

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

L/O/G/O

GV: HUỖNH THỊ THANH THƯỜNG

Email: hh.thanhthuong@gmail.com

thuonghtt@uit.edu.vn

4/18/2023

This document is available free of charge on

 **studocu**

Downloaded by Thành Búi ?i (thanh0985837774@gmail.com)

CHƯƠNG 3

THIẾT KẾ THUẬT TOÁN

Algorithm Design

GV: ThS. HUỖNH 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

- Vết 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.

Best Team



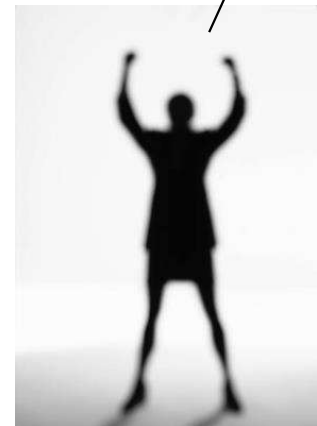
TỐT

=



+

TỐT



Phương pháp tham lam

❖ Mô hình hóa:

- Thường được vận dụng để giải bài toán **tối ưu tổ hợp**
- Xây dựng 1 phương án “tối ưu” X bằng cách:
 - Chọn lựa từng thành phần X_i của X cho đến khi đủ n thành phần (hoàn thành)
 - Với mỗi X_i , chọn X_i tối ưu
- Lời giải tối ưu (toàn cục) = tập hợp các lời giải tối ưu cục bộ.
- Từng bước tối ưu cục bộ, **hi vọng** sẽ tối ưu toàn cục

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  $\leftarrow \phi$  //initialize the solution to empty//
  for i  $\leftarrow$  1 to n
  {
    x  $\leftarrow$  SELECT(P)
    P = P-{x}
    if FEASIBLE(solution,x)
      solution  $\leftarrow$  UNION(solution,x)
  }
  return(solution)
}
```

Phương pháp tham lam

❖ Mô hình

```
greedy (P,n) ≡
{
    G = ∅ ;                               // Khởi tạo lời giải rỗng
    while ( P ≠ ∅ )                        // P chứa n đối tượng input
    {
        x= Chon(P); //chọn phần tử tốt nhất của P
        P = P-{x} hoặc cập nhật P để chọn ở bước kế
tiếp
        if( G ∪ {x} chấp nhận được )
            G = G ∪ {x};
    }
}
```

Phương pháp tham lam

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

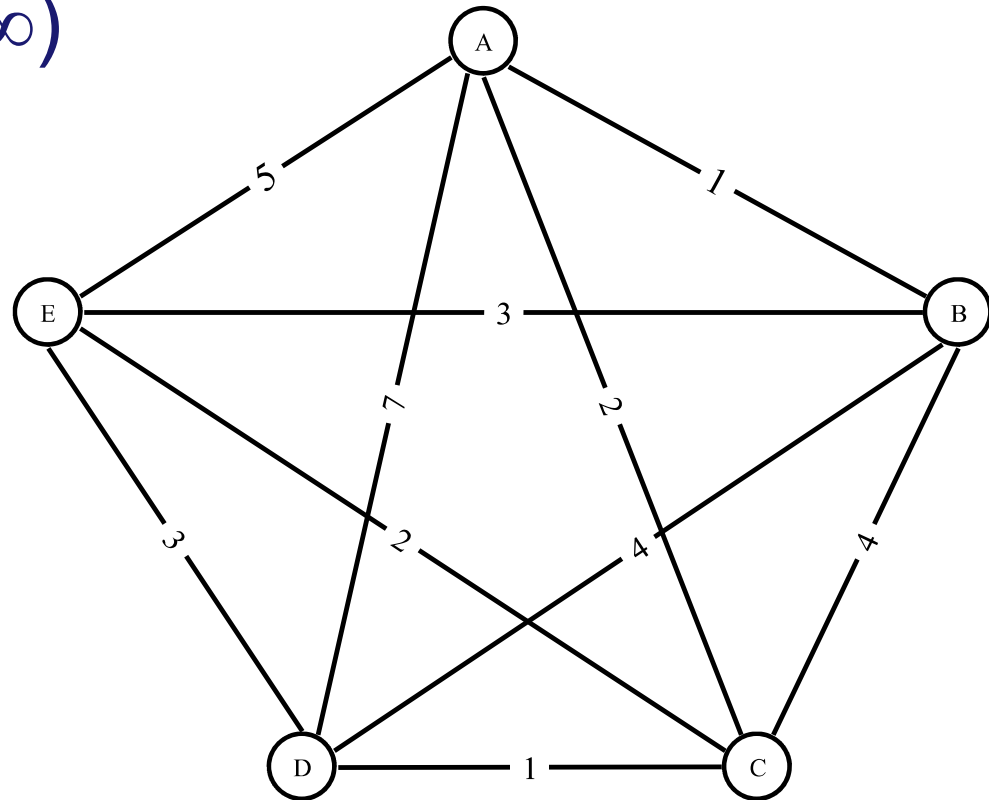
Traveling Salesman Problem - TSP



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

❖ Dựa theo kinh nghiệm con người:

Khi đi trên **những đoạn đường ngắn nhất** thì cuối cùng ta sẽ có một **hành trình ngắn nhất**.



Eureka !

➔ Nguyên lý tham lam

*lấy tiêu chuẩn hành trình ngắn nhất của bài toán làm tiêu chuẩn cho chọn lựa **cục bộ***

Traveling Salesman Problem

❖Thuật giải: Cách 1

$P = \{\text{các đỉnh trên đồ thị}\} / \{S\}$

$G = [S]$

$N = S$; // đỉnh hiện hành

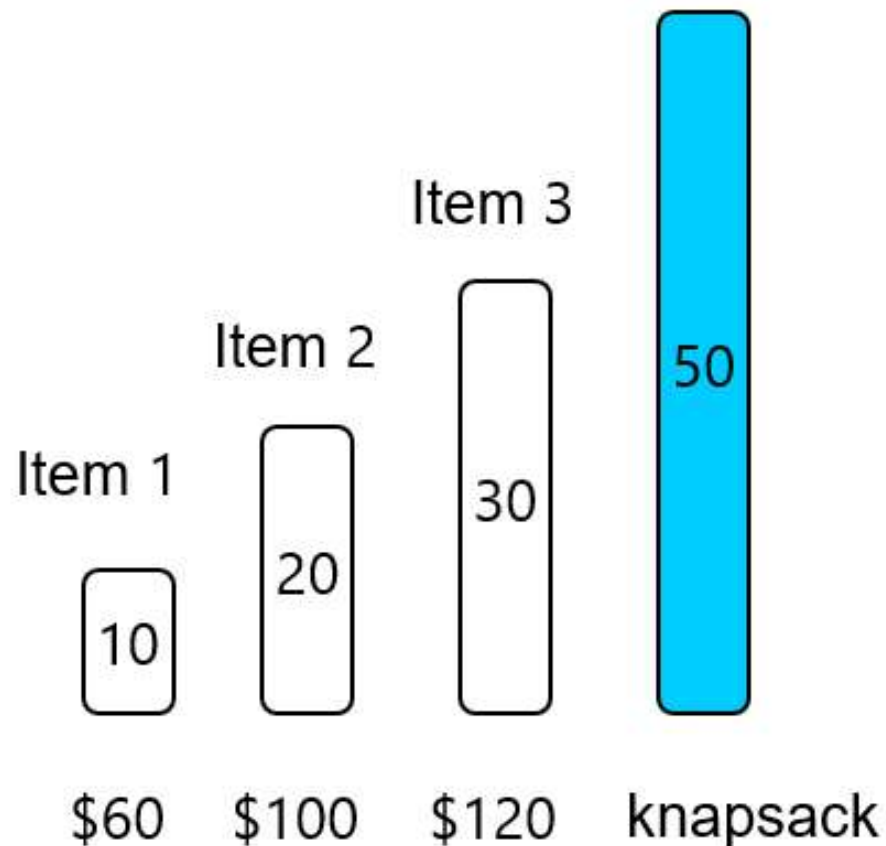
While ($P \neq \emptyset$)

- **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

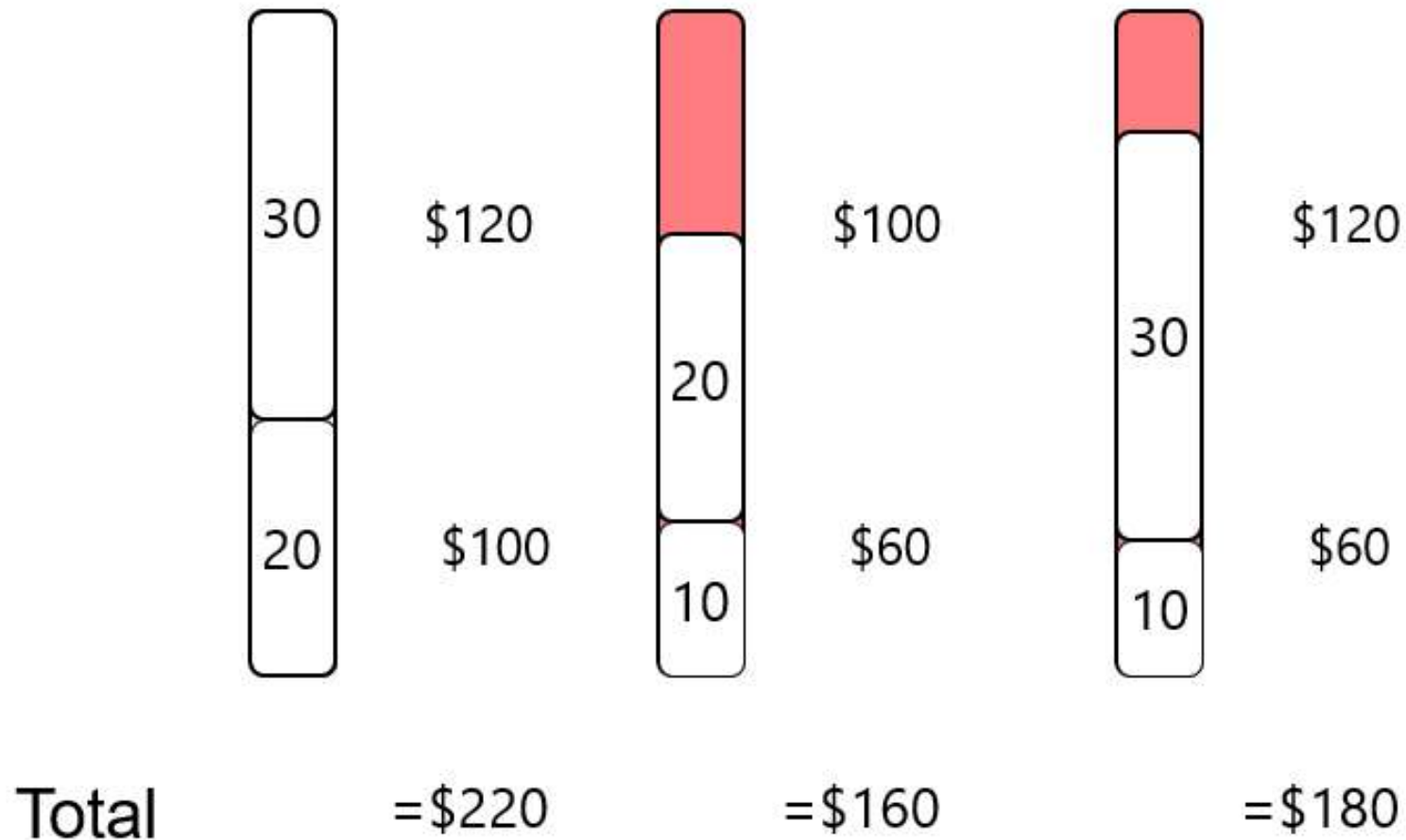
❖ Ví dụ 2: Knapsack 0/1 – Dạng 1

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



Knapsack 0/1

$$\begin{aligned} \text{maximize } z &= \sum_{1 \leq i \leq n} p_i x_i \\ \text{thỏa} \quad &\sum_{1 \leq i \leq n} w_i x_i \leq M \\ \text{và } x_i &\in \{0, 1\} \end{aligned}$$

Fractional Knapsack

❖ Ví dụ: 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 = $20 + 30 + 65 + 40 = 155$

Fractional Knapsack

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 = $65 + 50 + 30 = 145$

Fractional Knapsack

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 2: Fractional Knapsack – Dạng 2

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 .

Đồ vật có thể được tháo rời/bẻ ra làm nhiều phần.

Một phần x_i ($0 \leq x_i \leq 1$) của đồ vật i có giá trị là $p_i x_i$ (VD: $\frac{1}{2}$ đồ vật i có giá trị $\frac{1}{2} p_i$)

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.

Có thể lấy đi 1 phần của đồ vật

(Đã giải trên lớp)

Fractional Knapsack

<u>20</u> 30	\$80
20	\$100
10	\$60

Total = \$240



Fractional Knapsack

$$\begin{aligned} \text{maximize } z &= \sum_{1 \leq i \leq n} p_i x_i \\ \text{thỏa} \quad &\sum_{1 \leq i \leq n} w_i x_i \leq M \\ \text{và } &0 \leq x_i \leq 1 \end{aligned}$$

Fractional Knapsack

❖ Ví dụ: $n = 3$, $M = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$ và $(w_1, w_2, w_3) = (18, 15, 10)$

(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum p_i x_i$
1) $(1/2, 1/3, 1/4)$	16.5	24.25
2) $(1, 2/15, 0)$	20	28.2
3) $(0, 2/3, 1)$	20	31
4) $(0, 1, 1/2)$	20	31.5

Phương pháp tham lam

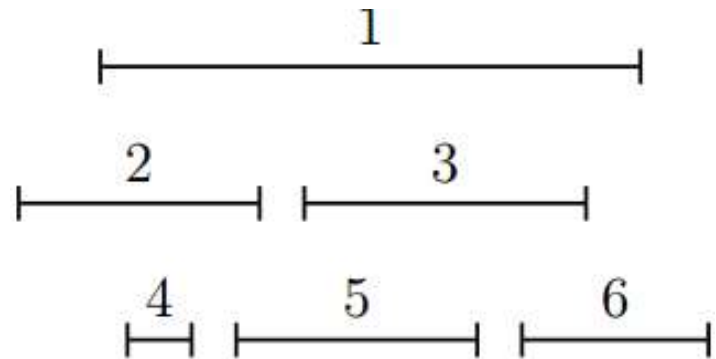
❖ Ví dụ 3: Bài toán chọn hoạt động (Activity-Selection Problem/ Interval Scheduling)

- Cho một tập các hoạt động $S = \{1, 2, \dots, n\}$
- 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

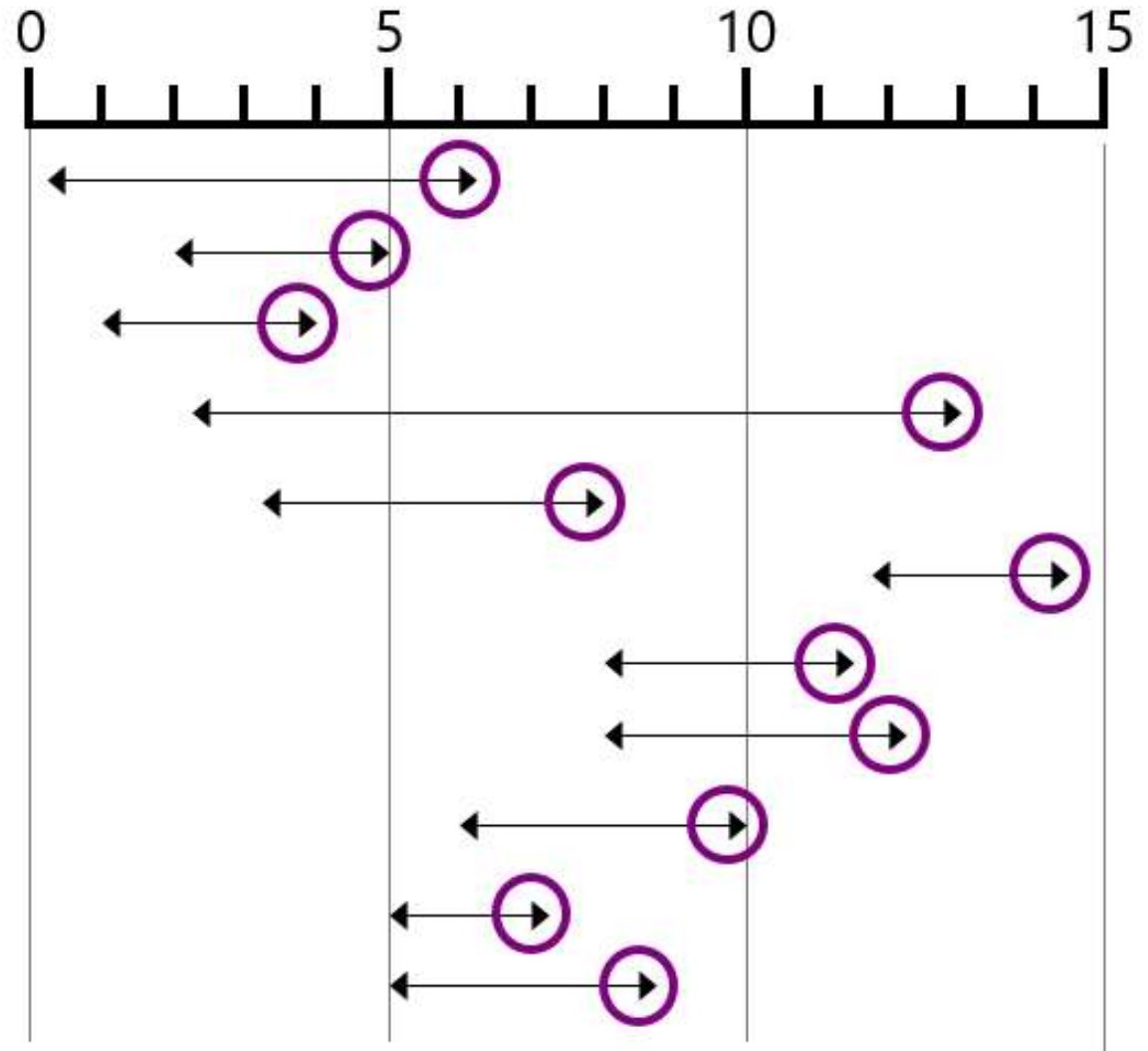
- Hai hoạt động i và j là “tương thích nhau” (compatible) nếu $[s_i, f_i)$ và $[s_j, f_j)$ không chạm nhau i.e. $f_i \leq s_j$ hoặc $f_j \leq s_i$.
- Yêu cầu: Tìm tập hợp lớn nhất các hoạt động tương thích nhau?



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

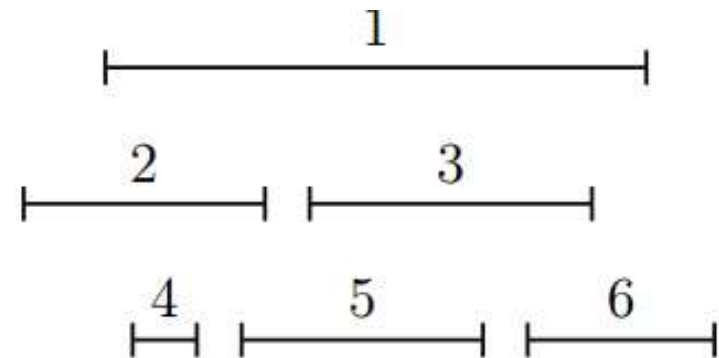
- Xem xét tất cả các phương án (tập con các hoạt động) chấp nhận được
- Chọn tập con (chấp nhận được) lớn nhất
- Độ phức tạp: $\Theta(2^n)$



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:

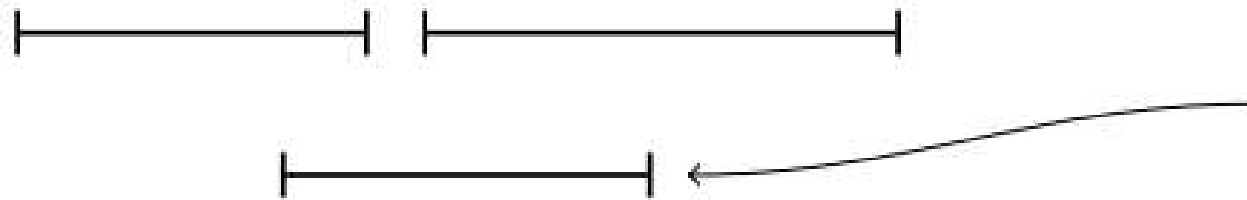
❖ Cách 1: Chọn hoạt động bắt đầu sớm nhất
(s_i nhỏ nhất)



Greedy Interval Scheduling

Những quy tắc có thể xem xét:

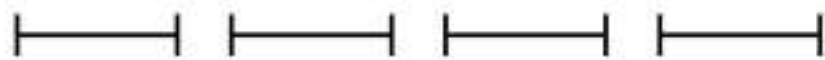
❖ Cách 2: Chọn hoạt động ngắn nhất ($f_i - s_i$ nhỏ nhất)



Smallest. Bad :(

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



Least # of incompatibles. Bad :(

Greedy Interval Scheduling

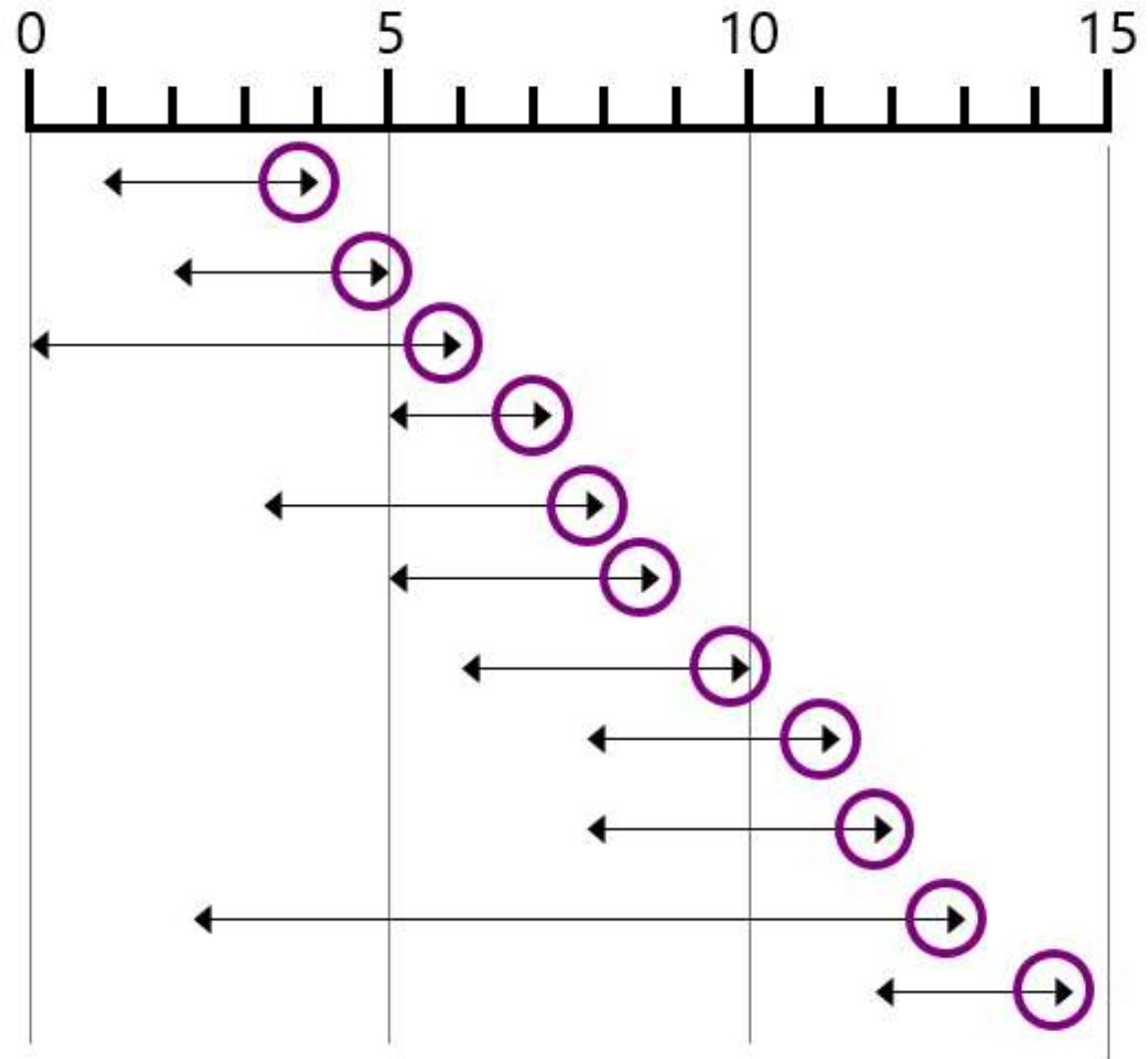
Những quy tắc có thể xem xét:

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

Greedy Interval Scheduling

Sort by finish time

Bài tập tại lớp: viết mã giả



Một số ví dụ khác

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

Kruskal's / Prim's algorithm

