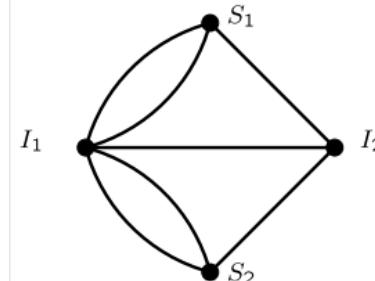
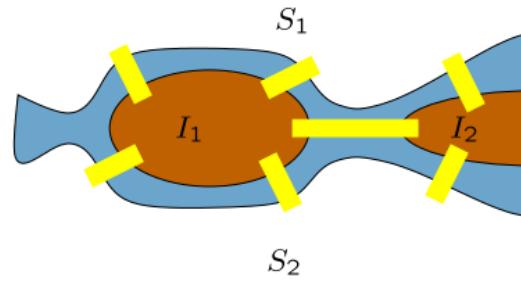


# LÝ THUYẾT ĐỒ THỊ

Phạm Nguyên Khang  
BM. Khoa học máy tính, CNTT  
[pnkhang@cit.ctu.edu.vn](mailto:pnkhang@cit.ctu.edu.vn)



Cần Thơ, 8/2021

# Nội dung

- Phần cơ bản
  - Định nghĩa & Phân loại
  - Một số đồ thị đặc biệt
  - Tính liên thông của đồ thị
  - Cây & cây có hướng

# Nội dung

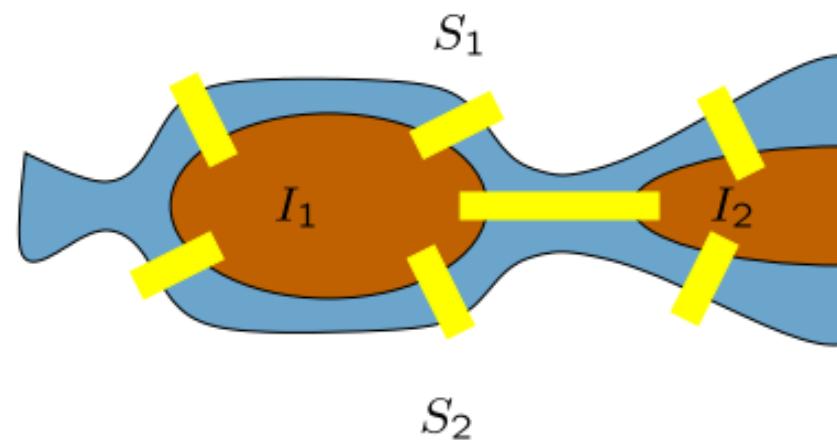
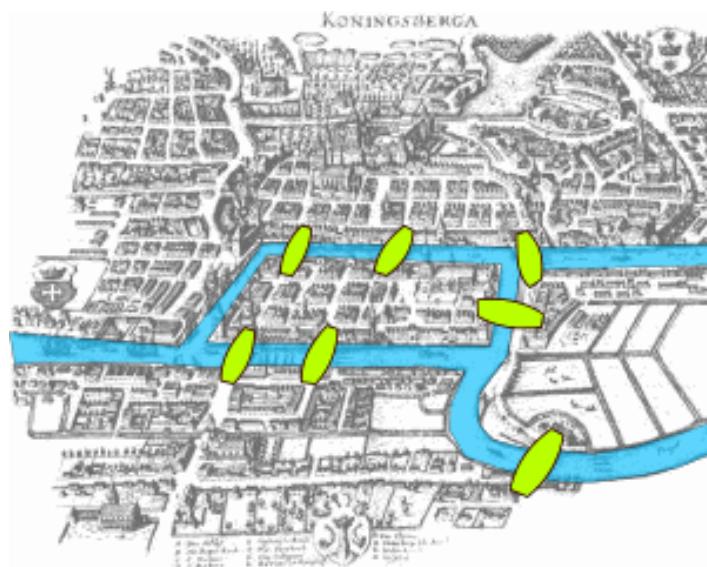
- Phần nâng cao
  1. Đồ thị Euler & ứng dụng
  2. Đồ thị Hamilton & ứng dụng
  3. Tìm đường đi ngắn nhất
  4. Xếp hạng đồ thị và bài toán tổ chức thi công
  5. Cây khung vô hướng có trọng lượng nhỏ nhất
  6. Cây khung có hướng có trọng lượng nhỏ nhất
  7. Luồng cực đại trên mạng

# Nội dung

- Giới thiệu
- Định nghĩa
  - Đô thị
  - Cung, đa cung, khuyên
  - Bậc của đô thị
- Phân loại đô thị
- Một số đô thị đặc biệt
- Tính liên thông của đô thị

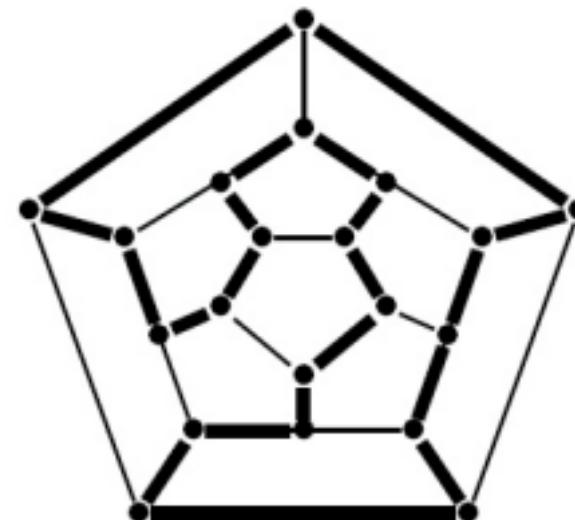
# Giới thiệu

- Lý thuyết đồ thị (LTĐT, Graph Theory)
  - Ngành khoa học phát triển từ rất lâu
  - Có nhiều ứng dụng hiện đại
  - Ý tưởng xuất phát từ nhà toán học Thụy Sĩ Leonhoard Euler (1707 – 1783) vào thế kỷ XVIII
  - Bài toán 7 cây cầu ở Königsberg



# Giới thiệu

- Hamilton (1805 - 1865) và bài toán du lịch

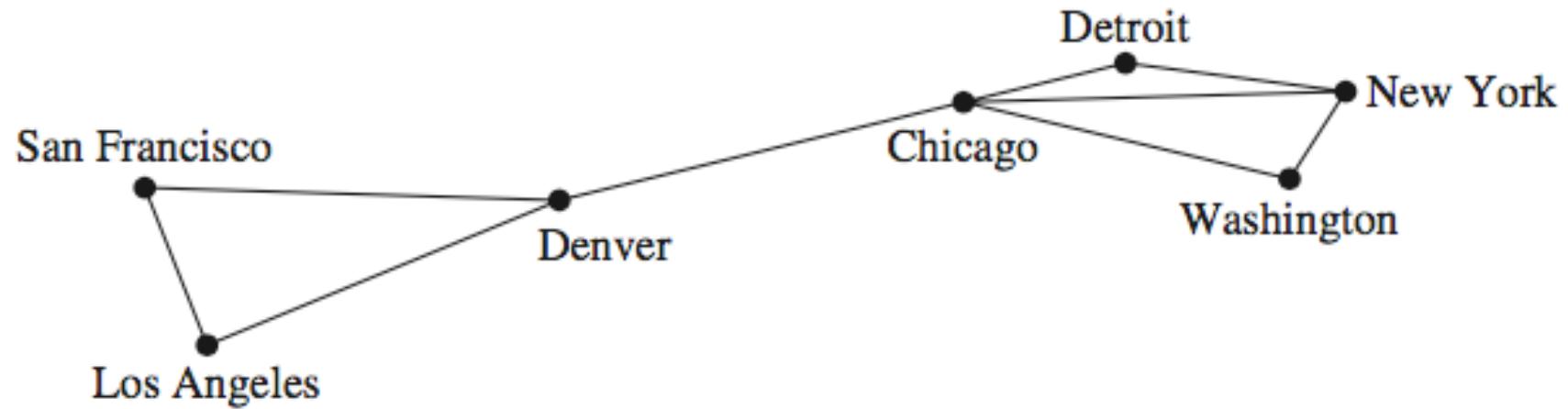


# Giới thiệu

- Ứng dụng của LTĐT
  - Thiết kế mạch điện
  - So sánh cấu trúc của các hợp chất hóa học
  - Xác định hai máy tính có thể kết nối được với nhau hay không
  - Tìm đường đi ngắn nhất trên bản đồ
  - Lập lịch thi, phân chia kênh cho các đài truyền hình

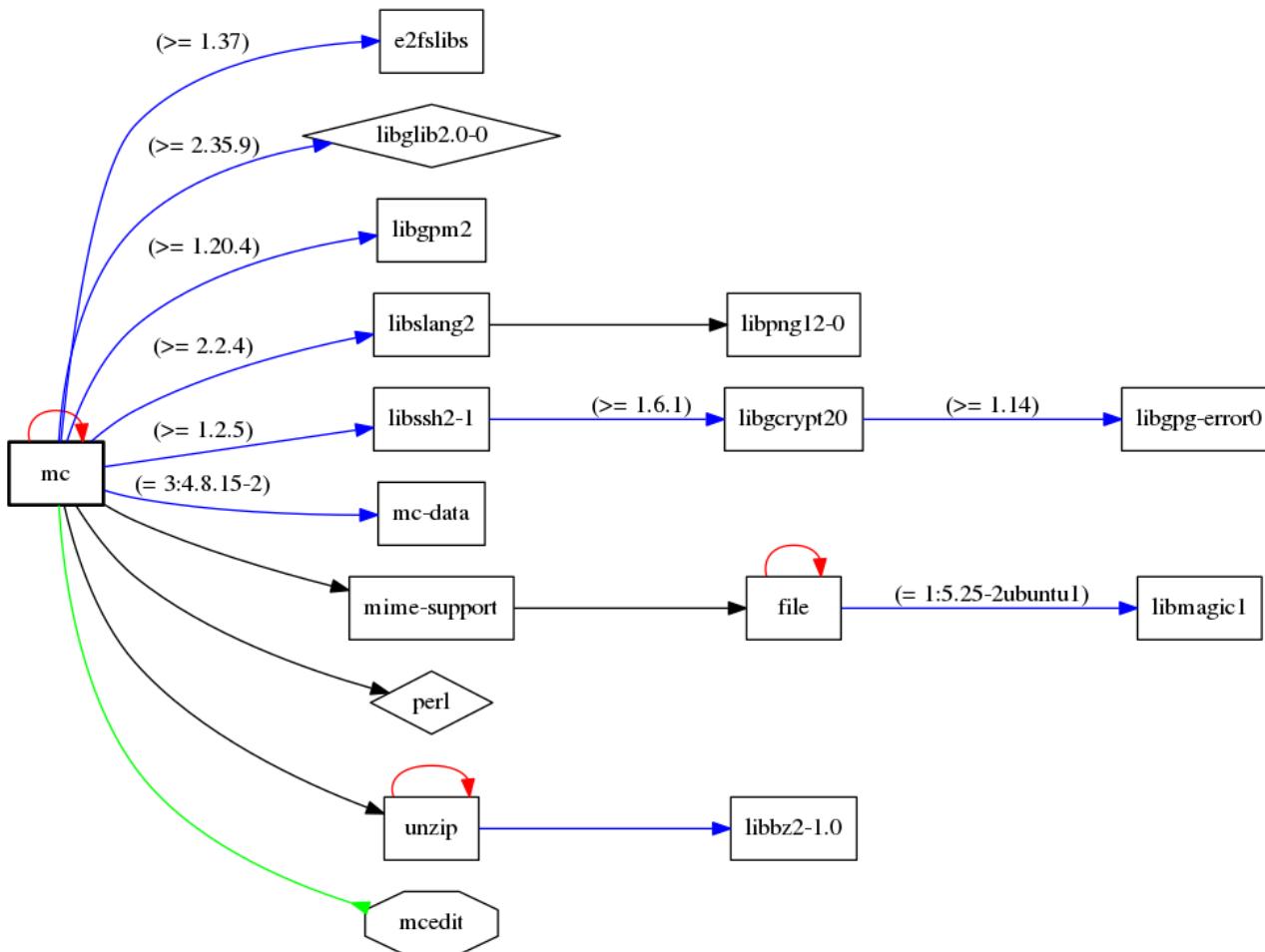
# Giới thiệu

- Mô hình mạng máy tính (computer network)



# Giới thiệu

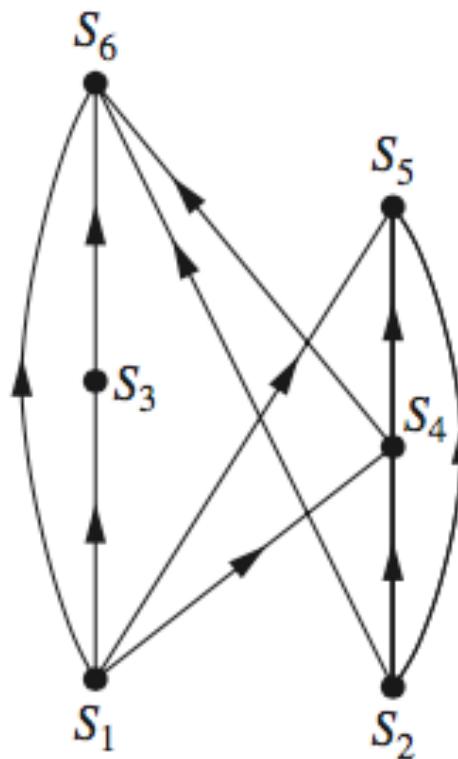
- Mỗi quan hệ phụ thuộc nhau giữa các gói phần mềm



# Giới thiệu

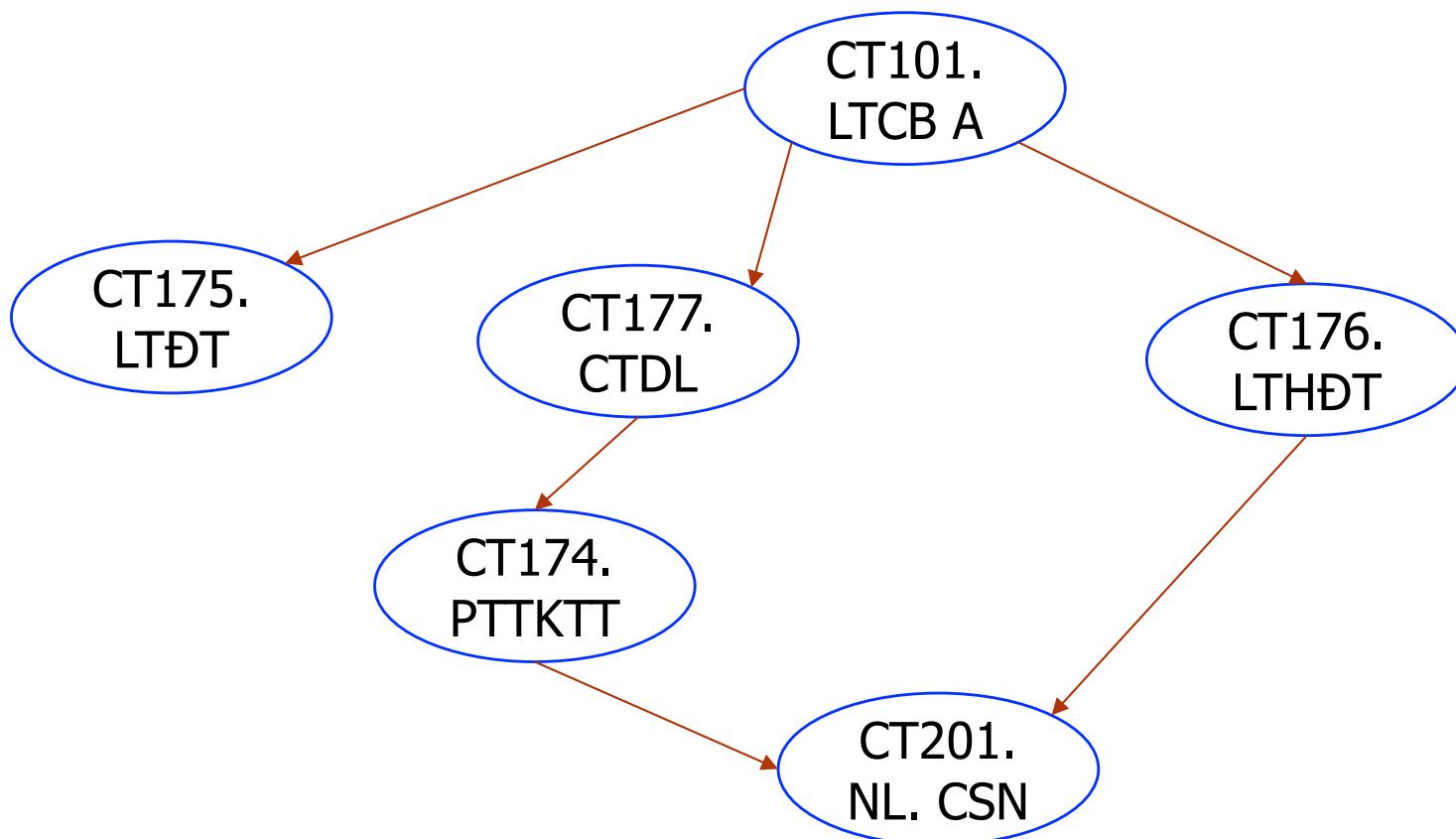
- Mỗi quan hệ phụ thuộc nhau giữa các lệnh

$S_1 \quad a := 0$   
 $S_2 \quad b := 1$   
 $S_3 \quad c := a + 1$   
 $S_4 \quad d := b + a$   
 $S_5 \quad e := d + 1$   
 $S_6 \quad e := c + d$



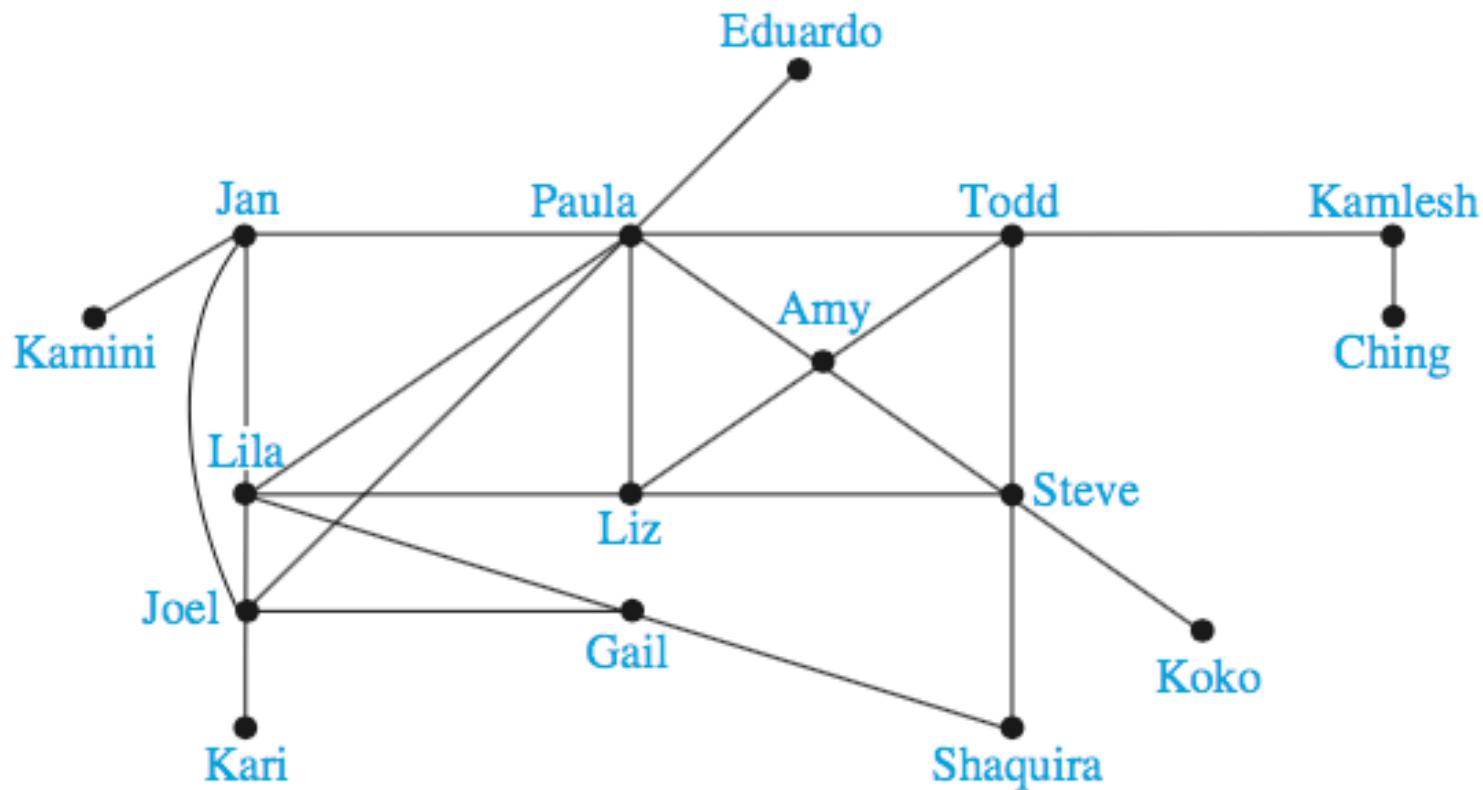
# Giới thiệu

- Mỗi quan hệ tiên quyết giữa các học phần



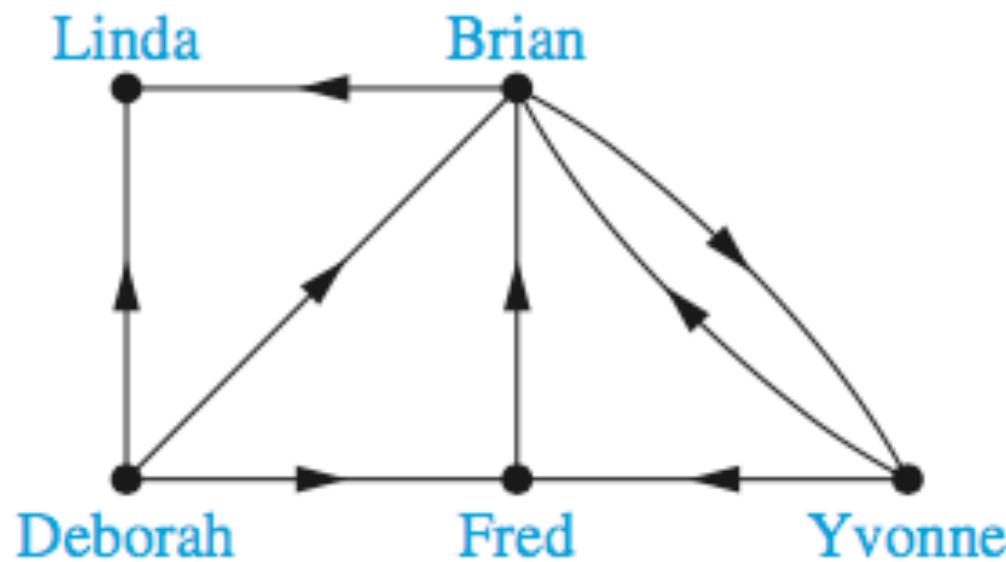
# Giới thiệu

- Mỗi quan hệ "quen biết nhau" qua mạng xã hội



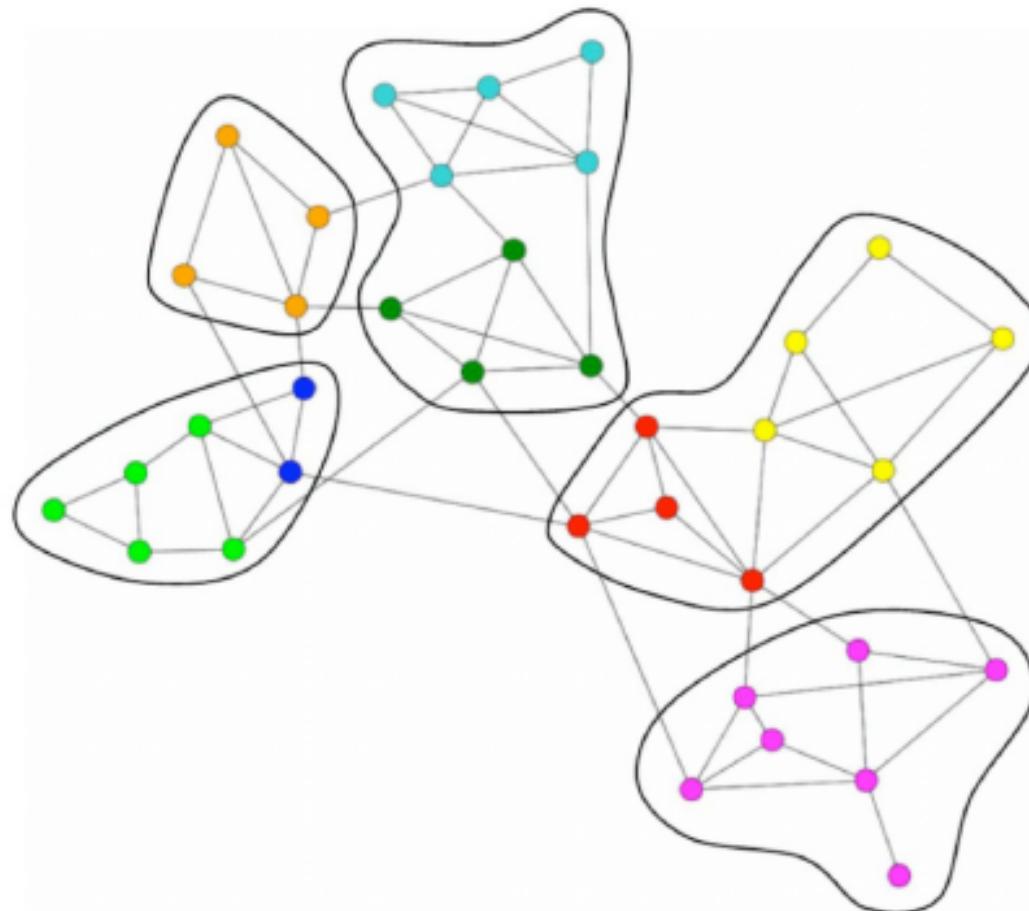
# Giới thiệu

- Mỗi quan hệ “ảnh hưởng nhau” trên mạng xã hội



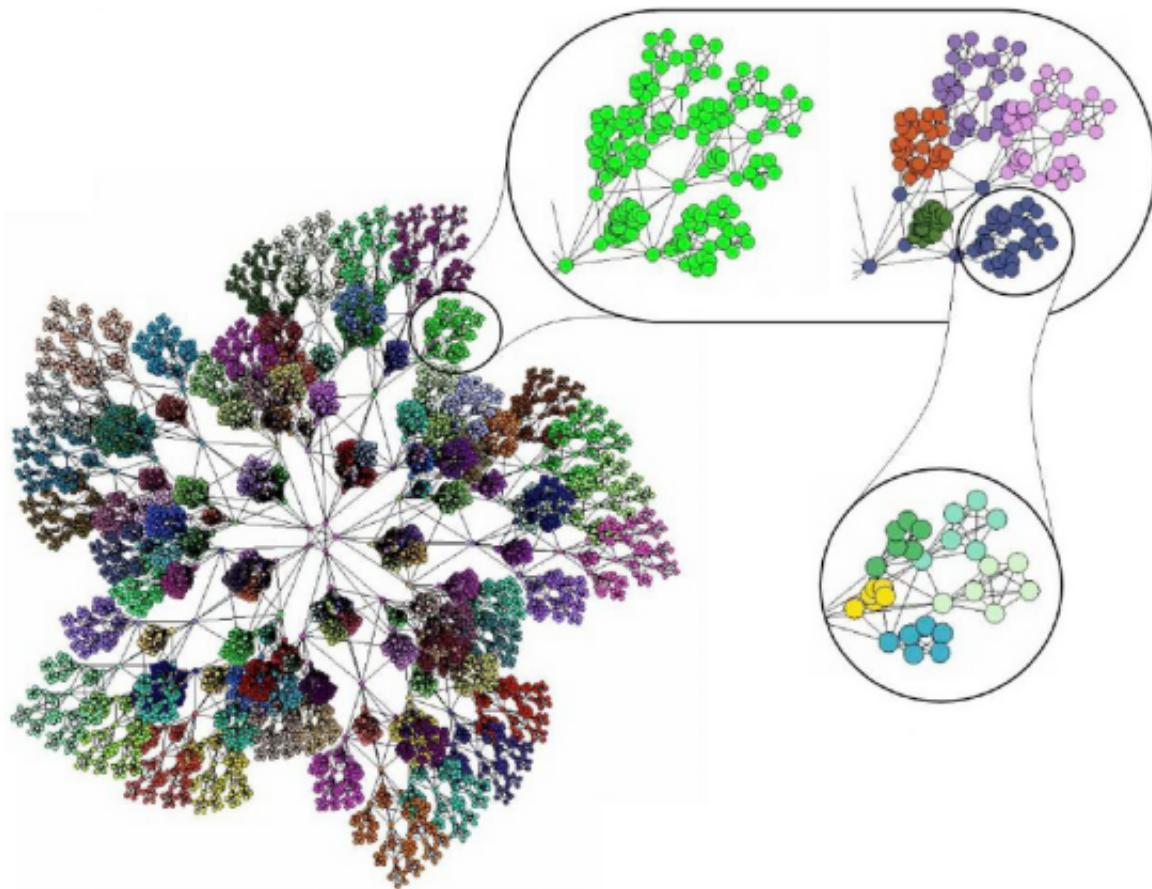
# Giới thiệu

- Tìm cộng đồng trên mạng xã hội

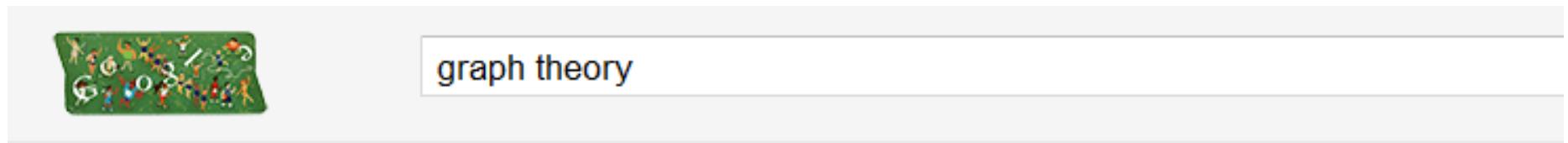


# Giới thiệu

- Kích thước có thể rất lớn



# Giới thiệu



graph theory

Search About 8,640,000 results (0.15 seconds)

Web [Graph theory - Wikipedia, the free encyclopedia](#)  
[en.wikipedia.org/wiki/Graph\\_theory](https://en.wikipedia.org/wiki/Graph_theory)  
In mathematics and computer science, **graph theory** is the study of graphs, which are mathematical structures used to model pairwise relations between objects ...  
[Glossary of graph theory](#) - [Loop](#) - [Spectral graph theory](#) - [List of graph theory topics](#)

Images

Maps

Videos

News

Shopping

Books

More

[Graph Theory Tutorials](#)  
[www.utm.edu/departments/math/graph/](https://www.utm.edu/departments/math/graph/)  
This is the home page for a series of short interactive tutorials introducing the basic concepts of **graph theory**. There is not a great deal of theory here, we will just ...

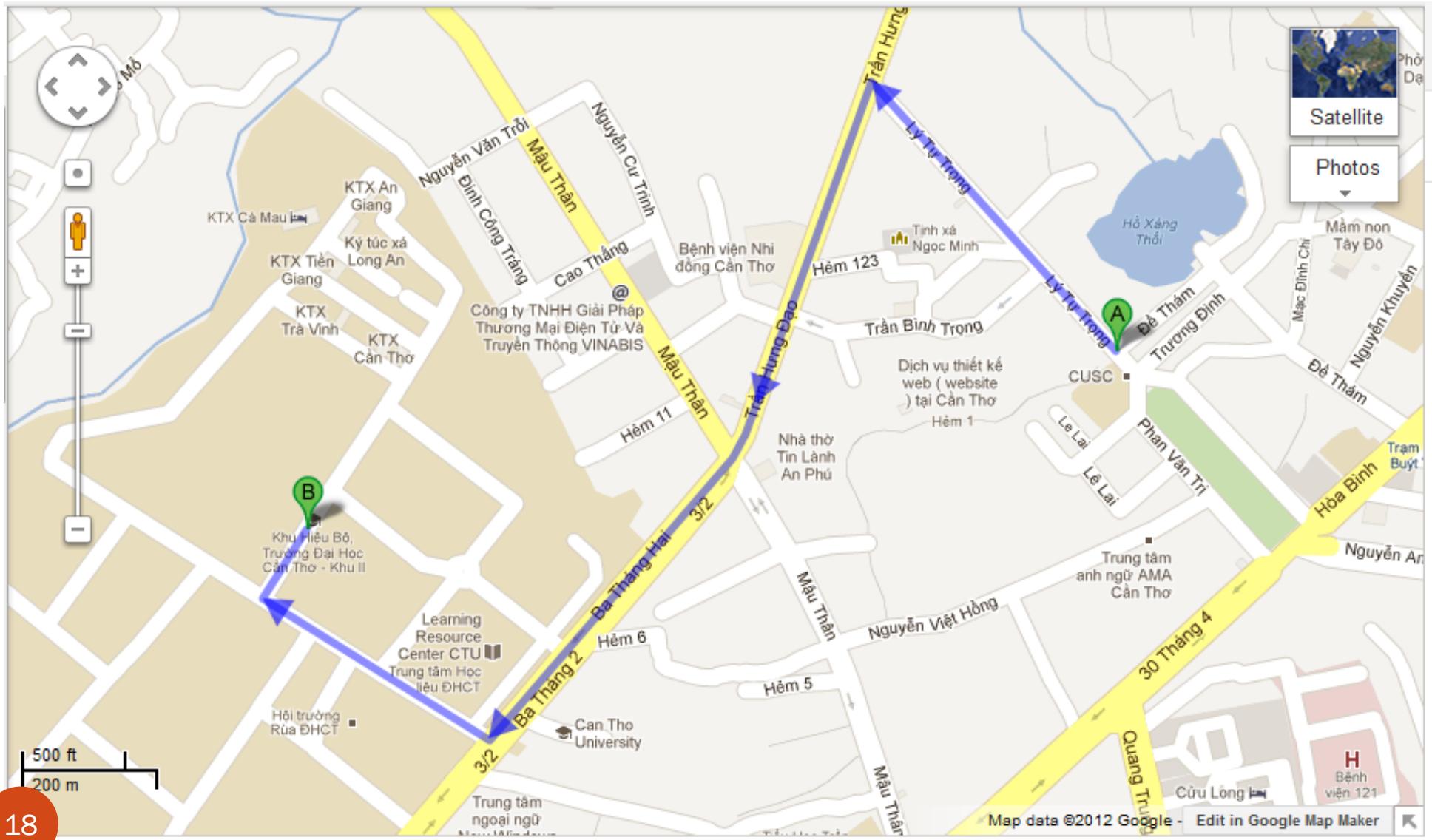
[graph theory -- graph theory textbooks and resources](#)  
[www.graphtheory.com/](http://www.graphtheory.com/)  
The website [www.graphtheory.com](http://www.graphtheory.com/) is sponsored by the mathematical textbooks of Professor Jonathan Gross of Columbia University. It provides comprehensive ...

16 Show search tools

# Giới thiệu



# Giới thiệu

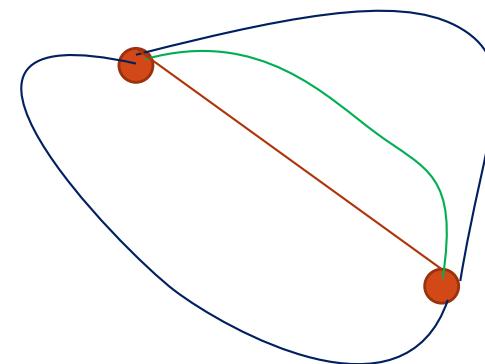
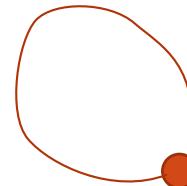


# Định nghĩa – Đồ thị

- **Đồ thị** (Graph)  $G$  là một bộ đôi  $\langle V, E \rangle$ , trong đó:
  - $V$ : **tập các đỉnh** (vertex set), và
  - $E$ : **tập các cung** (edge set), mỗi cung nối 2 đỉnh trong  $V$
- Cho cung  $e$  nối 2 đỉnh  $x$  và  $y$ 
  - Kí hiệu  $e = (x, y)$
  - $x, y$  được gọi là **đầu mút** (endpoints) của cung  $e$
  - $x$  và  $y$  được gọi là **kề** nhau (adjacent) hoặc **lân cận** của nhau (neighbours)
  - $e$  được gọi là **liên thuộc** (incident) với  $x$  và  $y$

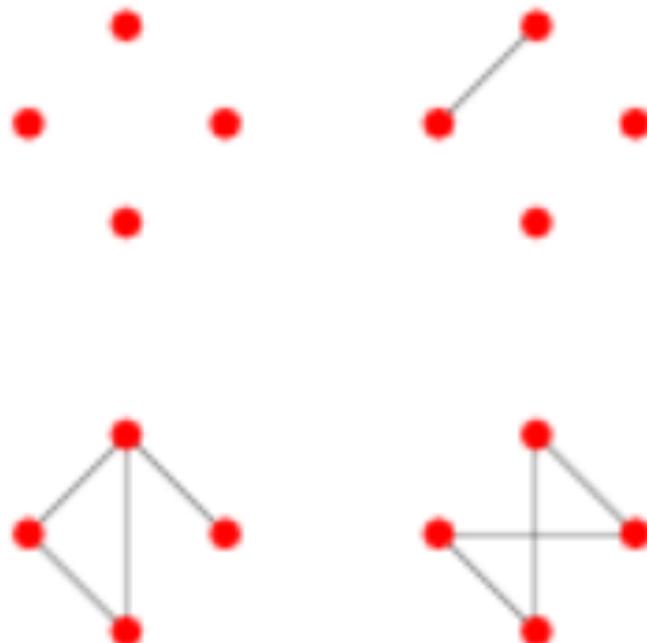
# Định nghĩa – Khuyên + Đa cung

- **Khuyên (loop):** cung có hai đầu mút trùng nhau
- **Đa cung (multi edges):** các cung có cùng chung đầu mút



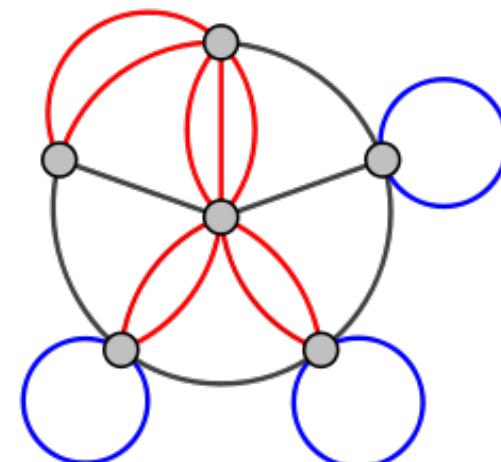
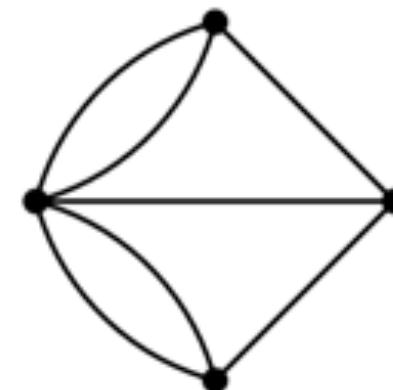
# Định nghĩa – Đơn đồ thị

- Đơn đồ thị vô hướng  
(simple graph):
  - Cung không có hướng:  $(x, y) \equiv (y, x)$
  - Không chứa khuyên
  - Không chứa đa cung



# Định nghĩa – Đa đồ thị + Giả đồ thị

- **Đa đồ thị (Multigraph)**
  - Cung không có hướng
  - Không chứa khuyên
- **Giả đồ thị (Pseudograph)**
  - Cung không có hướng
  - Có thể chứa đa cung và khuyên



# Bậc của đồ thị

- Bậc (degree) của đỉnh
  - Kí hiệu  $\deg(x)$  = số cung liên thuộc với đỉnh  $x$
  - Đỉnh có bậc = 0 gọi là đỉnh cô lập
  - Đỉnh có bậc = 1 gọi là đỉnh treo
- Định lý 1 (định lý bắt tay):
  - Tổng bậc của tất cả các đỉnh trong một đồ thị = 2 lần số cung

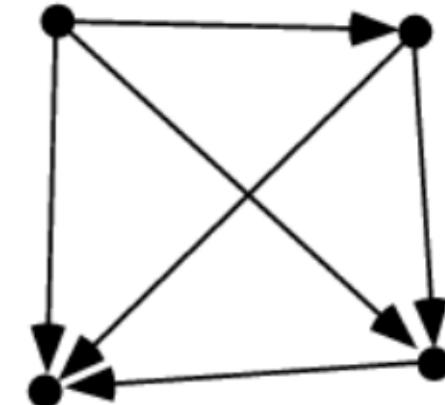
$$\sum_{v \in V} \deg(v) = 2|E| = 2e$$

# Bậc của đồ thị

- Định lý 2:
  - Số đỉnh bậc lẻ của một đồ thị vô hướng là số chẵn.
  - C/M: xem như bài tập
- Bài tập:
  - Cho đồ thị vô hướng  $G = \langle V, E \rangle$  có  $|E| = |V| - 1$ . G không có đỉnh cô lập.
    - a) Chứng minh rằng, G có ít nhất 1 đỉnh bậc 1.
    - b) Chứng minh rằng, G có ít nhất 2 đỉnh bậc 1.

# Định nghĩa – Đồ thị có hướng

- Đơn đồ thị có hướng
  - Cung có hướng  $(x, y) \neq (y, x)$
  - Không chứa chứa đa cung
    - Với mỗi cặp đỉnh  $x, y$  chỉ có thể tồn tại nhiều nhất 1 cung dạng  $(x, y)$ .
- Đa đồ thị có hướng (không khuyên)
  - Có thể chứa nhiều cung dạng  $(x, y)$
  - Không chứa khuyên
- Đa đồ thị có hướng (có khuyên)/quiver
  - Có thể chứa nhiều cung dạng  $(x, y)$
  - Có thể chứa khuyên

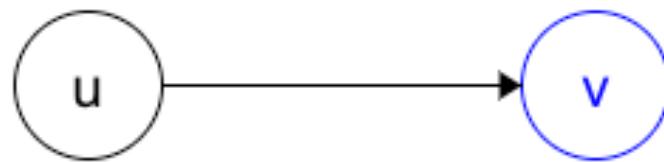


Tài liệu của Rosen  
không phân biệt 2 loại  
này.

Cả 2 đều được gọi là  
**directed multigraph**.

# Định nghĩa – Đồ thị có hướng

- Cho cung  $(u, v)$ 
  - $u$  kề tới  $v$  ( $u$  is adjacent to  $v$ )
  - $v$  kề từ  $u$  ( $v$  is adjacent from  $u$ )



- Trong lớp học này, để đơn giản và nhất quán trong việc cài đặt, ta quy ước:
  - **u kề với v** **(có cung đi từ u đến v)**
  - **v là đỉnh kề của đỉnh u**

# Bậc của đồ thị có hướng

- Bậc vào của đỉnh (in-degree)
  - Kí hiệu:  $\deg^-(v)$  = số cung đi đến  $v$
- Bậc ra của đỉnh (out-degree)
  - Kí hiệu:  $\deg^+(v)$  = số cung đi ra từ  $v$
- Tính chất:
  - $\deg(v) = \deg^-(v) + \deg^+(v)$
- Định lý 3:
  - Cho  $G = \langle V, E \rangle$  làm một đồ thị có hướng

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

# Bài tập

- Cho đồ thị vô hướng  $G = \langle V, E \rangle$   $n$  đỉnh  $m$  cung. Gọi  $\deg(v)$ ,  $v \in V$  là bậc của đỉnh  $v$ . Chứng minh rằng :

a.  $\min\{\deg(v)\}_{v \in V} \leq \left\lfloor \frac{2m}{n} \right\rfloor$

b.  $\max\{\deg(v)\}_{v \in V} \geq \left\lceil \frac{2m}{n} \right\rceil$

c. Nếu  $G$  là đơn đồ thị thì  $m \leq \frac{n(n-1)}{2}$

# Bài tập

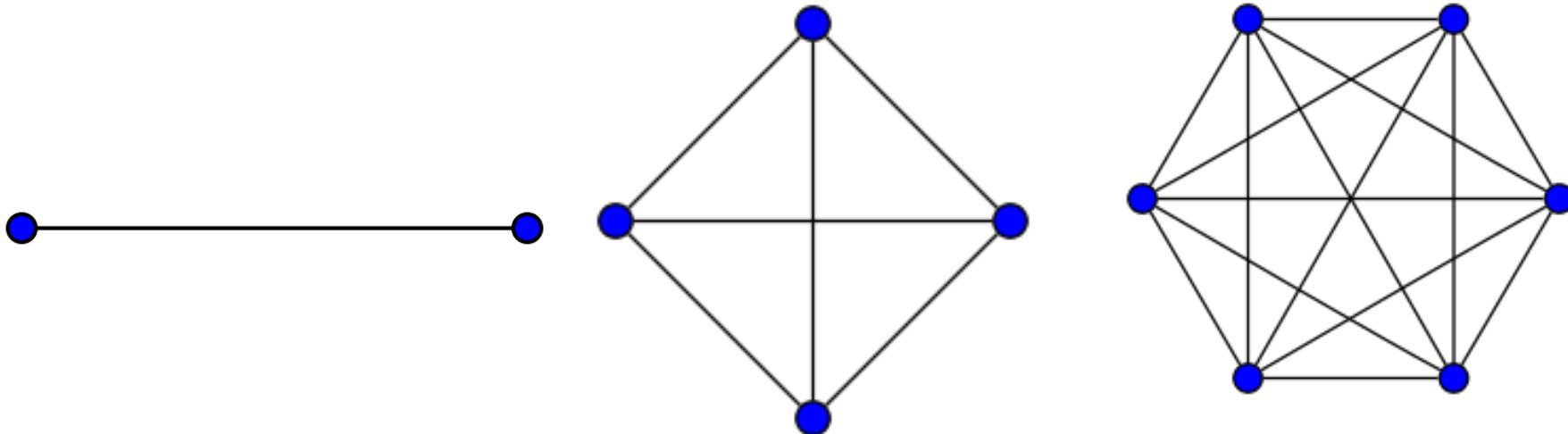
- Cho đơn đồ thị vô hướng  $G = \langle V, E \rangle$   $n$  đỉnh ( $n > 1$ ). CMR:
  - a. Bậc của một đỉnh luôn nhỏ hơn  $n$ .
  - b. Không tồn tại đồng thời một đỉnh có bậc 0 và một đỉnh có bậc  $n - 1$ .
  - c. Có ít nhất hai đỉnh có bậc bằng nhau

# Clique

- Clique trên đồ thị vô hướng là tập các đỉnh mà chúng đồi một kề nhau.
  - Clique tối đại (maximal clique): là một clique không thể thêm vào nó bất cứ đỉnh nào nữa.
  - Clique lớn nhất (maximum clique): là clique có nhiều phần tử nhất trong đồ thị.
- 
- Tập độc lập/tập ổn định (independent set/stable set): tập các đỉnh mà chúng không kề nhau
  - Tập độc lập tối đại (maximal independent set): tập độc lập mà nếu thêm vào một đỉnh sẽ làm mất tính độc lập của chúng
  - Tập độc lập lớn nhất (maximum independent set) là tập độc lập có số đỉnh nhiều nhất

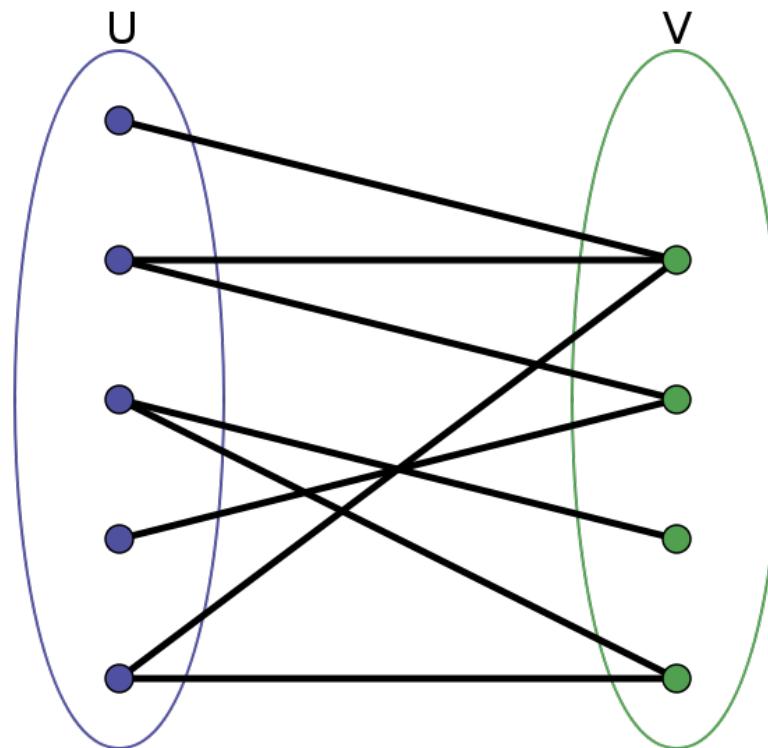
# Đồ thị đặc biệt

- Đồ thị đầy đủ (complete graph), Ký hiệu:  $K_n$ 
  - Đơn đồ thị vô hướng
  - Mọi cặp đỉnh đều có đúng 1 cung



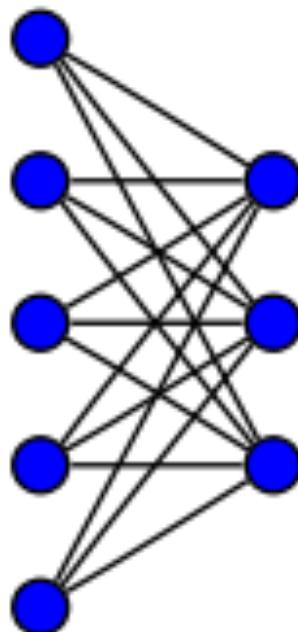
# Đồ thị đặc biệt

- Đồ thị phân đôi  
(bipartite graph/bigraph), Ký hiệu:  $K_{n,m}$ 
  - Tập đỉnh được chia thành hai tập không giao nhau:  $U$  và  $V$
  - Mỗi cung nối 1 đỉnh trong  $U$  và 1 đỉnh trong  $V$



# Đồ thị đặc biệt

- Đồ thị phân đôi đầy đủ
  - Còn gọi là đồ thị 2 clique (biclique)
  - Mỗi đỉnh của phần này nối với tất cả các đỉnh của phần kia



# Đồ thị đặc biệt

- Đồ thị đều bậc k (k-regular graph)
  - Là đồ thị mà tất cả các đỉnh của nó đều có bậc k.

# Đồ thị vô hướng nền

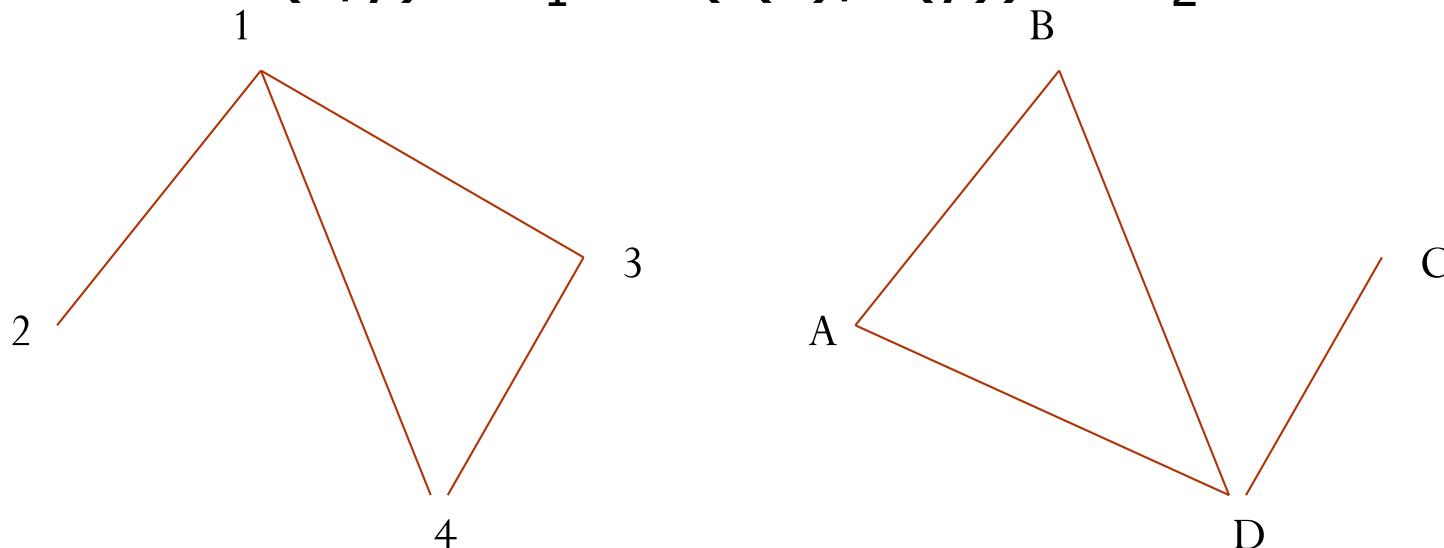
- **Đồ thị vô hướng nền** của một đồ thị có hướng là đồ thị vô hướng có được sau khi đã loại bỏ hướng của các cạnh.
- Đồ thị có hướng và đồ thị vô hướng nền của nó có cùng số cạnh

# Đồ thị con

- Cho đồ thị  $G = \langle V, E \rangle$ 
  - Đồ thị con  $G_s = \langle V_s, E_s \rangle$ ,  $E_s \subset E$  được xây dựng từ  $G$  là đồ thị có được sau khi loại bỏ các cung không thuộc  $E_s$ .
  - Đồ thị con  $G_s = \langle V_s, E_s \rangle$ ,  $V_s \subset V$ ,  $E_s \subset E$  được xây dựng từ  $G$  là đồ thị có được sau khi loại bỏ các đỉnh không nằm trong  $V_s$  và các cung liên thuộc với nó, cùng với các cung không nằm trong  $E_s$ .

# Sự đẳng cấu của đồ thị

- 2 đồ thị  $G_1 = \langle V_1, E_1 \rangle$  và  $G_2 = \langle V_2, E_2 \rangle$  được gọi là đẳng cấu nếu và chỉ nếu tồn tại song ánh:
  - $f: V_1 \rightarrow V_2$
  - $v_2 = f(v_1) \in V_2$
  - sao cho  $(x,y) \in E_1 \Leftrightarrow (f(x), f(y)) \in E_2$

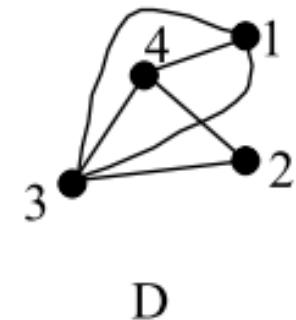
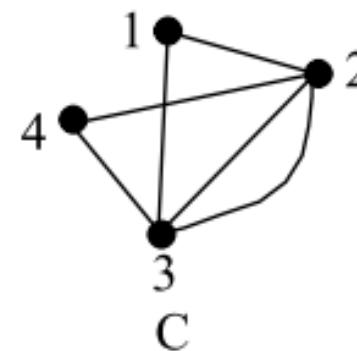
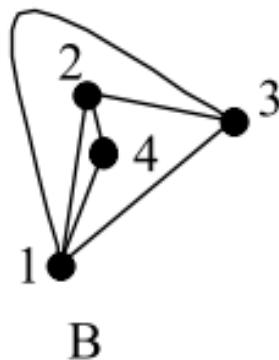
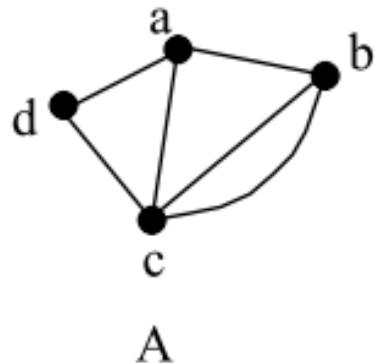


# Bài tập

- Vẽ các đồ thị sau
  - a)  $V = \{u, v, w, x\}$ ,  $E = \{uv, vw, wx, vx\}$
  - b)  $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ ,  $E = \{12, 22, 23, 34, 35, 67, 68, 78\}$
  - c)  $V = \{n, p, q, r, s, t\}$ ,  $E = \{np, nq, nt, rs, rt, st, pq\}$
- Với mỗi đồ thị xét xem nó có phải là đơn đồ thị không? Tại sao?

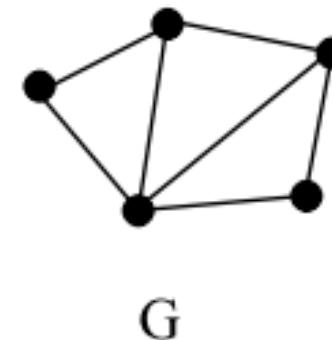
# Bài tập

- Trong các đồ thị B, C, D (các) đồ thị nào đẳng cấu với đồ thị A. Nếu đẳng cấu tìm song ánh tương ứng

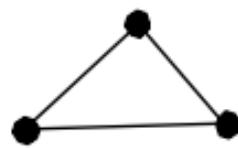


# Bài tập

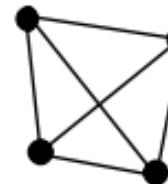
- Cho đồ thị  $G$  như hình vẽ



- Trong các đồ thị dưới đây, đồ thị nào là đồ thị con của đồ thị  $G$ .



P



Q



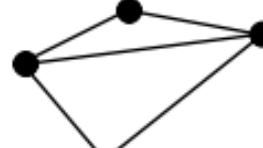
R



S



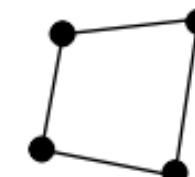
T



U



V



W

# Bài tập

- Cho  $G$  là đồ thị vô hướng có  $n$  đỉnh  $m$  cạnh ( $n \geq 1$ ,  $m \geq 0$ ). Với mọi số tự nhiên  $k \in \mathbb{N}$ , gọi:
  - $n_k$  là số đỉnh có bậc bằng  $k$ ;
  - $K$  là bậc lớn nhất (i.e.  $n_K > 0$  và  $\forall k > K, n_k = 0$ )

- CMR:  $\sum_{k=0}^K kn_k = 2m$  và  $\sum_{k=0}^K n_k = n$
- CMR:  $K \leq 2m$ . Cho 1 ví dụ trong trường hợp đẳng thức xảy ra.
- CMR: nếu  $G$  là đơn đồ thị thì  $K \leq n-1$ ; cho 1 ví dụ trong trường hợp dấu = xảy ra.

# Biểu diễn đồ thị trên máy tính

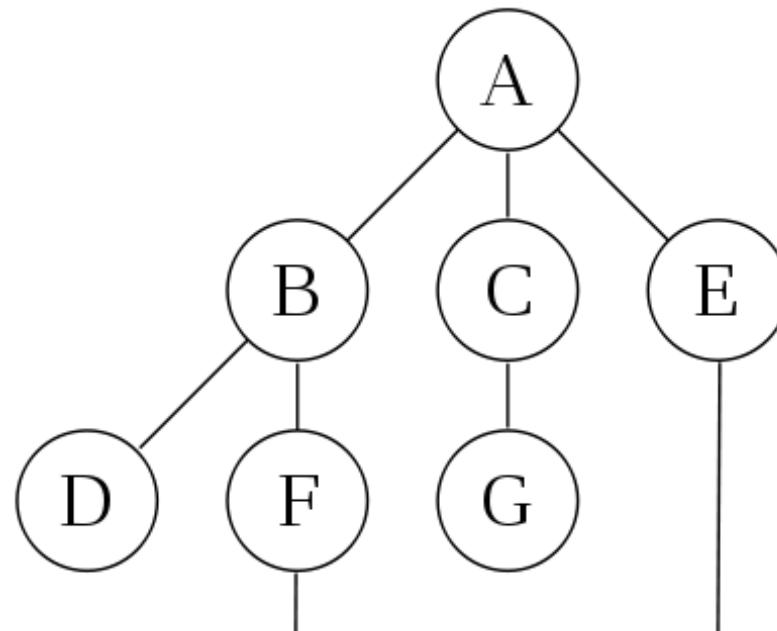
- Cấu trúc dữ liệu đồ thị
  - Danh sách cung
  - Ma trận kề (đỉnh – đỉnh)
  - Danh sách kề (danh sách con/danh sách cha)
  - Ma trận đỉnh cung (ma trận liên thuộc, ít sử dụng)
- Các phép toán trên cấu trúc dữ liệu đồ thị
  - `init_graph(G)`: khởi tạo đồ thị
  - `adjacent(G, x, y)`: kiểm tra x và y có kề nhau không
  - `degree(G, x)`: tính bậc của đỉnh x
  - `neighbors(G, x)`: trả về danh sách các đỉnh kề của x
  - `add_edge(G, e, x, y)`: thêm cung e = (x, y) vào G
  - `remove_edge(G, e, x, y)`: xoá cung e =(x, y) ra khỏi G

# Duyệt đồ thị

- Đi qua tất cả đỉnh của một đồ thị
  - Cập nhật và/hoặc kiểm tra giá trị của chúng trên đường đi
- Ý tưởng:
  - Bắt đầu từ 1 đỉnh bất kỳ đánh dấu nó đã duyệt
  - Duyệt các đỉnh kề của nó
  - Duyệt các đỉnh kề của đỉnh kề của nó
  - ...
- Phương pháp duyệt
  - Duyệt theo chiều rộng
  - Duyệt theo chiều sâu

# Duyệt đồ thị

- Duyệt theo chiều rộng (Breath first search – BFS)
  - Ý tưởng: “bay” từ đỉnh bắt đầu đến đỉnh cần duyệt
    - Đỉnh gần nhất được duyệt trước (ít cung nhất)



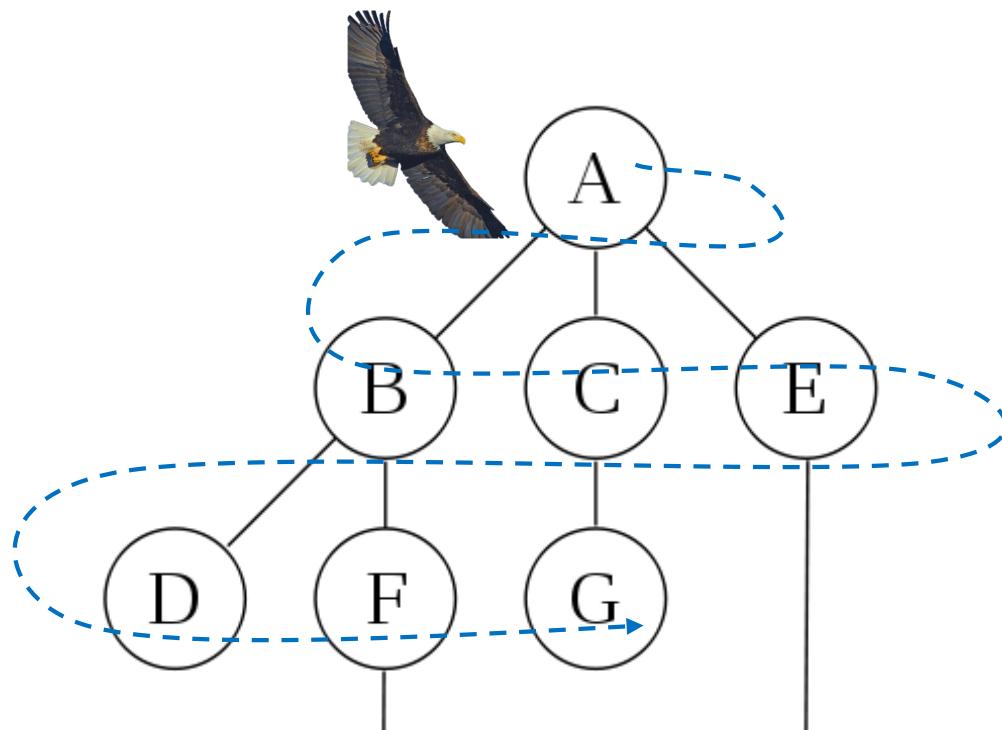
# Duyệt đồ thị

- Duyệt theo chiều rộng (Breath first search – BFS)
  - Ý tưởng: “bay” từ đỉnh bắt đầu đến đỉnh cần duyệt
    - Đỉnh gần nhất được duyệt trước (ít cung nhất)

Ê, đại bàng! Bay theo  
chiều ngang trước,  
sau đó đi xuống.



45



# Duyệt đồ thị

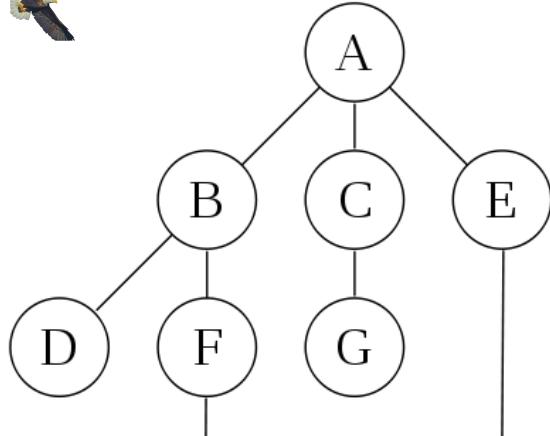
- Duyệt theo chiều rộng (Breath first search - BFS)
  - Sử dụng **hàng đợi**
  - Các đỉnh trong hàng đợi là các đỉnh *đã duyệt* nhưng *chưa xét các đỉnh kề của nó*
- **Thuật toán bfs 1 (Phiên bản gốc):**
  - Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)
  - **Duyệt s (vd: in s ra màn hình)**
  - **Đánh dấu s đã được duyệt** (vd:  $\text{mark}[s] = 1$ )
  - **while hàng đợi** chưa rỗng **do**
    - Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh u
    - **for** các đỉnh kề v của u **do**
      - **if** (v chưa được duyệt)
        - **Duyệt v (vd: in v ra màn hình)**
        - **Đánh dấu v đã duyệt**
        - **Đưa v vào hàng đợi**

# Duyệt đồ thị

- Duyệt theo chiều rộng (Breath first search - BFS)
  - Sử dụng **hàng đợi**
  - Các đỉnh trong hàng đợi là các đỉnh *sẽ được duyệt*
- **Thuật toán bfs 2: (Phiên bản cài đặt)**
  - Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)
  - **while** **hàng đợi** chưa rỗng **do**
    - Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**
    - **if** (**u** đã duyệt) **continue** (bỏ qua)
    - **Duyệt u (vd: in u ra màn hình)**
    - **Đánh dấu u đã duyệt**
    - **for** các đỉnh kề **v** của **u** **do**
      - Đưa **v** vào **hàng đợi**

# Duyệt đồ thị

- Duyệt theo chiều rộng (Breath first search - BFS)
  - Sử dụng **hàng đợi**
  - Các đỉnh trong hàng đợi là các đỉnh *sẽ được duyệt*
- **Thuật toán bfs 3: (Phiên bản bài tập lý thuyết)**
  - Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)
  - **while** **hàng đợi** chưa rỗng **do**
    - Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**
    - **if** (**u** đã duyệt) **continue** (bỏ qua)
    - **Duyệt u (vd: in u ra màn hình)**
    - **Đánh dấu u đã duyệt**
    - **for** các đỉnh kề **v** của **u** **do**
      - **if** (**v** chưa duyệt) (giảm số lần **v** nằm trong HD)
        - Đưa **v** vào **hàng đợi**



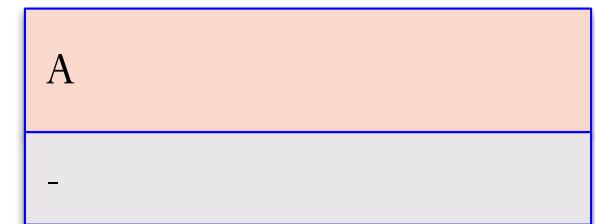
Đưa 1 đỉnh s bất kỳ vào hàng đợi (vd: đỉnh 1)

**while** hàng đợi chưa rỗng **do**

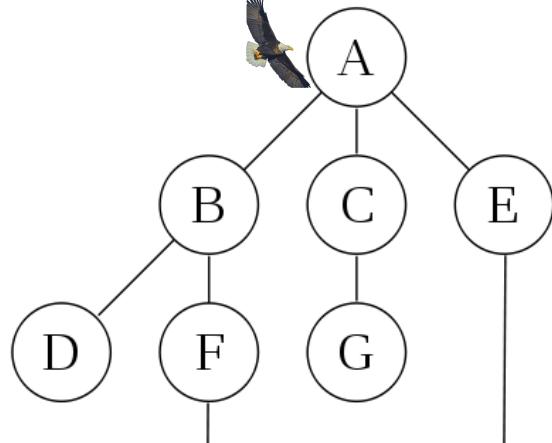
Lấy đỉnh ở đầu hàng đợi ra => gọi là đỉnh u  
Làm gì đó trên u

**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào hàng đợi



Đưa A vào hàng đợi



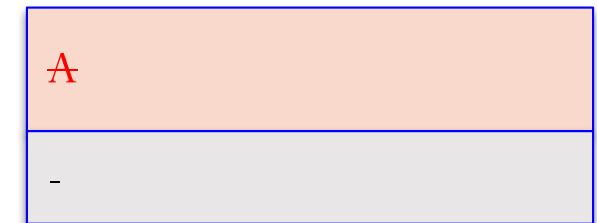
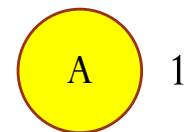
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

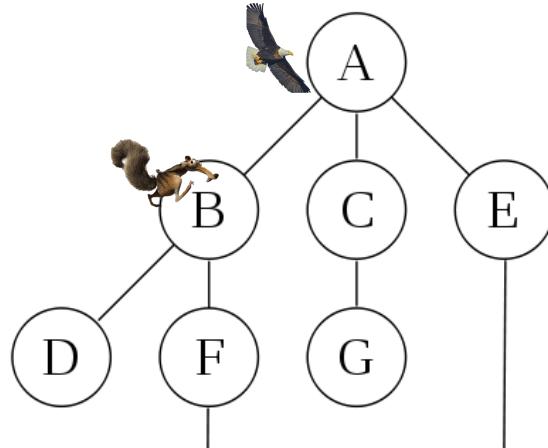
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Lấy A ra khỏi  
hàng đợi



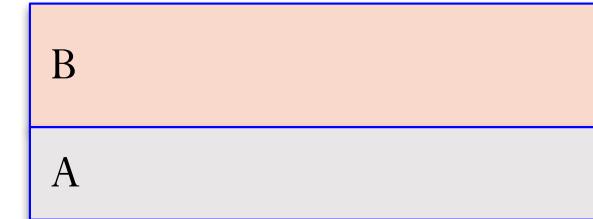
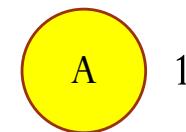
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

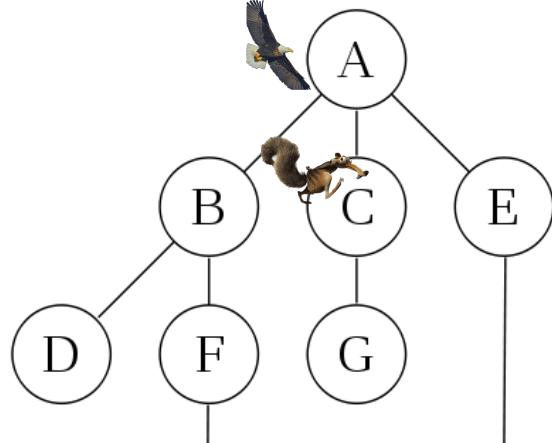
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét B, chưa  
duyệt => thêm  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

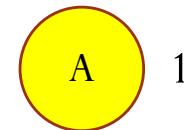
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

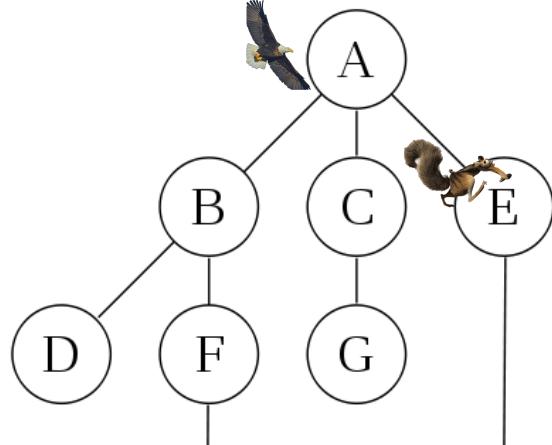
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

B, C

A, A



Xét C, chưa  
duyệt => thêm  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

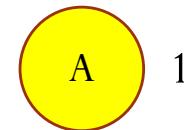
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

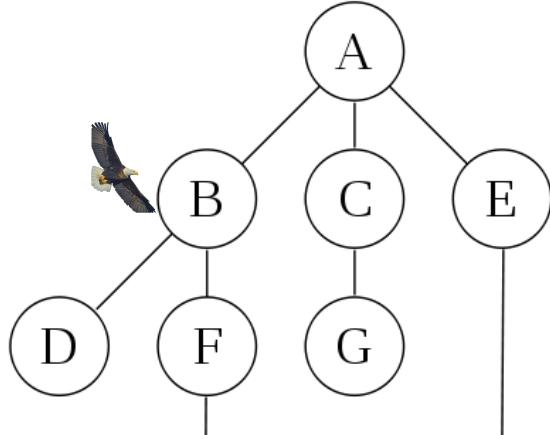
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

B, C, E

A, A, A



Xét E, chưa  
duyệt => thêm  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

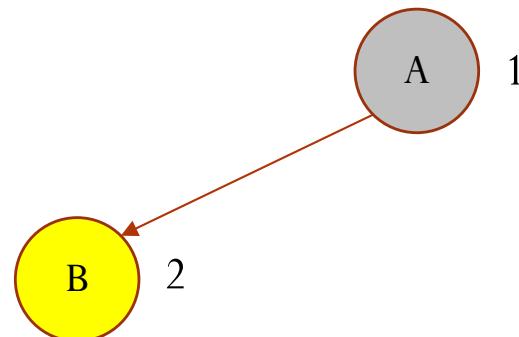
**for** các đỉnh kề **v** của **u** **do**

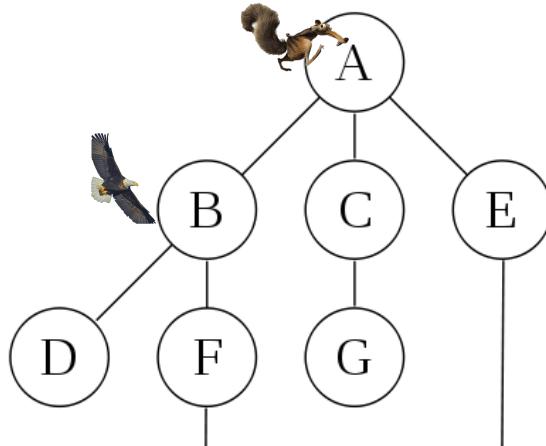
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

B, C, E

A, A, A

Lấy B ra khỏi  
hàng đợi





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

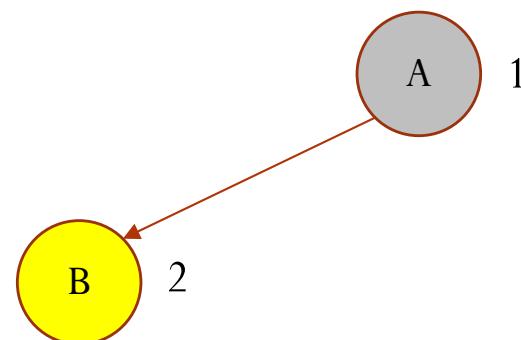
**for** các đỉnh kề **v** của **u** **do**

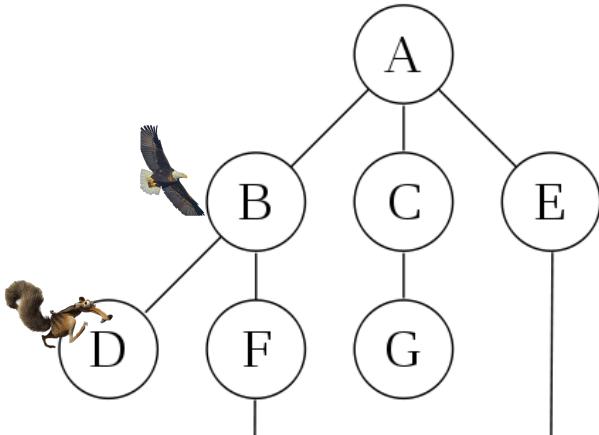
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

C, E

A, A

Xét A, đã duyệt  
=> bỏ qua





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

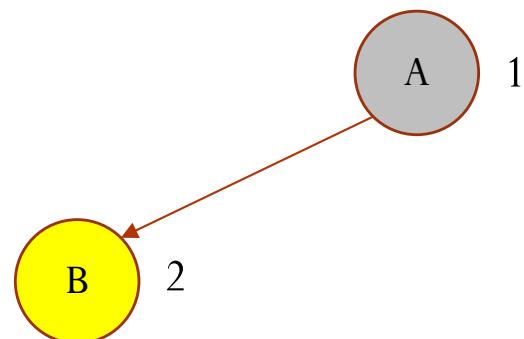
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

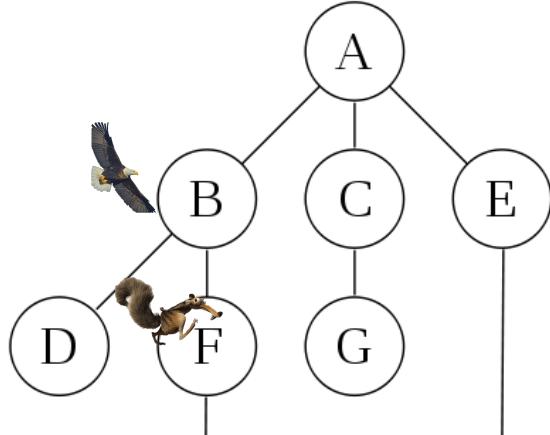
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

C, E, D

A, A, B



Xét D, chưa  
duyệt => đưa  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while hàng đợi** chưa rỗng **do**

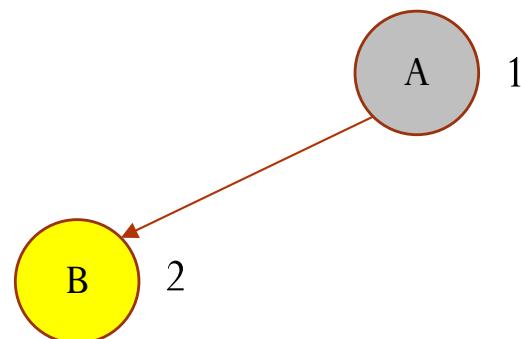
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh **kề v** của **u** **do**

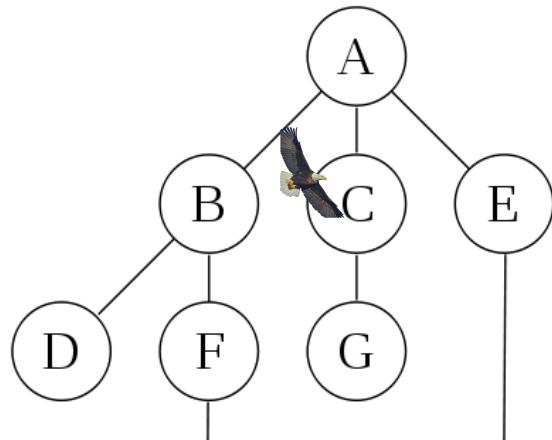
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

C, E, D, F

A, A, B, B



Xét F, chưa  
duyệt => đưa  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

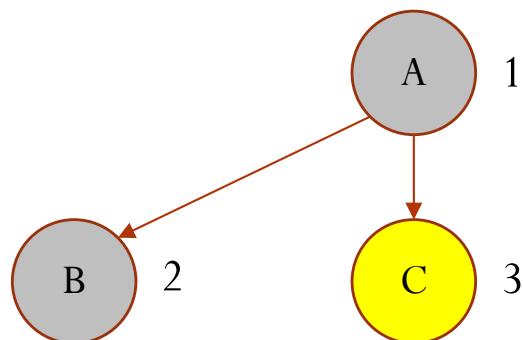
**for** các đỉnh kề **v** của **u** **do**

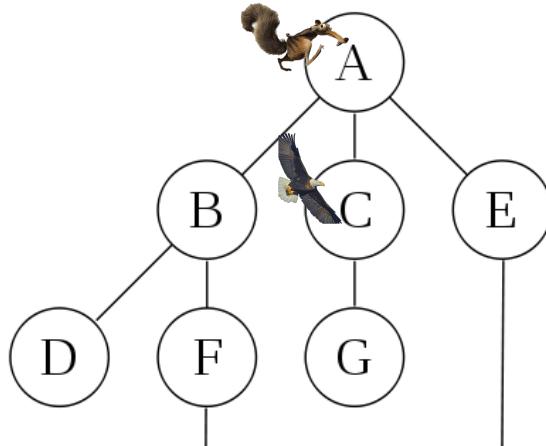
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

**C, E, D, F**

**A, A, B, B**

Lấy C ra khỏi  
hàng đợi





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

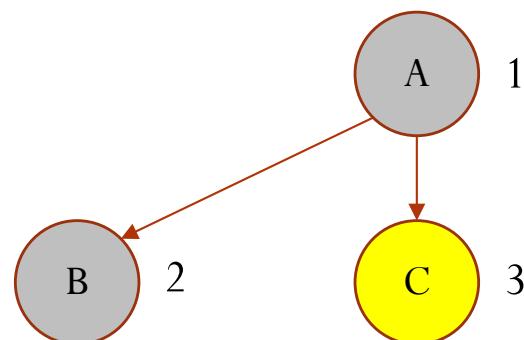
**for** các đỉnh kề **v** của **u** **do**

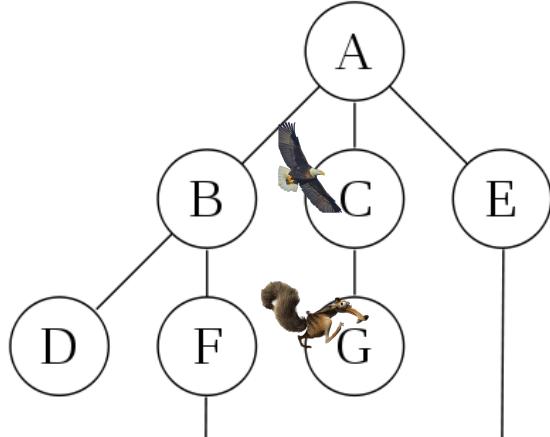
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

E, D, F

A, B, B

Xét A, đã duyệt  
=> bỏ qua





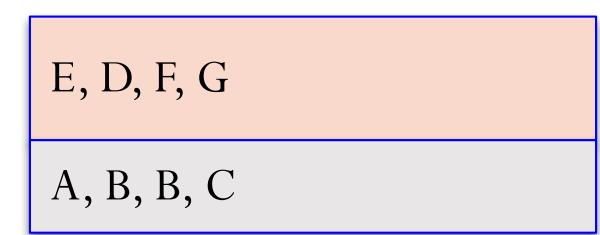
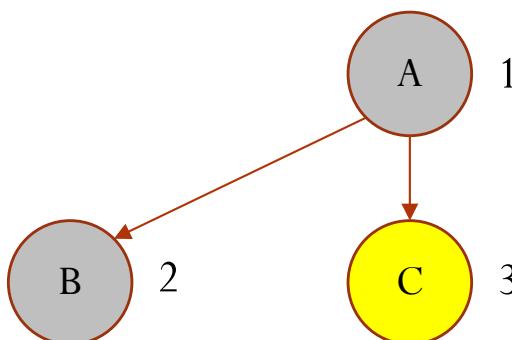
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

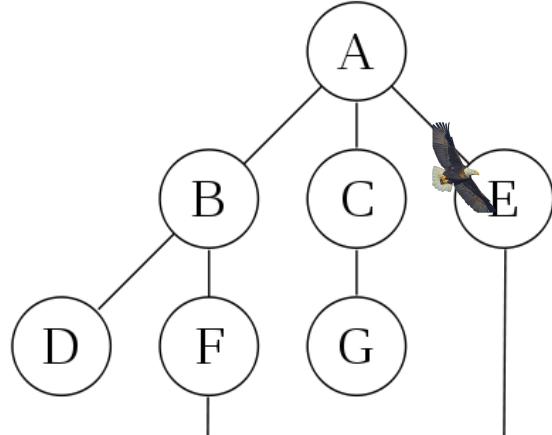
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét G, chưa  
duyệt => đưa  
vào HD



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

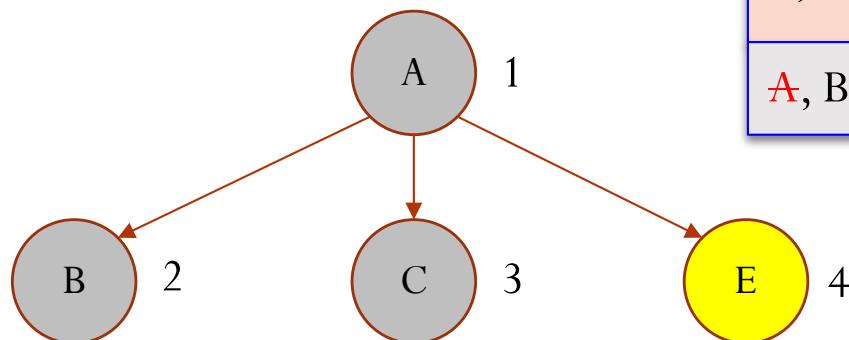
**for** các đỉnh kề **v** của **u** **do**

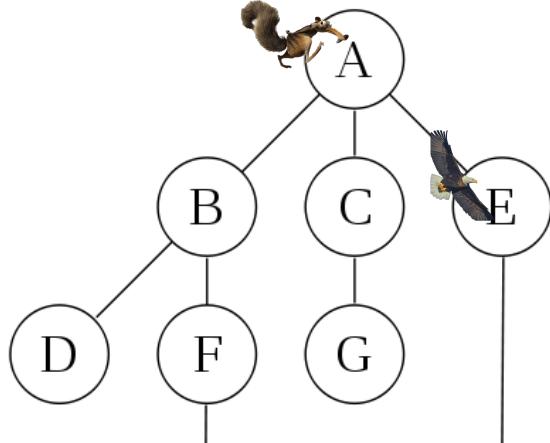
**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**

E, D, F, G

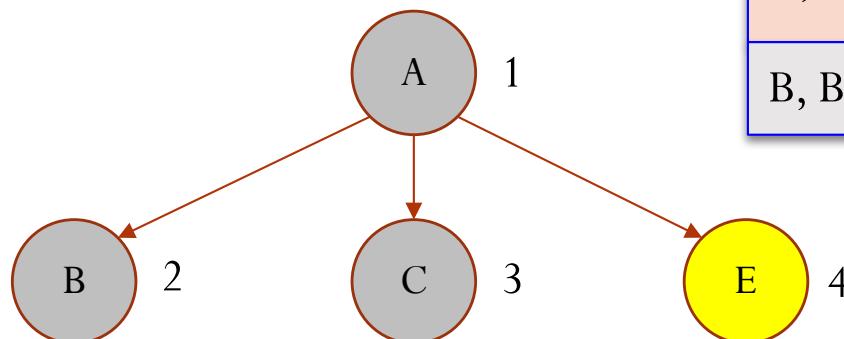
A, B, B, C

Lấy E ra khỏi  
hàng đợi





Xét A, đã duyệt  
=> bỏ qua



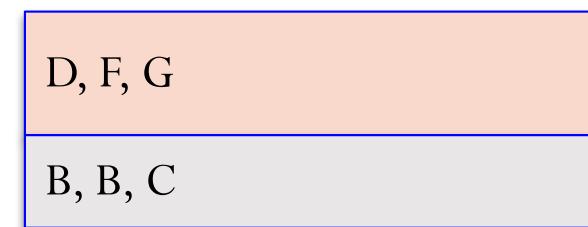
Đưa 1 đỉnh s bất kỳ vào hàng đợi (vd: đỉnh 1)

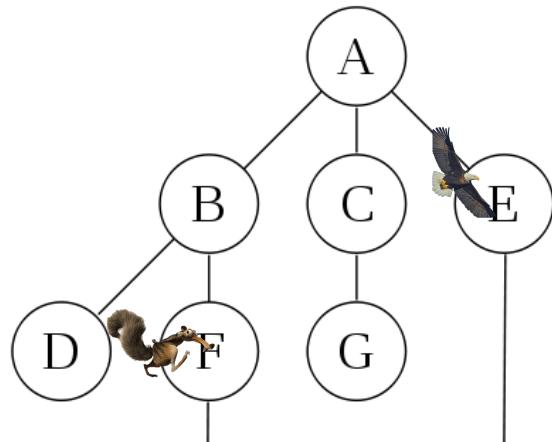
**while** hàng đợi chưa rỗng **do**

Lấy đỉnh ở đầu hàng đợi ra => gọi là đỉnh u  
Làm gì đó trên u

**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào hàng đợi





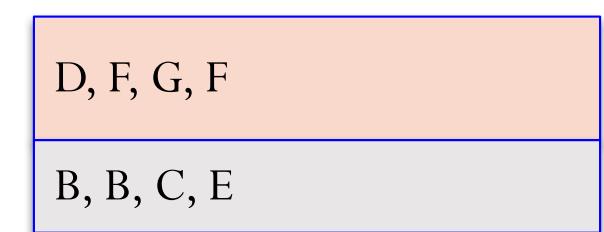
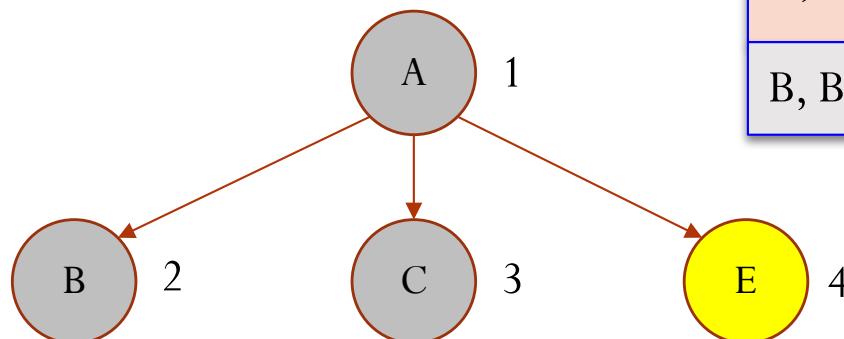
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

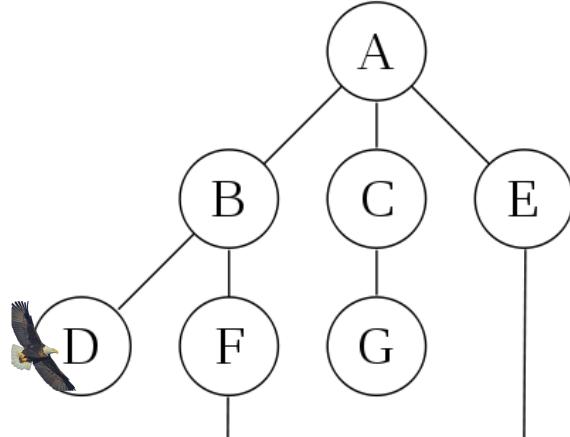
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét F, chưa  
duyệt (mặc dù  
đang nằm trong  
HĐ) => tiếp tục  
đưa vào HĐ



Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

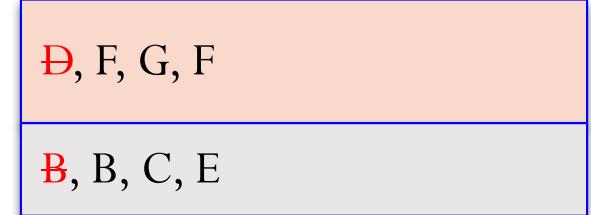
**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**

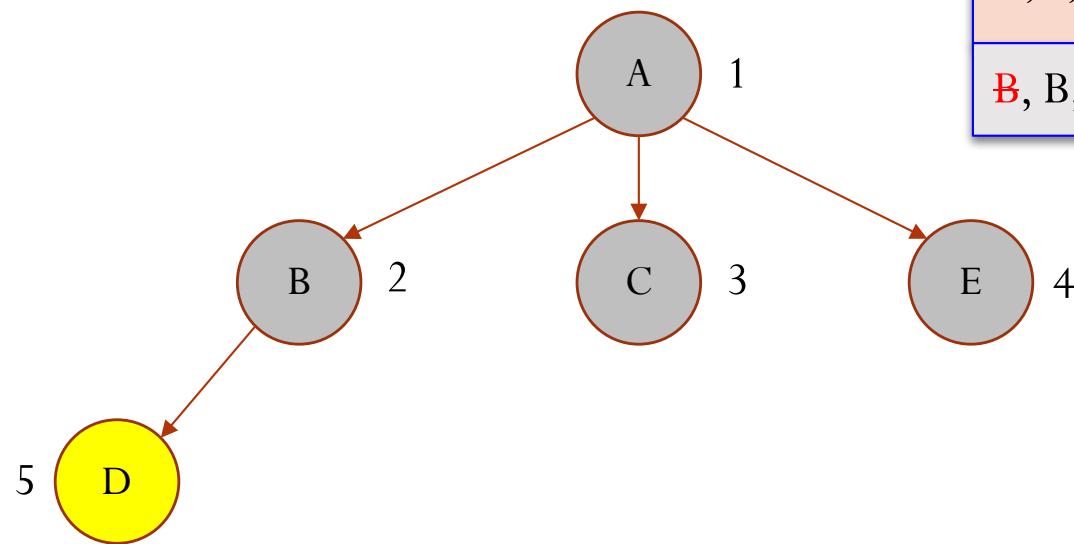
Làm gì đó trên **u**

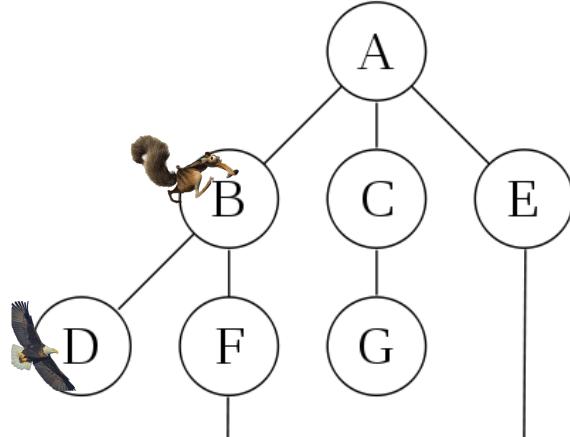
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Lấy D ra khỏi  
hàng đợi





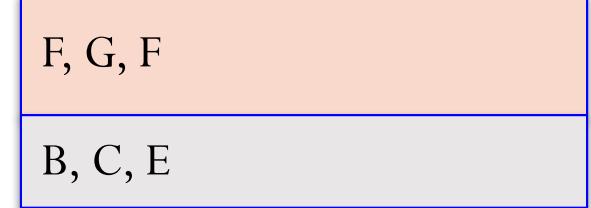
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

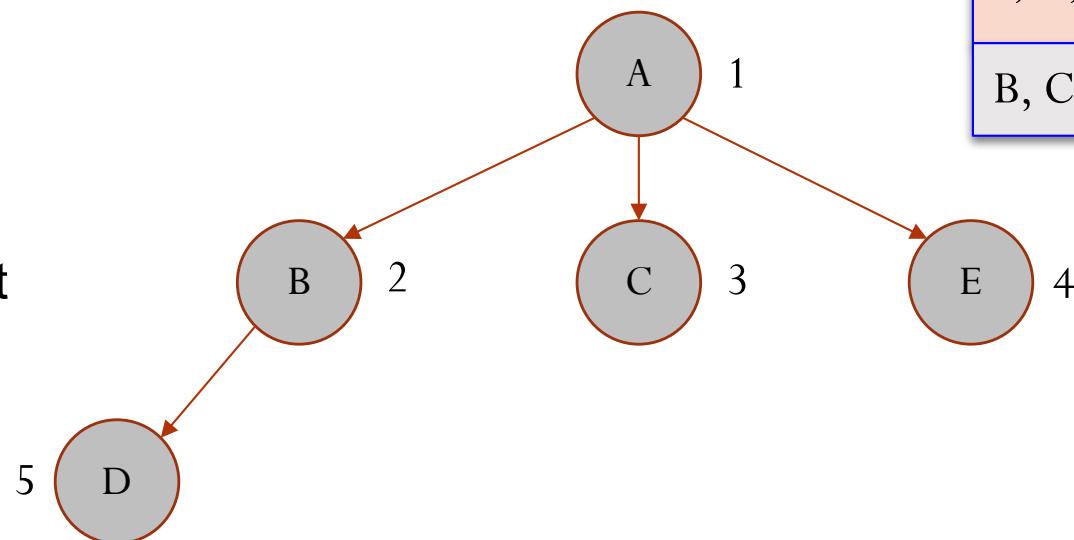
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

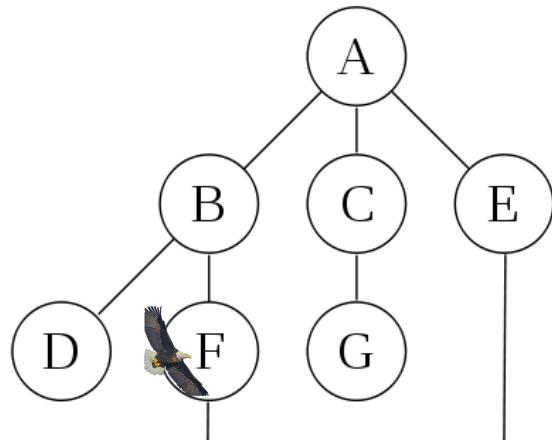
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét B, đã duyệt  
=> bỏ qua





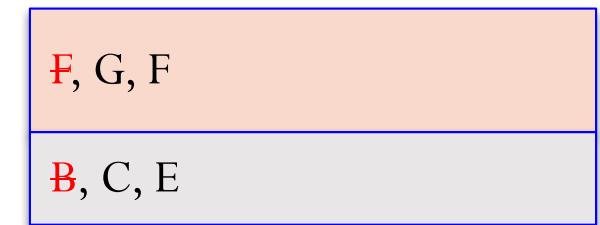
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

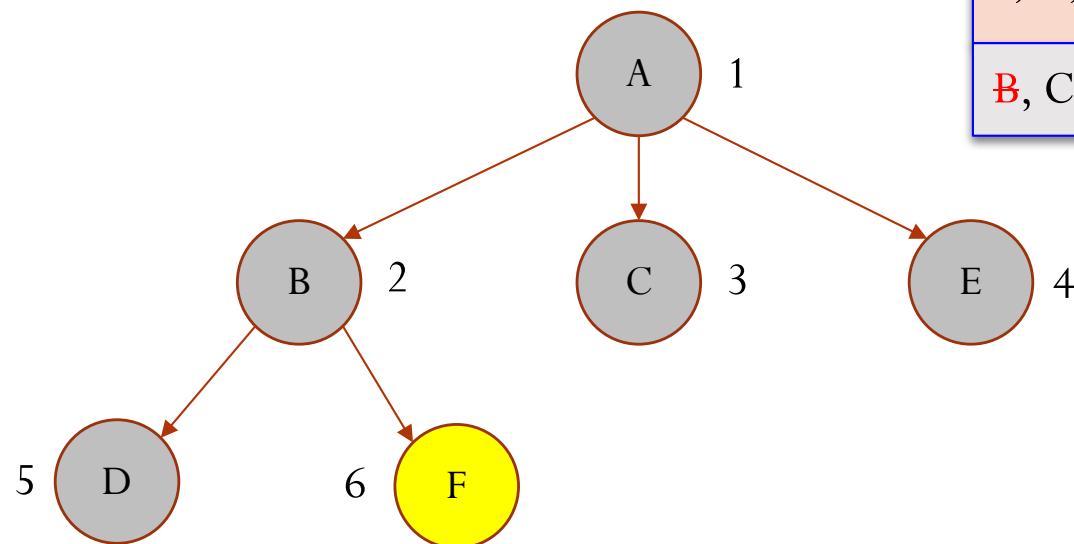
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

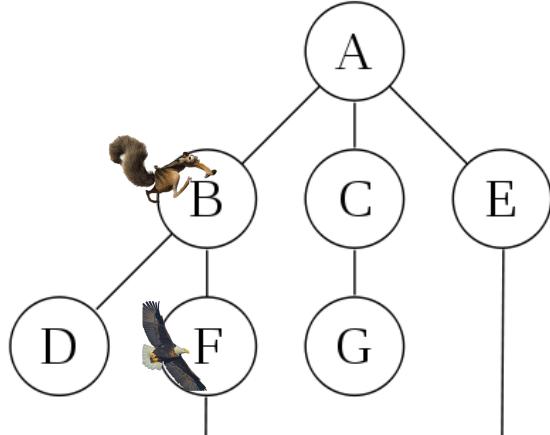
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Lấy **F** ra khỏi  
**hàng đợi**





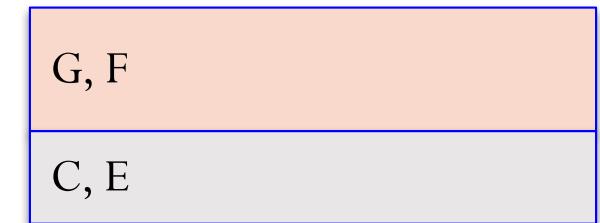
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

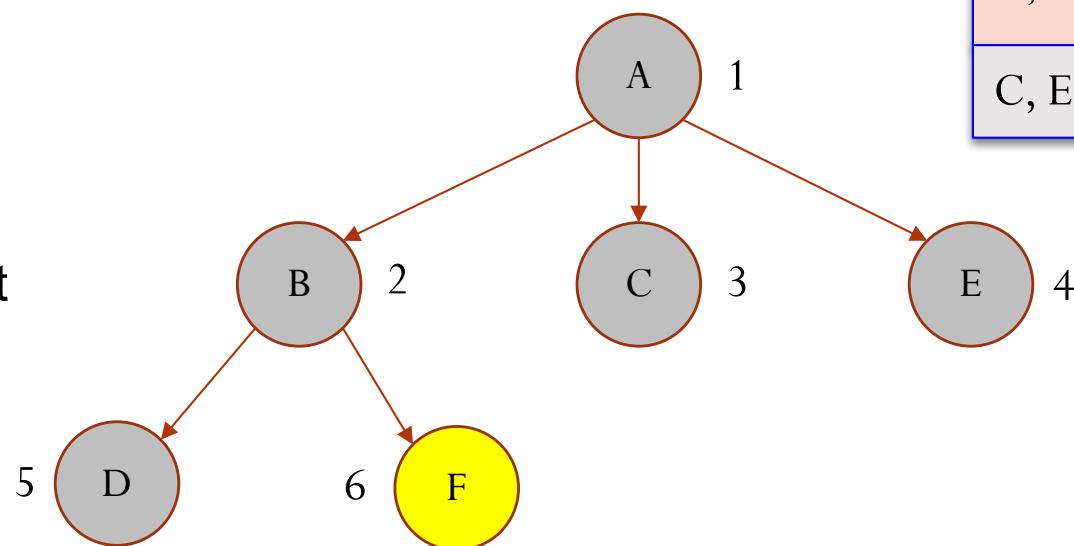
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

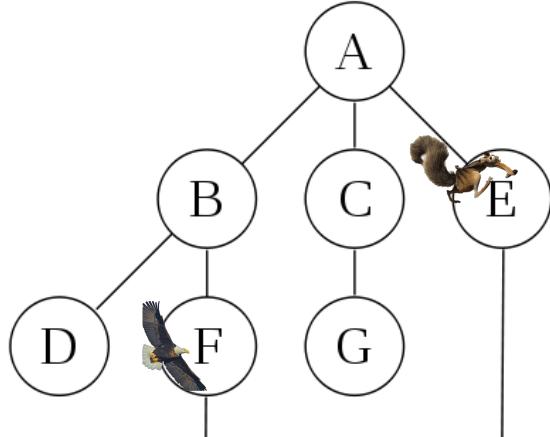
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét B, đã duyệt  
=> bỏ qua





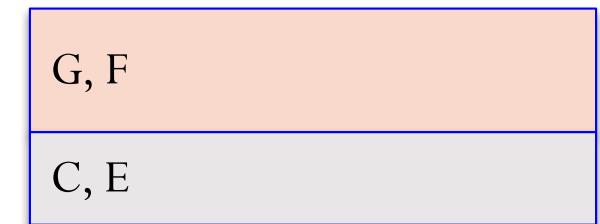
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

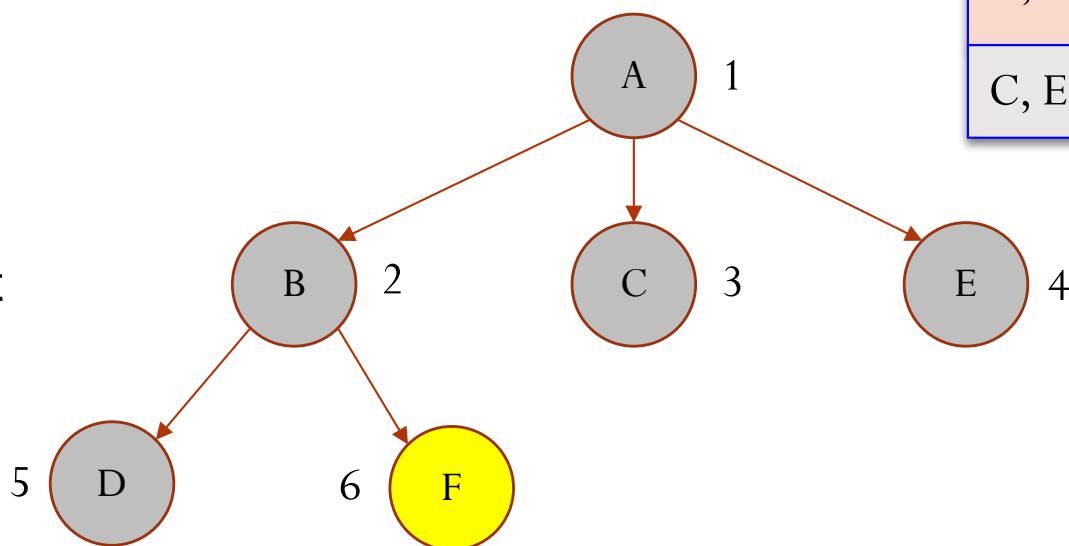
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

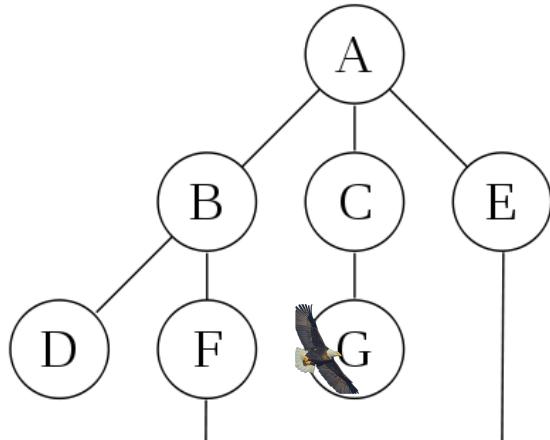
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét **E**, đã duyệt  
=> bỏ qua





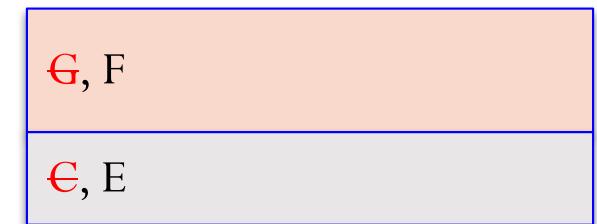
Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

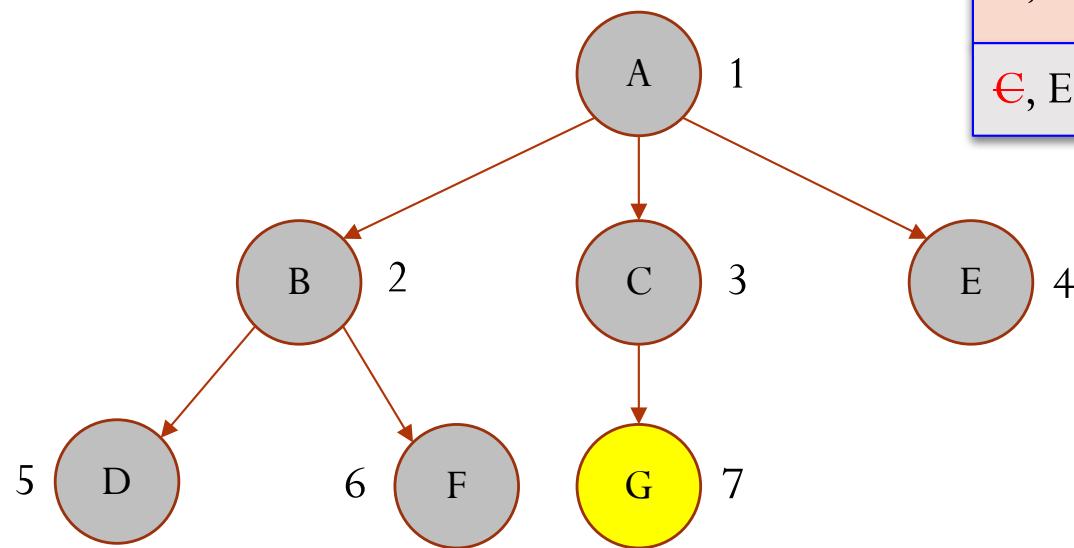
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

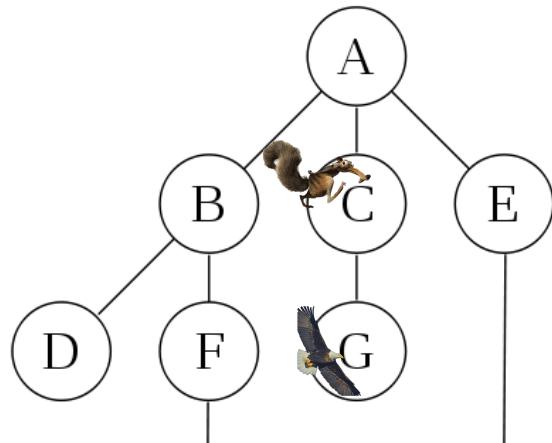
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Lấy **G** ra khỏi  
**hàng đợi**





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

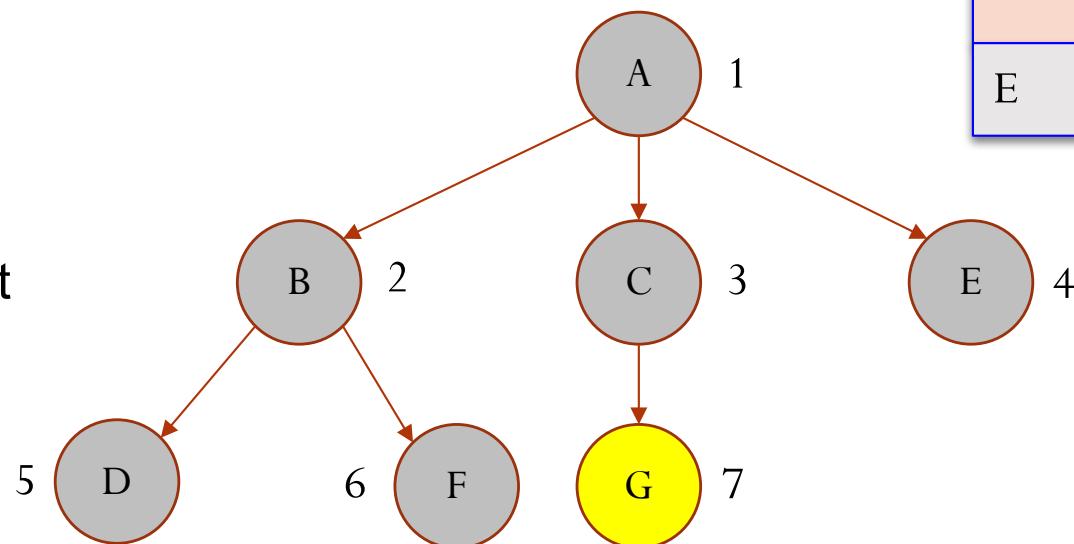
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

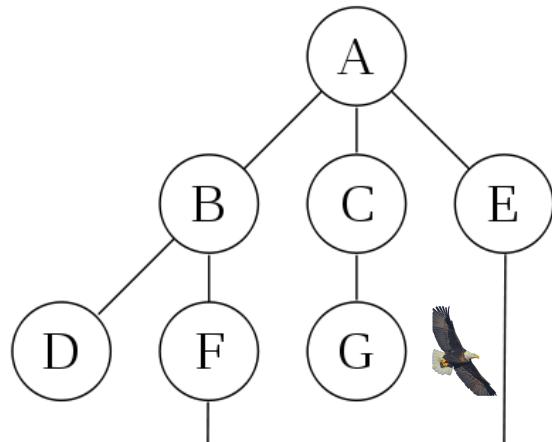
**for** các đỉnh **kề v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xét C, đã duyệt  
=> bỏ qua





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

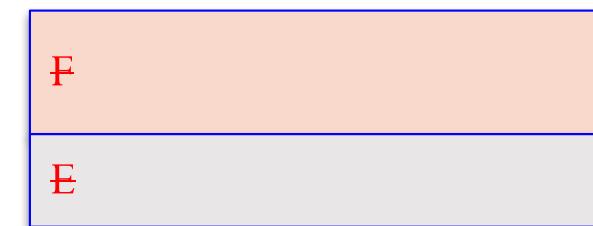
**while** **hàng đợi** chưa rỗng **do**

Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**

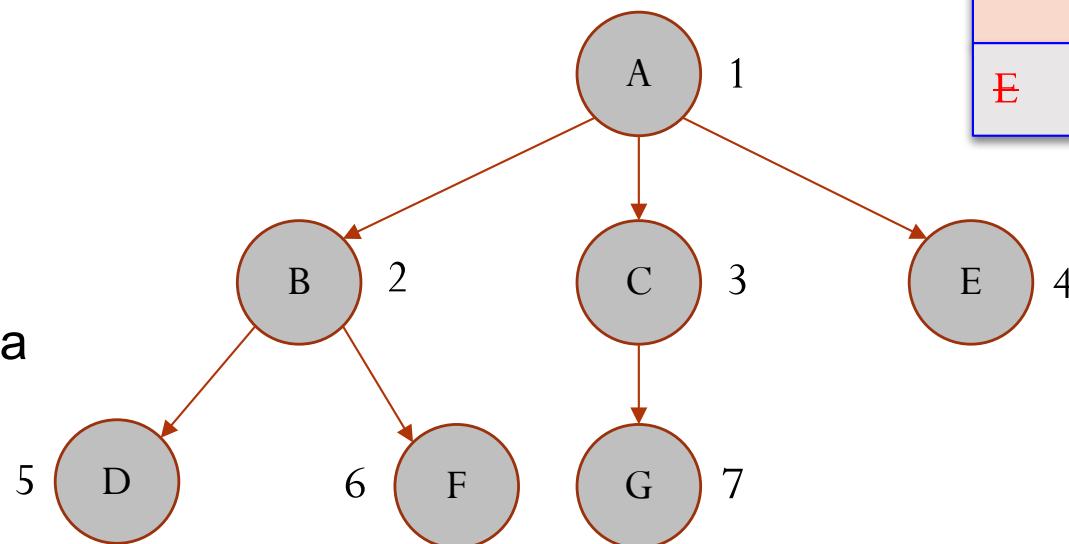
Làm gì đó trên **u**

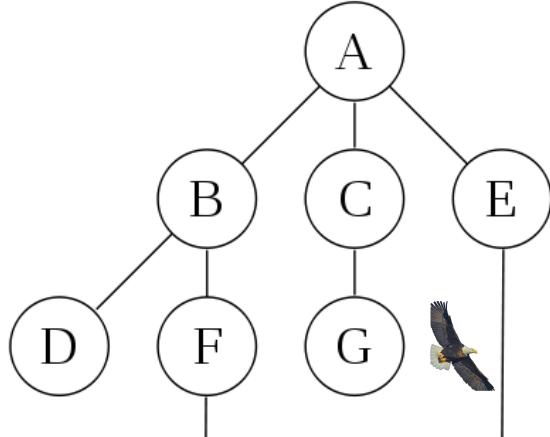
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Lấy **F** ra, **F** đã  
duyệt => bỏ qua





Đưa 1 đỉnh **s** bất kỳ vào **hàng đợi** (vd: đỉnh 1)

**while** **hàng đợi** chưa rỗng **do**

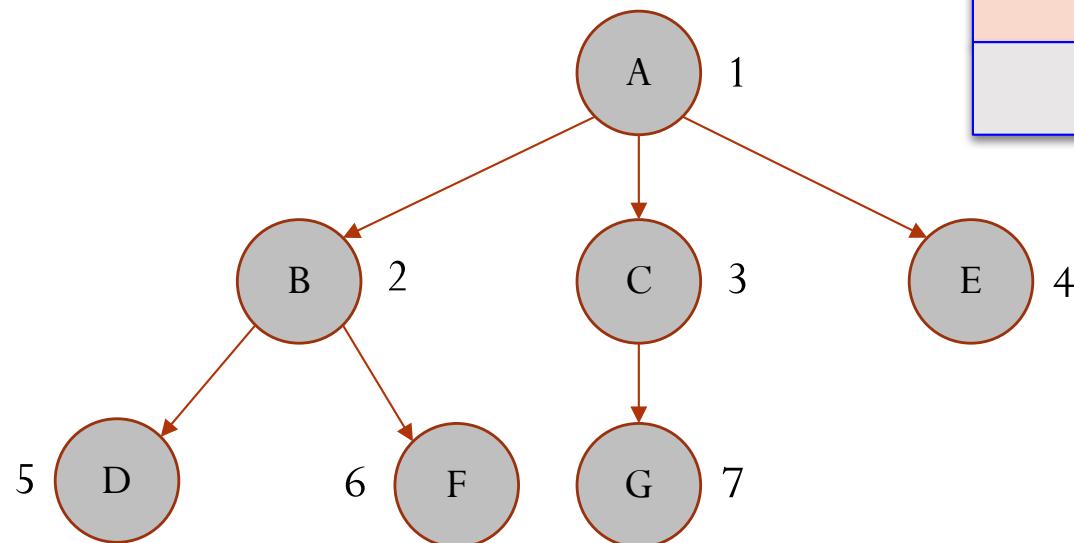
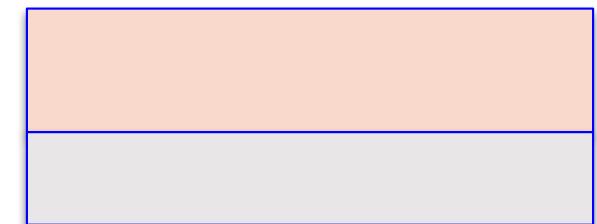
Lấy đỉnh ở đầu **hàng đợi** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **hàng đợi**



Xong!



**Hàng đợi rỗng  
=> Kết thúc**

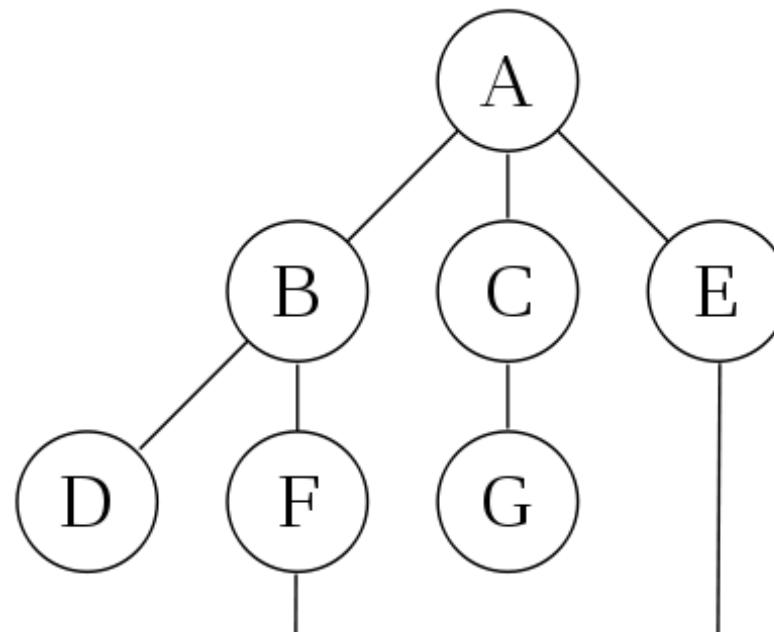
Thứ tự duyệt:  
**A, B, C, E, D, F, G**

Well done!  
Thank you, Scrat



# Duyệt đồ thị

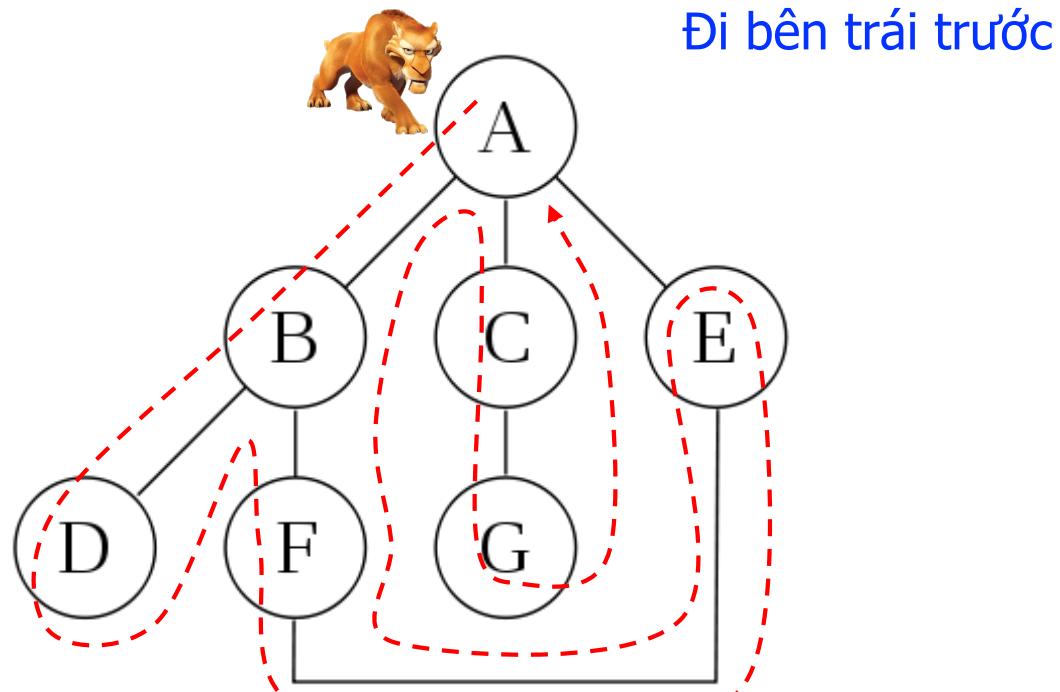
- Duyệt theo chiều sâu (Depth first search – DFS)
  - Ý tưởng: khám phá mê cung với dây và phẩn.
    - Đi xuống sâu nhất có thể, không đi được nữa thì quay lên



# Duyệt đồ thị

- Duyệt theo chiều sâu (Depth first search – DFS)
  - Ý tưởng: khám phá mê cung với dây và phẩn.

Hey! Diego, mang theo dây  
thừng và đi sâu nhất có thể,  
không đi được thì quay lên



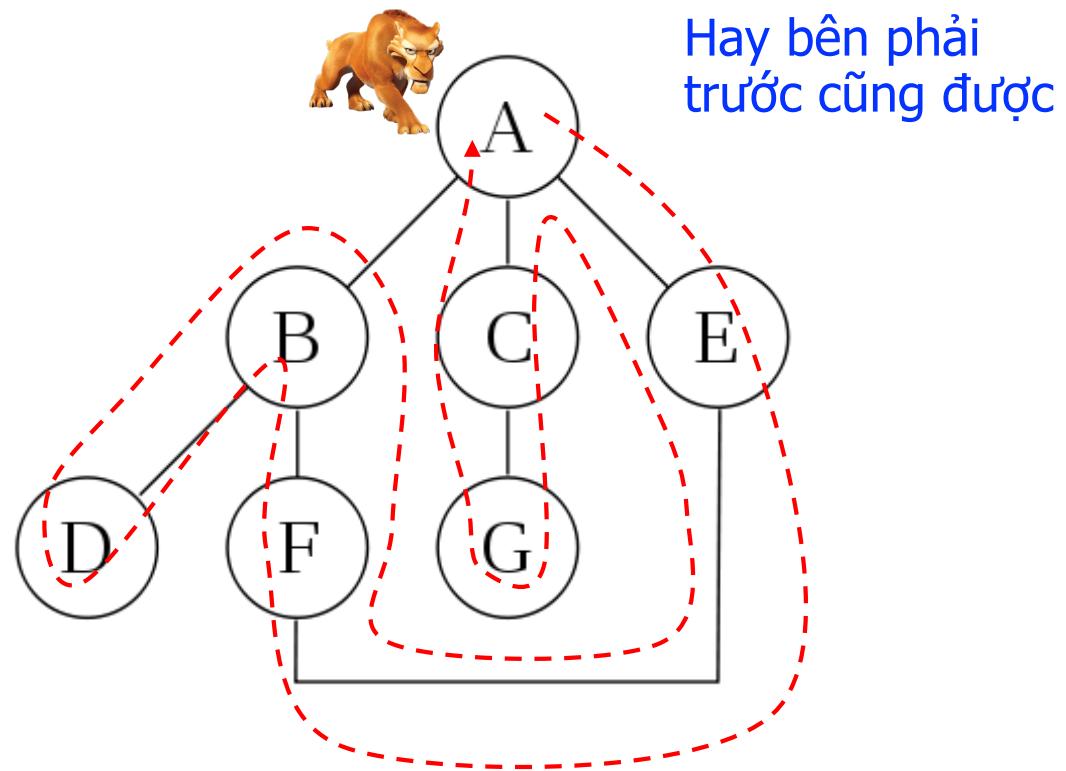
# Duyệt đồ thị

- Duyệt theo chiều sâu (Depth first search – DFS)
  - Ý tưởng: khám phá mê cung với dây và phẩn.

Hey! Diego, mang theo dây  
thừng và đi sâu nhất có thể,  
không đi được thì quay lên



75

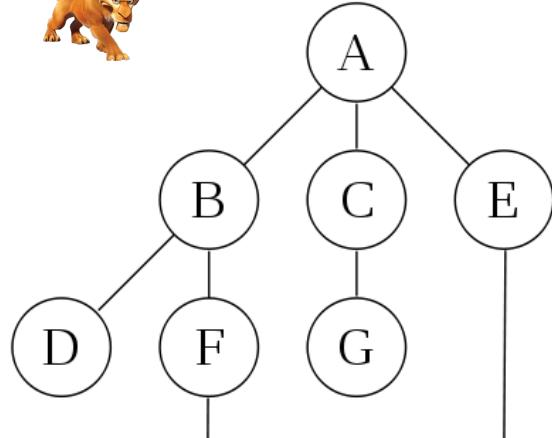


# Duyệt đồ thị

- Duyệt theo chiều sâu (Depth first search – DFS)
  - Sử dụng **ngăn xếp**
  - Các đỉnh trong ngăn xếp các đỉnh *sẽ được duyệt*. Một đỉnh có thể có mặt *nhiều lần* trong ngăn xếp.
- **Thuật toán dfs 1 (Phiên bản cài đặt)**
  - Đưa 1 đỉnh s bất kỳ vào **ngăn xếp** (vd: đỉnh 1)
  - **while** **ngăn xếp** chưa rỗng **do**
    - Lấy đỉnh ở đầu **ngăn xếp** ra => gọi là đỉnh u
    - **if** (u đã duyệt) **continue**; // *bỏ qua*
    - **Duyệt u (vd: in u ra màn hình)**
    - *Đánh dấu u đã duyệt*
    - **for** các đỉnh kề v của u **do**
      - Đưa v vào **ngăn xếp**

# Duyệt đồ thị

- Duyệt theo chiều sâu (Depth first search – DFS)
  - Sử dụng **ngăn xếp**
  - Các đỉnh trong ngăn xếp các đỉnh **sẽ được duyệt**. Một đỉnh có thể có mặt **nhiều lần** trong ngăn xếp.
- **Thuật toán dfs 2 (Phiên bản bài tập lý thuyết)**
  - Đưa 1 đỉnh s bất kỳ vào **ngăn xếp** (vd: đỉnh 1)
  - **while** **ngăn xếp** chưa rỗng **do**
    - Lấy đỉnh ở đầu **ngăn xếp** ra => gọi là đỉnh u
    - **if** (u đã duyệt) **continue**; // **bỏ qua**
    - **Duyệt u (vd: in u ra màn hình)**
    - **Đánh dấu u đã duyệt**
    - **for** các đỉnh kề v của u **do**
      - **if** (v chưa duyệt)
        - Đưa v vào **ngăn xếp**



Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

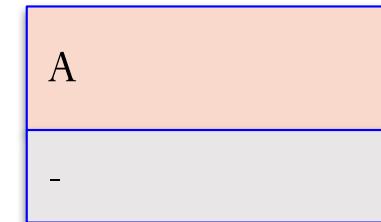
while ngăn xếp chưa rỗng do

Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

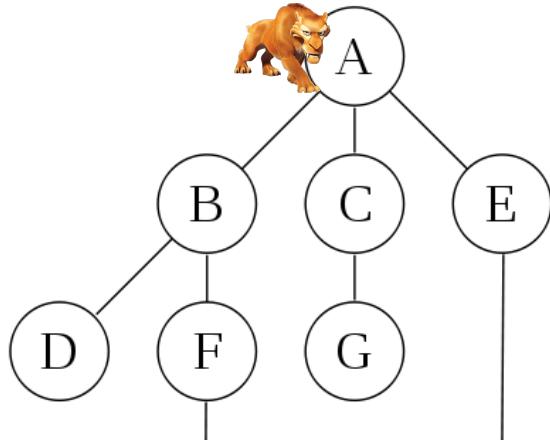
for các đỉnh kề v của u do

if (v chưa duyệt) Đưa v vào ngăn xếp

Đỉnh stack



Push A vào stack



Lấy A ra khỏi stack  
Đánh dấu A đã duyệt



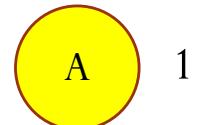
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

**while** ngăn xếp chưa rỗng **do**

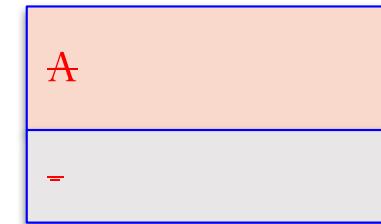
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

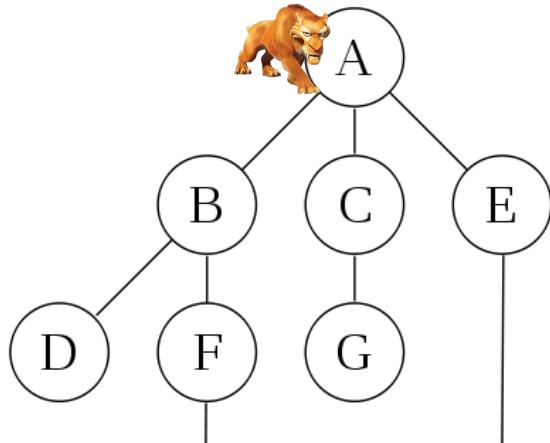
**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Lần lượt push B,  
C, E vào stack  
(chú ý thứ tự)

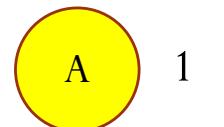
Đưa 1 đỉnh **s** bất kỳ vào **ngăn xếp** (vd: đỉnh 1)

**while** **ngăn xếp** chưa rỗng **do**

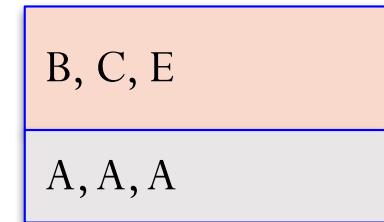
Lấy đỉnh ở đỉnh **ngăn xếp** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

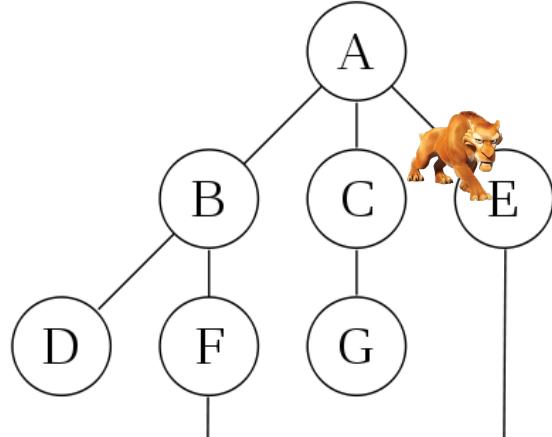
**for** các đỉnh kề **v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **ngăn xếp**



Đỉnh stack





Lấy E ra khỏi stack  
Đánh dấu E đã duyệt



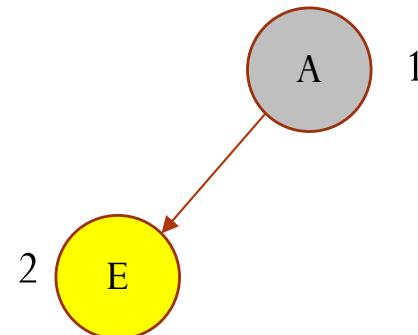
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

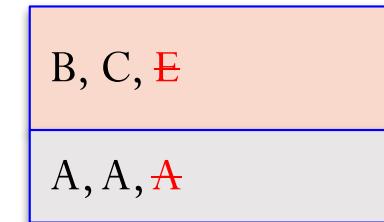
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

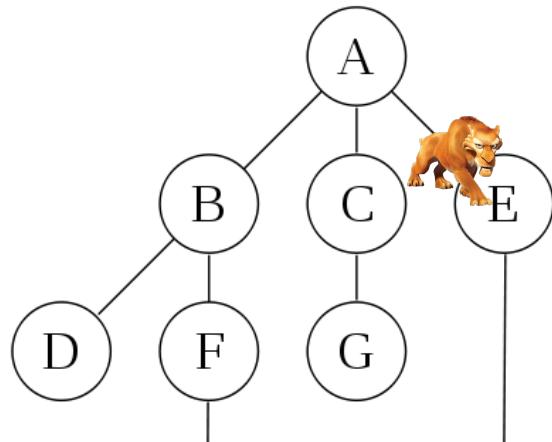
for các đỉnh kề v của u do

if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Push F vào stack (vì A đã duyệt, không bỏ vào stack nữa)

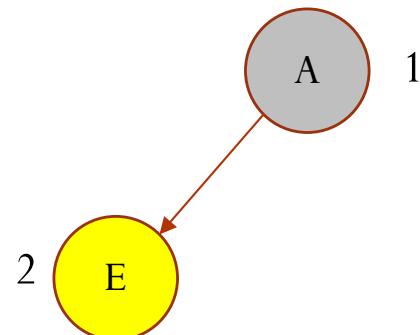
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

**while** ngăn xếp chưa rỗng **do**

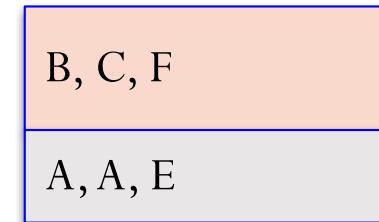
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

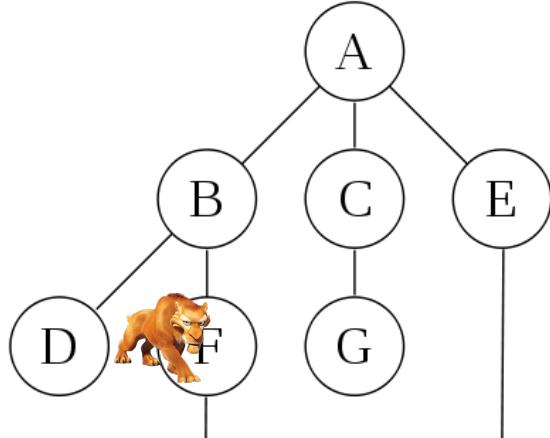
**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Lấy F ra khỏi stack  
Đánh dấu F đã duyệt

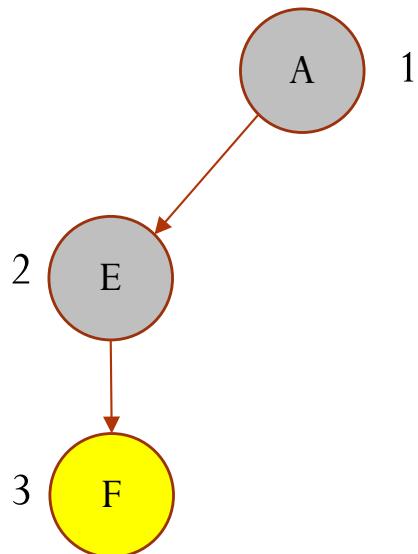
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

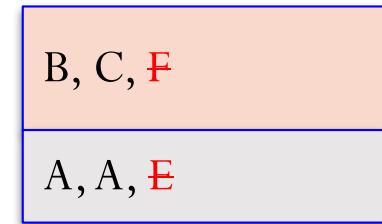
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

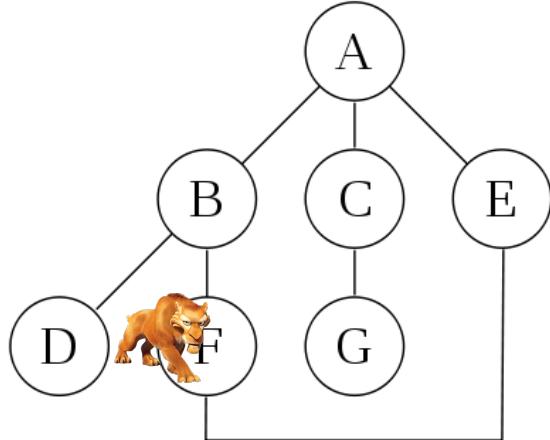
for các đỉnh kề v của u do

if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Push B vào stack (E  
đã duyệt)

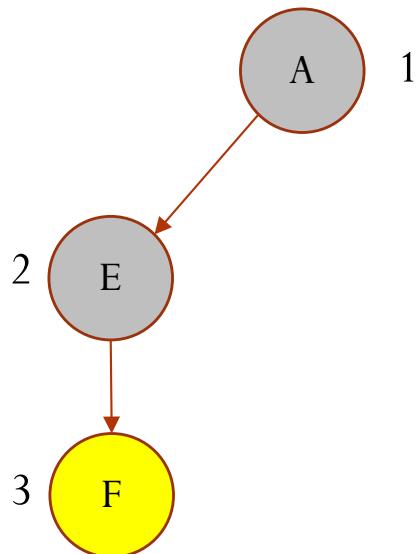
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

**while** ngăn xếp chưa rỗng **do**

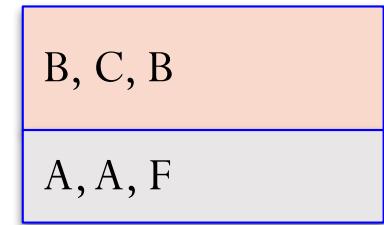
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

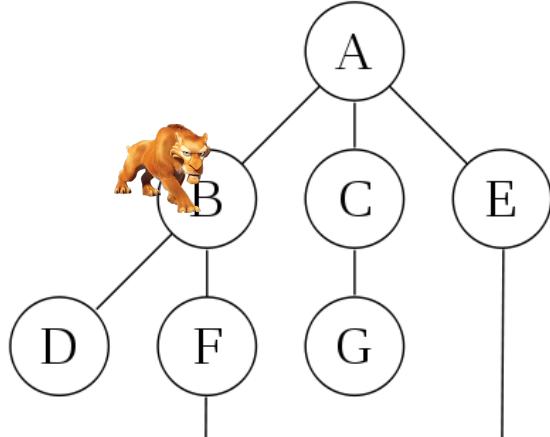
**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Lấy B ra khỏi stack  
Đánh dấu B đã duyệt

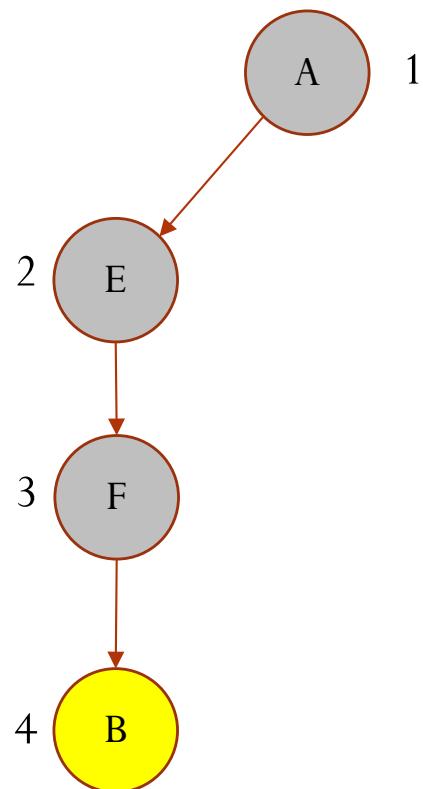
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

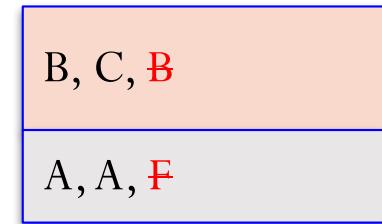
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

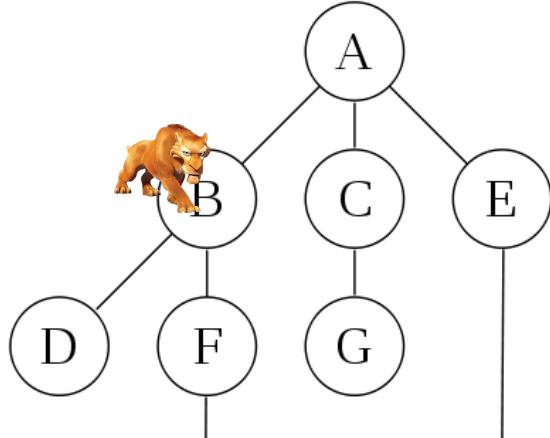
for các đỉnh kề v của u do

if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Push D vào stack (A, F đã duyệt)

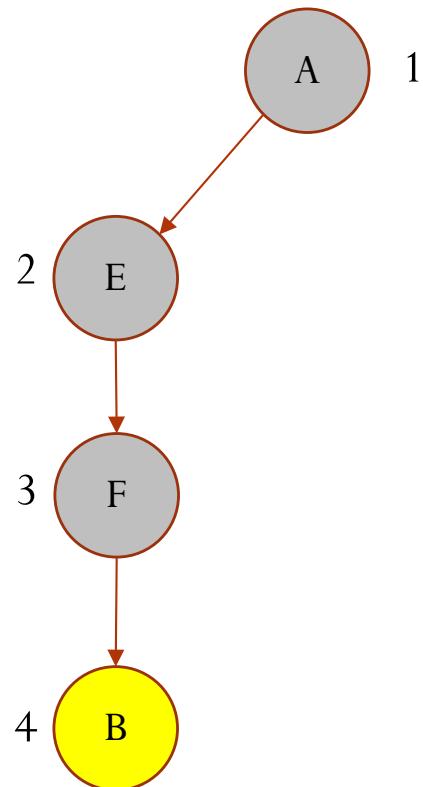
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

**while** ngăn xếp chưa rỗng **do**

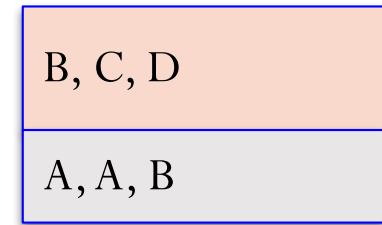
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

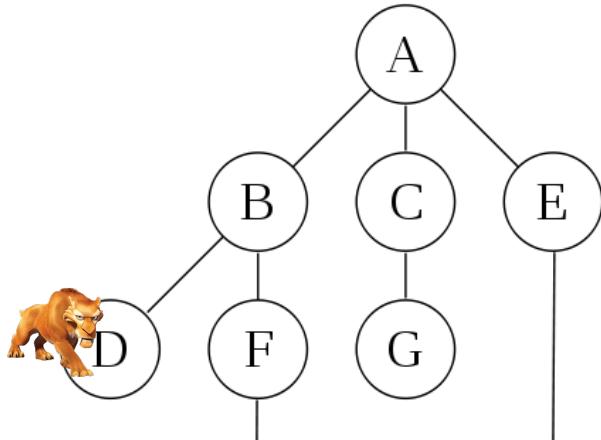
**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Lấy D ra khỏi stack  
Đánh dấu D đã duyệt

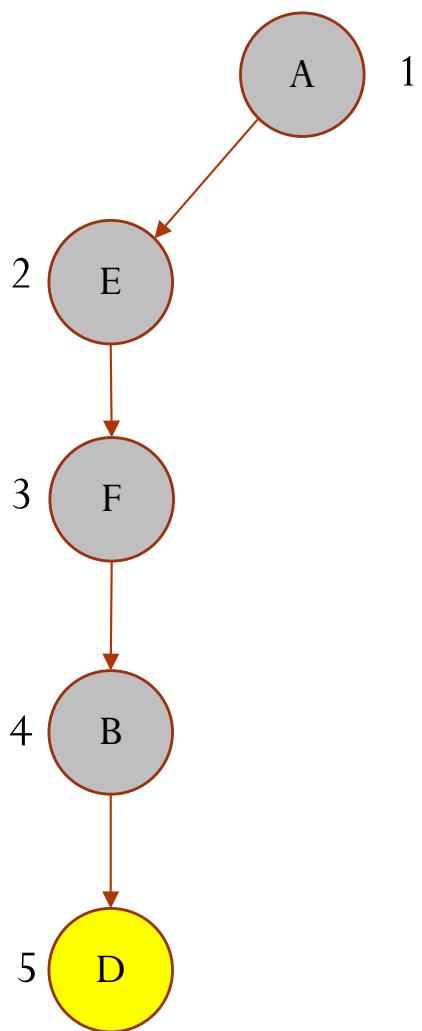
87

Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

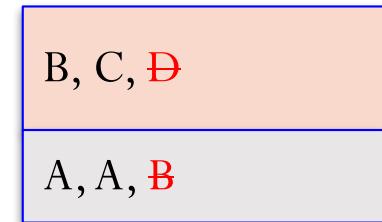
while ngăn xếp chưa rỗng do

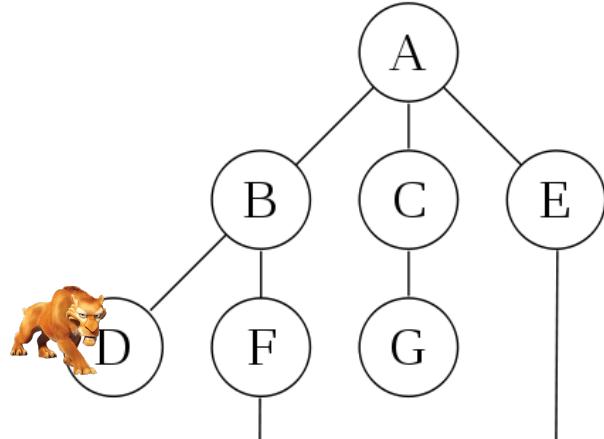
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

for các đỉnh kề v của u do  
if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Không push gì vào stack cả (B đã duyệt)

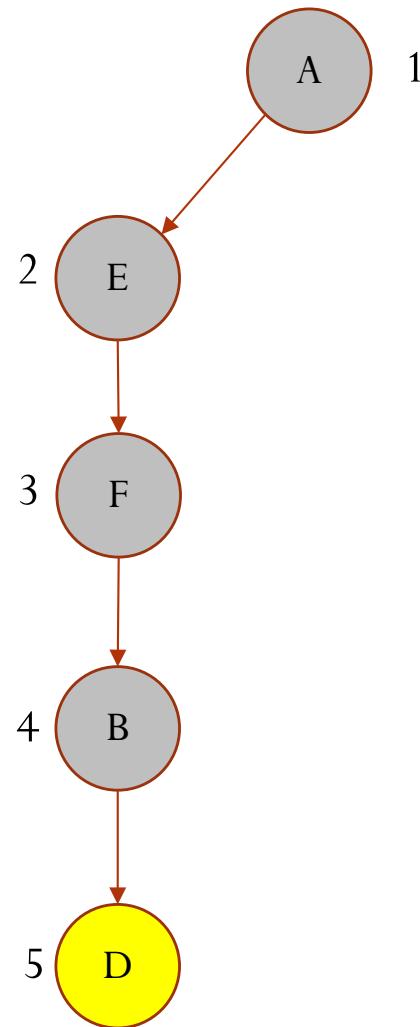
Đưa 1 đỉnh **s** bất kỳ vào **ngăn xếp** (vd: đỉnh 1)

**while** **ngăn xếp** chưa rỗng **do**

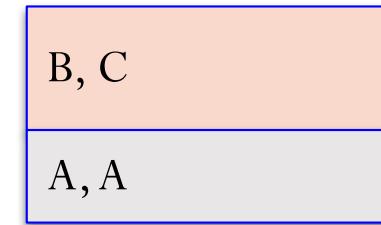
Lấy đỉnh ở đỉnh **ngăn xếp** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

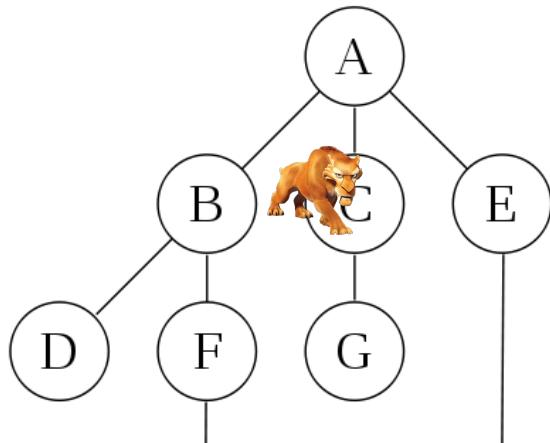
**for** các đỉnh **kề v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **ngăn xếp**



Đỉnh stack





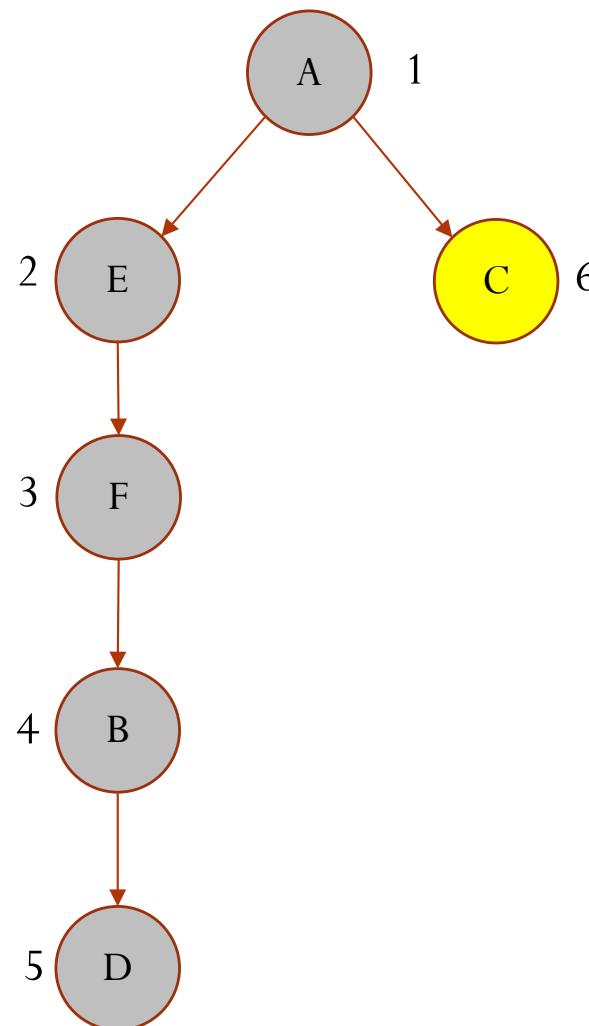
Lấy C ra khỏi stack  
Đánh dấu C đã duyệt

Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

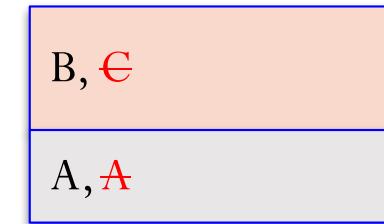
while ngăn xếp chưa rỗng do

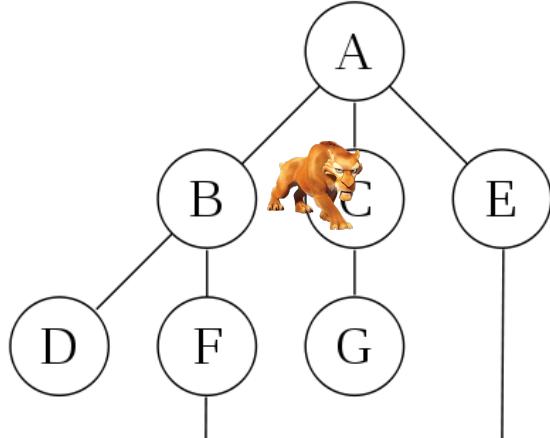
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

for các đỉnh kề v của u do  
if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Push G vào stack (A  
đã duyệt)

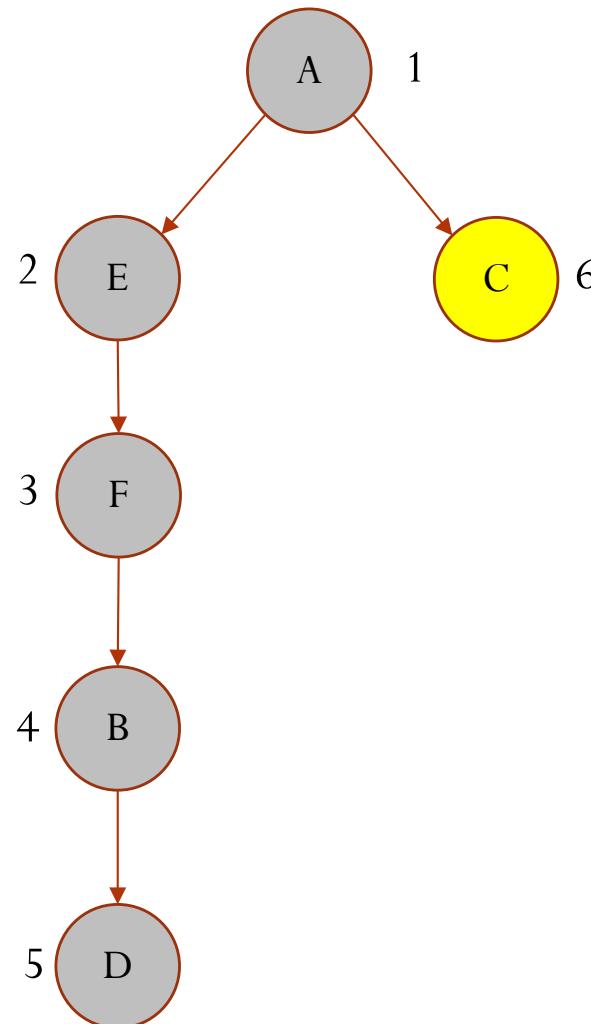
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

**while** ngăn xếp chưa rỗng **do**

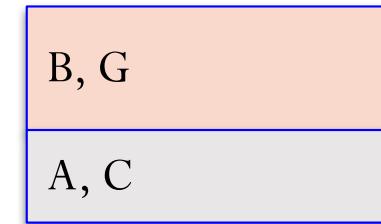
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

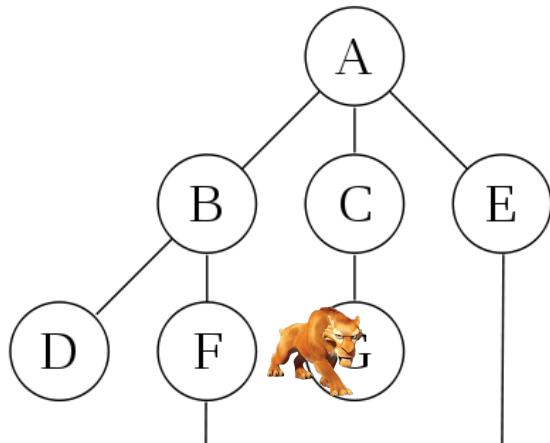
**for** các đỉnh kề v của u **do**

**if** (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





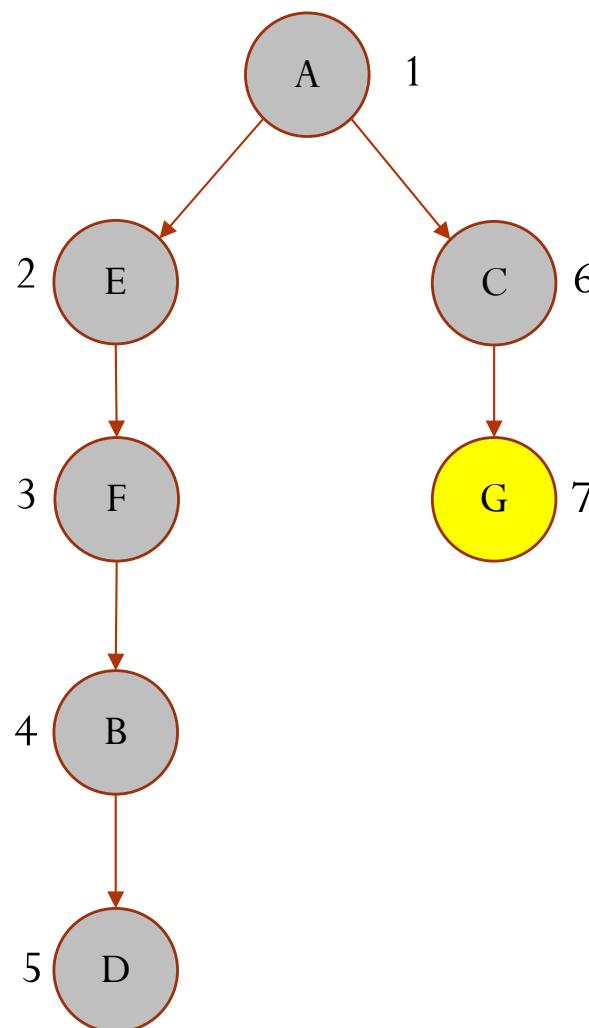
Lấy G ra khỏi stack  
Đánh dấu G đã duyệt

Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

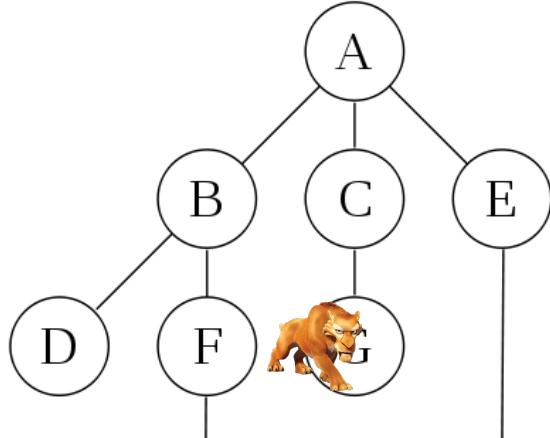
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

for các đỉnh kề v của u do  
if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Không push gì vào stack cả (C đã duyệt)

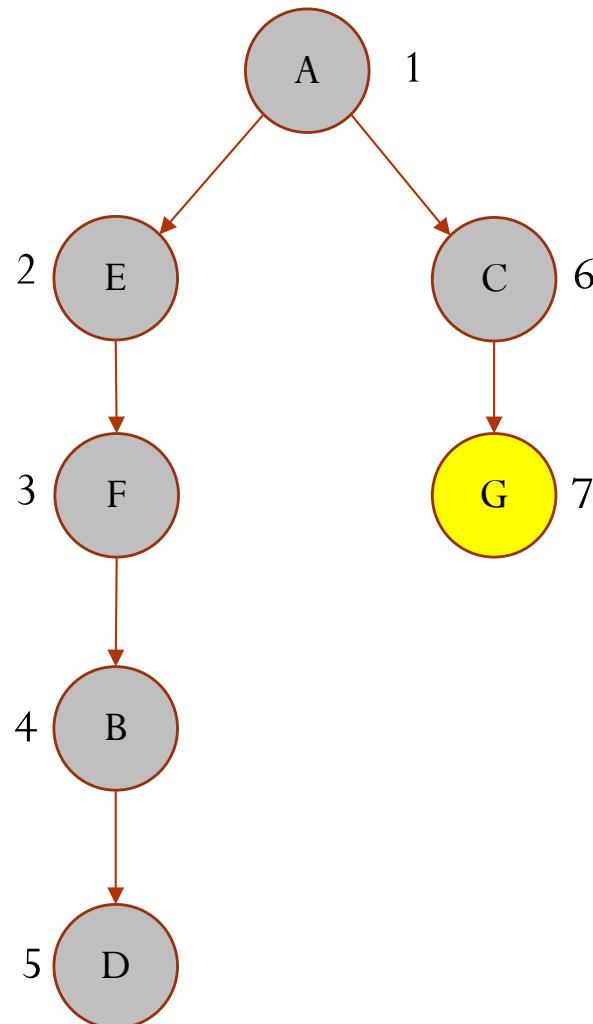
Đưa 1 đỉnh **s** bất kỳ vào **ngăn xếp** (vd: đỉnh 1)

**while** **ngăn xếp** chưa rỗng **do**

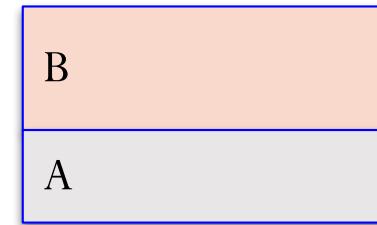
Lấy đỉnh ở đỉnh **ngăn xếp** ra => gọi là đỉnh **u**  
Làm gì đó trên **u**

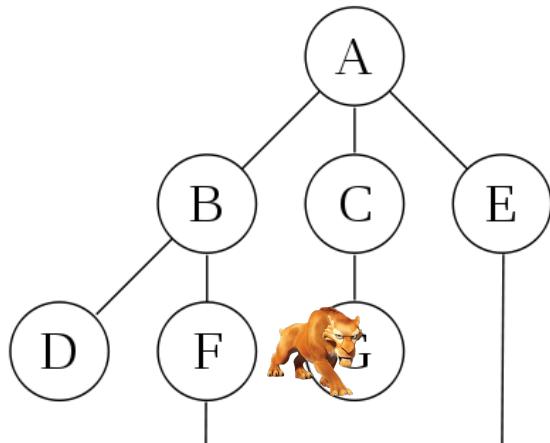
**for** các đỉnh **kề v** của **u** **do**

**if** (**v** chưa duyệt) Đưa **v** vào **ngăn xếp**



Đỉnh stack





Lấy B ra khỏi stack  
B đã được duyệt =>  
bỏ qua

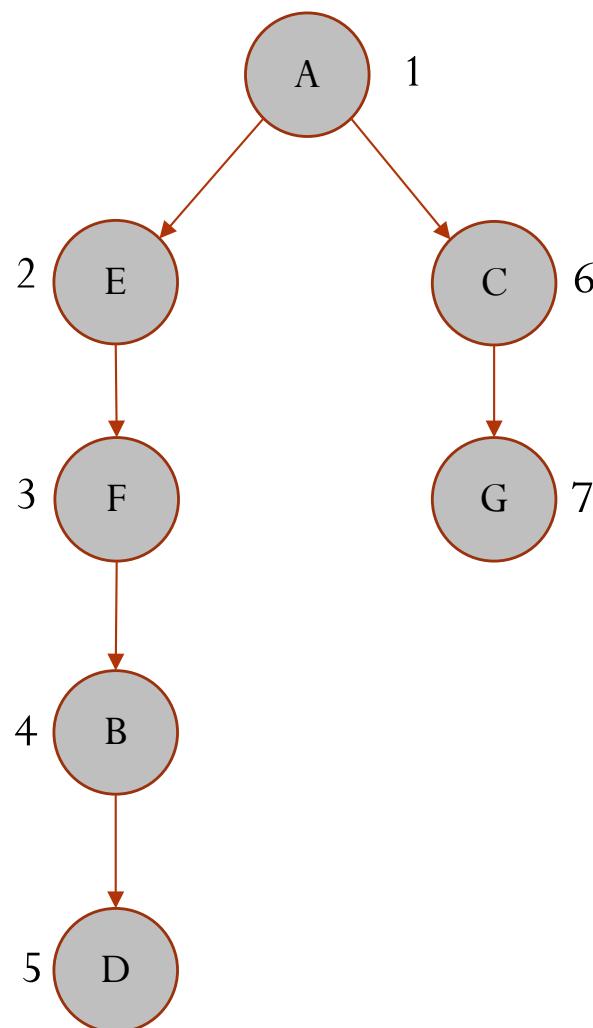
Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

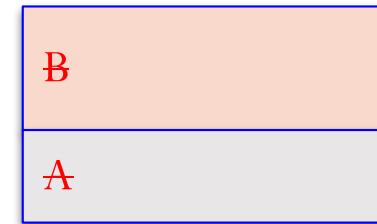
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

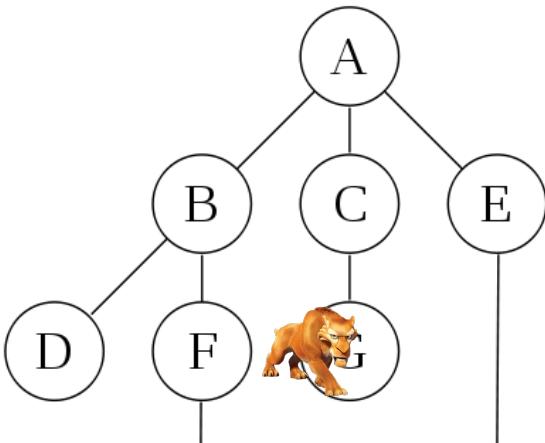
for các đỉnh kề v của u do

if (v chưa duyệt) Đưa v vào ngăn xếp



Đỉnh stack





Thứ tự duyệt:  
A, E, F, B, D, C, G

94

Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)

while ngăn xếp chưa rỗng do

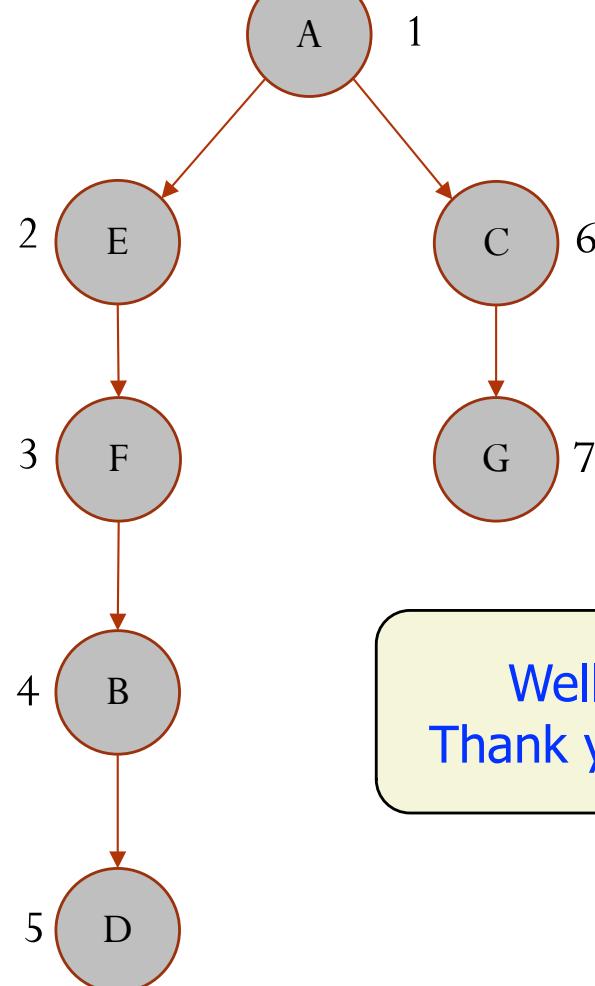
Lấy đỉnh ở đỉnh ngăn xếp ra => gọi là đỉnh u  
Làm gì đó trên u

for các đỉnh kề v của u do

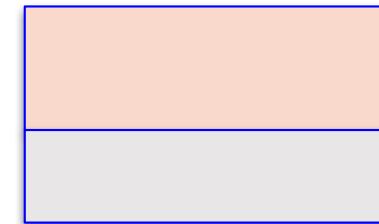
if (v chưa duyệt) Đưa v vào ngăn xếp



Xong!



Đỉnh stack



Stack rỗng =>  
Kết thúc

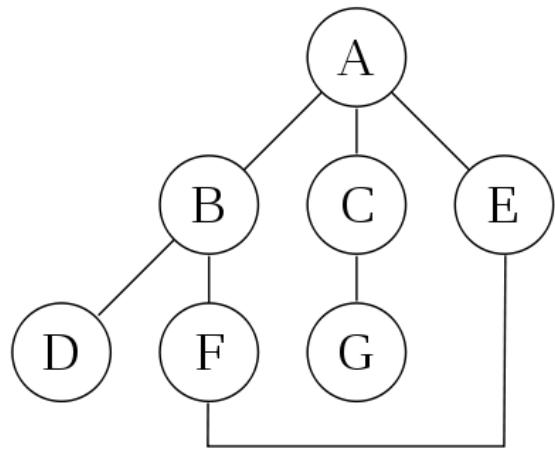
Well done!  
Thank you, Diego



# Duyệt đồ thị

- Duyệt theo chiều sâu (đệ quy)
  - Không sử dụng stack
  - Sử dụng đệ quy
- Thuật toán đệ quy:

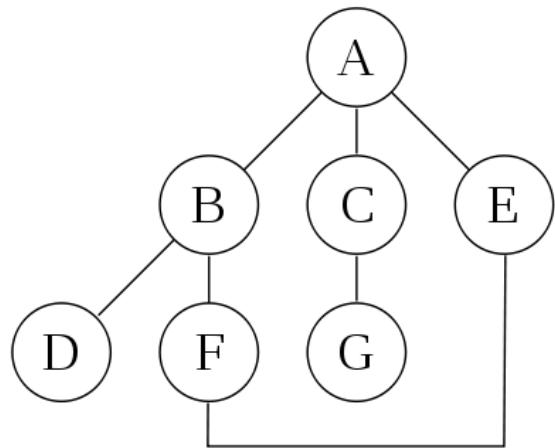
```
void DFS(u) {  
    if (u đã duyệt) return;  
    //Duyệt u: in u ra màn hình hoặc đánh số đỉnh u  
    Đánh dấu u đã duyệt  
    for (các đỉnh kề v của u) do  
        DFS(v); //gọi đệ quy duyệt các đỉnh kề v của u  
}
```



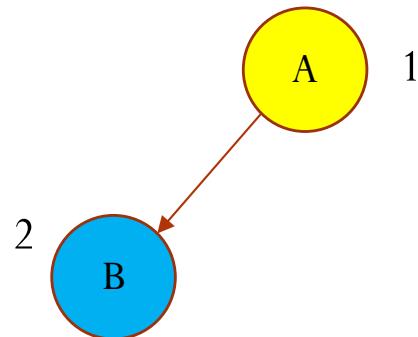
Đánh dấu A đã duyệt

A  
1

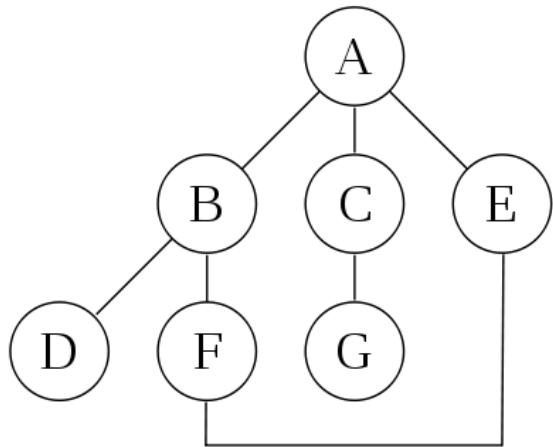
DFS(A)



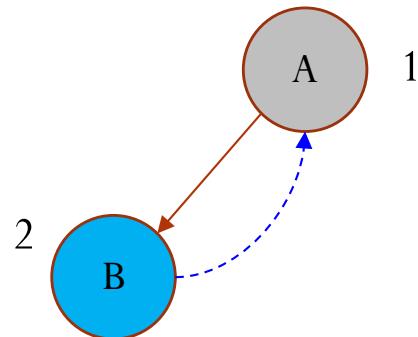
Đánh dấu B đã duyệt



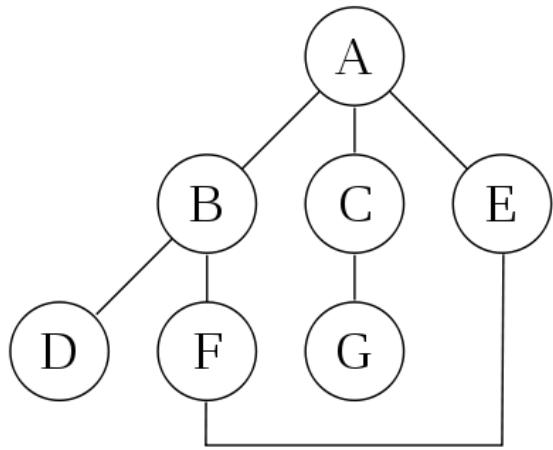
DFS(A)  
DFS(B)



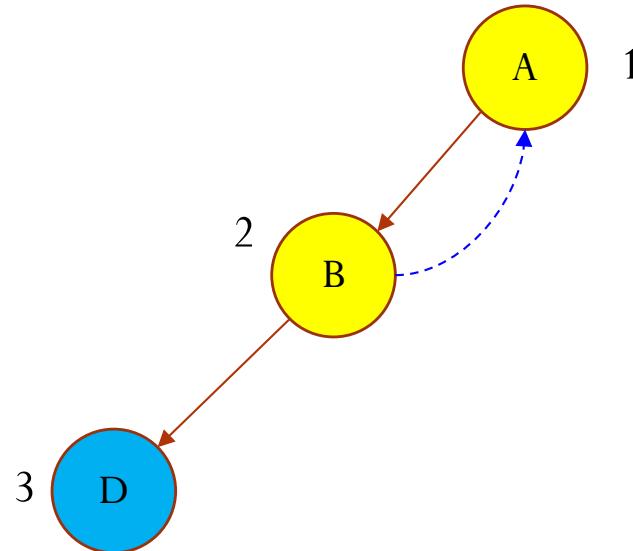
A đã được duyệt =>  
bỏ qua



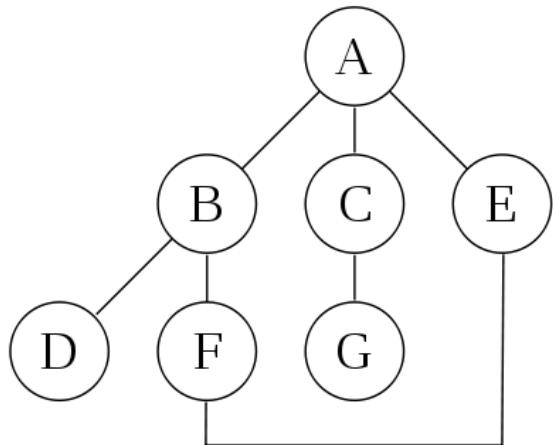
DFS(A)  
DFS(B)  
Bỏ qua A



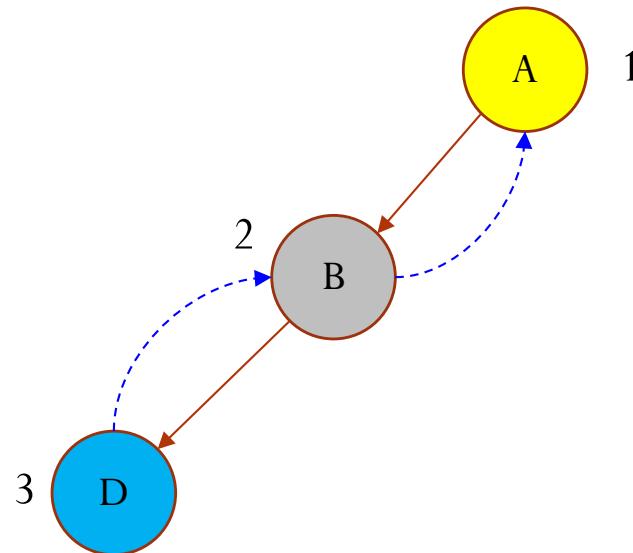
Đánh dấu D đã duyệt



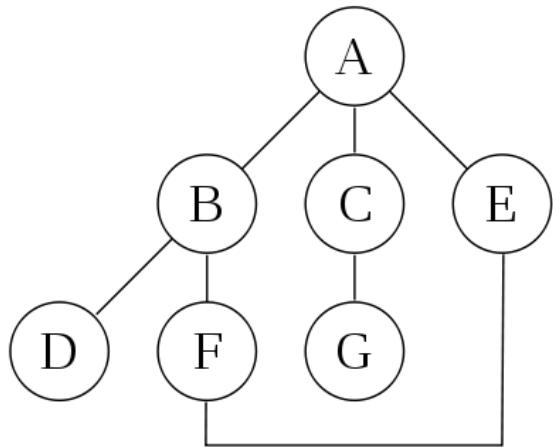
DFS(A)  
DFS(B)  
Bỏ qua A  
DFS(D)



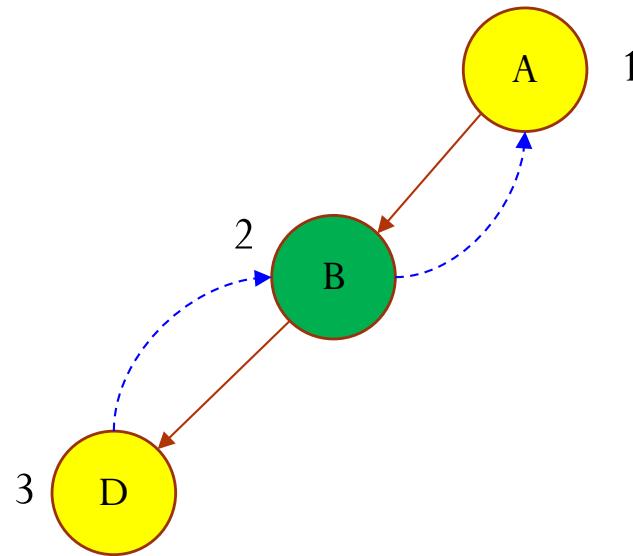
B đã được duyệt =>  
bỏ qua



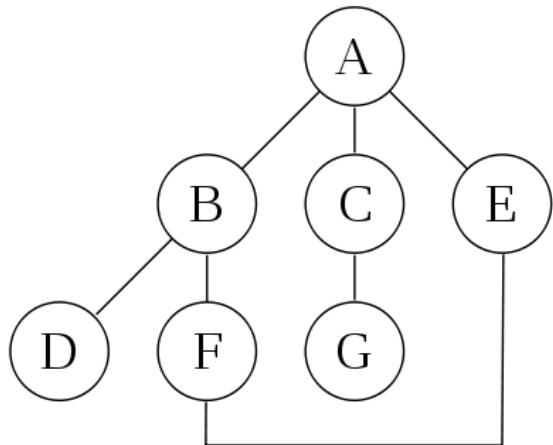
DFS(A)  
DFS(B)  
Bỏ qua A  
DFS(D)  
BỎ QUÁ B



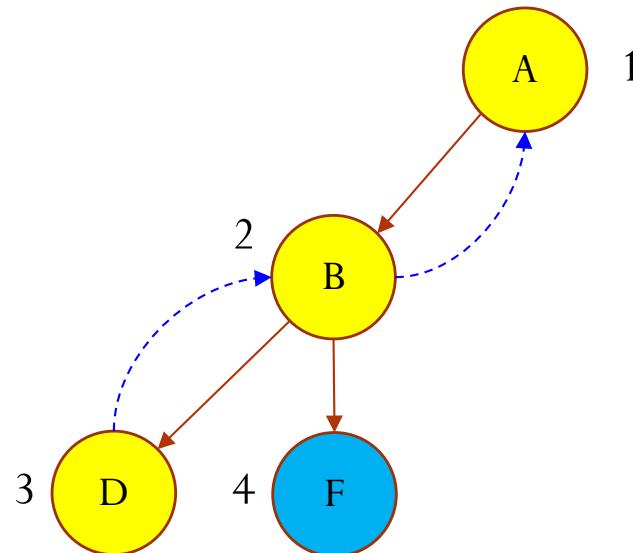
D đã xét xong =>  
quay về B



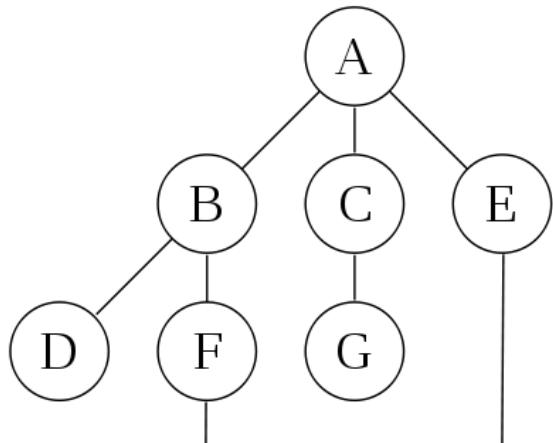
DFS(A)  
DFS(B)  
Bỏ qua A  
DFS(D)  
Bỏ qua B  
--->



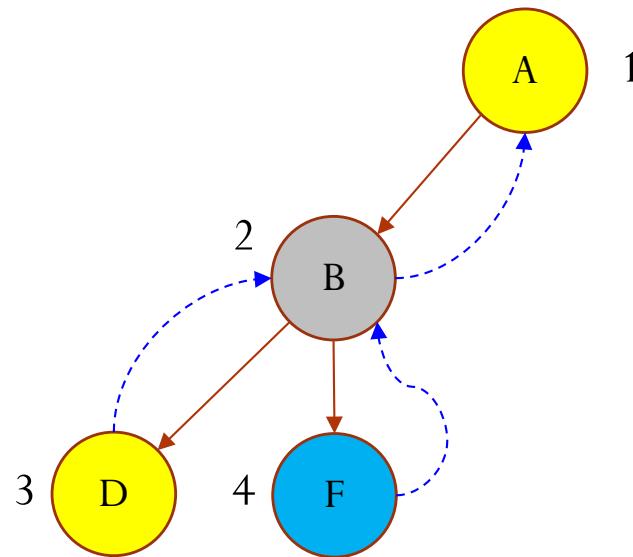
Đánh dấu F được duyệt



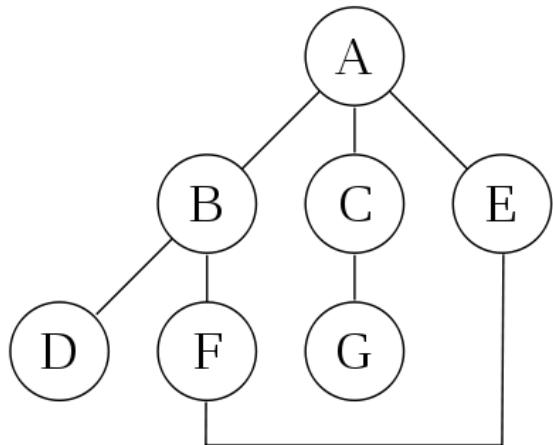
DFS(A)  
DFS(B)  
Bỏ qua A  
DFS(D)  
Bỏ qua B  
**DFS(F)**



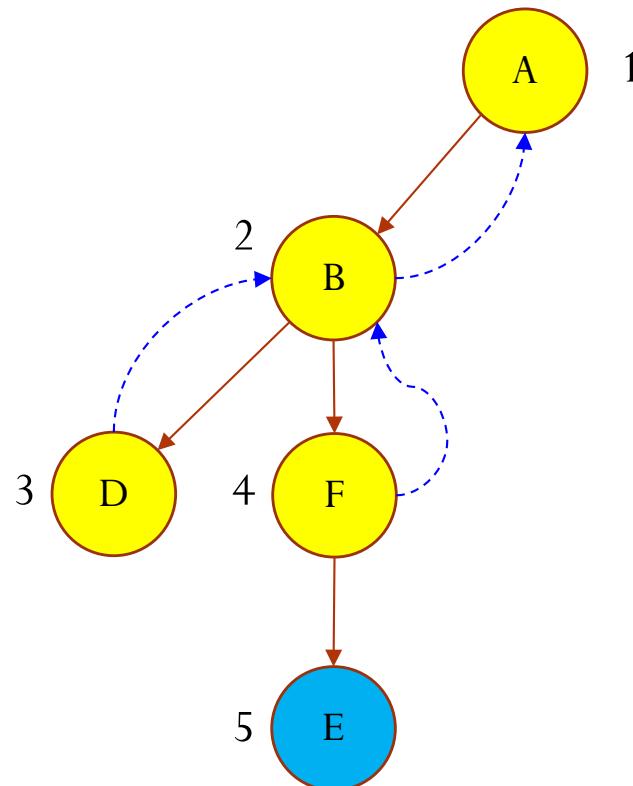
B đã được duyệt =>  
bỏ qua



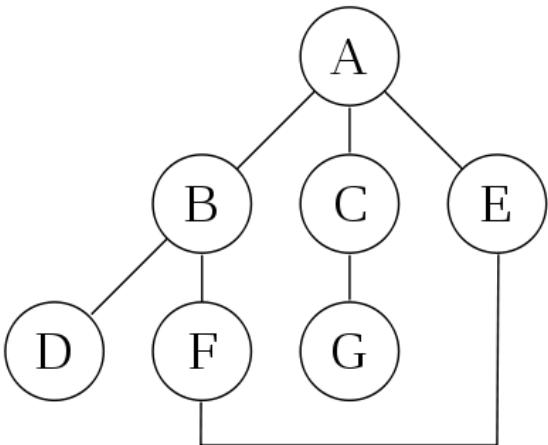
$\text{DFS}(A)$   
 $\text{DFS}(B)$   
 Bỏ qua A  
 $\text{DFS}(D)$   
 Bỏ qua B  
 $\text{DFS}(F)$   
 BỎ QUÁ B



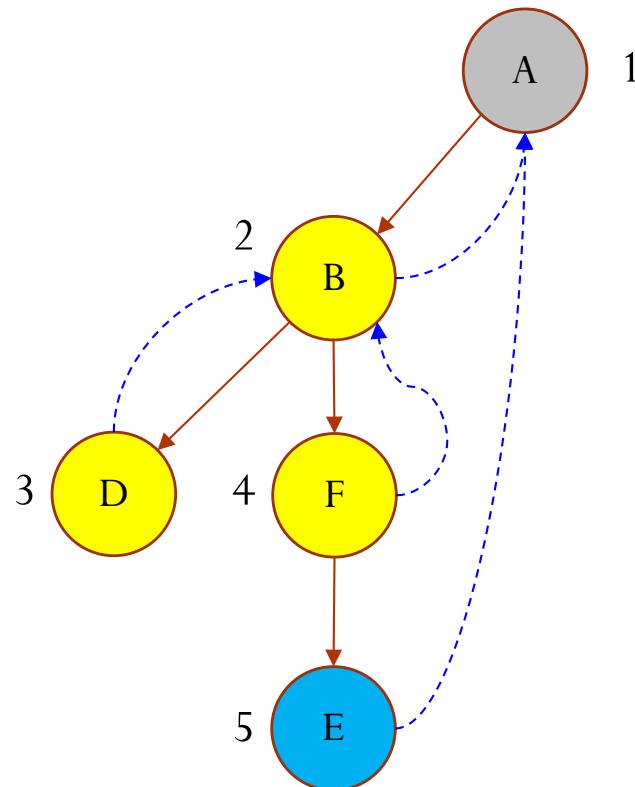
Đánh dấu E đã duyệt



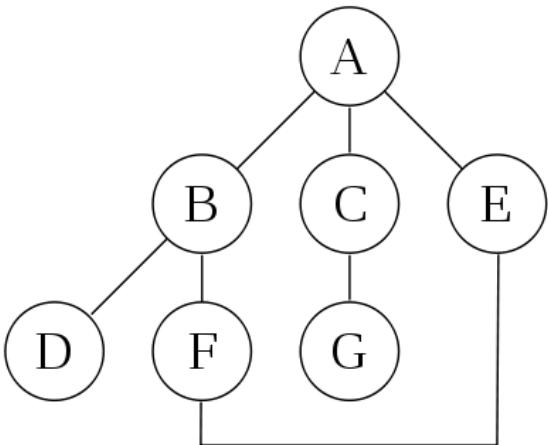
**DFS(A)**  
**DFS(B)**  
 Bỏ qua A  
**DFS(D)**  
 Bỏ qua B  
**DFS(F)**  
 Bỏ qua B  
**DFS(E)**



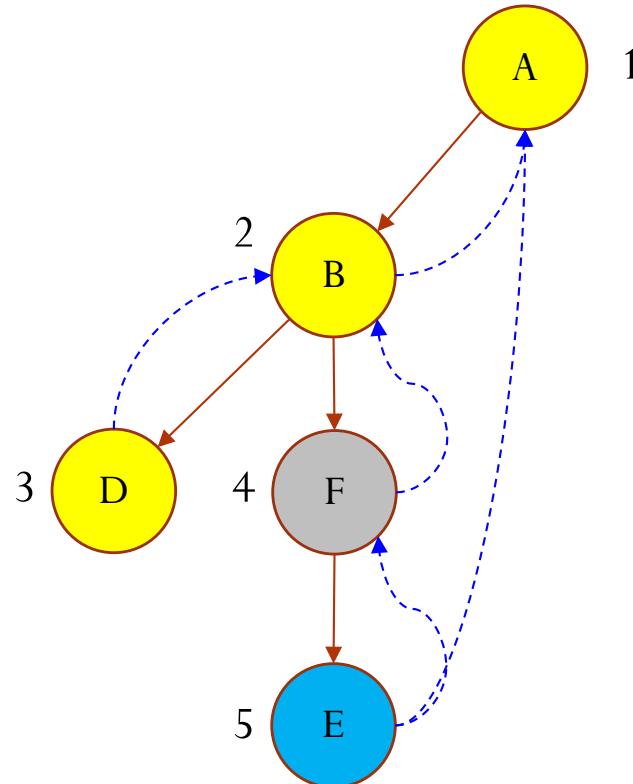
A đã được duyệt =>  
bỏ qua



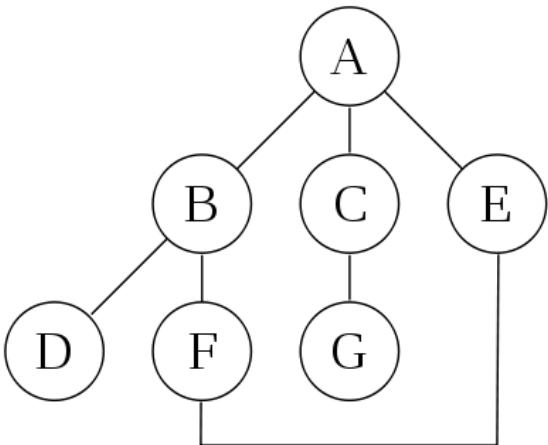
$\text{DFS}(A)$   
 $\text{DFS}(B)$   
 Bỏ qua A  
 $\text{DFS}(D)$   
 Bỏ qua B  
 $\text{DFS}(F)$   
 BỎ QUÁ B  
 $\text{DFS}(E)$   
 BỎ QUÁ A



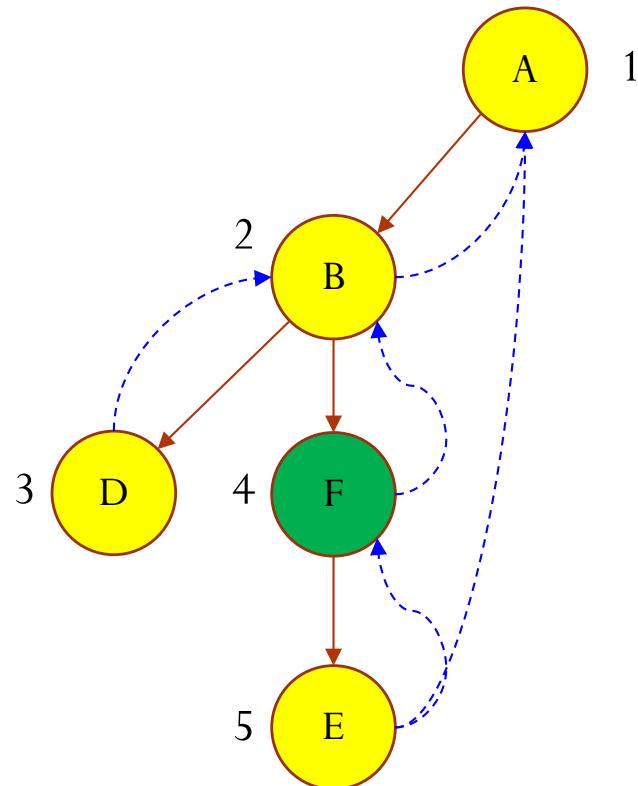
F đã được duyệt =>  
bỏ qua



DFS(A)
DFS(B)
BỎ QUA A
DFS(D)
BỎ QUA B
DFS(F)
BỎ QUA B
DFS(E)
BỎ QUA A
<b>BỎ QUA F</b>

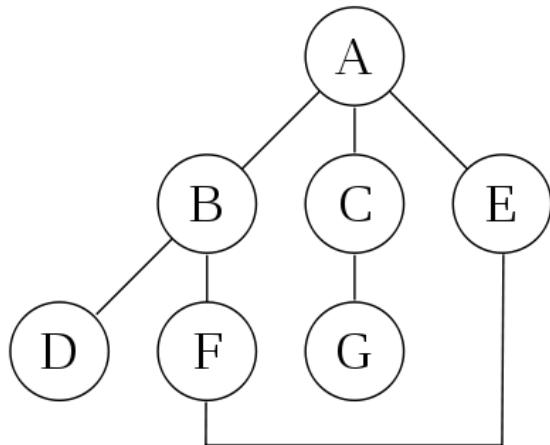


E đã xét xong =>  
quay về F

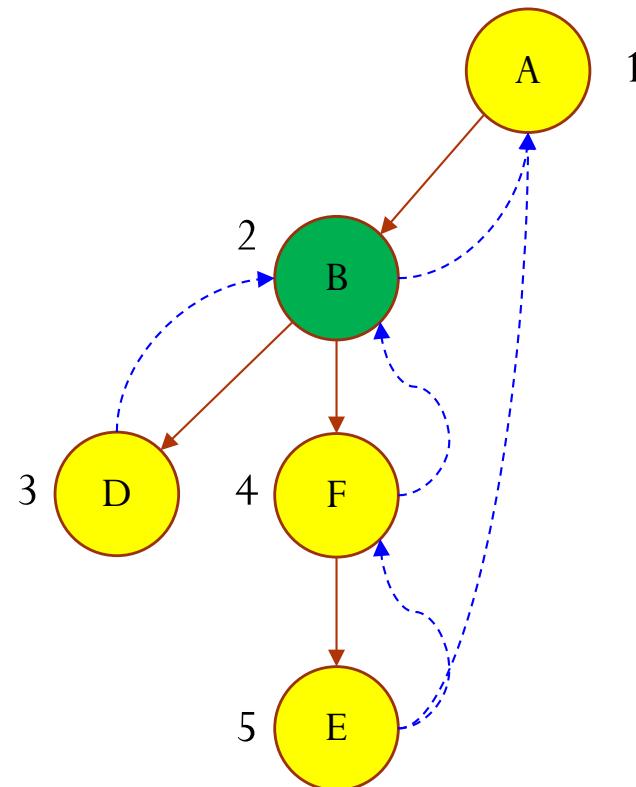


DFS(A)
DFS(B)
BỎ QUÁ A
DFS(D)
BỎ QUÁ B
DFS(F)
BỎ QUÁ B
DFS(E)
BỎ QUÁ A
BỎ QUÁ F

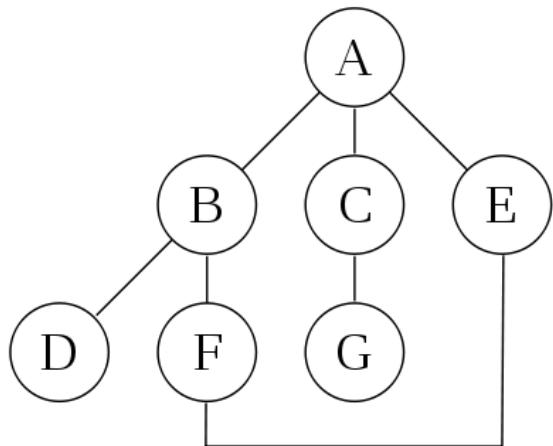
--->



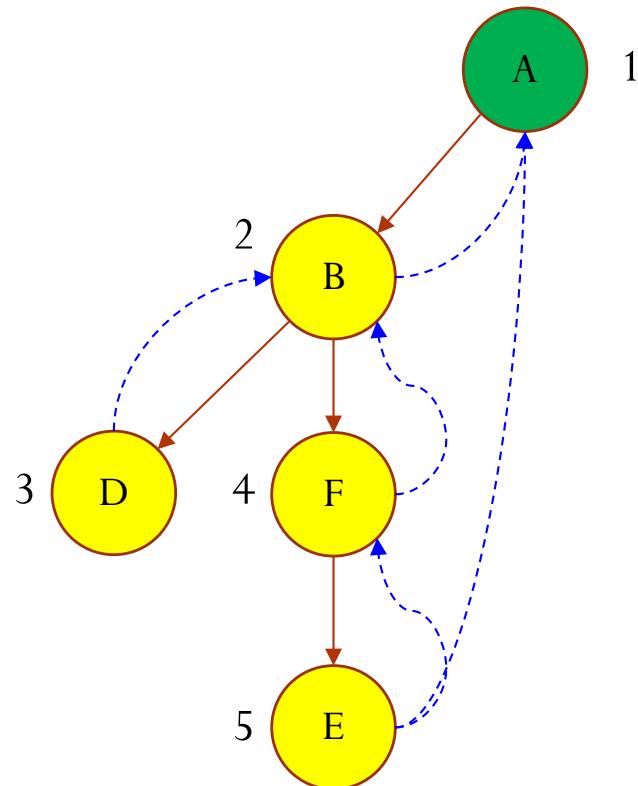
F đã xét xong =>  
quay về B



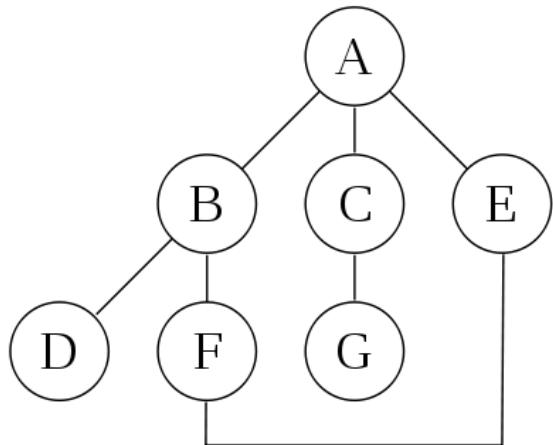
```
DFS(A)
DFS(B)
    BỎ qua A
DFS(D)
    BỎ qua B
DFS(F)
    BỎ qua B
DFS(E)
    BỎ qua A
    BỎ qua F
-->
```



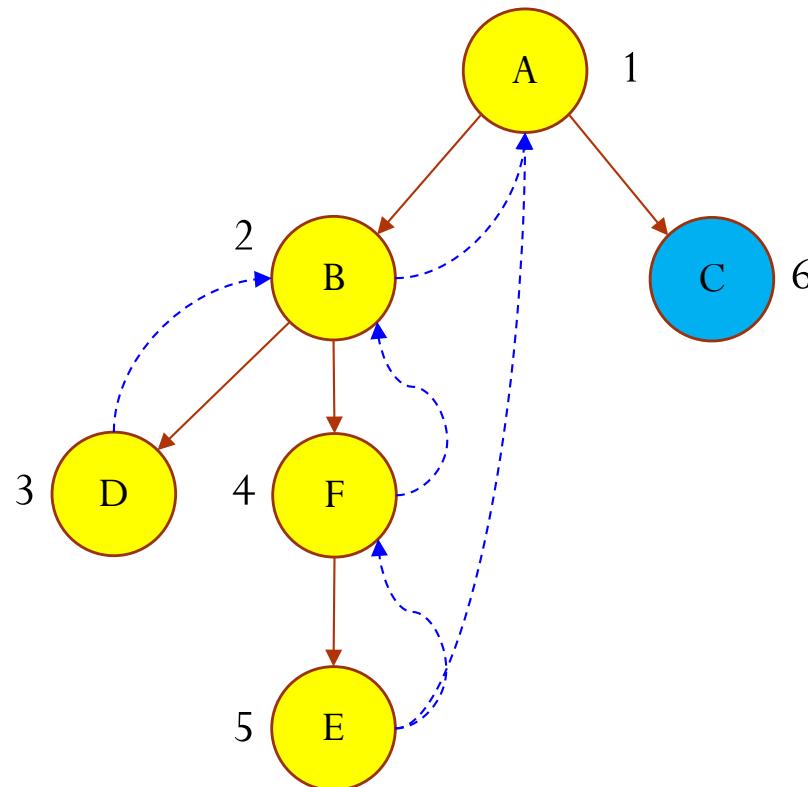
B đã xét xong =>  
quay về A



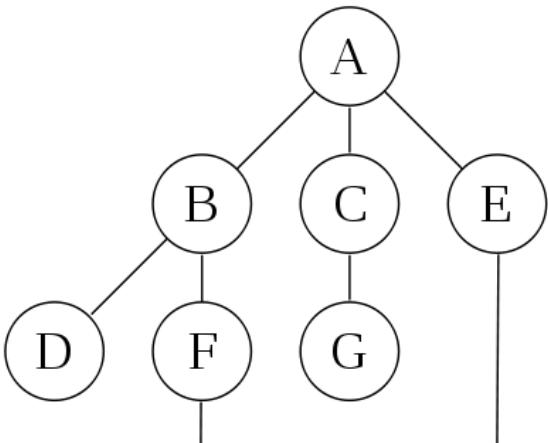
$\text{DFS}(A)$   
 $\text{DFS}(B)$   
 Bỏ qua A  
 $\text{DFS}(D)$   
 Bỏ qua B  
 $\text{DFS}(F)$   
 Bỏ qua B  
 $\text{DFS}(E)$   
 Bỏ qua A  
 Bỏ qua F  
 - - - - ->



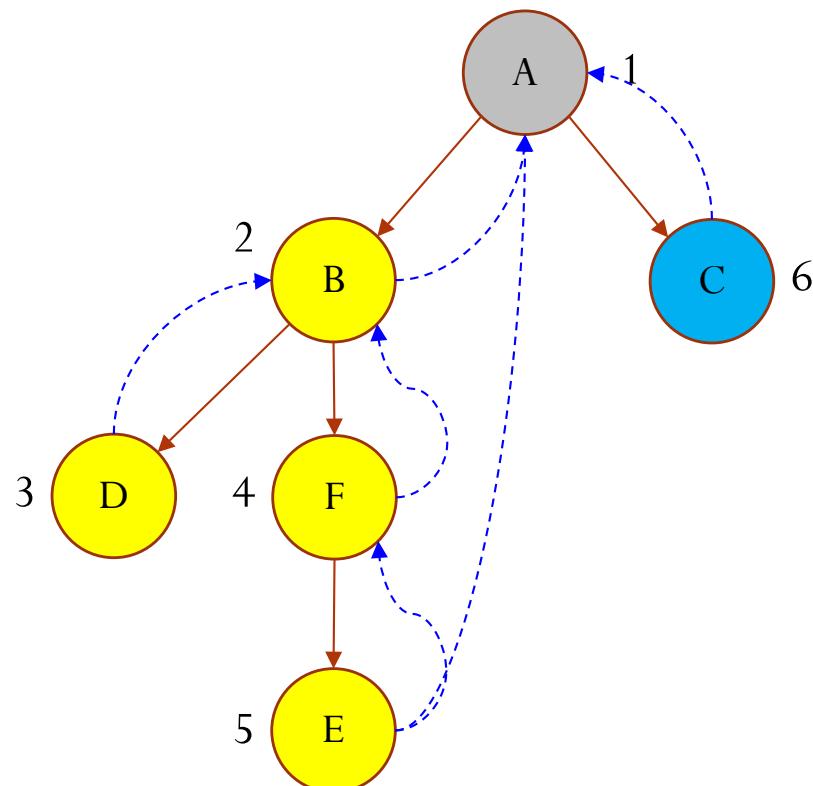
Đánh dấu C đã duyệt



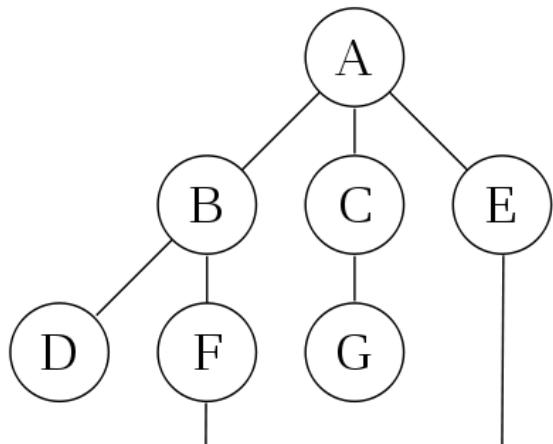
DFS(A)
DFS(B)
BỎ QUA A
DFS(D)
BỎ QUA B
DFS(F)
BỎ QUA B
DFS(E)
BỎ QUA A
BỎ QUA F
<b>DFS(C)</b>



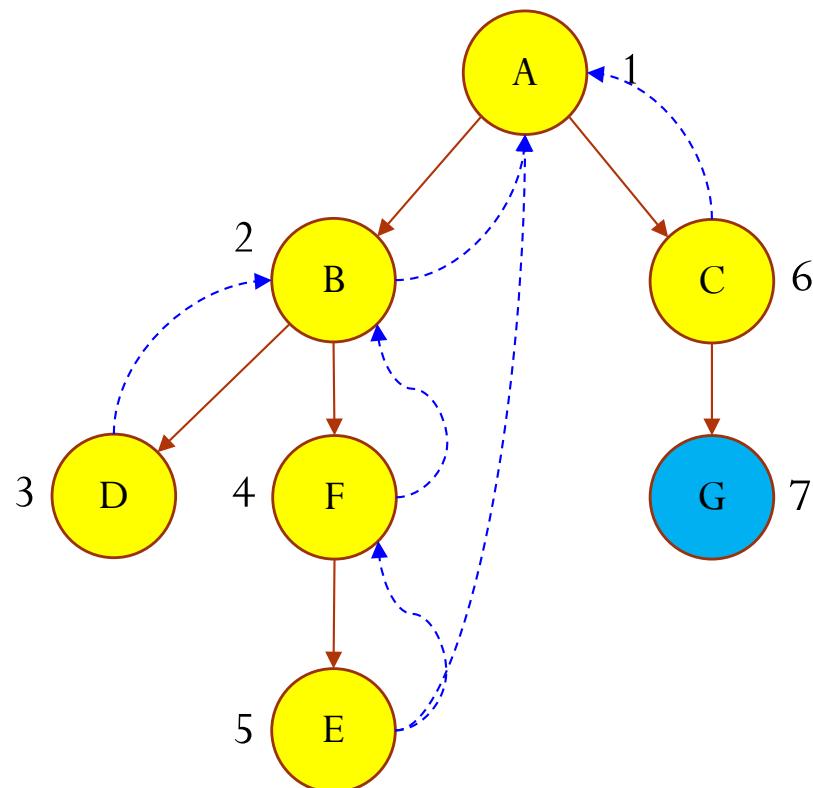
A đã được duyệt =>  
bỏ qua



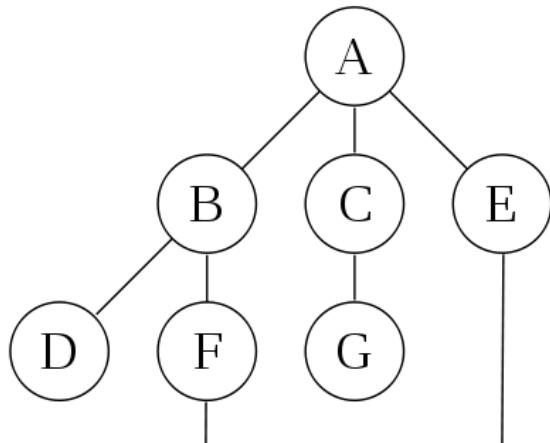
$\text{DFS}(A)$   
 $\text{DFS}(B)$   
 Bỏ qua A  
 $\text{DFS}(D)$   
 Bỏ qua B  
 $\text{DFS}(F)$   
 BỎ QUÁ B  
 $\text{DFS}(E)$   
 BỎ QUÁ A  
 BỎ QUÁ F  
 $\text{DFS}(C)$   
 BỎ QUÁ A



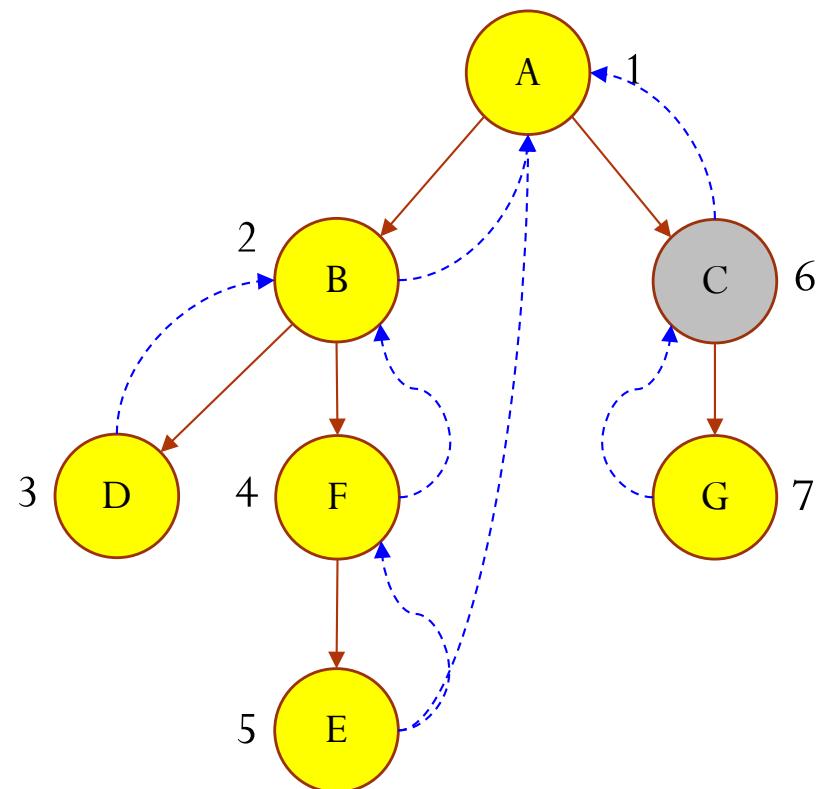
Đánh dấu G đã duyệt



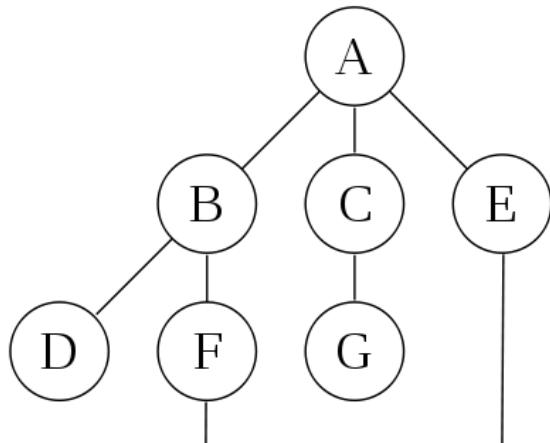
DFS(A)
DFS(B)
BỎ QUA A
DFS(D)
BỎ QUA B
DFS(F)
BỎ QUA B
DFS(E)
BỎ QUA A
BỎ QUA F
DFS(C)
BỎ QUA A
<b>DFS(G)</b>



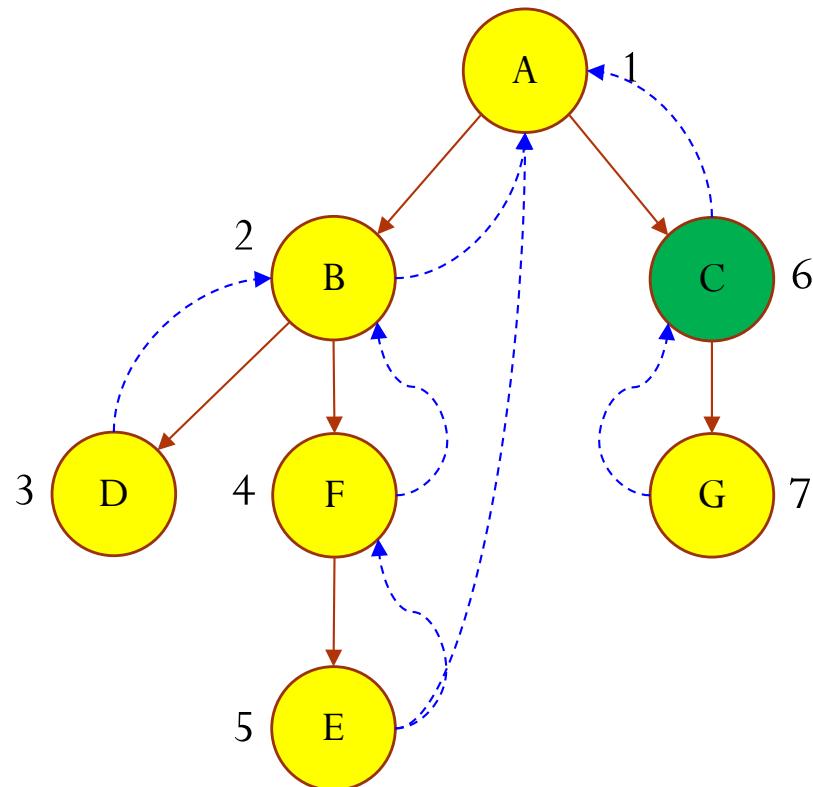
C đã được duyệt =>  
bỏ qua



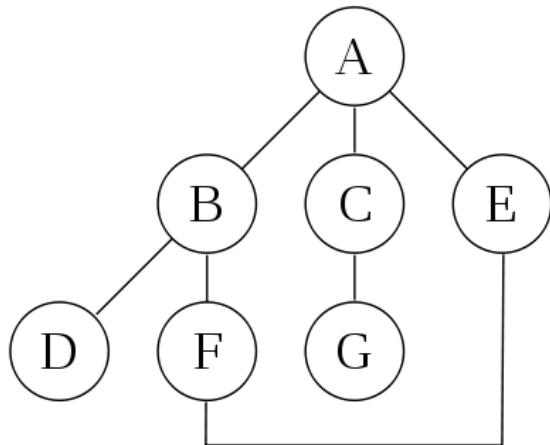
DFS(A)
DFS(B)
BỎ QUA A
DFS(D)
BỎ QUA B
DFS(F)
BỎ QUA B
DFS(E)
BỎ QUA A
BỎ QUA F
DFS(C)
BỎ QUA A
DFS(G)
<b>BỎ QUA C</b>



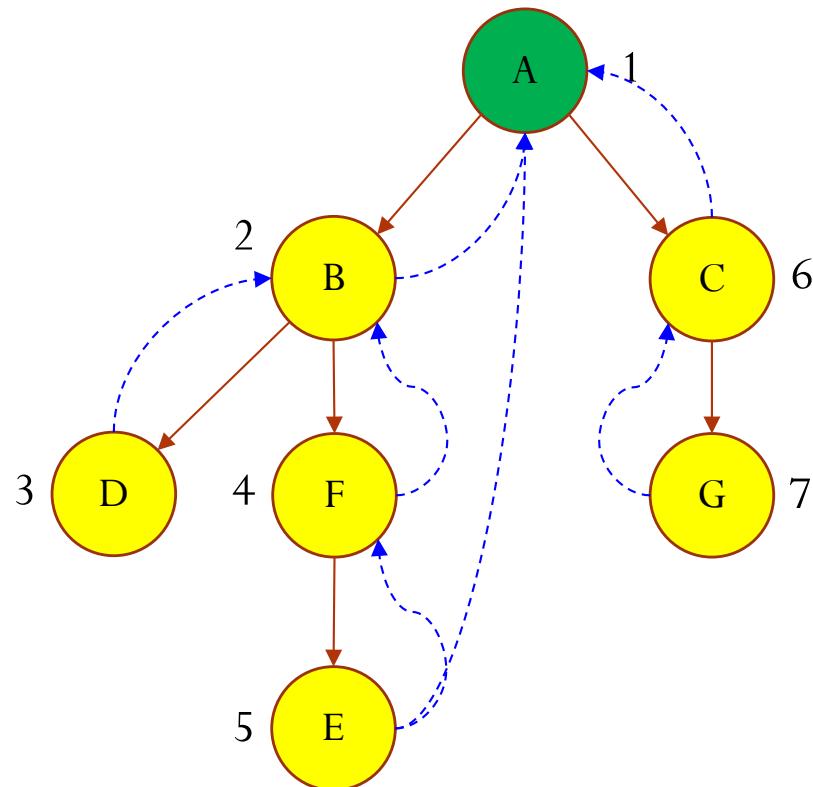
G đã xét xong =>  
quay về C



DFS(A)	
DFS(B)	
BỎ QUÁ A	
DFS(D)	
BỎ QUÁ B	
DFS(F)	
BỎ QUÁ B	
DFS(E)	
BỎ QUÁ A	
BỎ QUÁ F	
DFS(C)	
BỎ QUÁ A	
DFS(G)	
BỎ QUÁ C	
--->	

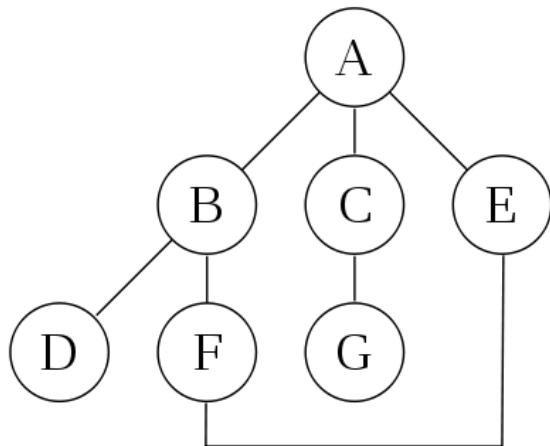


C đã xét xong =>  
quay về A

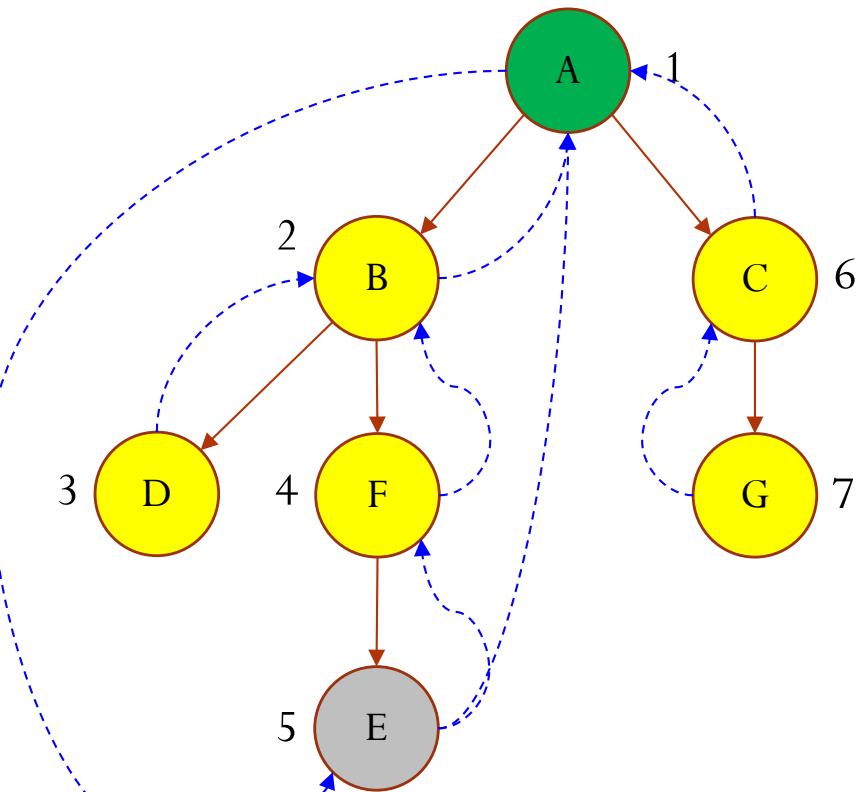


DFS(A)	
DFS(B)	BỎ QUA A
DFS(D)	BỎ QUA B
DFS(F)	BỎ QUA B
DFS(E)	BỎ QUA A
DFS(G)	BỎ QUA F
DFS(C)	BỎ QUA A
DFS(G)	BỎ QUA C

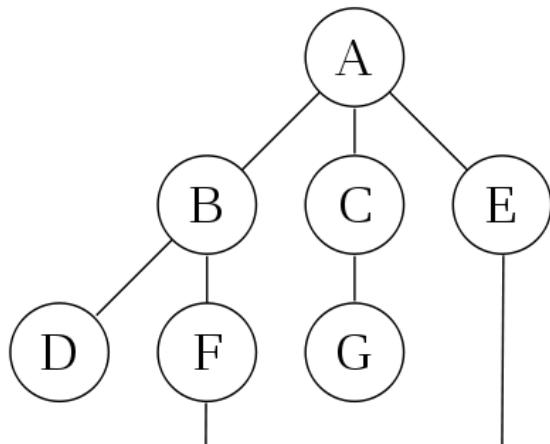
--->



E đã được duyệt =>  
bỏ qua

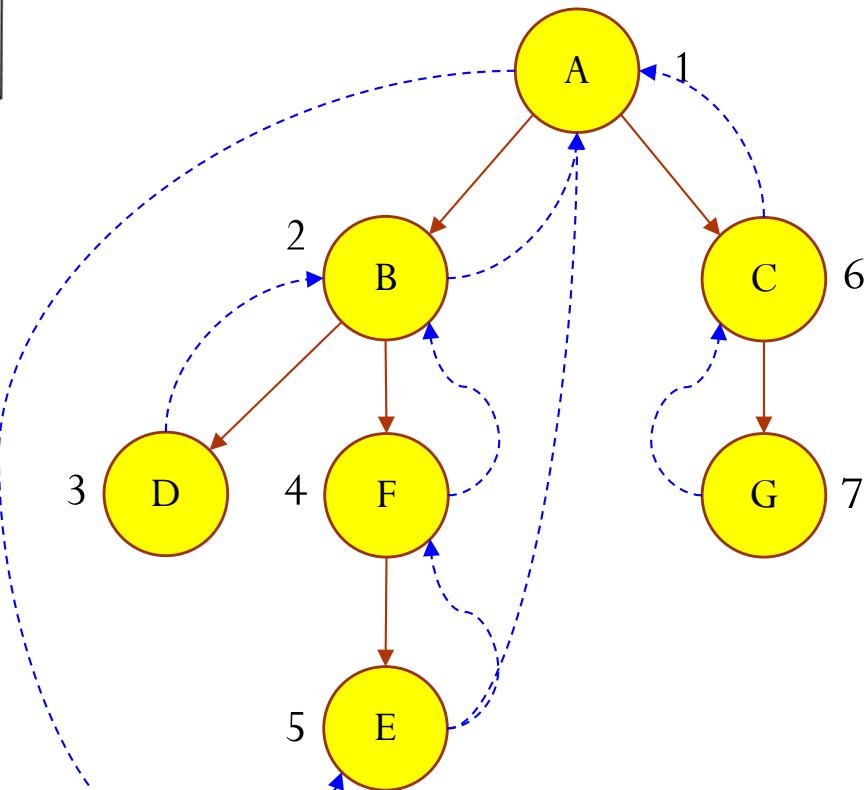


```
DFS(A)
DFS(B)
    BỎ qua A
DFS(D)
    BỎ qua B
DFS(F)
    BỎ qua B
DFS(E)
    BỎ qua A
    BỎ qua F
DFS(C)
    BỎ qua A
DFS(G)
    BỎ qua C
BỎ qua E
```



A đã xét xong  
=> Kết thúc

Thứ tự duyệt:  
A, B, D, F, E, C, G



DFS(A)	
DFS(B)	BỎ QUA A
DFS(D)	BỎ QUA B
DFS(F)	BỎ QUA B
DFS(E)	BỎ QUA A
DFS(G)	BỎ QUA F
DFS(C)	BỎ QUA A
DFS(G)	BỎ QUA C
DFS(E)	BỎ QUA E

# Duyệt theo chiều sâu

- Thứ tự duyệt của các đỉnh
  - Sử dụng stack: A, E, F, B, D, C, G
  - Đệ quy: A, B, D, F, E, C, G
- Để 2 phương pháp có thứ tự duyệt các đỉnh giống nhau, trong vòng lặp for của duyệt bằng stack
  - **for** các đỉnh kề v của u **do**
    - Đưa v vào **ngăn xếp**
  - Ta phải xét qua các đỉnh kề v của theo thứ tự **NGƯỢC LẠI** với thứ tự của vòng lặp trong thuật toán đệ quy

# Duyệt đồ thị

- Duyệt theo chiều sâu (đệ quy)
  - Phiên bản kiểm tra trong vòng lặp, phù hợp khi đồ thị không liên thông cần duyệt nhiều lần (*sẽ quay lại trong phần tính liên thông của đồ thị*)

```
void DFS(u) {  
    //Duyệt u: in u ra màn hình hoặc đánh số đỉnh u  
    Đánh dấu u đã duyệt  
    for (các đỉnh kề v của u) do  
        if (v chưa duyệt)  
            DFS(v); //gọi đệ quy duyệt các đỉnh kề v của u  
}
```

# Duyệt đồ thị

```
void DFS(u) {  
    if (u đã duyệt)  
        return;  
    //Duyệt u  
    Đánh dấu u đã duyệt  
  
    for (các đỉnh kề v của u)  
        DFS(v);  
}
```

```
void DFS(u) {  
  
    //Duyệt u  
    Đánh dấu u đã duyệt  
  
    for (các đỉnh kề v của u)  
        if (v chưa duyệt)  
            DFS(v);  
}
```