



## CHƯƠNG 3. MÔ HÌNH THAM LAM

### NỘI DUNG:

- 3. 1. Giới thiệu giải thuật
- 3. 2. Bài toán Activities Selection
- 3. 3. Bài toán N-Ropes
- 3. 4. Bài toán sắp đặt lại xâu ký tự
- 3. 5. Bài toán tìm cây khung nhỏ nhất
- 3. 6. CASE STUDY

### 3.1. Giới thiệu thuật toán

**Thuật toán tham lam (Greedy Algorithm):** là thuật toán xem xét quá trình giải quyết bài toán bằng việc tạo nên lựa chọn tối ưu cục bộ tại mỗi bước thực hiện với mong muốn tìm ra lựa chọn tối ưu toàn cục. Giải thuật tham lam thường không mang tính tổng quát. Tuy vậy, bằng việc xem xét các lựa chọn tối ưu cục bộ cho ta lời giải gần đúng với phương án tối ưu toàn cục cũng là giải pháp tốt trong thời gian chấp nhận được.

**Thuật toán Greedy bao gồm 5 thành phần chính:**

1. Một tập các ứng viên mà giải pháp thực hiện tạo ra.
2. Một hàm lựa chọn (selection function) để chọn ứng viên tốt nhất cho giải pháp.
3. Một hàm thực thi (feasibility function) được sử dụng để xác định các ứng viên có thể có đóng góp cho giải pháp thực thi.
4. Một hàm mục tiêu (objective function) dùng để xác định giá trị của phương pháp hoặc một phần của giải pháp.
5. Một hàm giải pháp (solution function) dùng để xác định khi nào giải pháp kết thúc.

### **3.2. Activities Selection Problem.**

Lựa chọn hành động là bài toán tối ưu tổ hợp diễn hình quan tâm đến việc lựa chọn các hành động không mâu thuẫn nhau trong các khung thời gian cho trước. Bài toán được phát biểu như sau:

Cho tập gồm n hành động, mỗi hành động được biểu diễn như bộ đôi thời gian bắt đầu si và thời gian kết thúc fi ( $i=1, 2, \dots, n$ ). Bài toán đặt ra là hãy lựa chọn nhiều nhất các hành động có thể thực hiện bởi một máy hoặc một cá nhân mà không xảy ra tranh chấp. Giả sử mỗi hành động chỉ thực hiện đơn lẻ tại một thời điểm.

**Input:**

- Số lượng hành động: 6
- Thời gian bắt đầu Start []= { 1, 3, 0, 5, 8, 5}
- Thời gian kết thúc Finish[] = { 2, 4, 6, 7, 9, 9}

**Output:** Số lượng lớn nhất các hành động có thể thực hiện bởi một người.

$$\text{OPT}[] = \{0, 1, 3, 4\}$$

**Lời giải.** Thuật toán tham lam giải quyết bài toán được thực hiện như sau:

**Algorithm Greedy-Activities- Selection( N, S[], F[] ):**

**Input:**

- N là số lượng hành động (công việc).
- S[] thời gian bắt đầu mỗi hành động.
- F[] thời gian kết thúc mỗi hành động.

**Ouput:**

- Danh sách thực thi nhiều nhất các hành động.

**Actions:**

**Bước 1** (sắp xếp). Sắp xếp các hành động theo thứ tự tăng dần của thời gian kết thúc.

**Bước 2** (Khởi tạo) Lựa chọn hành động đầu tiên làm phương án tối ưu ( OPT=1).  $N = N \setminus \{i\}$ ;

**Bước 3** (Lặp).

```
for each activities j ∈ N do {  
    if ( S[j] >= F[i] ) {  
        OPT = OPT ∪ j; i = j; N = N \ {i}  
    }  
}
```

**Bước 4** (Trả lại kết quả).

Return (OPT)

**EndActions.**

## Ví dụ. Thuật toán Greedy-ActivitiesN-Selection(int N, int S[], int F[]):

### Input:

- Số lượng hành động n = 8.
- Thời gian bắt đầu S[] = {1, 3, 0, 5, 8, 5, 9, 14}.
- Thời gian kết thúc F[] = {2, 4, 6, 7, 9, 9, 12, 18}.

### Output:

- OPT = { Tập nhiều nhất các hành động có thể thực hiện bởi một máy}.

### Phương pháp tiến hành::

Bước	i= ? j =?	(S[j] >=F[i]) ? i=?	OPT=?
			OPT = OPT ∪ {1}.
1	i=1; j=2	(3>=2): Yes; i=2.	OPT = OPT ∪ {2}.
2	i=2; j=3;	(0>=4): No; i=2.	OPT = OPT ∪ $\emptyset$ .
3	i=2; j=4;	(5>=4): Yes; i=4.	OPT = OPT ∪ {4}.
4	i=4; j=5;	(8>=7): Yes; i=5.	OPT = OPT ∪ {5}.
5	i=5; j=6;	(5>=9): No; i=5.	OPT = OPT ∪ $\emptyset$ .
6	i=5; j=7;	(9>=9): Yes; i=7.	OPT = OPT ∪ {7}.
7	i=7; j=8;	(14>=12): Yes; i=8.	OPT = OPT ∪ {8}.
OPT = { 1, 2, 4, 5, 7, 8}			

**3.3. Bài toán n-ropes.** Cho  $n$  dây với chiều dài khác nhau. Ta cần phải nối các dây lại với nhau thành một dây. Chi phí nối hai dây lại với nhau được tính bằng tổng độ dài hai dây. Nhiệm vụ của bài toán là tìm cách nối các dây lại với nhau thành một dây sao cho chi phí nối các dây lại với nhau là ít nhất.

**Input:**

- Số lượng dây: 4
- Độ dài dây  $L[] = \{4, 3, 2, 6\}$

**Output:** Chi phí nối dây nhỏ nhất.

$$\text{OPT} = 39$$

Chi phí nhỏ nhất được thực hiện như sau: lấy dây số 3 nối với dây số 2 để được tập 3 dây với độ dài 4, 5, 6. Lấy dây độ dài 4 nối với dây độ dài 5 ta nhận được tập 2 dây với độ dài 6, 9. Cuối cùng nối hai dây còn lại ta nhận được tập một dây với chi phí là  $6+9=15$ . Như vậy, tổng chi phí nhỏ nhất của ba lần nối dây là  $5 + 9 + 15 = 29$ .

Ta không thể có cách nối dây khác với chi phí nhỏ hơn 39. Ví dụ lấy dây 1 nối dây 2 ta nhận được 3 dây với độ dài  $\{7, 2, 6\}$ . Lấy dây 3 nối dây 4 ta nhận được tập hai dây với độ dài  $\{7, 8\}$ , nối hai dây cuối cùng ta nhận được 1 dây với độ dài 15. Tuy vậy, tổng chi phí là  $7 + 8 + 15 = 30$ .

**Thuật toán. Sử dụng phương pháp tham lam dựa vào hàng đợi ưu tiên.**

**Thuật toán Greedy-N-Ropes(int L[], int n):**

**Input:**

- n : số lượng dây.
- L[] : chi phí nối dây.

**Output:**

- Chi phí nối dây nhỏ nhất.

**Actions:**

**Bước 1 .** Tạo pq là hàng đợi ưu tiên lưu trữ độ dài n dây.

**Bước 2 (Lặp).**

OPT = 0; // Chi phí nhỏ nhất.

While (pq.size>1) {

    First = pq.top; pq.pop(); //Lấy và loại phần tử đầu tiên trong pq.

    Second = pq.top; pq.pop(); //Lấy và loại phần tử kế tiếp trong pq.

    OPT = First + Second; //Giá trị nhỏ nhất để nối hai dây

    Pq.push(First + Second); //Đưa lại giá trị First + Second vào pq.

}

**Bước 3( Trả lại kết quả).**

Return(OPT);

**EndActions.**

## Ví dụ. Thuật toán Greedy-N-Ropes(int L[], int n):

**Input:**

- Số lượng dây  $n = 8$ .
- Chi phí nối dây  $L[] = \{ 9, 7, 12, 8, 6, 5, 14, 4 \}$ .

**Output:**

- Chi phí nối dây nhỏ nhất.

**Phương pháp tiến hành::**

Bước	Giá trị First, Second	OPT=?	Trạng thái hàng đợi ưu tiên.
		0	4, 5, 6, 7, 8, 9, 12, 14
1	First=4; Second=5	9	6, 7, 8, 9, <b>9</b> , 12, 14
2	First=6; Second=7	22	8, 9, 9, 12, <b>13</b> , 14
3	First=8; Second=9	39	9, 12, 13, 14, 17
4	First=9; Second=12	60	13, 14, 17, <b>21</b>
5	First=13; Second=14	87	17, 21, <b>27</b>
6	First=17; Second=21	125	27, <b>38</b>
7	First=27; Second=38	190	<b>65</b>
OPT = 190			

### **3.4. Bài toán sắp đặt lại xâu ký tự**

**Bài toán.** Cho xâu ký tự  $s[]$  độ dài  $n$  và số tự nhiên  $d$ . Hãy sắp đặt lại các ký tự trong xâu  $s[]$  sao cho hai ký tự giống nhau đều cách nhau một khoảng là  $d$ . Nếu bài toán có nhiều nghiệm, hãy đưa ra một cách sắp đặt đầu tiên tìm được. Nếu bài toán không có lời giải hãy đưa ra thông báo “Vô nghiệm”.

Ví dụ.

**Input:**

- Xâu ký tự  $S[] = "ABB"$ ;
- Khoảng cách  $d = 2$ .

**Output:** BAB

**Input:**

- Xâu ký tự  $S[] = "AAA"$ ;
- Khoảng cách  $d = 2$ .

**Output:** Vô nghiệm.

**Input:**

- Xâu ký tự  $S[] = "GEEKSFORGEEKS"$ ;
- Khoảng cách  $d = 3$ .

**Output:** EGKEGKESFESFO

### 3.4. Bài toán sắp đặt lại xâu ký tự

Thuật toán Greedy-Arrang-String (S[], d):

**Input:**

- Xâu ký tự S[].
- Khoảng cách giữa các ký tự d.

**Output:** Xâu ký tự được sắp đặt lại thỏa mãn yêu cầu bài toán.

**Formats :** Greedy-Arrang-String(S, d, KQ);

**Actions:**

**Bước 1.** Tìm Freq[] là số lần xuất hiện mỗi ký tự trong xâu.

**Bước 2.** Sắp xếp theo thứ tự giảm dần theo số xuất hiện ký tự.

**Bước 3.** (Lặp).

i = 0; k = < Số lượng ký tự trong Freq[]>;

While ( (i < k) {

    p = Max(Freq); //Chọn ký tự xuất hiện nhiều lần nhất.

    For ( t = 0; t < p; t++ ) // điền các ký tự i, i+d, i+2d, .., i + pd

        if (i+(t\*d)>n) { < Không có lời giải>; return;>

        KQ[i + (t\*d)] = Freq[i].kytu;

    }

    i++;

}

**Bước 4( Trả lại kết quả):** Return(KQ);

**EndActions.**

## Greedy-Arrang-String ( $S[]$ , $d$ ): Input : $S[] = "GEEKSFORGEEKS; d= 3.$

**Bước 1.** Tìm tập ký tự và số lần xuất hiện mỗi ký tự.

Freq[]	
G	2
E	4
K	2
S	2
F	1
O	1
R	1

Sắp xếp theo thứ tự giảm dần của số lần xuất hiện.

**Bước 2.** Sắp xếp theo thứ tự giảm dần số lần xuất hiện.

Freq[]	
E	4
G	2
K	2
S	2
F	1
O	1
R	1

**Bước 3.** Lặp.

	KQ[ $I + p^*d$ ]												
$I = 0$	E			E			E			E			
$i=1$	E	G		E	G		E			E			
$i=2$	E	G	K	E	G	K	E			E			
$i=3$	E	G	K	E	G	K	E	S		E	S		
$i=4$	E	G	K	E	G	K	E	S	F	E	S		
$i=5$	E	G	K	E	G	K	E	S	F	E	S	O	
$i=6$	E	G	K	E	G	K	E	S	F	E	S	O	R

### 3.5. Thuật toán Boruvka xây dựng cây khung nhỏ nhất

**Thuật toán Boruvka:**

**Input :**  $G = \langle V, E, W \rangle$ , trong đó:

- Tập  $V = \{1, 2, \dots, N\}$  là tập đỉnh.
- Tập  $E = \{(u, v) : u \in V, v \in V\}$  là tập các cạnh.
- Tập  $W = \{w_i = d(e_i) : e_i \in E\}$  trọng số các cạnh

**Output:**

$$D(H) = \sum d(e) \rightarrow \text{Min. } (e \in T)$$

**Format:** Boruvka  $G = \langle V, E, W \rangle$ ;

**Actions:**

**Bước 1** (Khởi tạo):

- $D(H) = 0; T = \emptyset;$
- Khởi tạo  $N$  tập đỉnh rời nhau: SLT.

**Bước 3** (Lặp):

While ( $SLT > 1$ ) {

<Thành lập các cây độc lập có cạnh nhỏ nhất nối giữa các TPLT>;

$SLT = \langle \text{Số các cây được thành lập} \rangle$ ;

$T = T \cup \langle \text{Tập tất cả các cạnh của các cây độc lập} \rangle$ ;

$D(H) = D(H) + \langle \text{Độ dài của tất cả các cây độc lập} \rangle$ ;

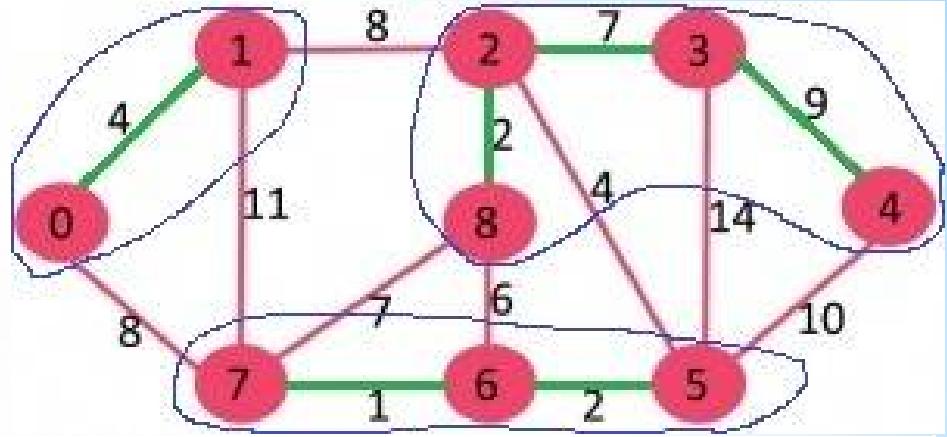
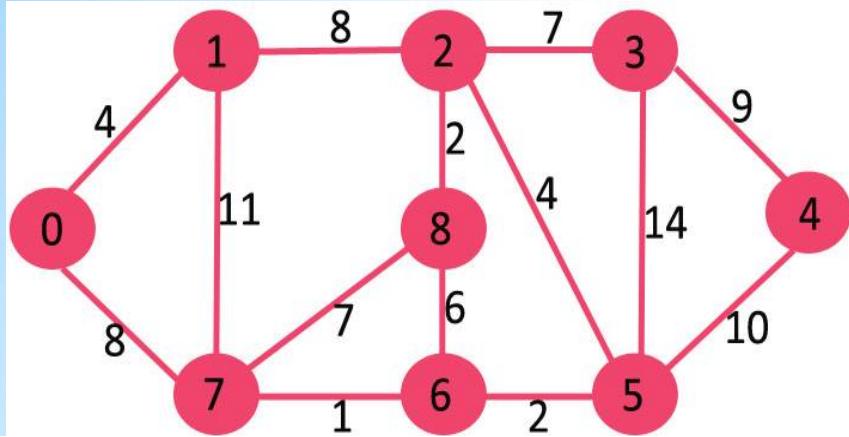
}

**Bước 3** (Trả lại kết quả).

Return ( $T, D(H)$ );

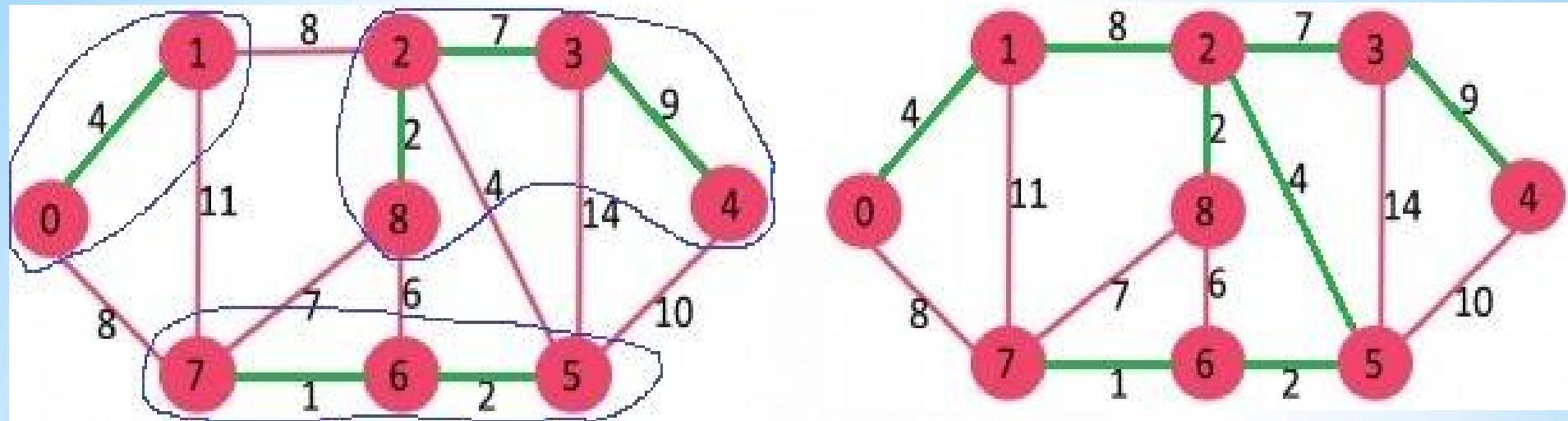
**EndActions.**

## Kiểm nghiệm thuật toán Boruvka



Thành phần LT	Cạnh có trọng số nhỏ nhất	Các cây được tạo
{0}	(0,1)	
{1}	(0,1)	
{2}	(2,8)	
{3}	(2,3)	
{4}	(3,4)	
{5}	(5,6)	
{6}	(6,7)	
{7}	(6,7)	
{8}	(2,8)	$H_1 = \{0, 1\}$ $H_2 = \{2, 3, 4, 8\}$ $H_3 = \{5, 6, 7\}$ $SLT = 3$ $T = \{(0,1), (2,3), (2,8), (3,4), (5,6), (6,7)\}$ $D(H) = 4 + 7 + 2 + 9 + 2 + 1 = 25$

## Kiểm nghiệm thuật toán Boruvka



Thành phần LT	Cạnh có trọng số nhỏ nhất	Các cây được tạo
{0, 1}	(1,2)	$H = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ $SLT = 1$ $T = \{(0,1), (2,3), (2,8), (3,4), (5,6), (6,7), (1,2), (2,5)\}$ $D(H) = 25 + 8 + 4 = 37$
{2, 3, 4, 8}	(2,5)	
{5, 6, 7}	(2,5)	

### **3.6. CASE STUDY:** Lập trình cho các bài toán dưới đây bằng phương pháp tham lam (Greedy Algorithm).

- Bài toán lựa chọn hành động (Activitues Selections Problem).
- Bài toán nối thừng (Connecting N-Ropes).
- Bài toán sắp xếp lại xâu với khoảng cách d các ký tự lặp (Rearrang String with d distance).
- Thuật toán Kruskal tìm cây khung nhỏ nhất (Kruskal's Minimum Spanning Tree Algorithm).
- Thuật toán PRIM tìm cây khung nhỏ nhất (PRIM's Minimum Spanning Tree Algorithm).
- Thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trên đồ thị có trọng số không âm.
- Giải bài toán tô màu đồ thị với số lượng các màu sử dụng tối thiểu (Graph Colouring Problem).
- Xây dựng mã Huffman dựa vào thuật toán tham lam.