ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

THIẾT KẾ VÀ ĐÁNH GIÁ THUẬT TOÁN

Bài 6
Phương pháp tham lam
(Greedy Method)

Nguyễn Thị Hồng Minh
minhnth@gmail.com

Nội dung

1. **Ý tưởng phương pháp**
2. **Mô hình thuật toán**
3. **Một số bài toán điển hình**
4. **Một số bài toán đồ thị giải bằng phương pháp tham**
   - Thuật toán Prim tìm cây bao trùm tối thiểu
   - Thuật toán Kruskal tìm cây bao trùm tối thiểu
   - Thuật toán Dijkstra tìm đường đi ngắn nhất

Ngoài lề

- “ ‘Greed,’ for lack of a better word, is good!”

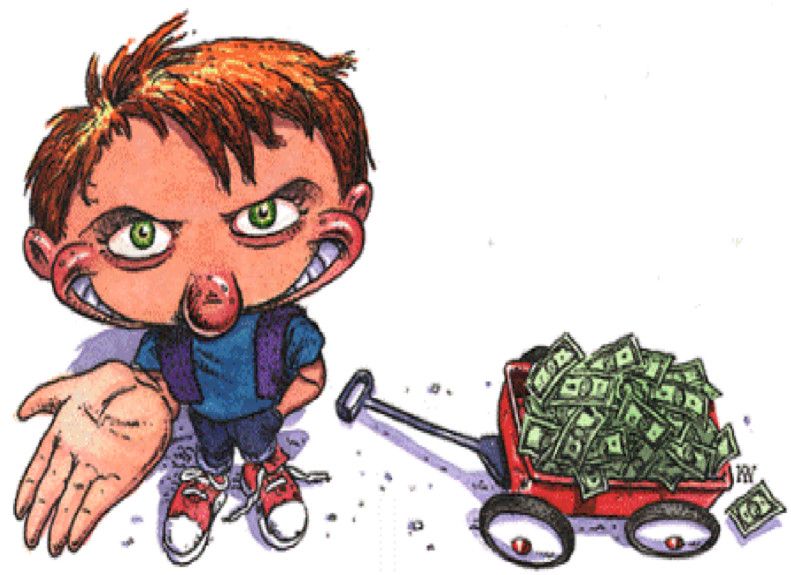
  Michael Douglas, diễn viên trong vai Gordon Gecko, phim *Wall Street*, 1987.

http://en.wikipedia.org/wiki/Gordon_Gekko

Ý tưởng phương pháp

- **Ý tưởng phương pháp tham lam**

  Xây dựng lời giải của bài toán với việc chấp nhận những lựa chọn "có vẻ" tốt nhất của từng giai đoạn. Tức là chấp nhận sự lựa chọn tối ưu cục bộ với hi vọng lựa chọn này sẽ dẫn đến lời giải tối ưu toàn cục cho bài toán.



Ý tưởng phương pháp

- **Ví dụ**

Xếp số tiền 78 (nghìn đồng) với các tờ tiền mệnh giá 50, 10, 5 và 1 nghìn đồng sao cho số tờ tiền là ít nhất?

Áp dụng chiến lược tham lam

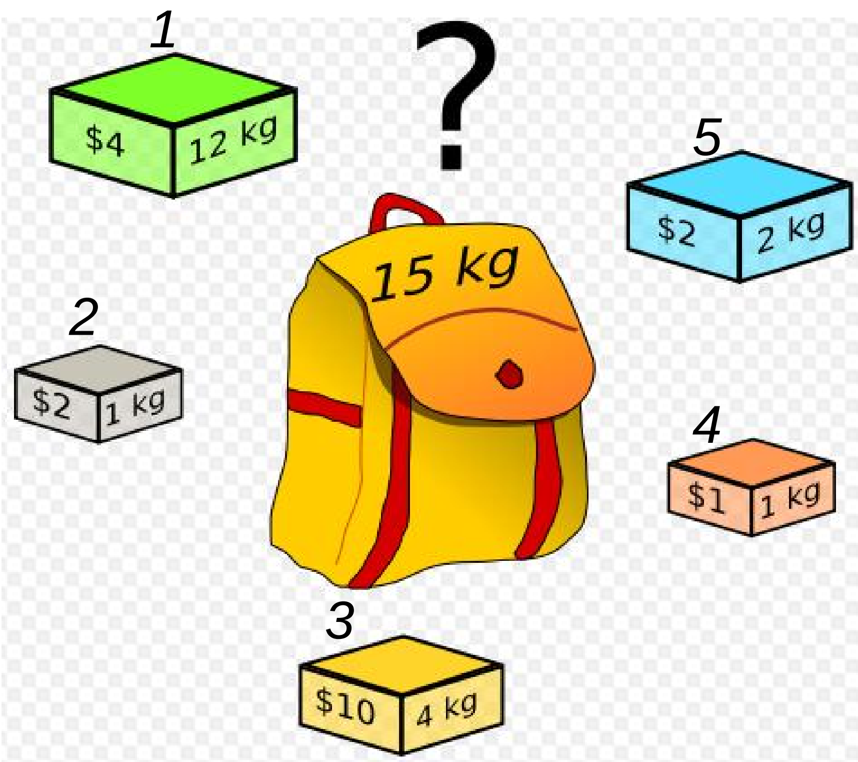
- Sắp các tờ tiền theo thứ tự giảm dần của mệnh giá
- Lấy tối đa số tờ mệnh giá lớn nhất có thể

=> 78 = 1 tờ 50 + 2 tờ 10 + 1 tờ 5 + 3 tờ 1

= 7 tờ

Ý tưởng phương pháp

- **Ví dụ:** Knapsack Problem



Áp dụng chiến lược tham lam

-Sắp các đồ vật theo thứ tự giảm dần của tỉ số giá trị/trọng lượng

3; 2; 5; 4; 1

- Lấy tối đa số đồ vật có tỉ số giá trị lớn nhất

15kg = 3 x 4kg + 3 x 1kg

=> 3 vật số 3, 3 vật số 2

Tổng giá trị: 36$

Ý tưởng phương pháp

- **Chiến lược tham luôn đưa ra kết quả tối ưu?**
  - Câu trả lời là "Không!"
  - Ví dụ

  Bài toán đổi tiền: Mệnh giá 10, 6, 1. Số tiền đổi là 55

  Tham: 55 = 5 x 10 + 5 x 1 => 10

  Không tham: 55 = 4 x 10 + 2 x 6 + 3 x 1 => 9

  55 = 3 x 10 + 4 x 6 + 1 x 1 => 8

Ý tưởng phương pháp

- **Hai tính chất mang lại hiệu quả của chiến lược tham**
  - Tính chất của sự lựa chọn tham lam: một giải pháp tối ưu toàn cục có thể đạt được bằng cách thực hiện những chọn lựa tối ưu cục bộ.
  - Tính chất cấu trúc con tối ưu: một giải pháp tối ưu của bài toán chứa trong nó các giải pháp tối ưu của bài toán con.
- **Để đạt được những hiệu quả đó thì sự lựa chọn ở mỗi bước cần thỏa mãn:**
  - Khả thi (Feasible): Thỏa mãn các ràng buộc của bài toán
  - Tối ưu cục bộ (Locally optimal): Là lựa chọn tốt nhất trong các lựa chọn khả thi
  - Không thể thay đổi (Irrevocable): Một khi đã chọn, thì lựa chọn không thể thay đổi ở các bước tiếp theo

Lược đồ phương pháp

- Các thành phần cơ bản của một thuật toán tham:
  - Input: *A* - Tập hợp các đối tượng.
  - Ouput: *S*- Nghiệm tối ưu của bài toán.

    $S=\{x_1,x_2,\ldots,x_n\}$; $x_i \in A$; $f(S)$ có giá trị tối ưu
  - Các hàm chức năng:
    - o Hàm lựa chọn để xác định đối tượng tốt nhất tham gia vào lời giải ở mỗi giai đoạn – x = *BestSelect*(A)
    - o Hàm xác định tính khả thi nếu đối tượng tốt nhất được lựa chọn có thể tham gia vào nghiệm của bài toán - *Acceptable*(S,x).
    - o Hàm đưa lựa chọn được chấp nhận vào nghiệm - *Integrate*(S,x).
    - o Hàm xác định một lời giải đã được xây dựng xong - *Is_solution*(S).

Lược đồ phương pháp

- **Lược đồ**

```
Greedy (A) ≡     //Tìm lời giải tối ưu trên bộ dữ liệu A
    S = ∅ ;
    while (A ≠ ∅ and !Is_solution(S))
        x = BestSelect(A);
        A = A \ {x};
        if (Acceptable(S, x))
            Integrate(S, x);

    endw;
    return S;
End.
```

Bài toán đổi tiền

- **Phân tích bài toán**

  Input: $A = \{a_1, a_2, \ldots a_n\}; T$ //a_i các mệnh giá tiền, T tổng số tiền cần đổi

  Ouput: $S = \{k_1, k_2, ..k_n\} \mid \sum_{i=1}^{n} k_i a_i = T; \sum_{i=1}^{n} k_i = \min$ //k_i Số tờ tiền mệnh giá a_i

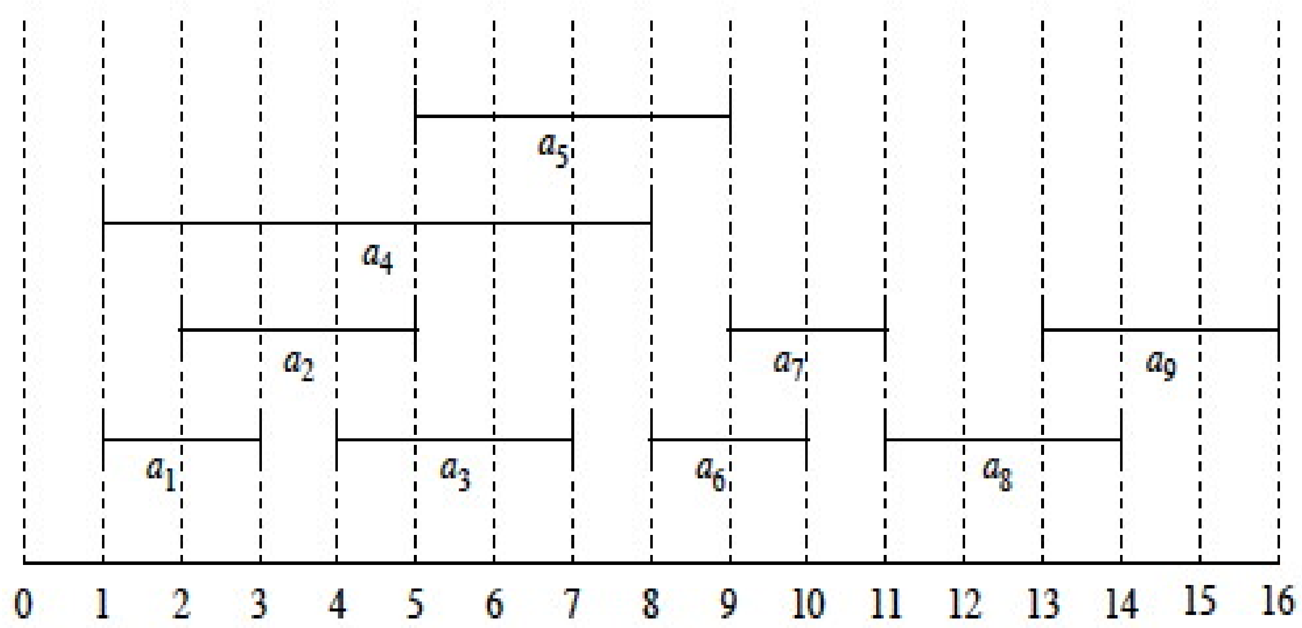
- **Lược đồ**

```
ChangeMoney (A,T) ≡
    for (i=1..n) k_i=0 ;
    i=1;
    while (i<=n and T>0)
        k_i= T/a_i; T -= k_i*a_i;
        i += 1;
    endw;
    if (T>0) return -1;  //Không có lời giải
    else return {k_1,k_2,…k_n}
End.
```

Bài toán lập lịch sử dụng tài nguyên

- **Activity-selection problem:** *n* hoạt động cùng sử dụng 1 tài nguyên. Mỗi hoạt động thứ *i* diễn ra trong thời gian s_i tới f_i ($s_i < f_i$) và toàn quyền sử dụng tài nguyên trong thời gian đã sắp xếp. Tìm dãy nhiều nhất các hoạt động có thể thực hiện được?
- **Ví dụ**

i	s_i	f_i
1	1	3
2	2	5
3	4	7
4	1	8
5	5	9
6	8	10
7	9	11
8	11	14
9	13	16



Bài toán lập lịch sử dụng tài nguyên

- **Phân tích**:

  Input: $A = \{a_i = (s_i, f_i); s_i < f_i ; i=1\ldots n\}$

  Ouput: $S = \{a_1'; a_2';\ldots a_m' \mid a_i' \in A; f_i' <= s'_{i+1}; m - \max\}$
- **Chiến lược cho lựa chọn tham lam**
  - Nếu chọn hoạt động kết thúc sớm nhất sẽ có cơ hội chọn được nhiều hoạt động phía sau.
  - Sắp xếp dãy các hoạt động theo thứ tự tăng dần của thời gian kết thúc ($f_i < f_{i+1}$, $i=1..n-1$)
  - Tại mỗi bước xây dựng chuỗi hoạt động, chọn hoạt động có thể tiếp tục sau dãy hoạt động đã chọn và có thời gian kết thúc sớm nhất.

    Ở bước i, dãy đã chọn $a_1'; a_2';\ldots a_{i-1}'$ cần chọn $a_i' \in A_i = A \backslash \{a_1'; a_2';\ldots a_{i-1}'\}$, chọn a_i' là $a_i \in A$ sao cho $s_i >= f_{i-1}'$, f_i = min trong Ai
  - Bắt đầu với a_1.

Bài toán lập lịch sử dụng tài nguyên

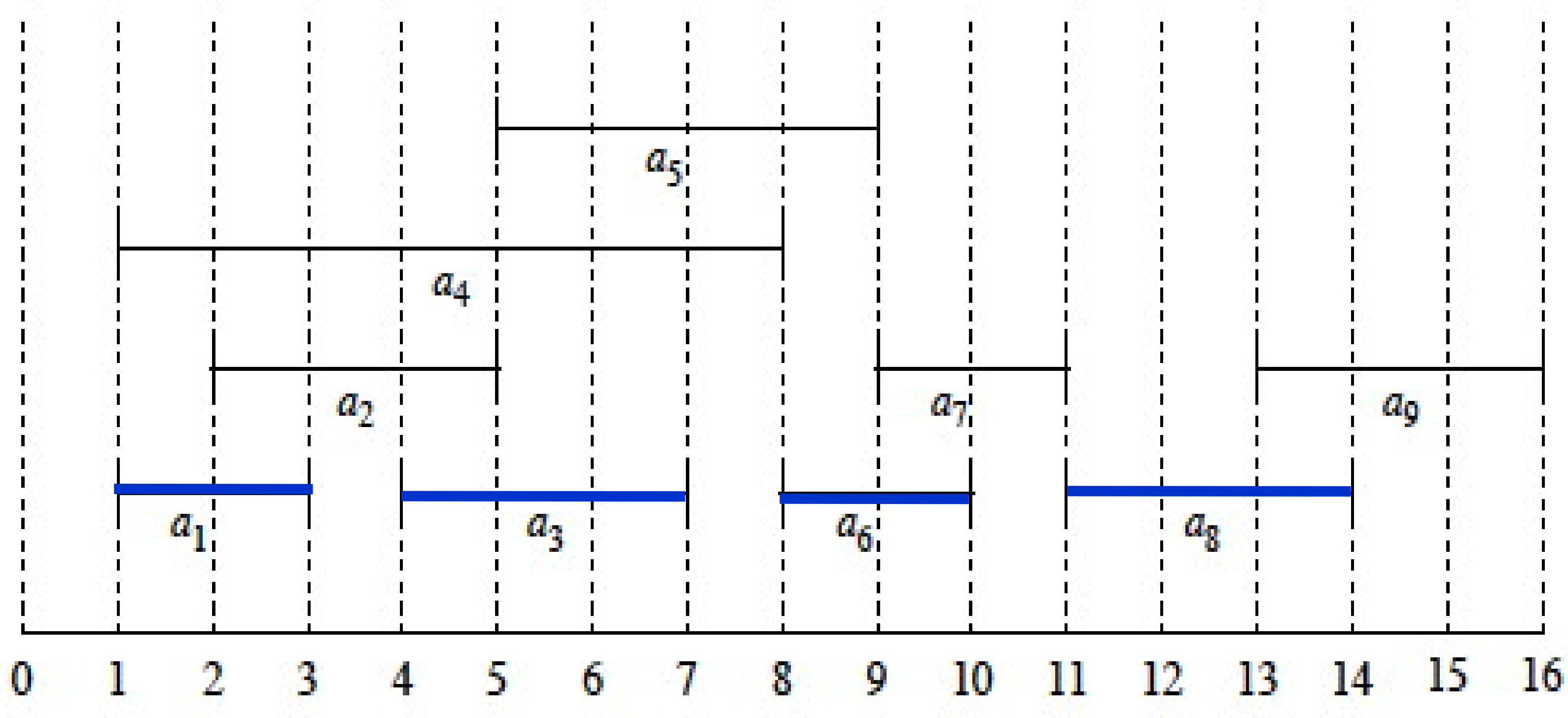
- **Thuật toán:**

```
Activity_Selector(A) ≡
    S={a1}
    j=1;
    for (i=2..n)
       if (si>=fj)
          S = S ∪ {ai};   //∪: Ghép vào sau dãy S đã chọn
          j = i;
       endi;
    endf;
 End.
```

Chú ý: Dãy các hoạt động $A=\{a_1,a_2,\ldots a_n\}$ đã được sắp xếp để $f_i<f_{i+1}$

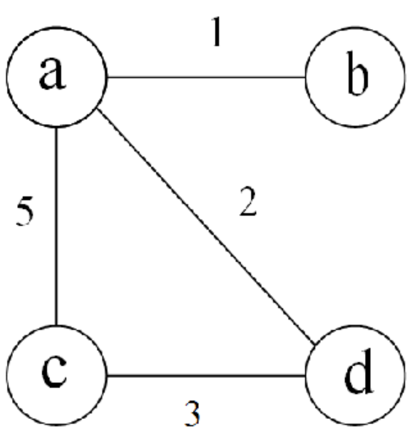
Bài toán lập lịch sử dụng tài nguyên

- **Minh họa**

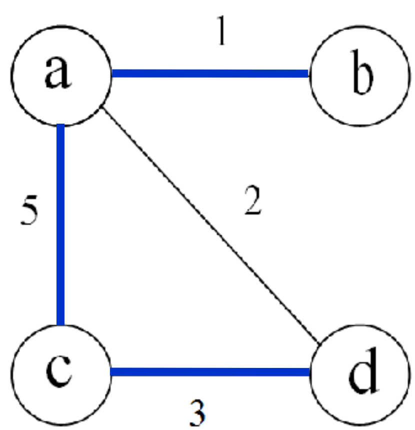


Bài toán cây bao trùm tối thiểu

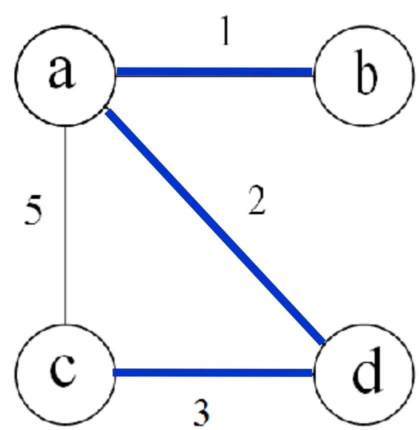
- **Bài toán cây bao trùm tối thiểu** (Minimum Spanning Tree – MST)
  - Cây bao trùm (Spanning Tree) của một đồ thị G là một đồ thị con liên thông, không có chu trình (cây) có chứa tất cả các đỉnh của G.
  - Cây bao trùm tối thiểu (Minimum Spanning Tree) của một đồ thị liên thông có trọng số G là cây bao trùm có tổng trọng số các cạnh là nhỏ nhất.



G



$W(T_1)=9$



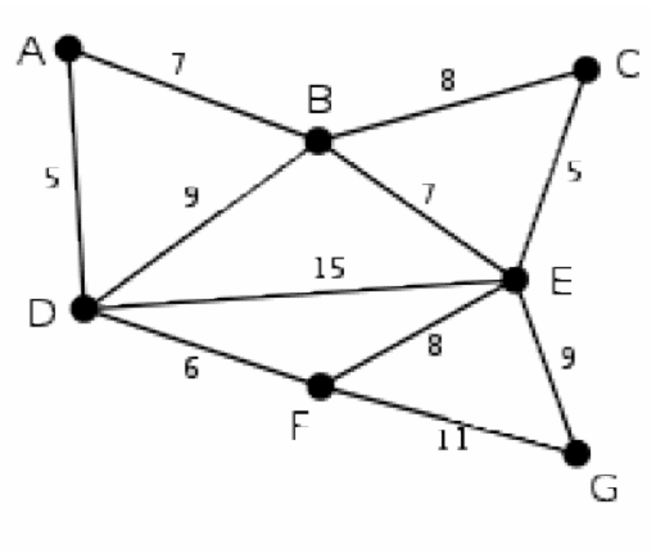
$W(T_1)=6$

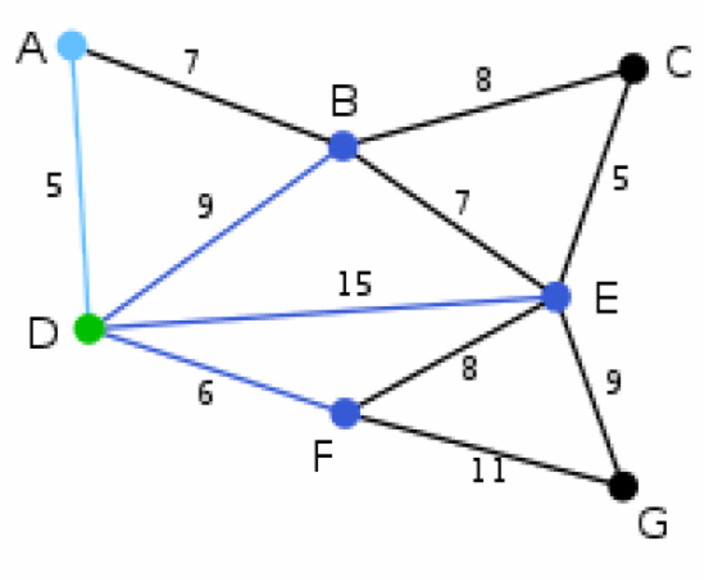
Bài toán cây bao trùm tối thiểu

- **Thuật toán Prim**
  - Bắt đầu với cây chỉ có 1 đỉnh T_0.
  - Phát triển cây theo các bước, mỗi bước thêm một đỉnh vào cây đã có bằng một cạnh. Dãy các cây được phát triển T_1, T_2 , . . . , T_{n-1}.
  - Chiến lược tham lam: Tại mỗi bước dựng cây T_{i+1} từ cây T_i với việc thêm vào đỉnh "gần nhất".
  - Đỉnh gần nhất với T_i: Đỉnh không thuộc T_i và được nối với T_i bằng cạnh có trọng số nhỏ nhất.
  - Thuật toán dừng lại khi tất cả các đỉnh đã được thêm vào.

Bài toán cây bao trùm tối thiểu

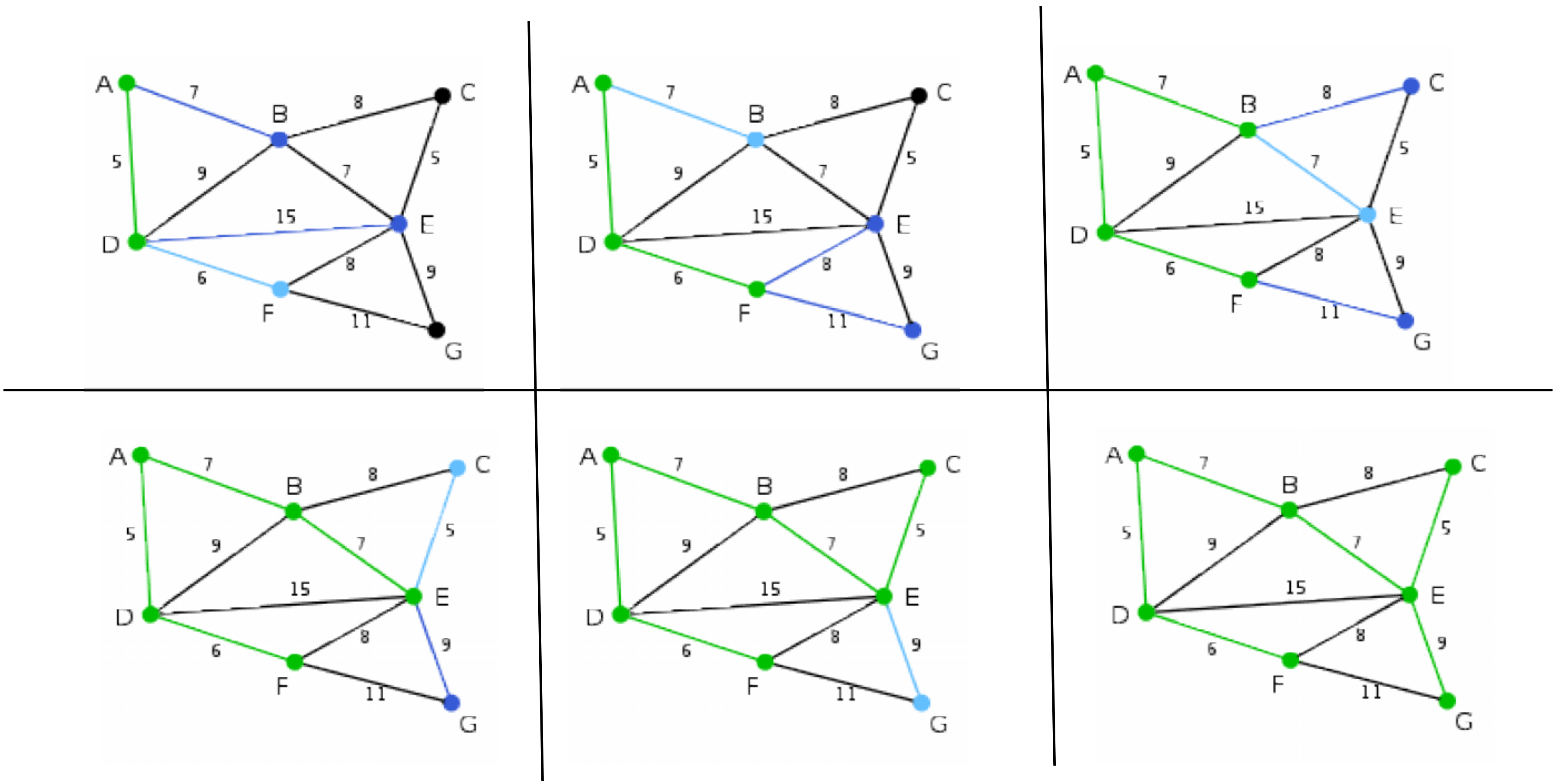
- **Thuật toán Prim**





Bài toán cây bao trùm tối thiểu

- **Thuật toán Prim**



http://en.wikipedia.org/wiki/Prim’s_algorithm

Bài toán cây bao trùm tối thiểu

- **Lược đồ thuật toán Prim**

Prim(G) ≡

// Input: G = (V, E)

// Output: E_T , tập các cạnh của cây bao trùm tối thiểu của G

$V_T = \{v_0\}$

$E_T = \emptyset$

for i=1..|V| − 1

tìm cạnh có trọng số nhỏ nhất $e^*=(v^*,u^*)$ trong tất cả các cạnh (v, u) mà $v \in V_T$ và $u \in V - V_T$;

$V_T = V_T \cup \{u^*\}$

$E_T = E_T \cup \{e^*\}$

endf;

return E_T;

End.

Bài toán cây bao trùm tối thiểu

- **Chứng minh tính đúng**

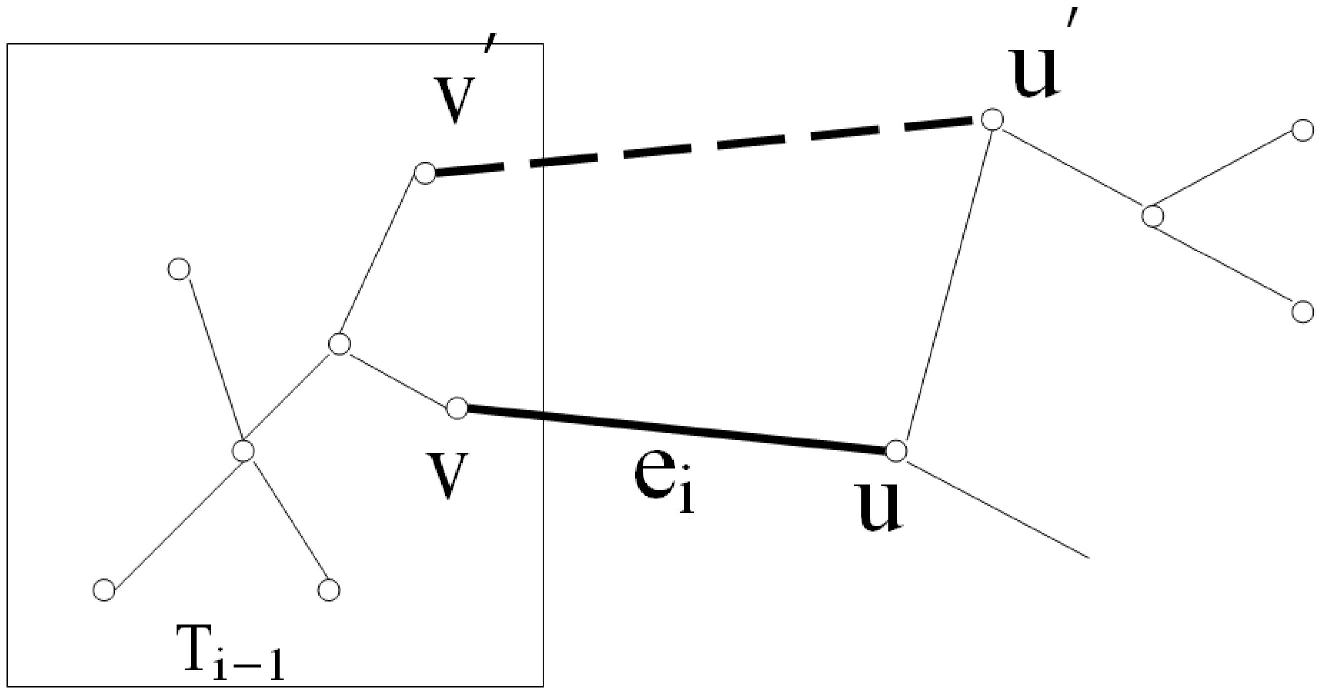
  Sử dụng phương pháp quy nạp và phản chứng

  - Chứng minh bằng quy nạp: mỗi cây con T_i, $i = 0, \ldots, n-1$, sinh ra bằng thuật toán Prim là một phần của cây bao trùm tối thiểu. Giả sử T_{i-1} là một phần của cây bao trùm tối thiểu thì T_i (sinh ra bằng cách thêm vào cạnh gần nhất) cũng là một phần của cây bao trùm tối thiểu.
  - Phản chứng: giả sử cây bao trùm tối thiểu của G không chứa T_i, t.l có cạnh $e_i = (v, u)$ là cạnh có trọng số nhỏ nhất nối một đỉnh thuộc T_{i-1} với 1 đỉnh không thuộc T_{i-1}, và e_i không thuộc cây tối thiểu T (theo Prim).
  - Nếu thêm e_i vào T sẽ tạo ra một chu trình, chu trình sẽ chứa cạnh (v', u') nối một đỉnh v' □ $T_i - 1$ với 1 đỉnh $u' \notin T_{i-1}$. Nếu xóa đi cạnh (v′, u′) của chu trình này sẽ nhận được một cây bao trùm khác chứa e_i có trọng số nhỏ hơn T.

    Như vậy trái với giả thiết cây bao trùm tối thiểu không chứa T_i => đpcm

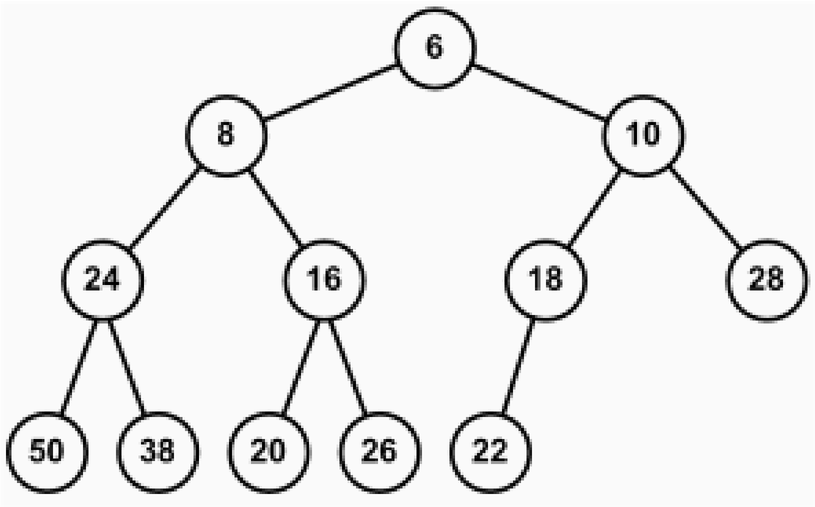
Bài toán cây bao trùm tối thiểu

- **Chứng minh tính đúng**



Bài toán cây bao trùm tối thiểu

- **Đánh giá độ phức tạp thuật toán**
  - Xác định đỉnh "gần nhất":
    - Sử dụng ma trận kề biểu diễn đồ thị: Duyệt đỉnh với độ phức tạp O(V^2)
    - Sử dụng cây nhị phân đống nhỏ nhất (min heap) và danh sách kề biểu diễn đồ thị: O(E log V)

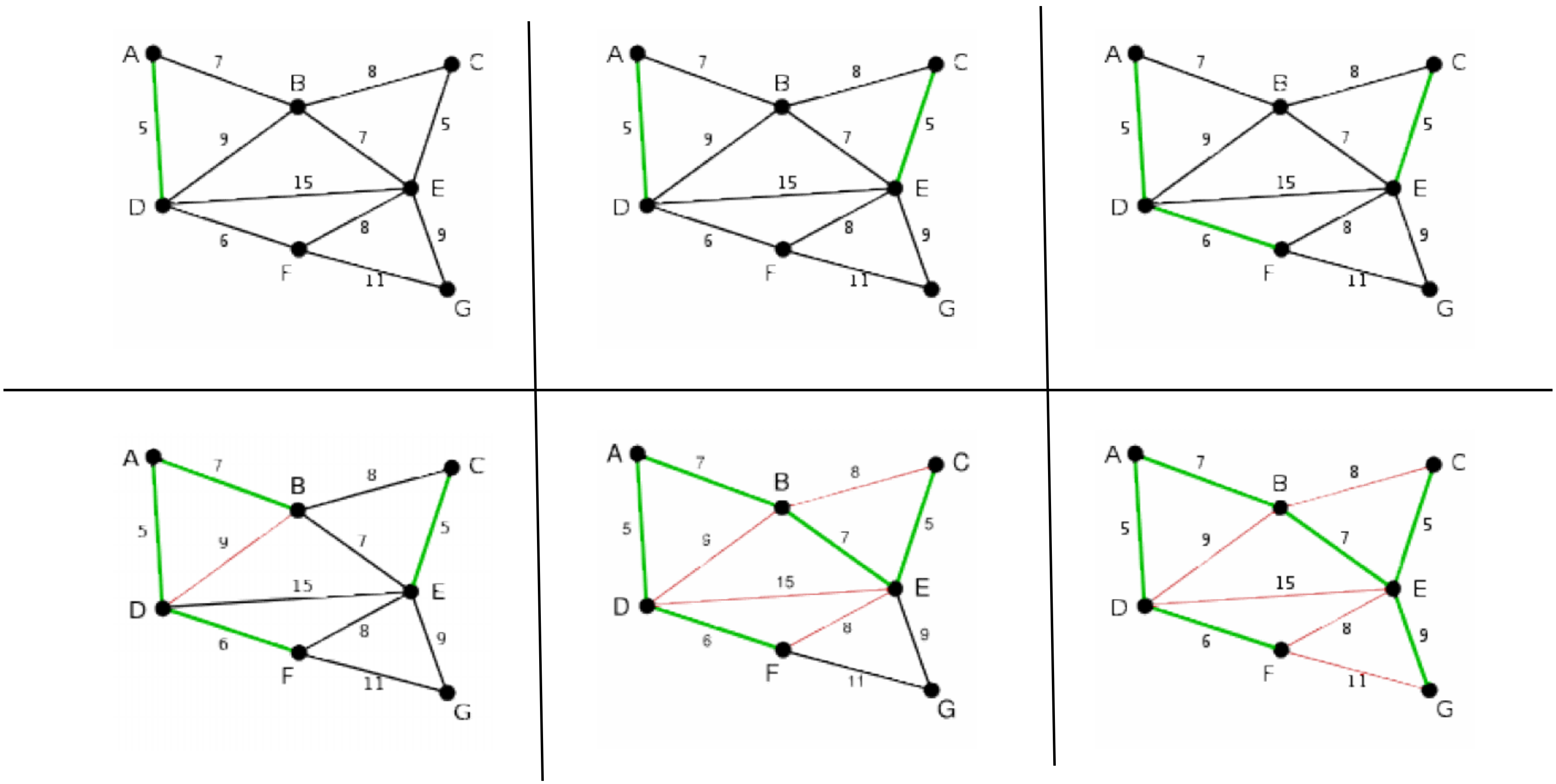


Bài toán cây bao trùm tối thiểu

- **Thuật toán Kruskal**
  - Các cạnh được sắp xếp theo thứ tự tăng dần của trọng số.
  - Bắt đầu bằng một rừng (forest) rỗng.
  - Xây dựng MST theo các bước, mỗi bước thêm một cạnh
    - Trong quá trình dựng MST luôn có một "rừng": các cây không liên thông.
    - Thêm vào cạnh có trọng số nhỏ nhất trong các cạnh chưa thêm vào cây và không tạo thành chu trình.
    - Như vậy tại mỗi bước một cạnh có thể:
      - Mở rộng một cây đã có
      - Nối hai cây thành một cây mới
      - Tạo cây mới
  - Thuật toán dừng lại khi tất cả các đỉnh đã được thêm vào.

Bài toán cây bao trùm tối thiểu

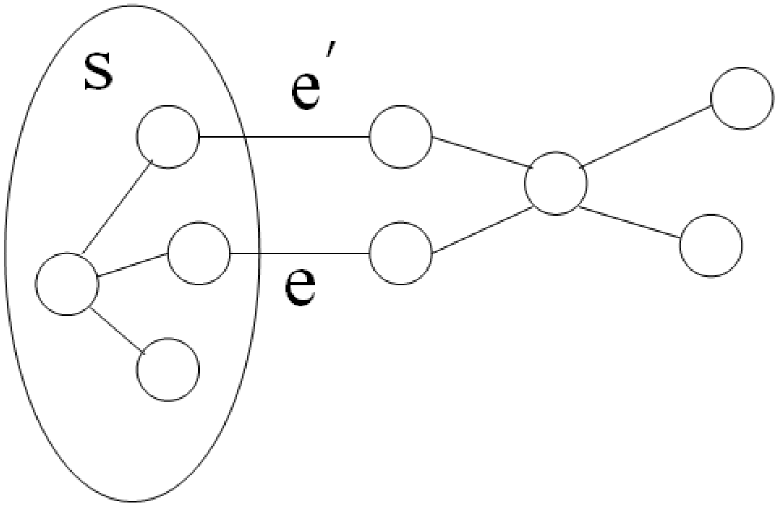
- **Thuật toán Kruskal**



http://en.wikipedia.org/wiki/Kruskal%27s_algorithm

Bài toán cây bao trùm tối thiểu

- **Thuật toán Kruskal**
  - Chứng minh tính đúng của thuật toán
    - Thuật toán sinh được cây bao trùm
    - Cây bao trùm là nhỏ nhất



  - Độ phức tạp thuật toán
    - $O(E \log E) = O(E \log V)$

Một số chủ đề tiểu luận giữa kì

1. Chứng minh tính đúng của thuật toán (quy nạp và bất biến vòng lặp): nguyên tắc, ví dụ.
2. Phương pháp đệ quy: khái niệm, nhận dạng bài toán, lược đồ phương pháp, đánh giá độ phức tạp, ví dụ.
3. Khử đệ quy: các dạng khử đệ quy, ví dụ.
4. Phương pháp chia để trị: ý tưởng, mô hình, lược đồ phương pháp, đánh giá độ phức tạp, ví dụ.
5. Phương pháp vét cạn: nguyên lí, mô hình, lược đồ phương pháp, ví dụ.
6. Phương pháp quay lui: ý tưởng, lược đồ phương pháp, chứng minh tính đúng, ví dụ.
7. Phương pháp nhánh cận: ý tưởng, nguyên lí đánh giá cận, mô hình, lược đồ phương pháp, ví dụ.
8. Phương pháp tham lam: ý tưởng, lược đồ phương pháp, ví dụ.
9. Bài toán cây bao trùm tối thiểu bằng các phương pháp: vét cạn, nhánh cận, tham lam
10. Bài toán tìm đường đi ngắn nhất: vét cạn, nhánh cận, tham lam
11. Bài toán người đưa hàng: vét cạn, quay lui, tham lam.
12. Bài toán xếp balo: vét cạn, đệ quy, quay lui, tham lam
13. Chủ đề tự chọn…