^2 \quad < n_0\left(1 - \frac{1}{k}\right)^{t+1}$$

- Do $1 - x \leq e^{-x}$, ta được

$$n_t \leq n_0\left(1 - \frac{1}{k}\right)^t < n_0(e^{-1/k})^t = ne^{-t/k}$$

- Khi $t = k\ln n$, thì $n_t < ne^{-\ln n} = 1$, tức là không còn phần tử nào cần phủ.



Hình: Chỉ ra rằng $1 - x \leq e^{-x}$ với mọi x, đẳng thức nếu và chỉ nếu $x = 0$.

Tỉ suất của thuật toán tham lam

- Tỉ lệ giữa nghiệm của thuật toán tham lam và nghiệm tối ưu thay đổi theo dữ liệu vào nhưng luôn nhỏ hơn $\ln n$.
- Có một số input tỉ lệ rất gần với $\ln n$.
- Ta gọi tỉ lệ lớn nhất là ***tỉ suất của thuật toán tham lam***.

Nhận xét

Dưới một số giả sử được thừa nhận rộng rãi, ta có thể chứng minh được rằng: không có thuật toán thời gian đa thức với tỉ suất xấp xỉ nhỏ hơn.