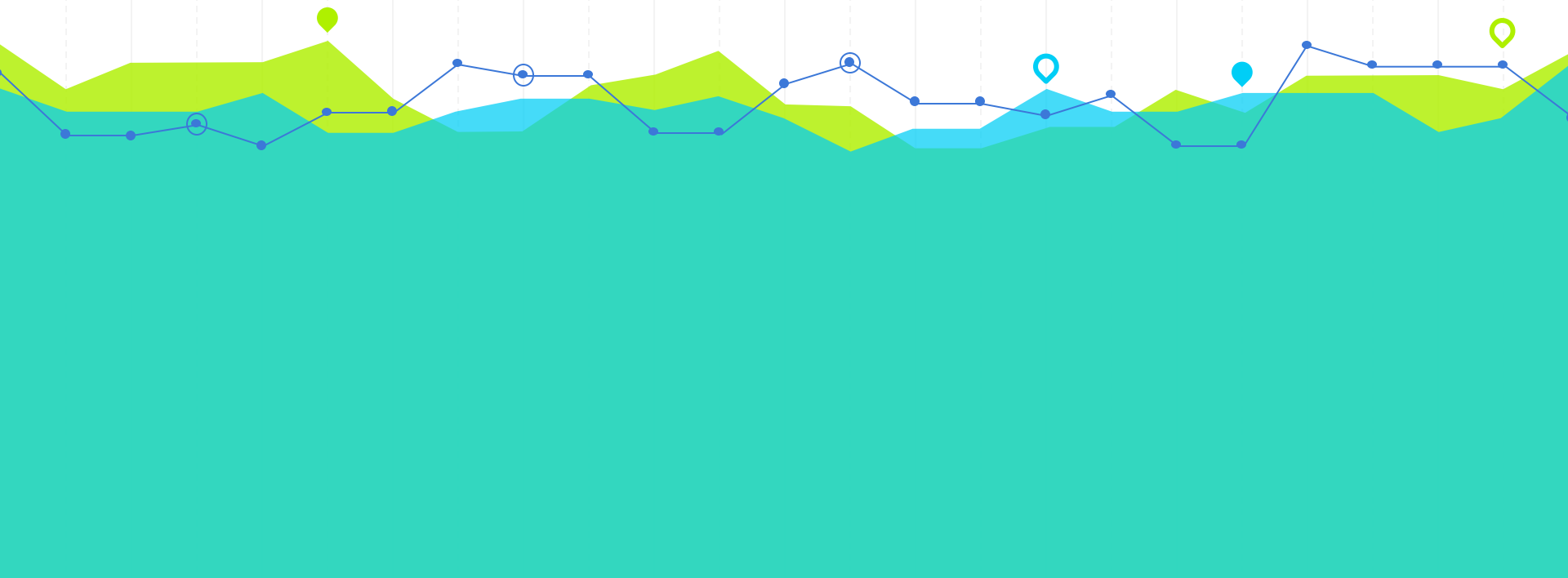


HELLO!



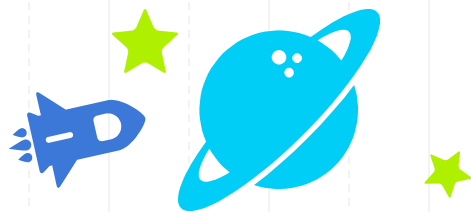
“Greed, for lack of better word, is good”

Gordon Gecko, Wall Street (1987)



GREEDY ALGORITHMS





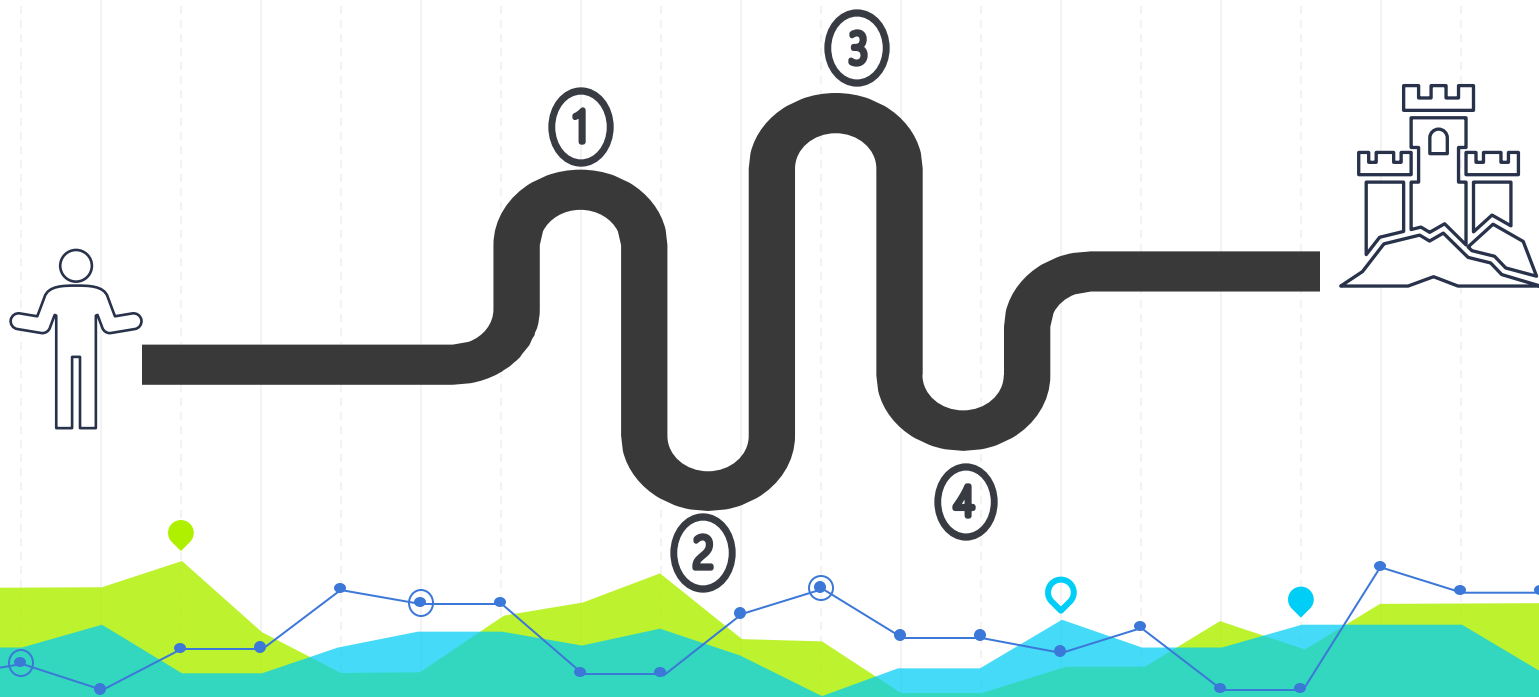
KHÁI NIỆM



Tối ưu ở từng bước
(*Locally optimal*)

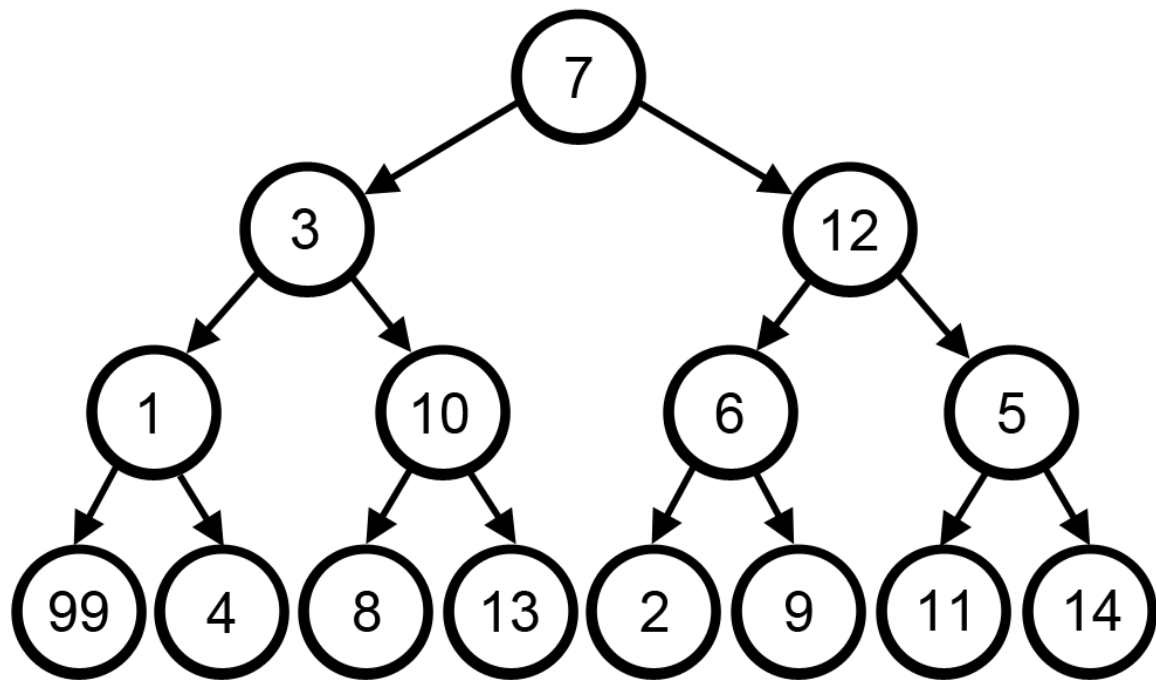
Hy vọng

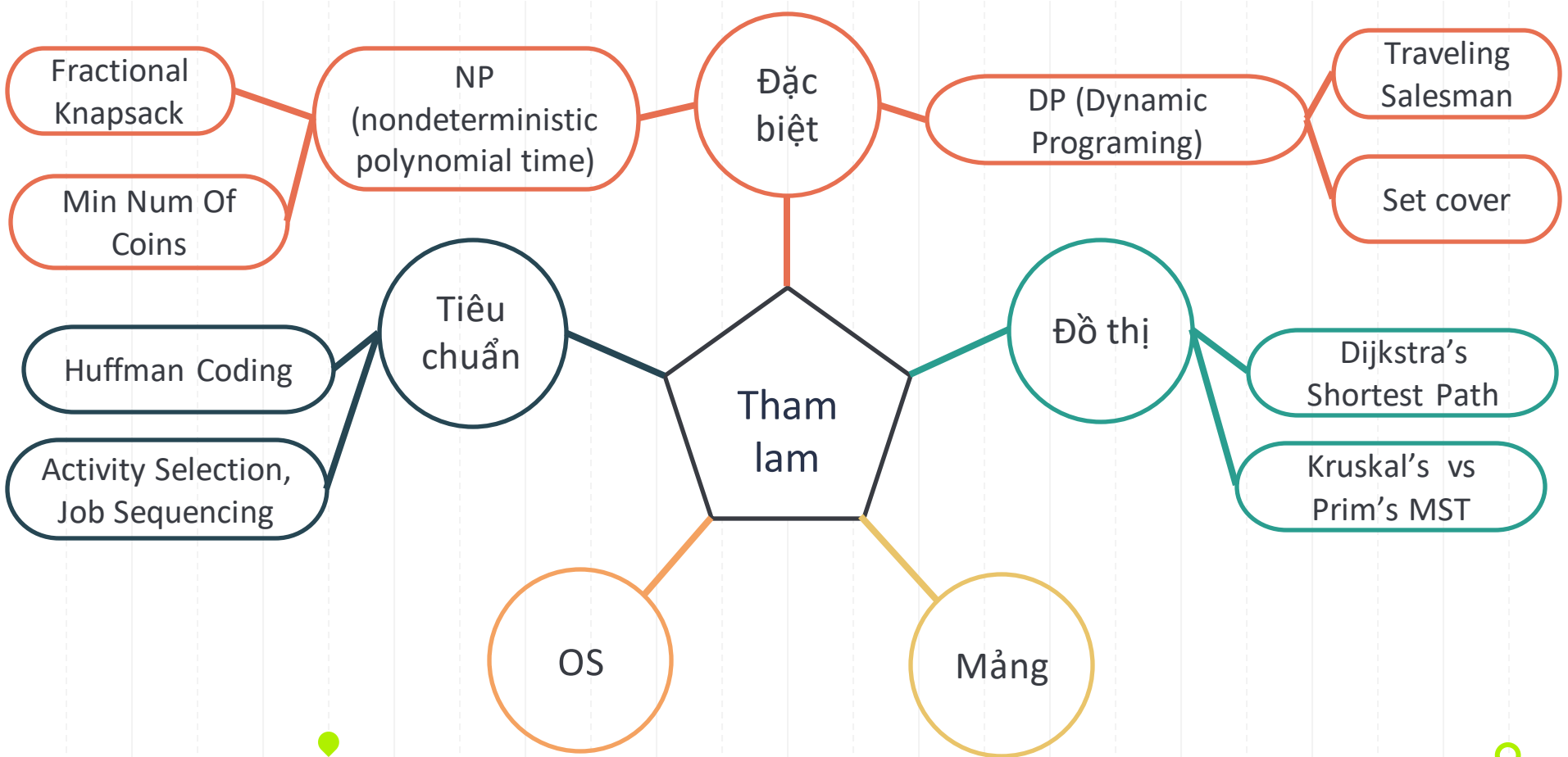
Tối ưu toàn bộ vấn đề
(*Global optimal*)

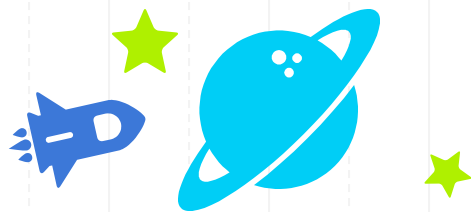




Root \rightarrow node lá
Đường đi có tổng
lớn nhất?

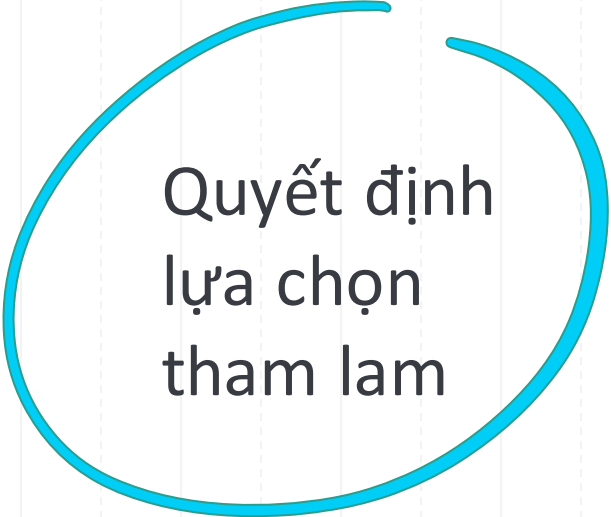







CẤU TRÚC

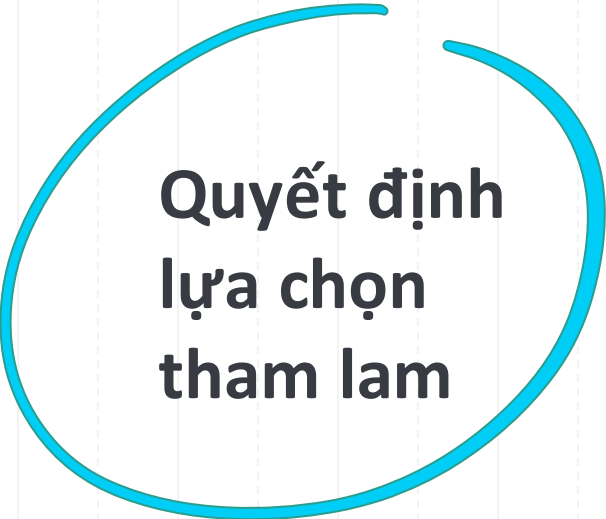




Quyết định
lựa chọn
tham lam



Cấu trúc con
tối ưu




**Quyết định
lựa chọn
tham lam**



Tối ưu tổng thể có thể
đạt được bằng cách lựa
chọn tối ưu ở mỗi bước





Giải pháp tối ưu
tổng thể nằm
trong giải pháp
tối ưu con

Cấu trúc con
tối ưu



Quyết định
lựa chọn
tham lam

Cấu trúc con
tối ưu

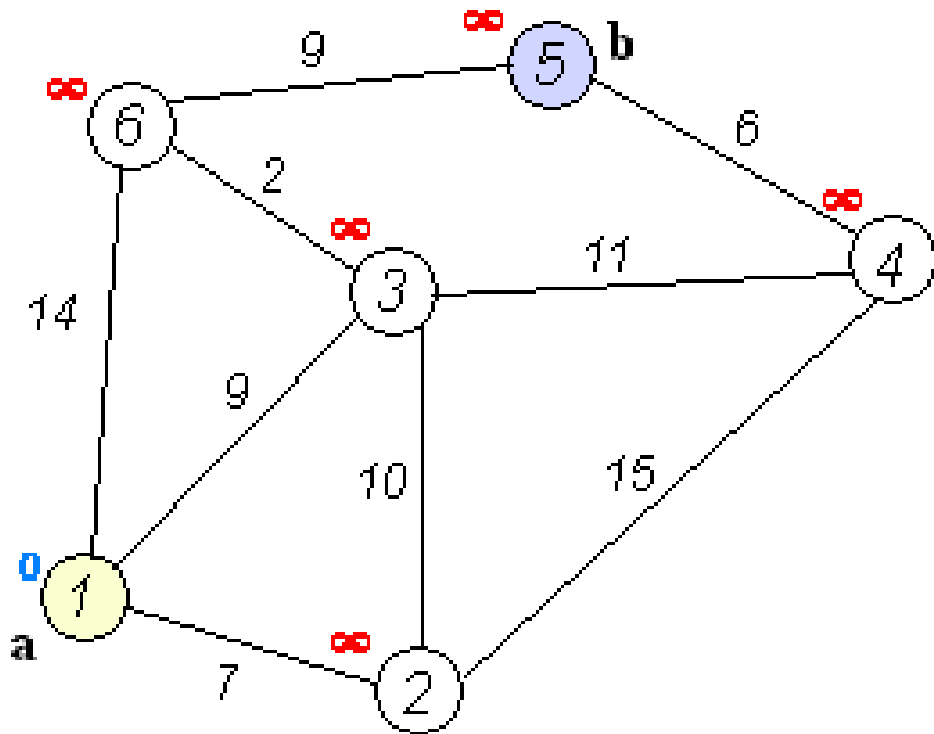
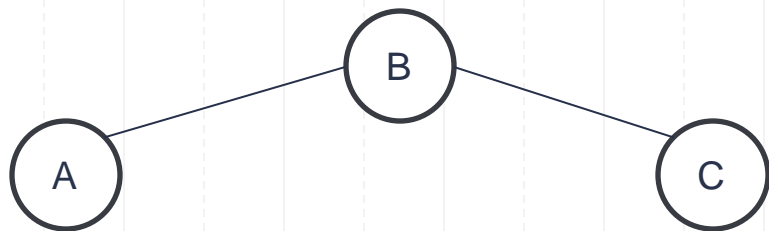
“
THUẬT TOÁN THAM LAM
”



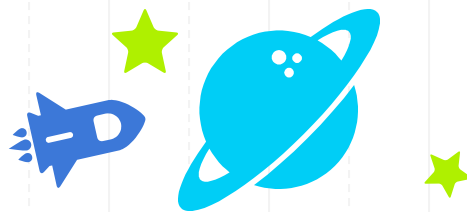
Quyết định lựa chọn tham lam:

Mỗi đỉnh sử dụng thông tin cục bộ ở tất cả các đỉnh và “chọn” đỉnh hướng về nó có giá trị nhỏ nhất.

Cấu trúc con tối ưu:



Dijkstra's algorithm to find the shortest path



DẠNG THUẬT TOÁN PHỔ QUÁT



THÀNH PHẦN of THAM LAM



Tập hợp ứng viên để tìm ra giải pháp

Selection function

Xác định ứng viên tốt nhất và đưa vào giải pháp



Xác định một ứng viên có thể đóng góp giải pháp hay không?

Objective function

Chỉ định một giá trị cho giải pháp (*maximized/Minimized*)



Chỉ ra khi nào tìm được giải pháp hoàn chỉnh

Candidate set



Feasibility function



Solution function

```
// n defines the input set
Algorithm Greedy(a, n)
{
    // initialize solution set
    solution= NULL;

    for i = 1 to n do
    {
        //Selection Function
        x = Select(a);

        // Feasibility solution
        if Feasible(solution, x) then

            // Include x in the solution set
            solution = Union(solution, x);
    }
    return solution;
}
```

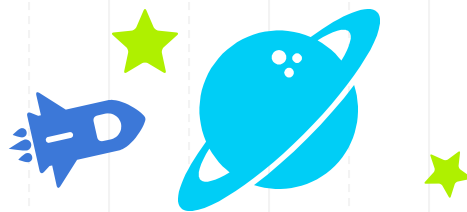
THUẬT TOÁN PHỔ QUÁT

Select

Dựa trên một Hàm mục tiêu (*Objective function*) chọn đầu vào từ **a**, gán giá trị cho **a** và loại bỏ nó.

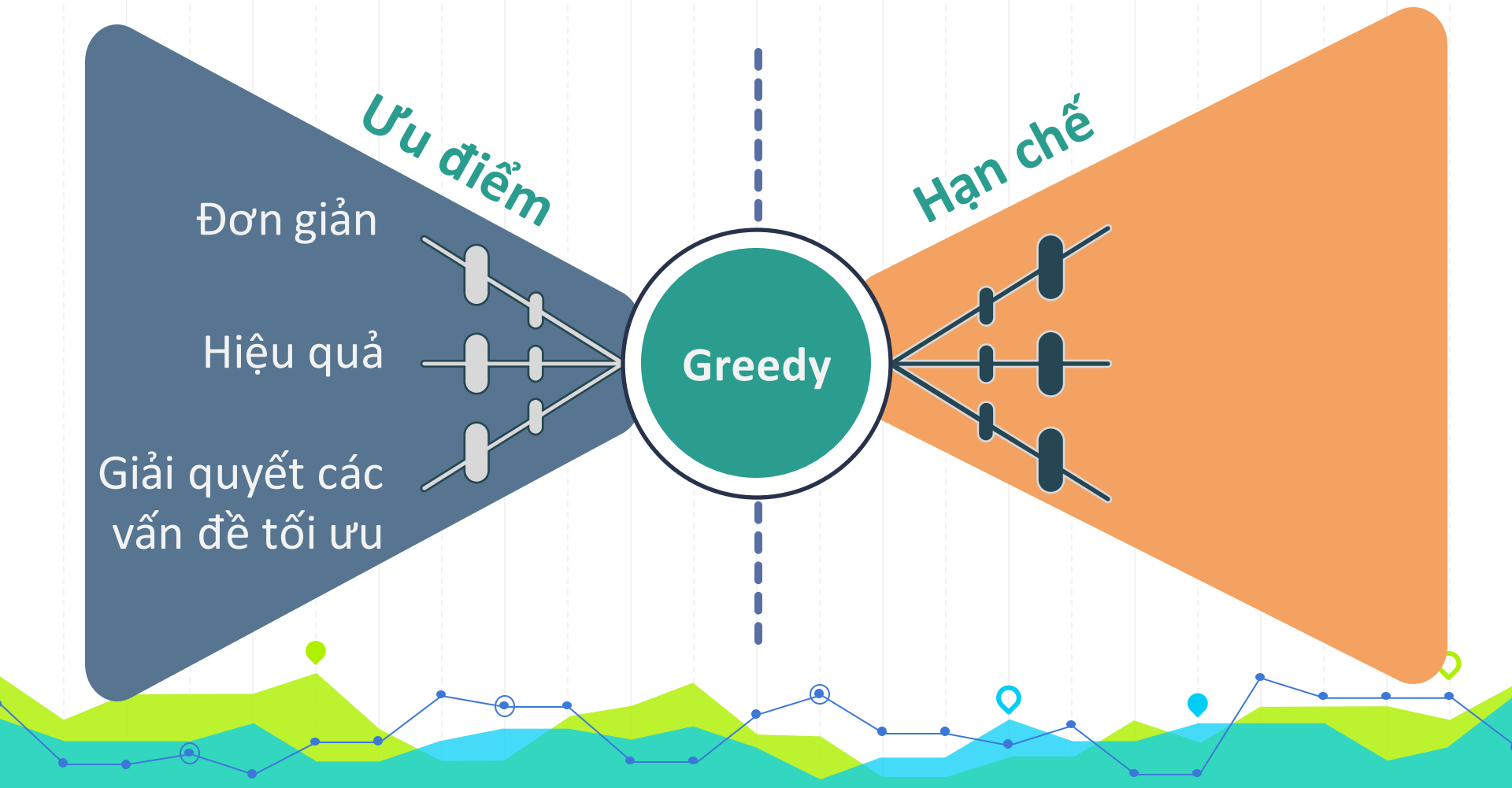
Feasible

Một hàm boolean để quyết định **x** có thể được đưa vào tập giải pháp mà không vi phạm bất kỳ ràng buộc nào không.



ƯU ĐIỂM và HẠN CHẾ

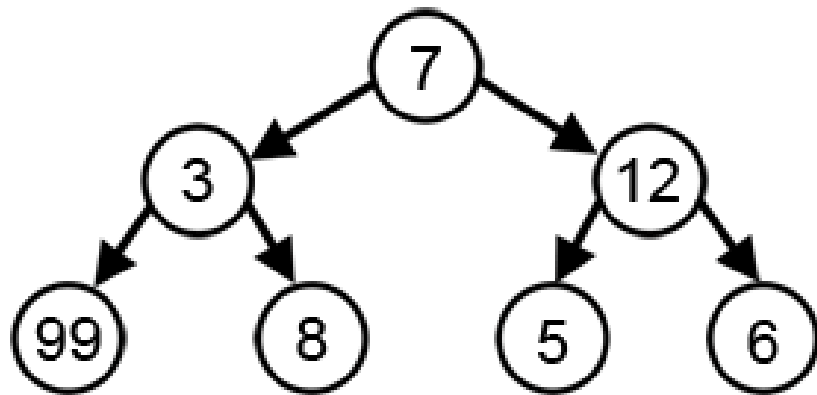


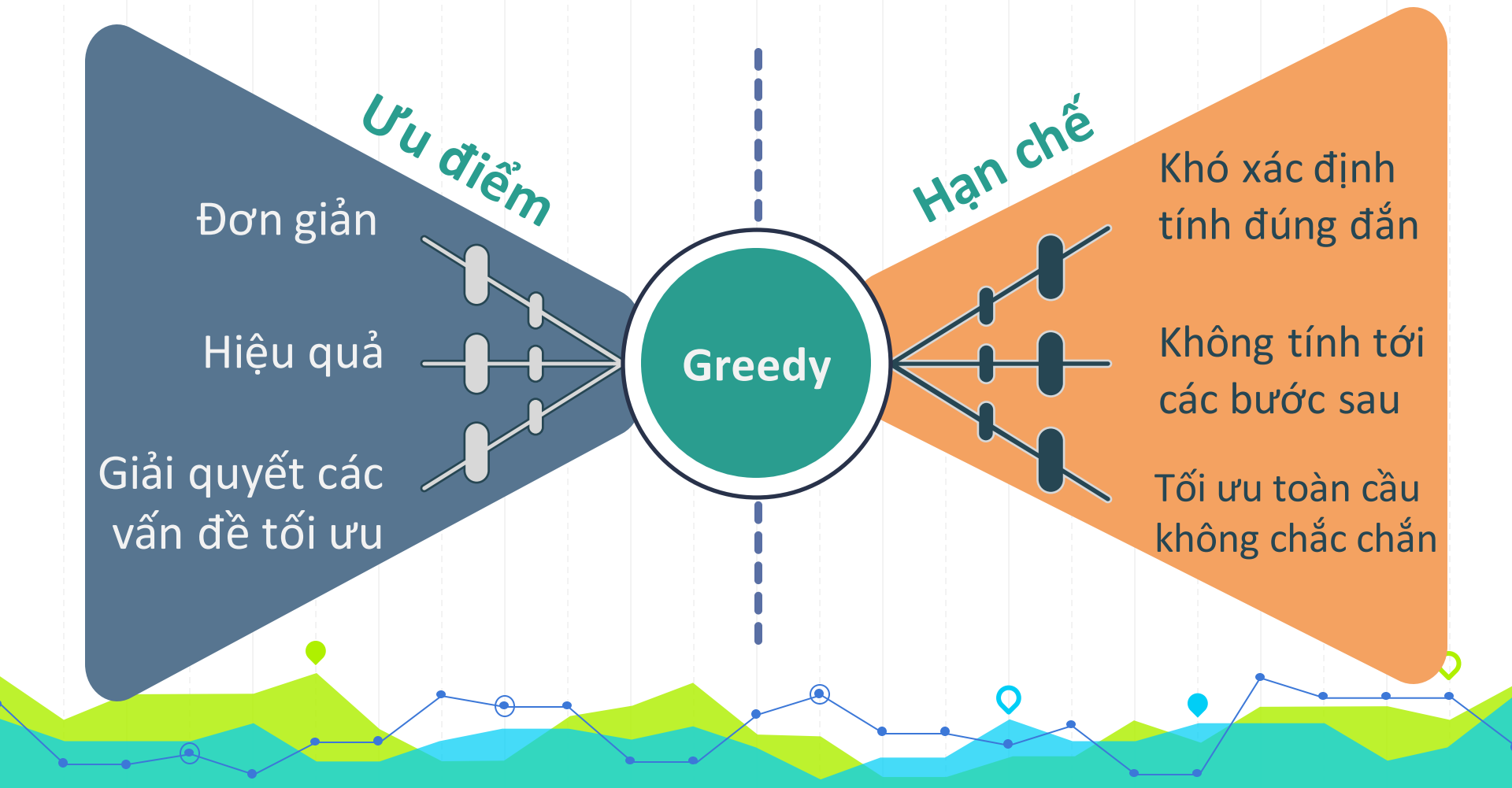


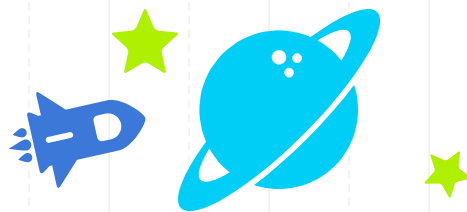


Thuật toán tham lam tìm kiếm
đường đi có tổng lớn nhất?

- *Thuật toán:* $7 \rightarrow 12 \rightarrow 6$
- *Thực tế:* $7 \rightarrow 3 \rightarrow 99$







NHẬN DẠNG BÀI TOÁN



**Quyết định lựa chọn
Tham lam & Cấu trúc
con tối ưu**

Minimization Problem
"smallest – shortest –
minimize"

Maximization Problem
"largest – longest –
maximize"

**Optimization
Problem**



- Kruskal's Minimum Spanning Tree
- Dijkstra's Shortest Path Algorithm
- Minimum product subset of an array
- Job Sequencing Problem
- Activity Selection Problem
- Huffman Coding
- First Fit algorithm in Memory Management
- Shortest Job First Scheduling
- Minimum number of coins required
- Set cover problem

Negative Weight Edges
"Shortest path"

Single/Multiple Sources
Multiple Destinations

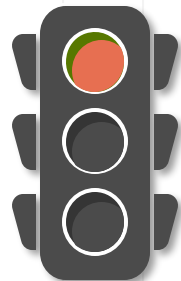
- Traveling Salesman
- Vertex Cover

**Thứ tự lựa chọn quan trọng/
Giới hạn cho giải pháp**

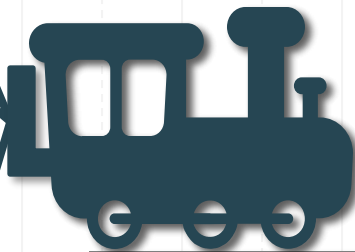
- Min Num Of Coins
- 0-1 Knapsack Problem

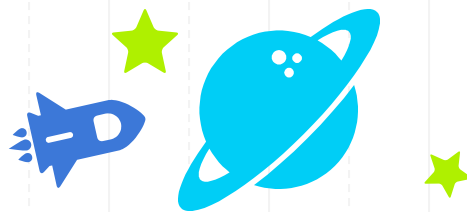


**Cẩn thận
lựa chọn**



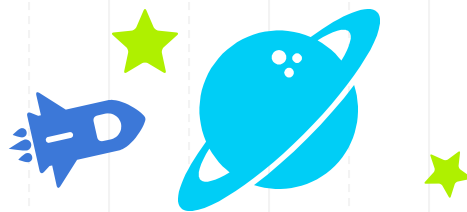
**Greedy
Algorithm**





ĐỘ PHỨC TẠP

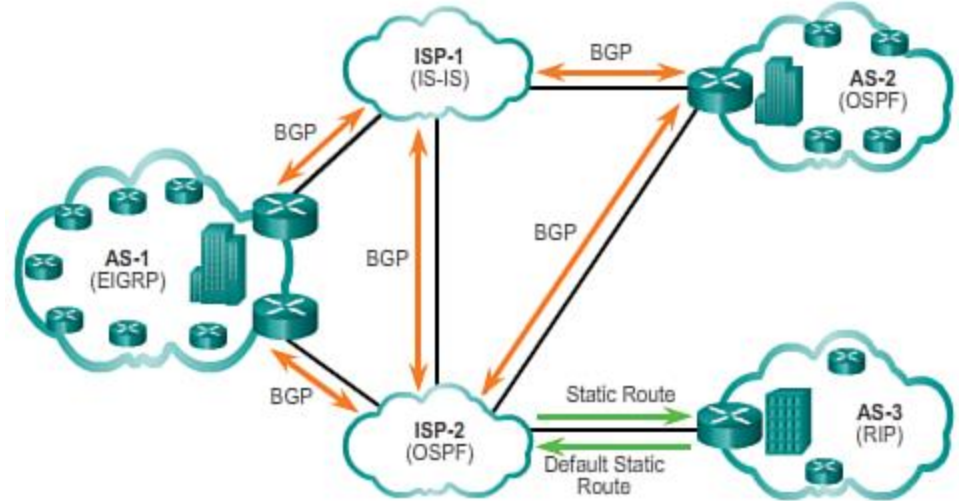




ỨNG DỤNG THỰC TẾ

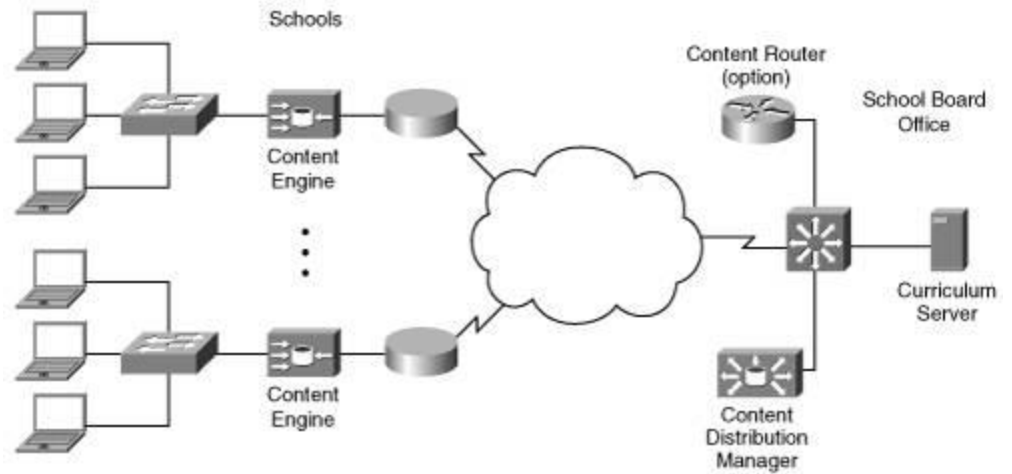
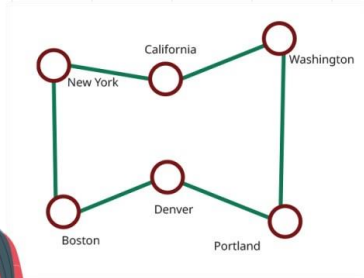


Dijkstra's Algorithm



Prim's Kruskal's Algorithm

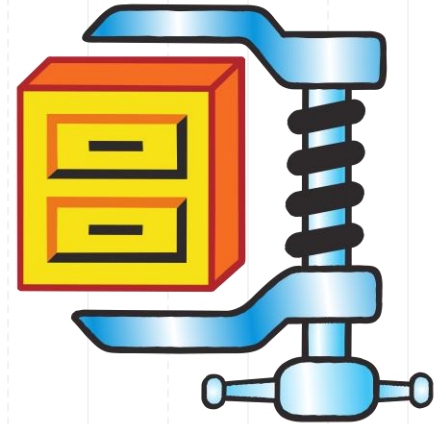
I want to find the shortest route to visit each city and return back to the place where I started.



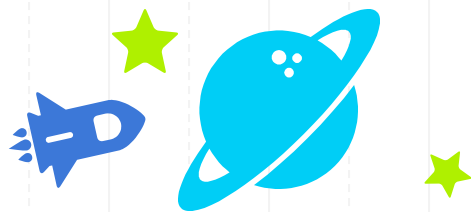
Huffman Coding



IMAGE FORMAT COMPARISON



WinZip



VÍ DỤ





Bài tập 1: Khi Tham lam tốt

Đề bài: bạn được cung cấp n hoạt động với thời gian bắt đầu và thời gian kết thúc. Chọn số lượng tối đa hoạt động mà có thể thực hiện bởi một người. Giả định rằng mỗi người chỉ có thể làm tối đa một công việc một lúc.



Ý tưởng:

- Sắp xếp theo thời gian kết thúc tăng dần của các hoạt động
- Chọn thời gian kết thúc sớm nhất và đảm bảo thời gian bắt đầu của nó muộn hơn thời gian kết thúc của các hoạt động được chọn trước đó.

Ví dụ: Cho 3 hoạt động được sắp xếp theo thời gian bắt đầu và thời gian kết thúc.

`start[] = {10, 12, 20};`

`finish[] = {20, 25, 30};`



```
def MaxActivities(arr, n):
    selected = []
    Activity.sort(key = lambda x:x[1])
    i = 0
    selected.append(arr[i])
    for j in range(1, n):
        if arr[j][0] >= arr[i][1]:
            selected.append(arr[j])
            i = j
    return selected

Activity = [[5, 9], [1, 2], [3, 4], [0,
6],[5, 7], [8, 9]]
n = len(Activity)
selected = MaxActivities(Activity, n)
print("Following activities are selected
:")
print(selected)
```

Following activities are selected:
[[1, 2], [3, 4], [5, 7], [8, 9]]

Độ phức tạp: $O(n \log n)$



Bài tập 2: Khi Tham lam thất bại

Đề bài: Một kẻ trộm đột nhập vào một cửa hiệu tìm thấy có n mặt hàng có trọng lượng và giá trị khác nhau, nhưng hắn chỉ mang theo một cái túi có sức chứa về trọng lượng tối đa là M . Vậy kẻ trộm nên bỏ vào ba lô những món nào và số lượng bao nhiêu để đạt giá trị cao nhất trong khả năng mà hắn có thể mang đi được.



Ý tưởng:

- Chọn giá trị càng cao và khối lượng thấp.
- Tỷ lệ giá trị của vật, cao là tốt (Giá trị/Khối lượng)
- Chọn theo tỷ lệ giảm dần, khối lượng còn lại của túi có thể đựng.

Ví dụ: Tên trộm có một túi đồ có thể đựng được 37 kg, cùng với đó là 4 món đồ với giá trị và khối lượng tương ứng.

Đồ vật	Khối lượng	Giá trị
A	15	30
B	10	25
C	2	2
D	4	6

Bước 1:

Đồ vật	Khối lượng	Giá trị	Tỉ lệ
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

Bước 2:

$$\rightarrow 3 * A (30kg) + 1 * D (4kg) + C (2kg) = 36 \text{ kg} \leq 37kg$$



Ví dụ: Tên trộm có một túi đồ có thể đựng được 10 kg, cùng với đó là 3 món đồ với giá trị và khối lượng tương ứng.

Đồ vật	Khối lượng	Giá trị
A	7	9
B	6	6
C	4	4



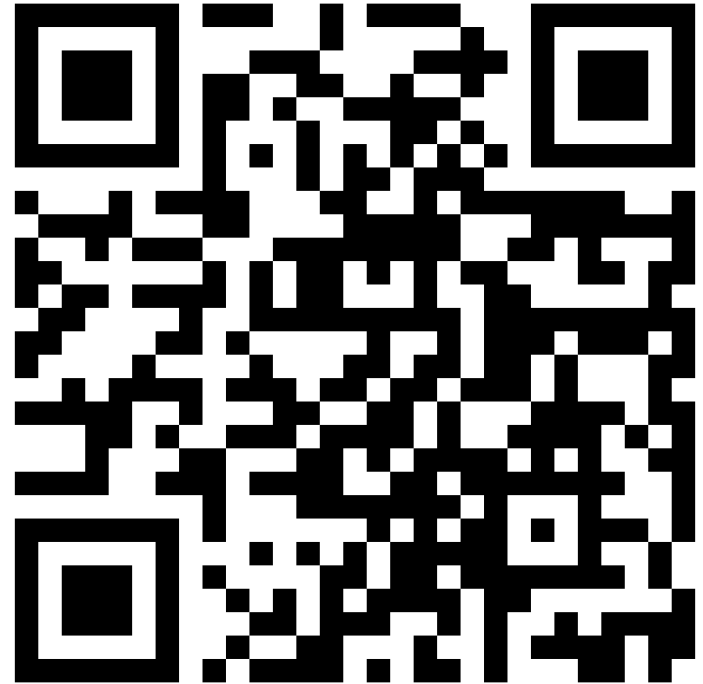
Đồ vật	Khối lượng	Giá trị	Tỉ lệ
A	7	9	9/7
B	6	6	1
C	4	4	1

- **Tham lam:** A (7kg) – giá trị 9
- **Thực tế:** B (6kg) + C (4kg) → giá trị 10

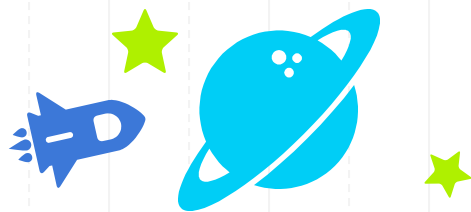


Room Name: SON5423

- Làm theo nhóm
- Tên nhóm: **Nxx**
VD: N17, N02



b.socrative.com/login/student



GIẢI ĐÁP



```
function Dijkstra(Graph, source):
```

```
    create vertex priority queue Q
```

```
    for each vertex v in Graph:
```

```
        dist[v] ← INFINITY
```

```
        prev[v] ← UNDEFINED
```

```
        add v to Q
```

```
    dist[source] ← 0
```

```
    while Q is not empty:
```

```
        u ← vertex in Q with min dist[u]
```

```
        remove u from Q
```

```
        for each neighbor v of u:
```

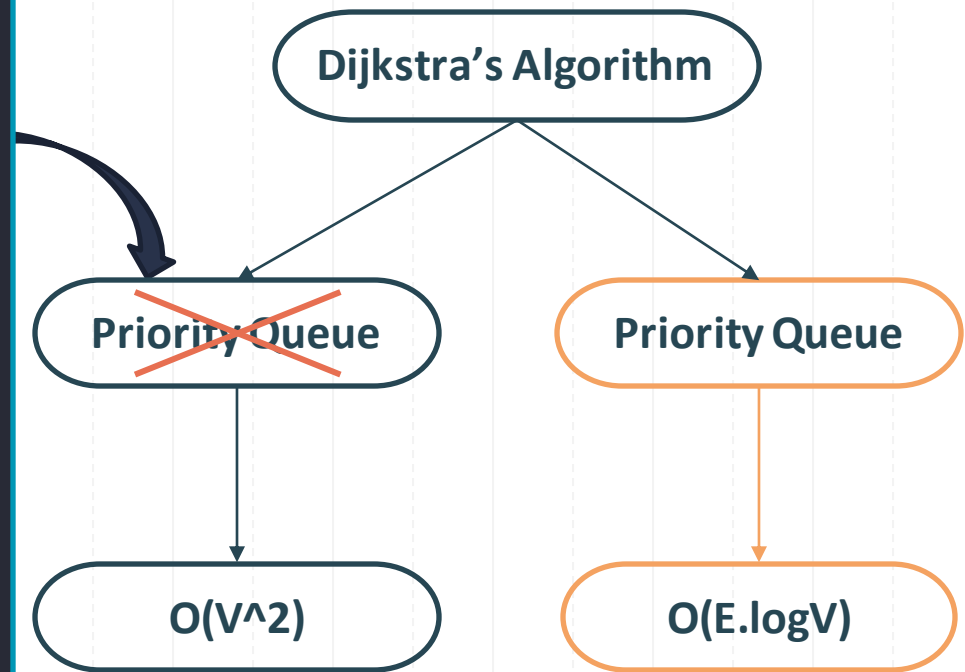
```
            alt ← dist[u] + length(u, v)
```

```
            if alt < dist[v]
```

```
                dist[v] ← alt
```

```
                prev[v] ← u
```

```
    return dist, prev
```



Dijkstra's Algorithm

~~Priority Queue~~

$O(V^2)$

Priority Queue

$O(E \cdot \log V)$

```
function Dijkstra(Graph, source):
```

```
    dist[source] ← 0
```

```
    create vertex priority queue Q
```

```
    for each vertex v in Graph:
```

```
        if v ≠ source
```

```
            dist[v] ← INFINITY
```

```
            prev[v] ← UNDEFINED
```

```
        Q.add_with_priority(v, dist[v])
```

```
    while Q is not empty:
```

```
        u ← Q.extract_min()
```

```
        for each neighbor v of u:
```

```
            alt ← dist[u] + length(u, v)
```

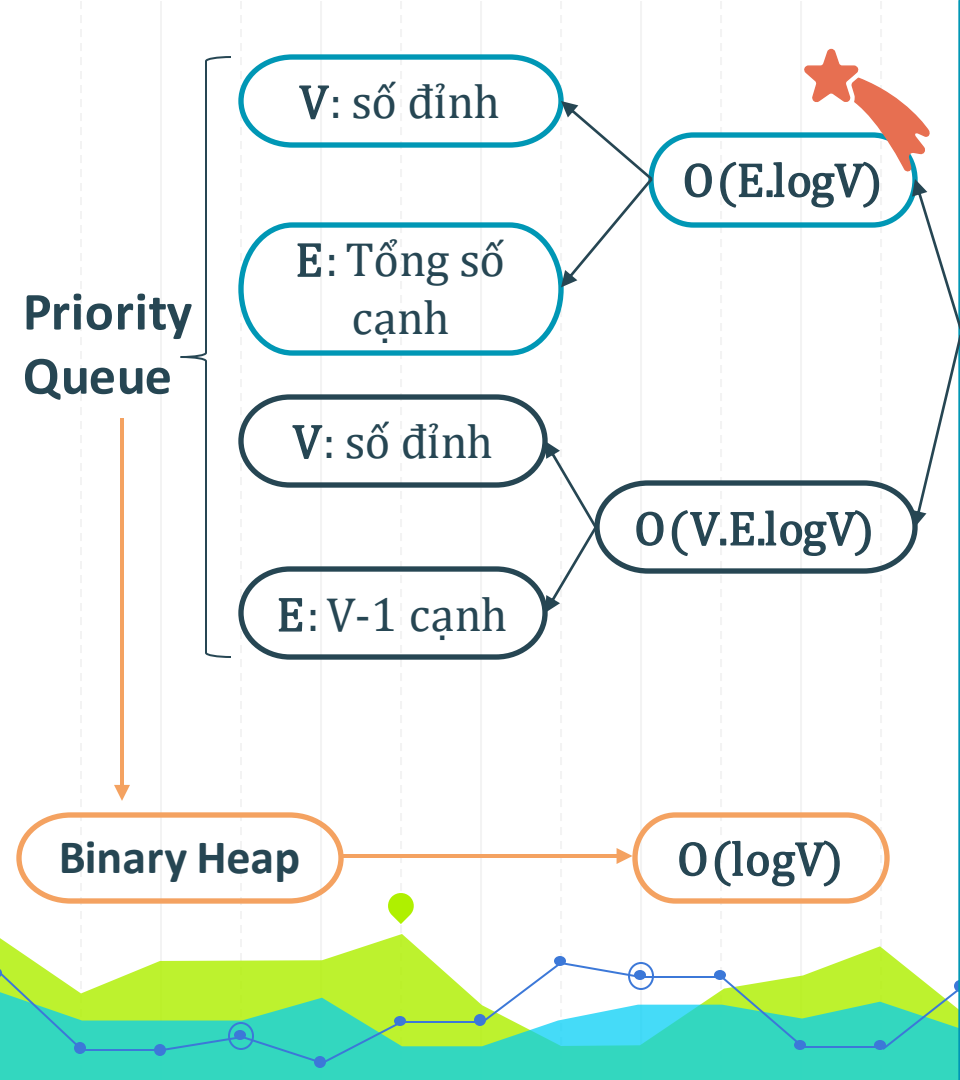
```
            if alt < dist[v]
```

```
                dist[v] ← alt
```

```
                prev[v] ← u
```

```
                Q.decrease_priority(v, alt)
```

```
    return dist, prev
```

```
function Dijkstra(Graph, source):
```

```
    dist[source] ← 0
```

```
    create vertex priority queue Q
```

```
    for each vertex v in Graph:
```

```
        if v ≠ source
```

```
            dist[v] ← INFINITY
```

```
            prev[v] ← UNDEFINED
```

```
    Q.add_with_priority(v, dist[v])
```

```
    while Q is not empty:
```

```
        u ← Q.extract_min()
```

```
        for each neighbor v of u:
```

```
            alt ← dist[u] + length(u, v)
```

```
            if alt < dist[v]
```

```
                dist[v] ← alt
```

```
                prev[v] ← u
```

```
            Q.decrease_priority(v, alt)
```

```
    return dist, prev
```

Bài toán trồng hoa

Đề: Mỗi loài hoa lại có thời gian phát triển từ lúc trồng tới lúc nở hoa khác nhau. Giúp ông nông dân tìm ra ngày sớm nhất mà tất cả loài hoa đều nở hoa.

Ngày trồng	Loài hoa	Thời gian nở
1	Hoa Hồng (1)	3
2	Hoa Cúc (2)	4
3	Hoa Lan (3)	2
4	Hoa Mười giờ (4)	1

Ý tưởng:

- Chọn loài hoa theo thứ tự sắp xếp thời gian sẽ nở giảm dần.

Ngày trồng	Loài hoa	Thời gian sẽ nở (ngày)
1	Hoa Cúc (2)	4
2	Hoa Hồng (1)	3
3	Hoa Lan (3)	2
4	Hoa Mười giờ (4)	1

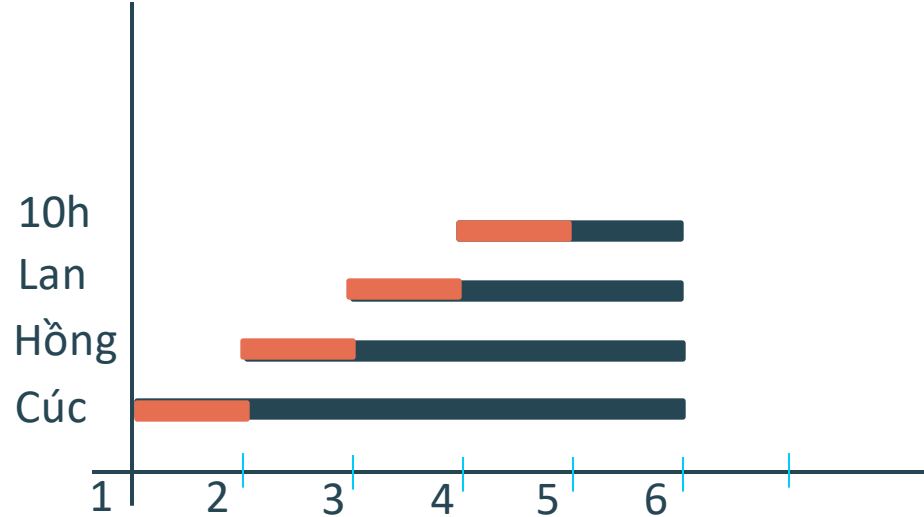
→ Kết quả: 6 ngày (2-1-3-4)



SAI LẦM:

Ngày trồng	Loài hoa	Thời gian sẽ nở (ngày)	Tổng
1	Hoa Cúc (2)	4	5
2	Hoa Hồng (1)	3	5
3	Hoa Lan (3)	2	5
4	Hoa Mười giờ (4)	1	5

→ Kết quả: 5 ngày (2-1-3-4)



Thông tin nhóm



Thành viên:

Họ và tên	MSSV
Trương Thế Tấn	19522180
Nông Thanh Hồng	19521551
Nguyễn Thành Luân	19521809
Đinh Trọng Tùng Sơn	19522132

THANK YOU!

