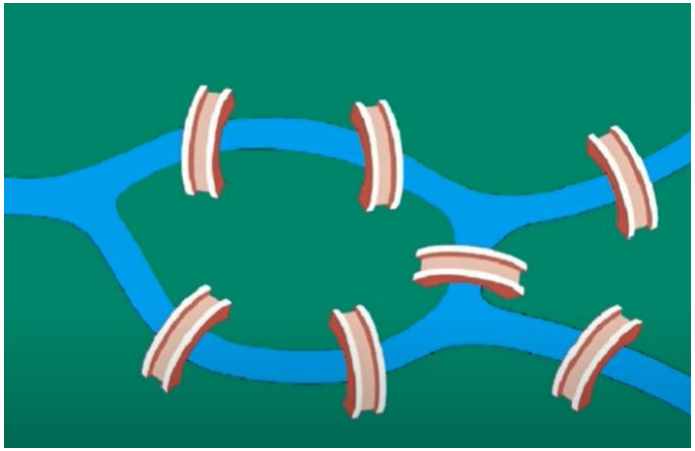
LÝ THUYẾT ĐỒ THỊ

1. Giới thiệu và khái niệm về đồ thị
2. Thuật toán duyệt đồ thị
3. Thuật toán tìm cây khung nhỏ nhất
4. Thuật toán đường đi ngắn nhất
5. Thuật toán và ứng dụng Luồng cực đại
6. Tài liệu tham khảo

Nhóm thực hiện: Nhóm 11
Thành viên:
+ **Hứa Thanh Tân KHTN2019**
+ **Ngô Quang Vinh KHTN2019**
+ **Ngô Hữu Mạnh Khanh KHTN2019**

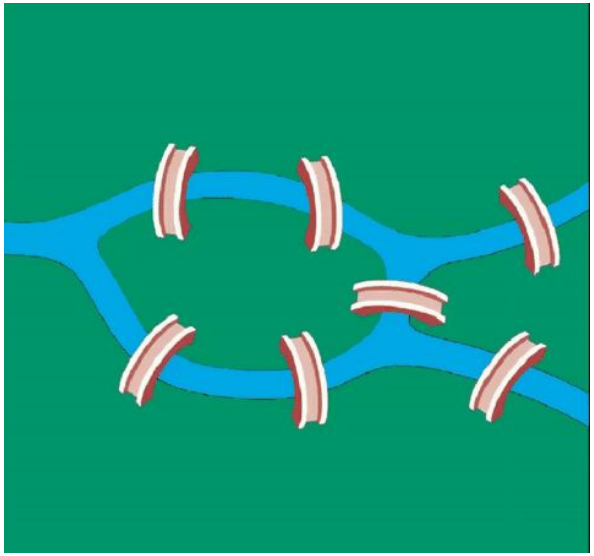
1. Giới thiệu và khái niệm về đồ thị

a. Giới thiệu



- Bài toán 7 câu cầu của Euler:

Tìm một tuyến đường mà đi qua mỗi cây cầu một lần và chỉ đúng một lần



=> Không tồn tại lời giải cho bài toán này

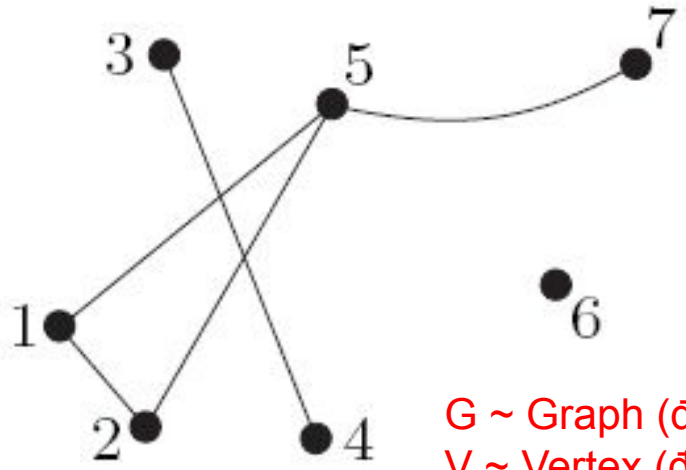
=> Tạo thành cơ sở phát triển của lý thuyết đồ thị

1. Giới thiệu và khái niệm về đồ thị

b. Khái niệm

- $G = (V, E)$ với $V \neq \varnothing$
  - V: tập các đỉnh
  - E: tập các cạnh

- Cạnh $e \in E$
  - ứng với 2 đỉnh $v, w \in V$
  - v, w gọi là 2 đỉnh kề
  - $v \equiv w$: e được gọi là vòng (khuyên) tại v



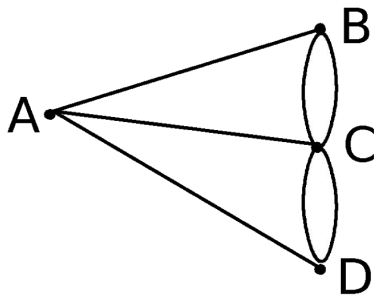
G ~ Graph (đồ thị)
V ~ Vertex (đỉnh)
E ~ Edge (cạnh)

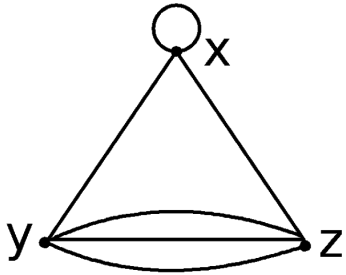
$$V = \{1, \ldots, 7\}$$
$$E = \{\{1,2\}, \{1,5\}, \{2,5\}, \{3,4\}, \{5,7\}\}$$

1. Giới thiệu và khái niệm về đồ thị

b. Khái niệm

- *Cạnh bội* (song song)
  - Hai cạnh phân biệt cùng tương ứng với một cặp đỉnh
- *Đơn đồ thị*
  - Đồ thị không có vòng và cạnh song song
- *Đa đồ thị*
  - Các đồ thị không phải là đơn đồ thị





- Hai đỉnh v, u trong đồ thị G được gọi là liên thông nếu tồn tại một đường đi nối chúng với nhau.
- Đồ thị G gọi là liên thông nếu hai đỉnh phân biệt bất kỳ trong đồ thị đều liên thông. Ngược lại thì ta gọi là đồ thị không liên thông.

1. Giới thiệu và khái niệm về đồ thị

c. Dấu hiệu nhận biết các bài toán đồ thị

- Không có quy chuẩn chung nào cho các bài toán đồ thị -> tùy thuộc vào cách nhìn nhận bài toán của từng người.
- Tồn tại nhiều trạng thái (node), và giữa các trạng thái có mối liên hệ với nhau (đường đi).
- Có một số keyword như: đường ngắn nhất, cycle, đỉnh, cạnh, ...

d. Cách giải quyết các bài toán đồ thị

- Đọc kĩ đề và nhận diện dạng đồ thị -> cây, rừng, đồ thị đầy đủ?
- Đề yêu cầu gì? -> thuật toán (Dijkstra, Bellman Ford, Floyd, …)

=> vận dụng Computational thinking đã học để tiếp tục giải quyết bài toán

2. Duyệt đồ thị

a. BFS (Duyệt theo chiều rộng)

- Xuất phát từ một đỉnh đi tới tất cả các đỉnh kề
- Tiếp tục đem đỉnh khác (từ tập đã được lưu) ra xét và đi cho đến khi không còn đỉnh nào có thể đi

=> vét cạn không gian bài toán và tìm được lời giải (nếu có)

Bài tập đã giải sử dụng BFS:
Water Supply (tuần 3)

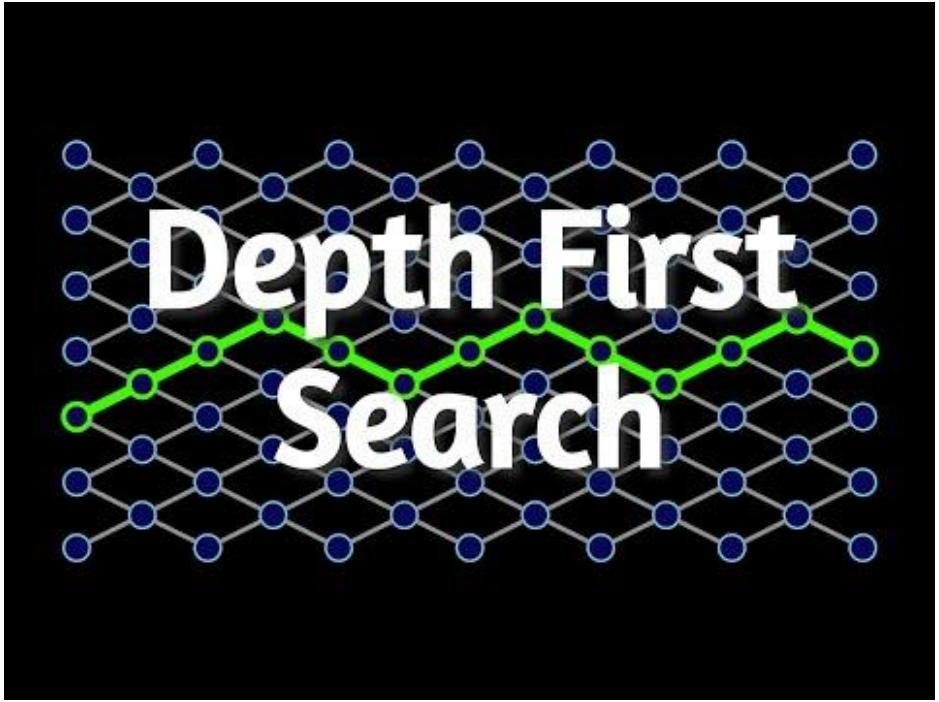
Tham khảo: https://github.com/vinhqngo5/CS112.L11.KHTN_Team011/blob/master/week3/Docs/Water_supply.ipynb

2. Duyệt đồ thị

b. DFS (Duyệt theo chiều sâu)

- Xuất phát từ một đỉnh và đi mãi cho đến khi không thể đi tiếp, sau đó quay lại đỉnh đầu.
- Trong quá trình quay lại:

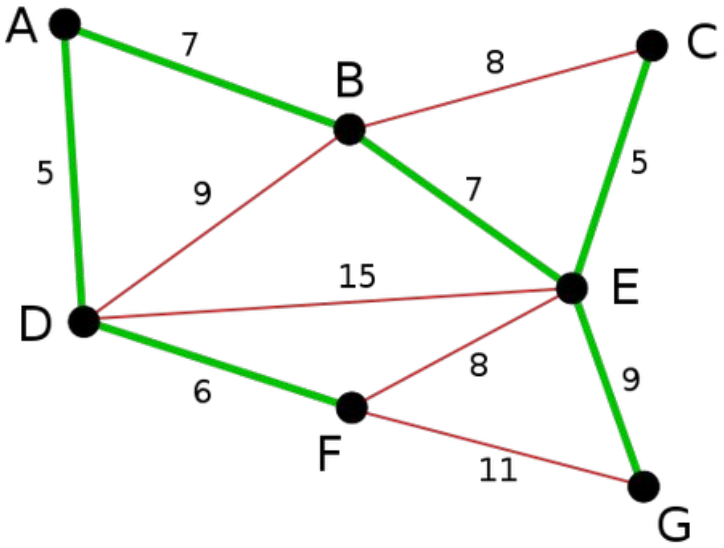
+ Nếu gặp đường đi khác thì đi cho đến khi không đi được nữa
+ Nếu không tìm ra đường đi nào khác thì ngừng tìm kiếm



Bài tập đã giải sử dụng DFS:
Water Supply (tuần 3)

Tham khảo: https://github.com/vinhqngo5/CS112.L11.KHTN_Team011/blob/master/week3/Docs/Water_supply.ipynb

3. Cây khung nhỏ nhất



Cây khung nhỏ nhất là bài toán tìm đồ thị liên thông sao cho tổng trọng số là nhỏ nhất

Ứng dụng:

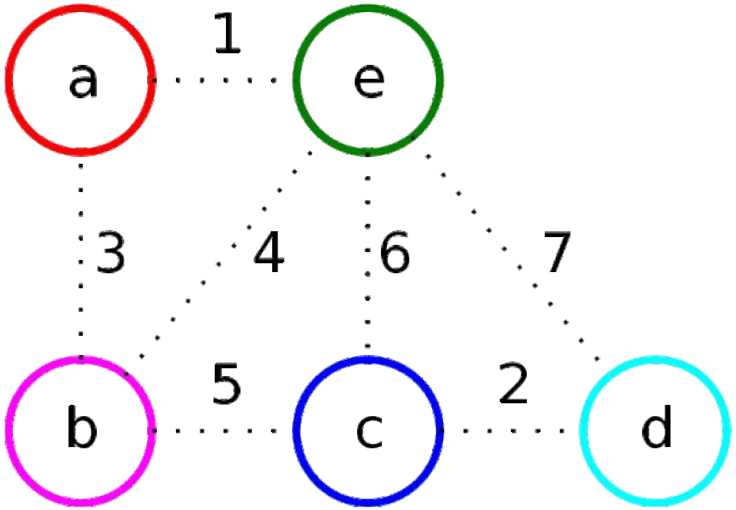
- Thiết kế hệ thống mạng, hệ thống điện, nước.
- Lên kế hoạch đi du lịch

3. Cây khung nhỏ nhất (Prim algorithm)

```
#Prim Algorithm
T = [] # save spanning tree
U = [0] # save vertices
while (U != V)
    let (u,v) be the lowest cost edge
        such that u ∈ U and v ∈ V - U
    add (u,v) to T
    add v to U
MST = T
```

3. Cây khung nhỏ nhất (Kruskal algorithm)

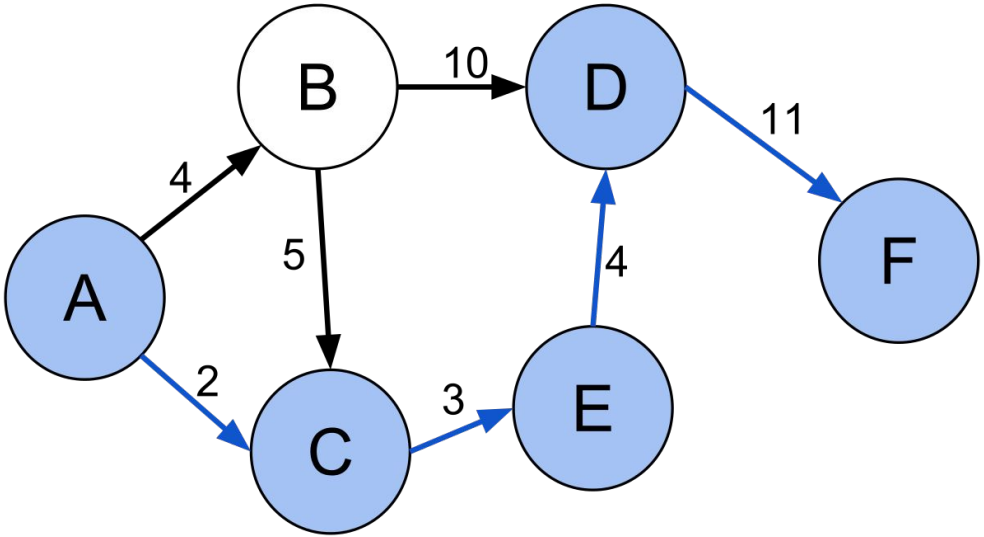
Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6




```
#Kruskal Algorithm
Sort E edges by increasing weight
T = [] # save spanning tree
for (i = 0; i < length(edgeList);i++)
    if adding e = edgeList[i] does not form a cycle
        add e to T
    else ignore e
MST = T
```

4. Đường đi nhỏ nhất

Cho đồ thị, tìm đường đi ngắn nhất từ một điểm đến các đỉnh còn lại



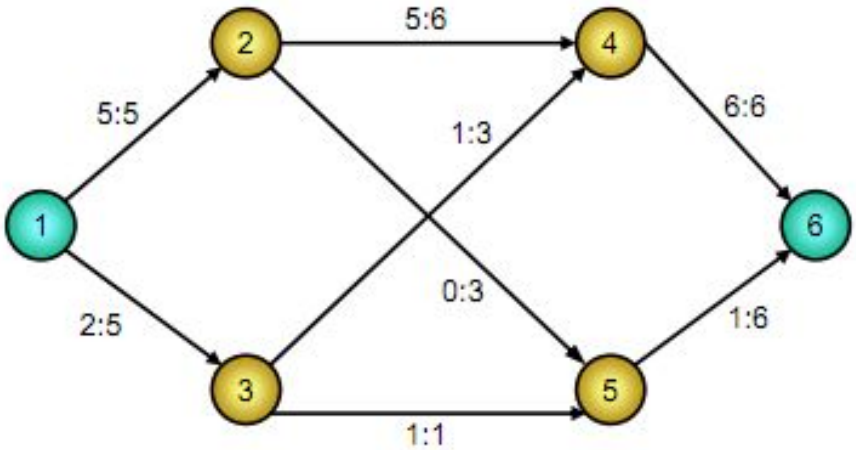
Ứng dụng:

- Tìm đường đi ngắn nhất
- Giải thuật định tuyến (Link-state Routing)

4. Đường đi nhỏ nhất

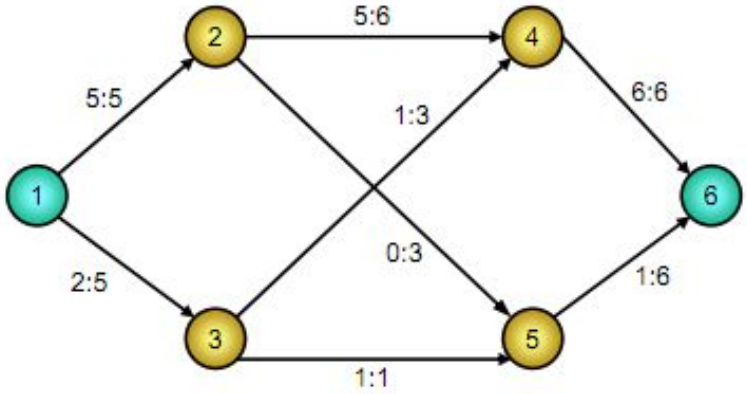
```
#Dijkstra Algorithm
sptSet = [1, 0, 0, ...] # check v is processed
dist = [0, INF, INF, ...] # save shortest path
For v in V:
    u = sptSet[u] == False and have min distance
    sptSet[u] = True
    for i in V:
        if sptSet[i] = False and
        dist[i] > dist[u] + weight(u,i):
            dist[i] = dist[u] + weight(u,i)
Shortest_path = dist
```

5. Luồng cực đại



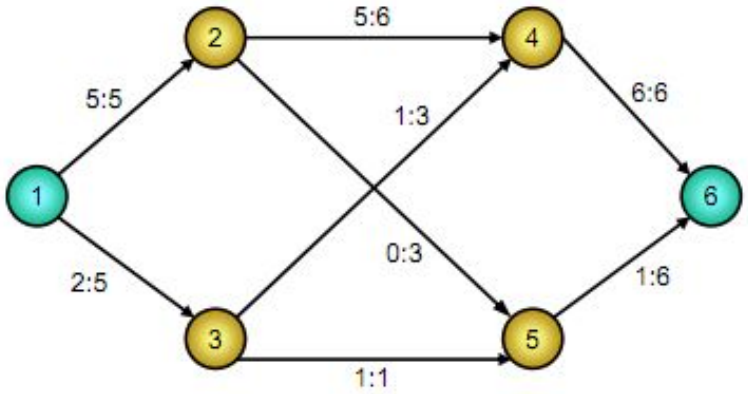
5. Luồng cực đại

a. Luồng cực đại trong chiến tranh



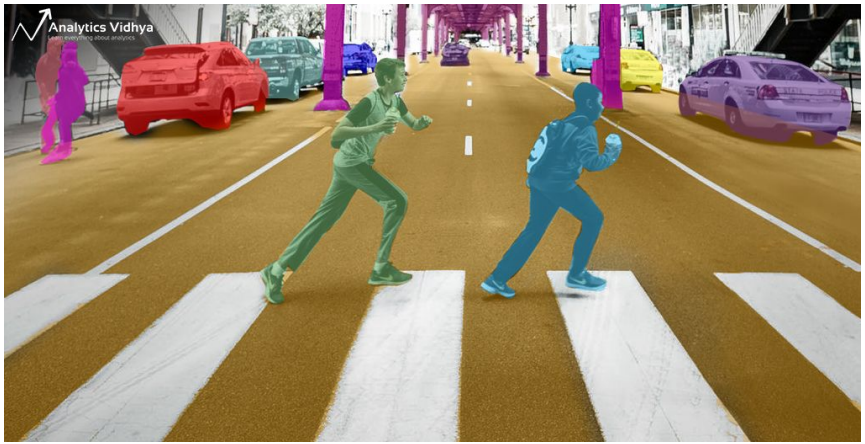
5. Luồng cực đại

b. Bài toán xếp lịch



5. Luồng cực đại

c. Ứng dụng trong Computer Vision (Image segmentation)

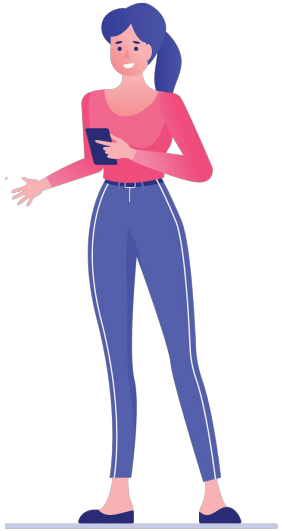


$$q(A,B) = \sum_{i \in A} a_i + \sum_{i \in B} b_i - \sum_{\substack{i,j \text{ adjacent} \\ |A \cap \{i,j\}|=1}} p_{ij},$$

Tài liệu tham khảo:

- Sách giải thuật và lập trình - thầy Lê Minh Hoàng
- Tài liệu chuyên tin
- Tài liệu trường Đại học Công nghệ Thông tin
- https://www.geeksforgeeks.org/
- https://csacademy.com/app/graph_editor/ (vẽ đồ thị)
- https://visualgo.net/en (trực quan hóa thuật toán)
- PIMA (Project in Mathematics and Applications)
- https://www.wikipedia.org/
- https://www.coursera.org/learn/algorithms-on-graphs
- Các bài toán đã được giải trên lớp

THANKS FOR WATCHING



Scan for Q&A