PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN (Design and Analysis of Algorithms)

L/O/G/O

GV: HUỲNH THỊ THANH THƯƠNG

Email: hh.thanhthuong@gmail.com

thuonghtt@uit.edu.vn

CHƯƠNG 3

THIẾT KẾ THUẬT TOÁN

Algorithm Design

GV: ThS. HUỲNH THỊ THANH THƯƠNG
Email:
thuonghtt@uit.edu.vn

Nội dung

- Phương pháp chia để trị, giảm để trị, biến đổi để trị
- Phương pháp tham lam

  (Greedy method)
- Phương pháp quay lui, nhánh cận, cắt tỉa
- Phương pháp quy hoạch động
- Các phương pháp khác

Bài toán tối ưu tổ hợp

❖ Cách giải quyết

- Vét cạn
- Các thuật toán Quy hoạch tuyến tính

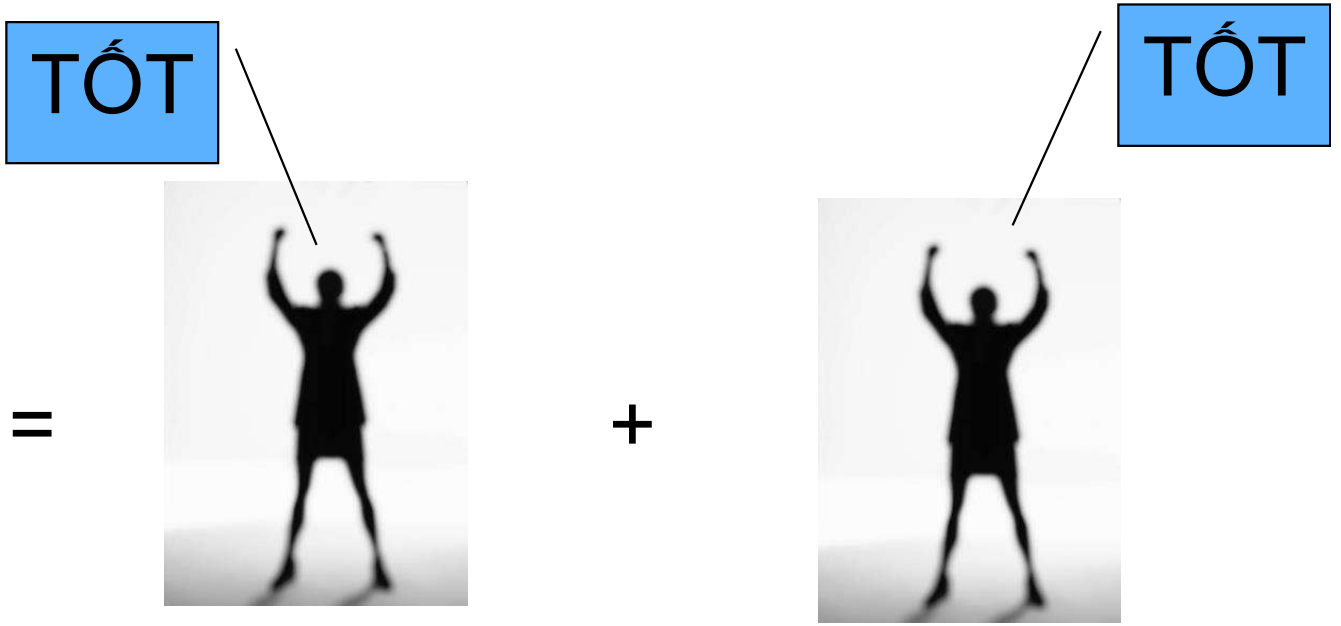
(Toán học: ngành tối ưu)

- Tối ưu cục bộ → phương pháp tham lam

- Đi qua 1 số bước với 1 tập các khả năng lựa chọn cho mỗi bước
- Tại mỗi bước, chọn một khả năng được xem là tốt nhất tại lúc đó

Phương pháp tham lam

Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán, dựa vào đó chọn lựa hành động tốt nhất của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.



Properties of Greedy Algorithms

Các bài toán có thể giải được bằng phương pháp tham lam có 2 tính chất/đặc trưng sau (dấu hiệu nhận biết):

- Optimal Substructure

- Greedy choice property = thiết kế tiêu chuẩn tối ưu cục bộ

Phương pháp tham lam

❖Ưu điểm:

- Đơn giản, dễ cài đặt
- Tốc độ nhanh (thời gian đa thức)

Phương pháp tham lam

❖Khuyết điểm:

- Chưa chắc cho lời giải chính xác (thường cho phương án tốt chứ chưa hẳn là tối ưu).
- Không phải luôn chấp nhận được (có thể cho lời giải tệ)
- Khó chứng minh tính đúng, nếu chứng minh được thì nó là thuật toán

Greedy Algorithm

```
GREEDY(P,n)
{                          // P(1:n) contains the n inputs//
   solution ← ϕ //initialize the solution to empty//
   for i ← 1 to n
   {
         x ← SELECT(P)
         P = P-{x}
         if FEASIBLE(solution,x)
                  solution ← UNION(solution,x)
   }
   return(solution)
}
```

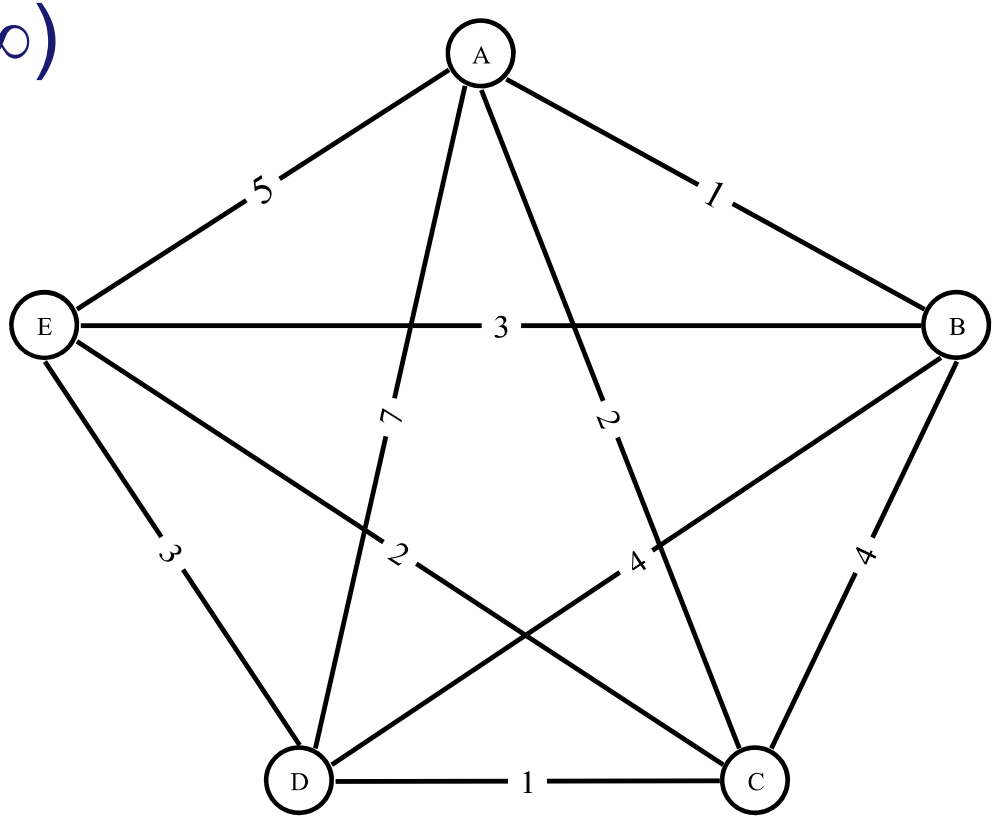
Phương pháp tham lam

❖ Ví dụ 1: Bài toán người giao hàng (du lịch, đưa thư)

Traveling Salesman Problem - TSP

Traveling Salesman Problem

- Bài toán người giao hàng
  - Giả thiết: đồ thị đủ (nếu thực tế không có đường đi thì cho trọng số là ∞)
  - Ý tưởng ?
  - Viết thuật toán ?
  - Độ phức tạp ?
  - Cài đặt ?



Traveling Salesman Problem

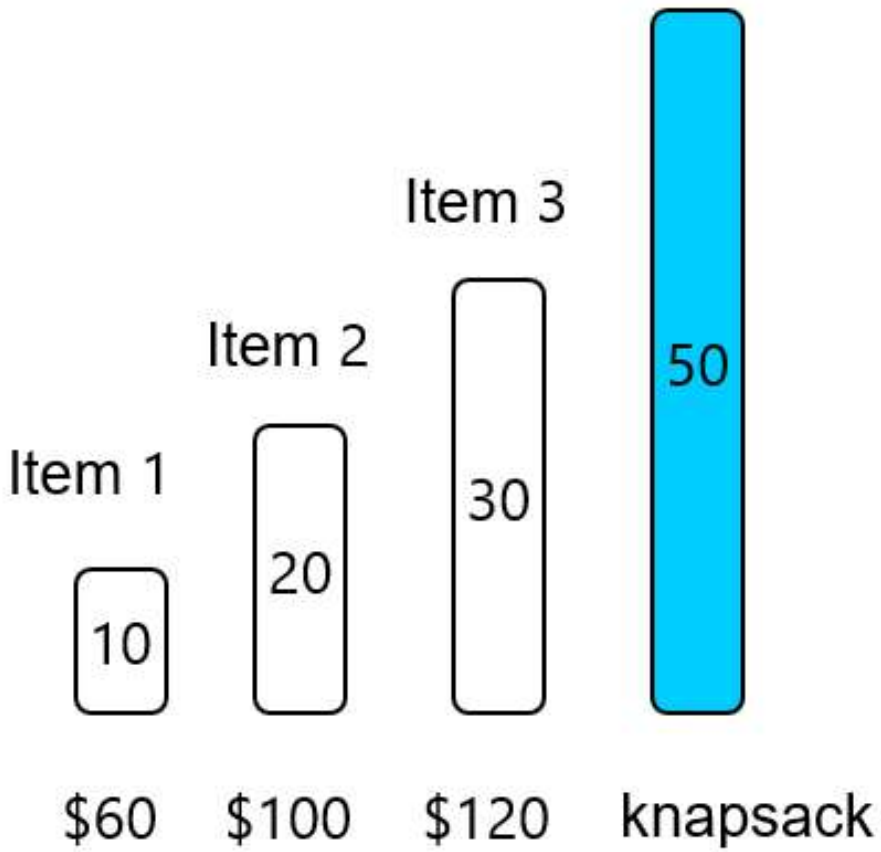
❖Thuật giải: Cách 1

P = {các đỉnh trên đồ thị}/ {S}
G = [S]
N = S; // đỉnh hiện hành
While (P ≠ ∅)
- Chọn đỉnh M trong P có khoảng cách tới N là nhỏ nhất
- Cập nhật các đối tượng để chọn: P = P/ {M}
- Cập nhật M vào G
- N = M;

Thêm S vào G

Phương pháp tham lam

❖ Ví dụ 2: Bài toán ba lô (Knapsack Problem) – ăn trộm



Phương pháp tham lam

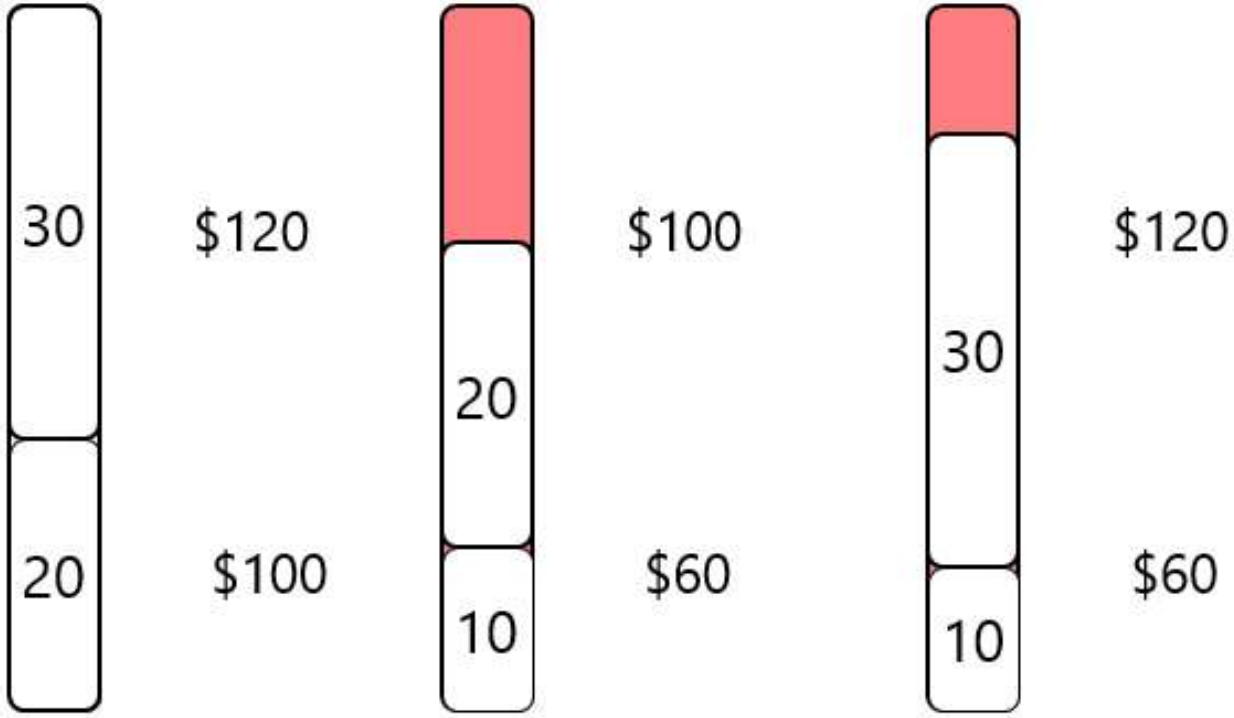
❖ Ví dụ 2: Knapsack 0/1 – Dạng 1

Cho n đồ vật và một cái ba lô có thể đựng trọng lượng tối đa *M*, mỗi đồ vật *i* có trọng lượng w_i và giá trị là p_i .

Chọn một cách lựa chọn các đồ vật cho vào túi sao cho trọng lượng không quá M và tổng giá trị là lớn nhất.

Mỗi đồ vật hoặc là lấy đi hoặc là bỏ lại

Knapsack 0/1



Total =$220 =$160 =$180

Knapsack 0/1

maximize z = $\sum_{1 \le i \le n} p_i x_i$

thỏa $\sum_{1 \le i \le n} w_i x_i \le M$

và $x_i \in \{0, 1\}$

Knapsack 0/1

❖Ví dụ 2: Ba lô có sức chứa là 100 kg và 5 đồ vật

Price ($)	20	30	65	40	50		
weight (kg)	10	20	30	40	50		

❖ Cách 1: chọn vật có trọng lượng nhỏ nhất trước (bad)

- Total Weight = 10 + 20 + 30 + 40 = 100
- Total Price = 20 + 30 + 65 + 40 = 155

Knapsack 0/1

Price ($)	20	30	65	40	50		
weight (kg)	10	20	30	40	50		

- Cách 2: chọn vật có giá trị cao nhất trước (bad)

  - Total Weight = 30 + 50 + 20 = 100
  - Total Price = 65 + 50 + 30 = 145

Knapsack 0/1

Price ($)	20	30	65	40	50
weight (kg)	10	20	30	40	50
price/weight	2	1,5	2,1	1	1

❖Cách 3: chọn vật có price/ weight lớn nhất trước

- Total weight = 30 + 10 + 20 + 40 = 100
- Total Price = 65 + 20 + 30 + 40 = 155

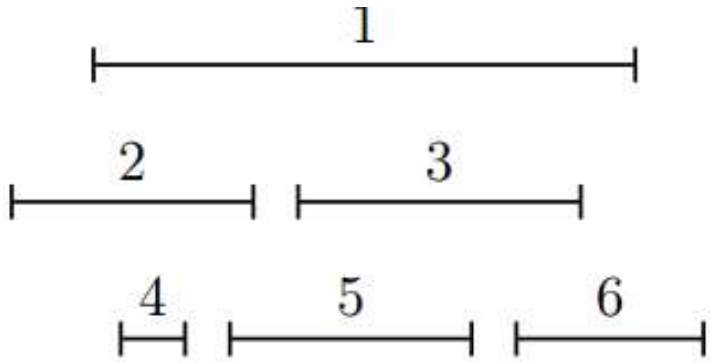
Phương pháp tham lam

❖ Ví dụ 3: Bài toán chọn hoạt động (Activity-Selection Problem/ Interval Scheduling)

- Cho một tập các hoạt động S = {1, 2, …, n}
- Một hoạt động i có thời điểm bắt đầu là s_i và thời điểm chấm dứt là f_i, $s_i < f_i$
- Nếu hoạt động i được chọn thì i tiến hành trong thời gian $[s_i, f_i)$

Phương pháp tham lam

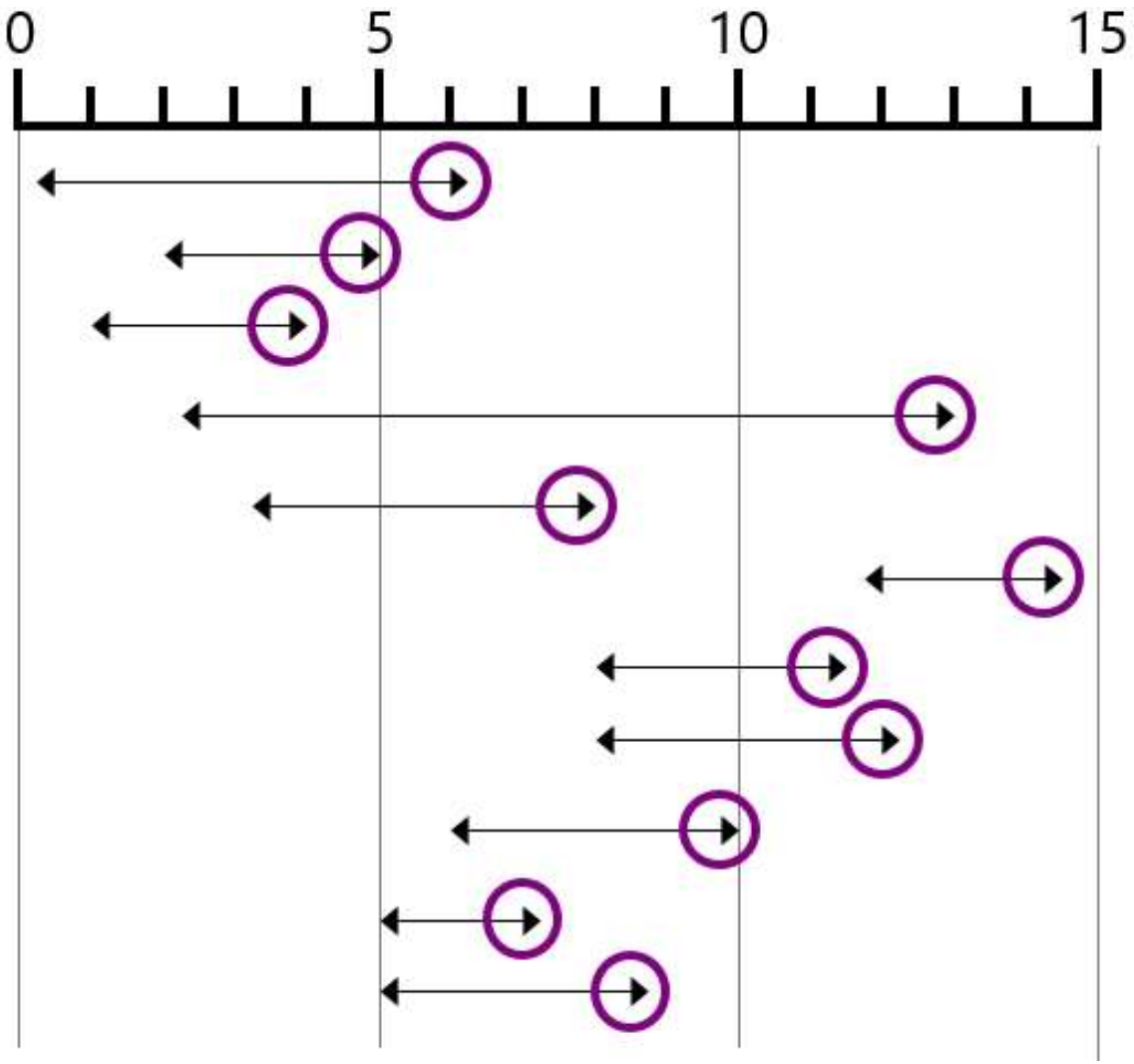
- Bài toán chọn hoạt động
  - Hai hoạt động i và j là "tương thích nhau" (compatible) nếu $[s_i, f_i)$ và $[s_j, f_j)$ không chạm nhau i.e. $f_i \le s_j$ hoặc $f_j \le s_i$.
  - Yêu cầu: Tìm tập hợp lớn nhất các hoạt động tương thích nhau?



Bài toán thực tế?

Phương pháp tham lam

i	s_i	f_i
1	0	6
2	3	5
3	1	4
4	2	13
5	3	8
6	12	14
7	8	11
8	8	12
9	6	10
10	5	7
11	5	9



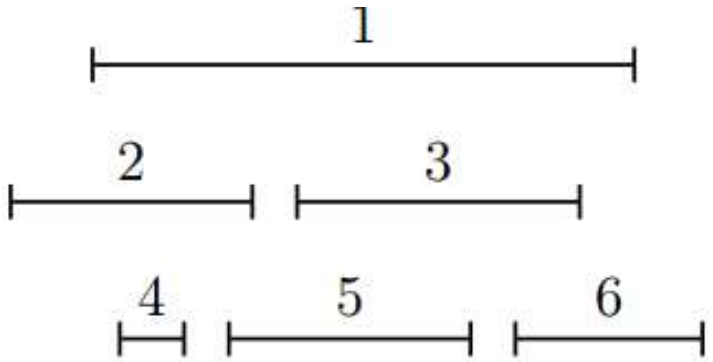
Interval Scheduling: Brute Force

- Xem xét tất cả các phương án (tập con các hoạt động) chấp nhận được
- Chọn tập con (chấp nhận được) lớn nhất
- Độ phức tạp: $\Theta(2^n)$

Greedy Interval Scheduling

❖ Ý tưởng:

1. Dùng 1 quy tắc đơn giản (?) để chọn 1 hoạt động i
2. Bỏ qua tất cả hoạt động không tương thích với i
3. Lặp lại cho đến khi tất cả hoạt động đều được xem xét



Greedy Interval Scheduling

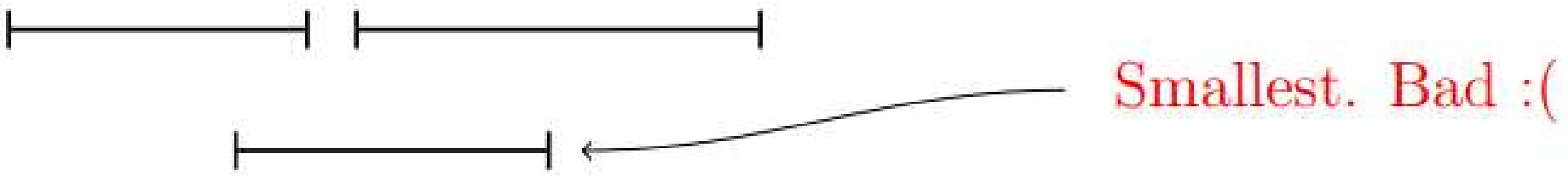
Những quy tắc có thể xem xét:

- Cách 1: Chọn hoạt động bắt đầu sớm nhất (s_i nhỏ nhất)

Greedy Interval Scheduling

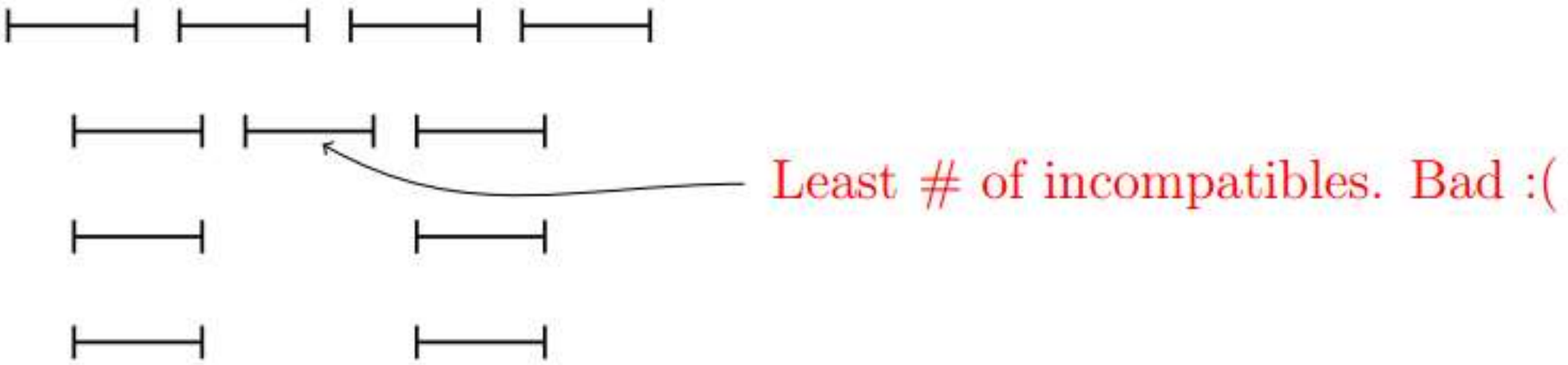
Những quy tắc có thể xem xét:

- Cách 2: Chọn hoạt động ngắn nhất ($f_i - s_i$ nhỏ nhất)



Greedy Interval Scheduling

❖Cách 3: Với mỗi hoạt động, tìm số hoạt động tương thích với nó và chọn hoạt động nào có số tương thích lớn nhất



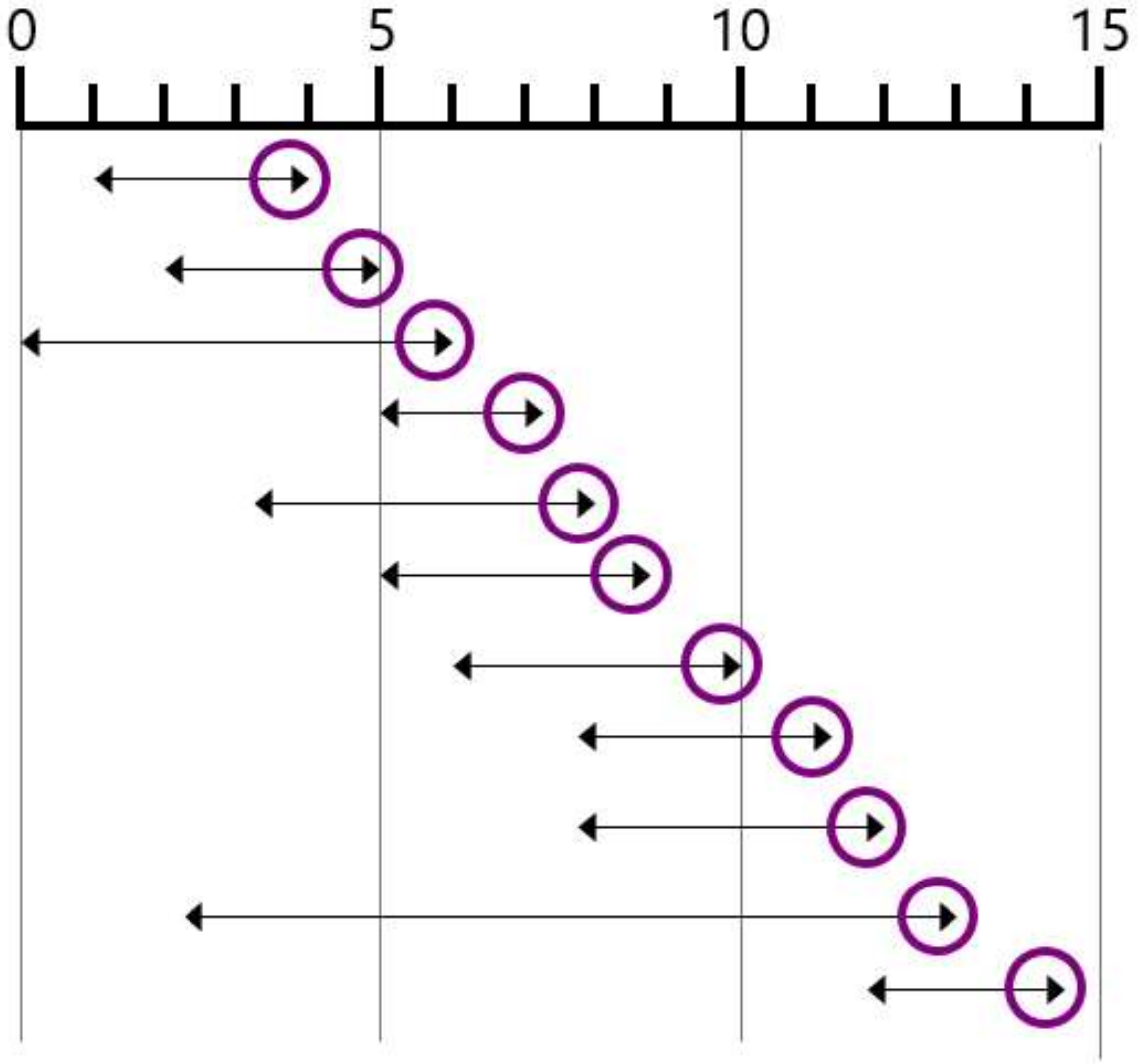
Greedy Interval Scheduling

Những quy tắc có thể xem xét:

- Cách 4: Chọn hoạt động hoàn thành sớm nhất (f_i nhỏ nhất)

Greedy Interval Scheduling

Sort by finish time



Ví dụ 4: Bài toán phân công công việc

Có n công việc, được phân công cho m máy như nhau thực hiện đồng thời. Hãy viết chương trình dùng kỹ thuật THAM LAM để tìm một phương án phân công sao cho thời gian để các máy hoàn thành hết n công việc là ngắn nhất.

INPUT

Dòng đầu tiên chứa hai số nguyên không âm n và m, giá trị không quá 1 tỷ

Dòng thứ hai chứa n số nguyên dương $x_0, x_1, ..., x_{n-1}$ trong đó x_i là thời gian cần thiết để một máy hoàn thành công việc i, giá trị không quá 1 tỷ

OUTPUT

Xuất trên một dòng n số nguyên dương $y_0, y_1, ..., y_{n-1}$, giá trị mỗi số trong đoạn [0, m-1] trong đó giá trị của y_i là số thứ tự của máy được phân công để thực hiện công việc i.

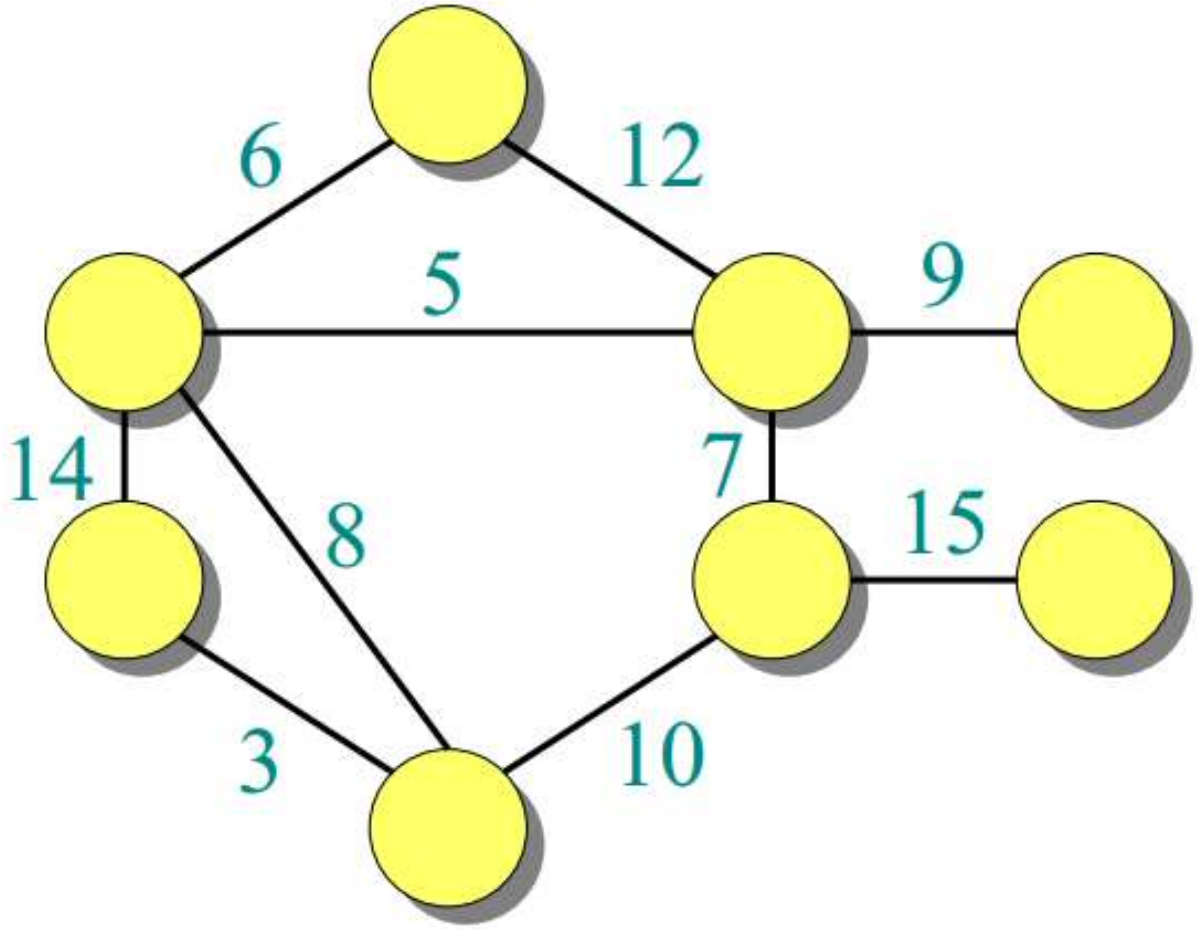
VÍ DỤ

Input	Output
8 3 79 1 80 59 75 51 7 29	1 1 0 2 2 1 2 0 Một đáp án khác (theo kỹ thuật tham lam): 1 0 0 2 2 1 0 0

Một số ví dụ khác

- Minimum spanning tree : Prim's algorithm và Kruskal's algorithm
- Euler cycle
- Shortest paths: Dijkstra's algorithm, …

Ví dụ 5: Kruskal's / Prim's algorithm



Ví dụ 6

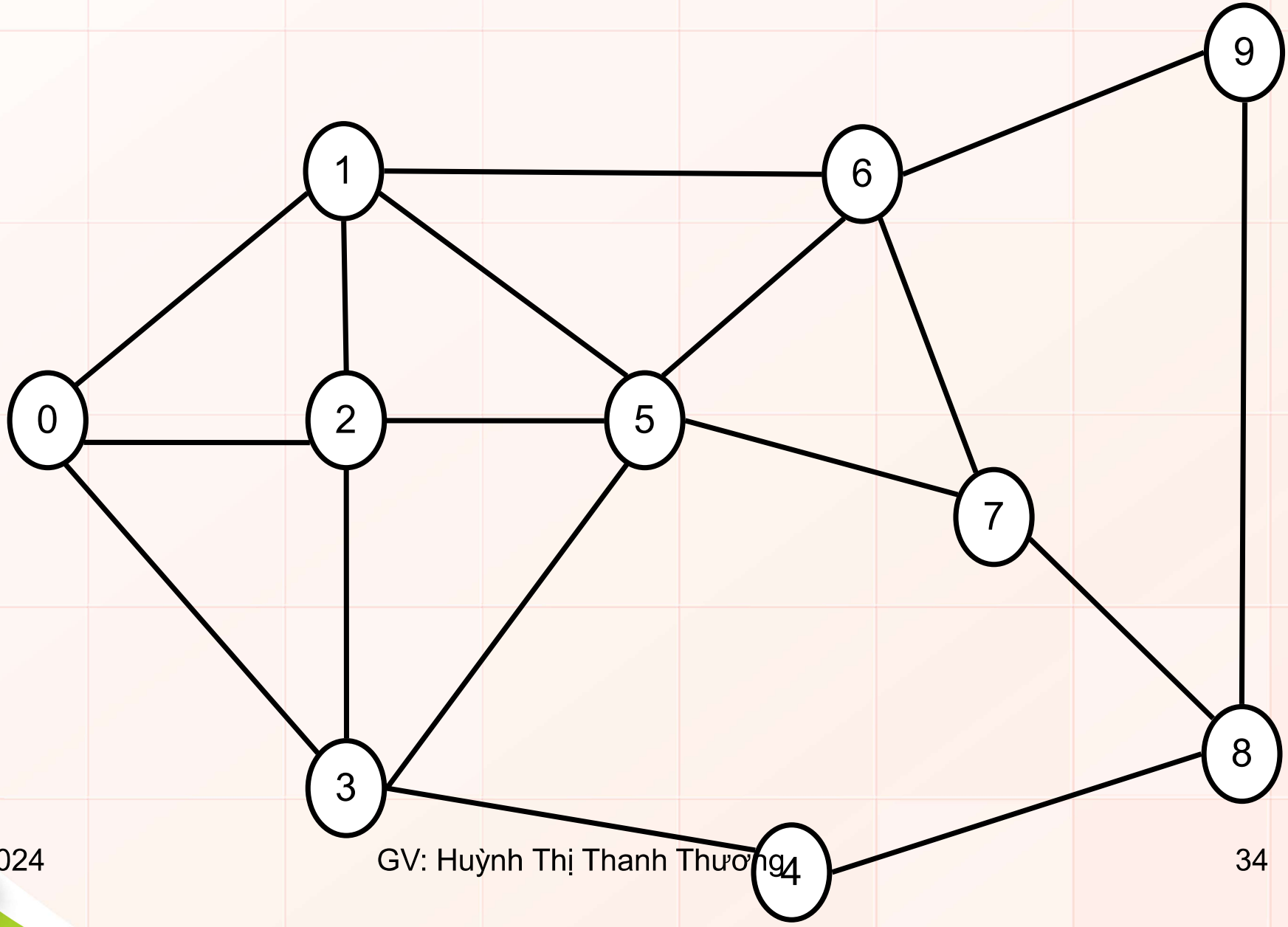
- <u>Bài toán tô màu</u>

  Giả sử ta có bản đồ các quốc gia trên thế giới, ta muốn tô màu các quốc gia này sao cho 2 nước có cùng ranh giới được tô khác màu nhau.

  Yêu cầu tìm cách tô sao cho số màu sử dụng là ít nhất.

Ví dụ 6

Đỉnh:	0	1	2	3	4	5	6	7	8	9
Bậc:	3	4	4	4	2	5	4	3	3	2



Thuật giải

Giải pháp được diễn đạt tương tự như thuật toán nhưng không đòi hỏi các tiêu chuẩn như thuật toán.

- ✓ **Tính xác định**
- ✓ **Tính đúng**: chấp nhận các thuật giải đơn giản có thể cho kết quả đúng hay gần đúng nhưng có khả năng thành công cao hơn.

Thuật giải heuristic

Thuật giải Heuristic phải có những đặc trưng:

- Thường tìm được lời giải tốt, mặc dù không phải là tốt nhất → kết quả chấp nhận được
- Thực hiện dễ dàng nhanh chóng so với thuật giải tối ưu → hiệu quả
- Giải pháp tự nhiên, mang tính thông minh nhất định tương tự con người, gần gũi với cách giải của con người

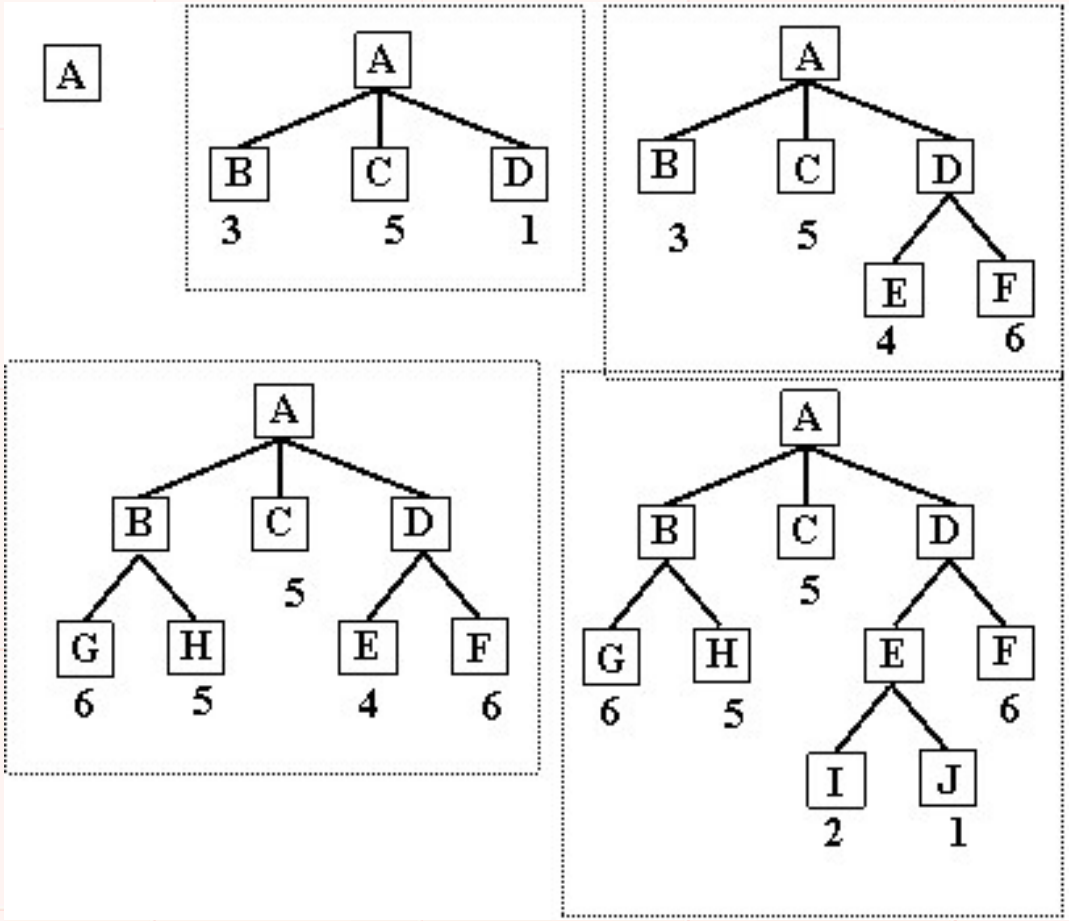
Best-First Search - BFS

❖ Tư tưởng:

- Kết hợp 2 pp tìm kiếm theo chiều sâu và chiều rộng, được hướng dẫn bởi hàm đánh giá.
- Tại mỗi bước, chọn đi theo trạng thái có khả năng cao nhất trong số các trạng thái đã được xét cho đến thời điểm đó.
  - ưu tiên đi vào nhánh có khả năng nhất, đồng thời vẫn "quan sát" những nhánh khác
  - nếu càng đi sâu vào một hướng mà phát hiện càng đi thì càng tệ → không đi tiếp mà chọn đi theo một hướng tốt nhất trong số những hướng chưa đi

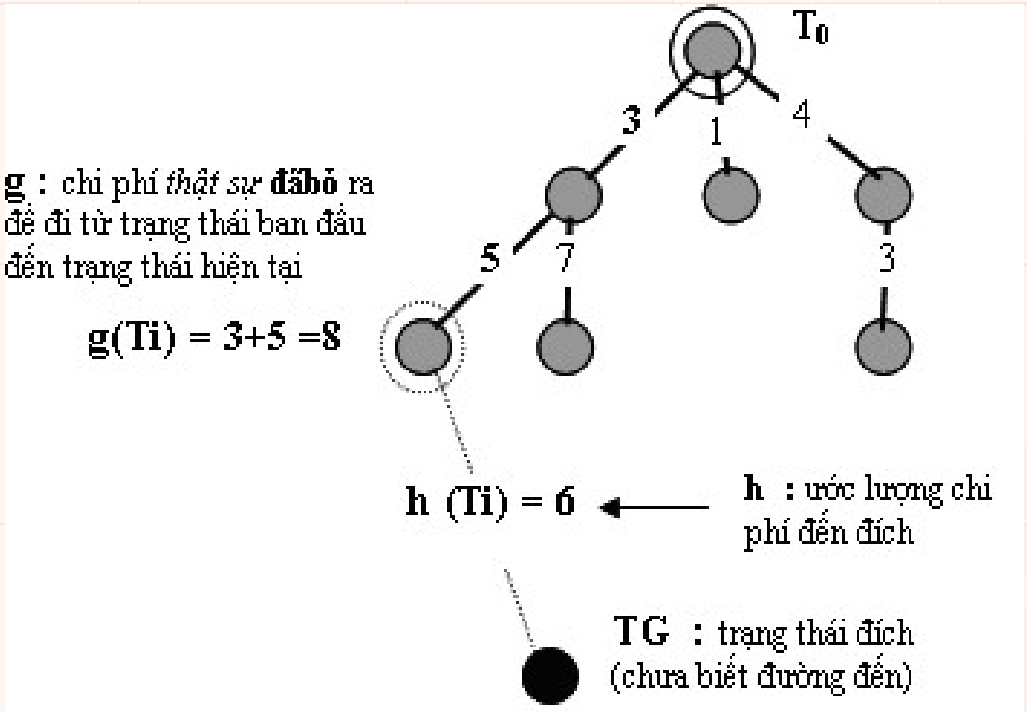
Best-First Search

❖ Minh họa:



Hàm heuristic

- Độ tốt **f** của một trạng thái được tính dựa theo 2 hai giá trị:
  - **h:** một ước lượng về chi phí từ trạng thái hiện hành cho đến đích.
  - **g:** "chiều dài quãng đường" đã đi từ trạng thái ban đầu cho đến trạng thái hiện tại (chi phí thực sự)
  - quy ước là **g** và **h** đều không âm và càng nhỏ nghĩa là càng tốt



Search Algorithms

- Hàm đánh giá f(n) cho mỗi trạng thái/node

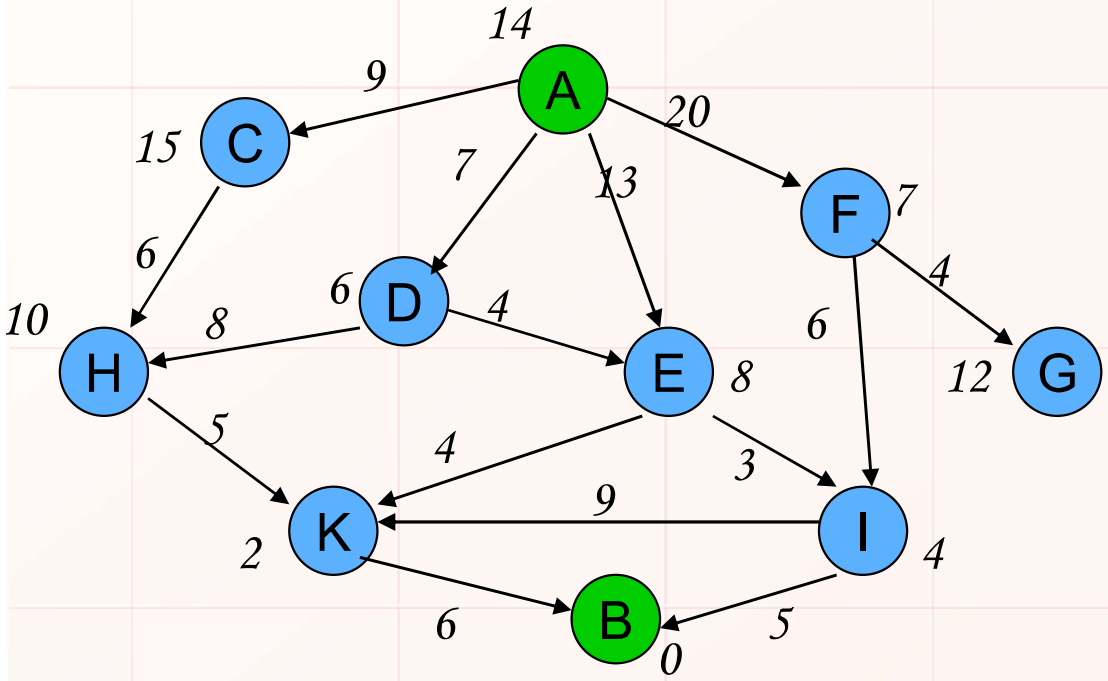
$$f(n) = g(n) + h(n)$$

COST HEURISTIC

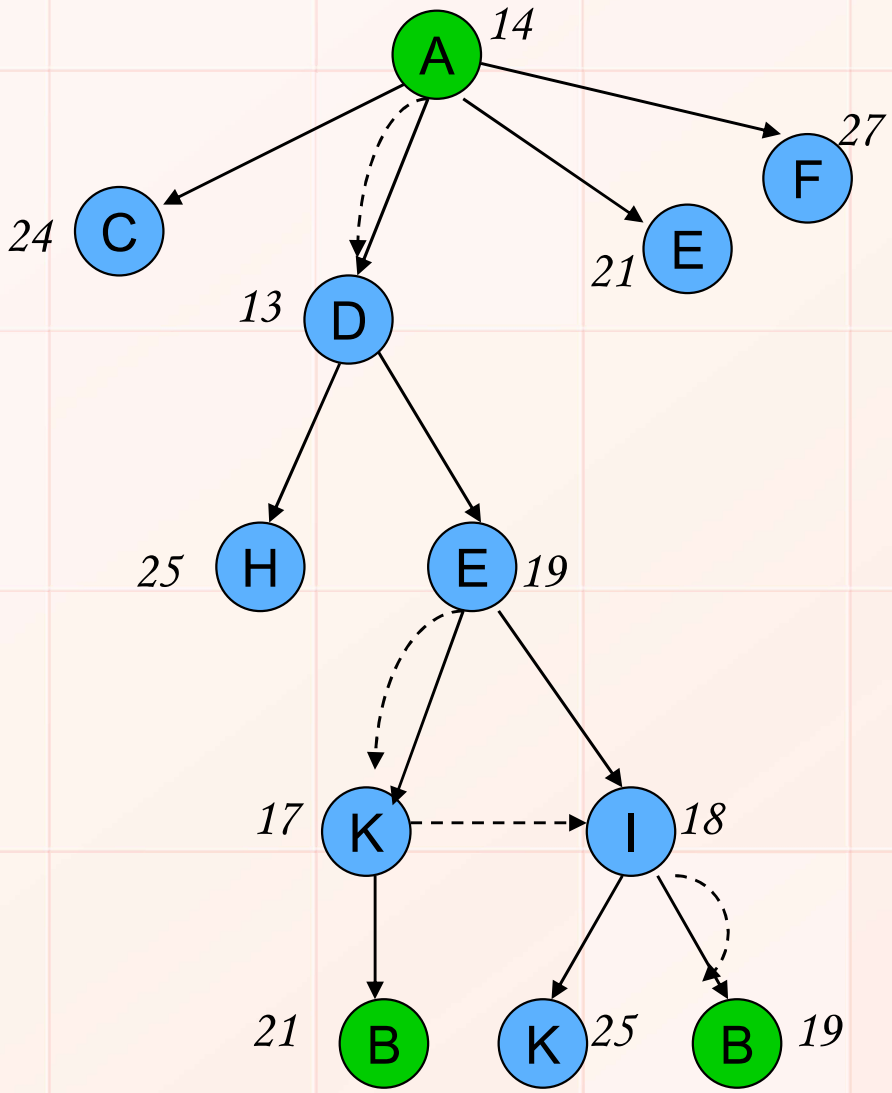
→ chọn node tốt nhất trước – “best-first search”

Uniform cost search: h(n) = 0
Greedy search: g(n) = 0, h(n) arbitrary
A search: g(n), h(n) arbitrary
A* search: g(n), h(n) admissible

Tìm đường đi ngắn nhất



Đồ thị không gian trạng thái với hàm đánh giá



Cây tìm kiếm theo giải thuật A*

A*

Mỗi đỉnh p tương ứng với 1 số độ tốt f(p) = g(p) + h(p)

Bước 1: Open := {s};

Close :={};

g(s) :=0; ,

f(s) = h(s);

Bước 2: While (Open ≠ {})

2.1 Chọn p thuộc Open có f(p) nhỏ nhất

2.2 Nếu p là trạng thái kết thúc thì thoát , thông báo kết quả

2.3 Chuyển p qua Close, và mở các q sau p

A*

2.4 Xét các đỉnh kề q của p

TH1: **q∉Open và q ∉ Close**

g(q) = g(p) + cost(p,q);

f(q) = g(q) + h(q); *// h(q) là giá trị heuristic*

prev(q) = p

Thêm q vào Open

TH2: **q∈Open**

if(g(q) > g(p) + cost(p,q)) *// nếu đến được q bằng path ngắn hơn thì cập nhật lại q trong Open*

g(q) = g(p) + cost(p,q);

f(q) = g(q) + h(q);

prev(q) = p

A*

TH3: **q ∈ Close**

if(g(q) > g(p) + cost(p,q)) *// nếu đến được q bằng path ngắn hơn*

Bỏ q khỏi Close

g(q) = g(p) + cost(p,q);

f(q) = g(q) + h(q);

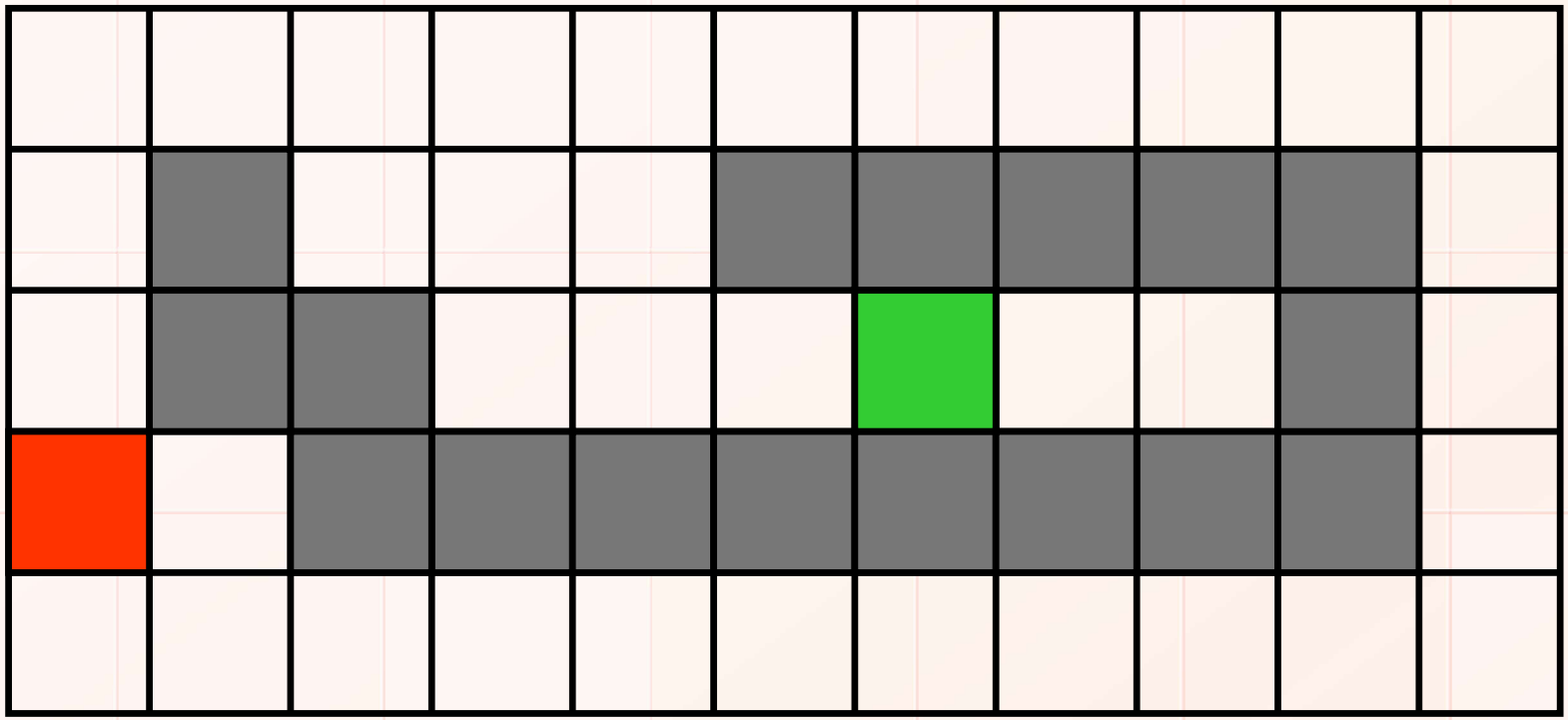
prev(q) = p

Thêm q vào Open

Cập nhật các đỉnh chịu ảnh hưởng từ sự thay đổi của q

Bước 3: Không tìm được

Robot Navigation



A* - Ứng dụng

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5