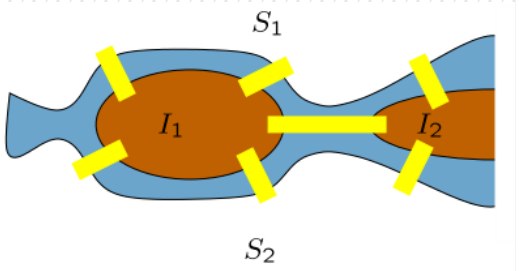
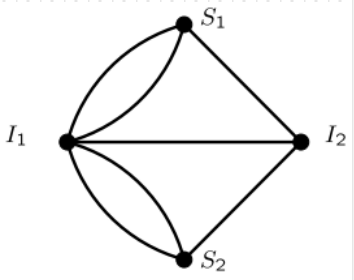
# LÝ THUYẾT ĐỒ THỊ

Phạm Nguyên Hoàng
BM. Khoa học máy tính, Khoa CNTT&TT
pnhoang@ctu.edu.vn




Cần Thơ, 2019

# 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

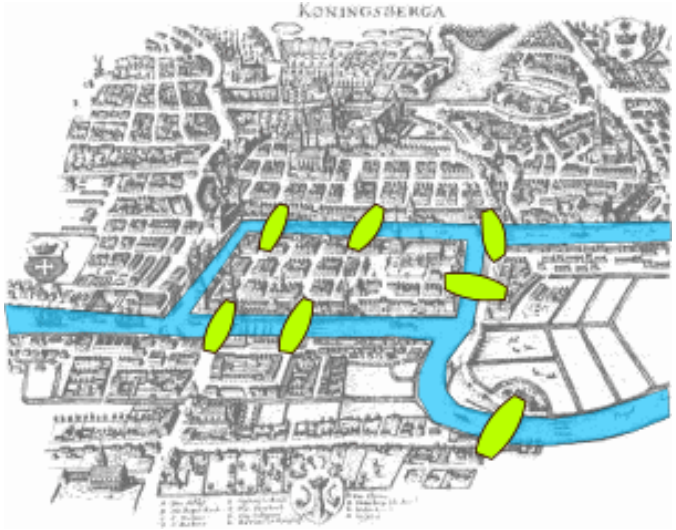
- Phần nâng cao
  1. Đồ thị Euler & ứng dụng
  2. Đồ thị Hamilton & ứng dụng
  3. Tìm đường đi ngắn nhất
  4. Xếp hạng đồ thị và bài toán tổ chức thi công
  5. Cây khung vô hướng có trọng lượng nhỏ nhất
  6. Cây khung có hướng có trọng lượng nhỏ nhất
  7. Luồng cực đại trên mạng

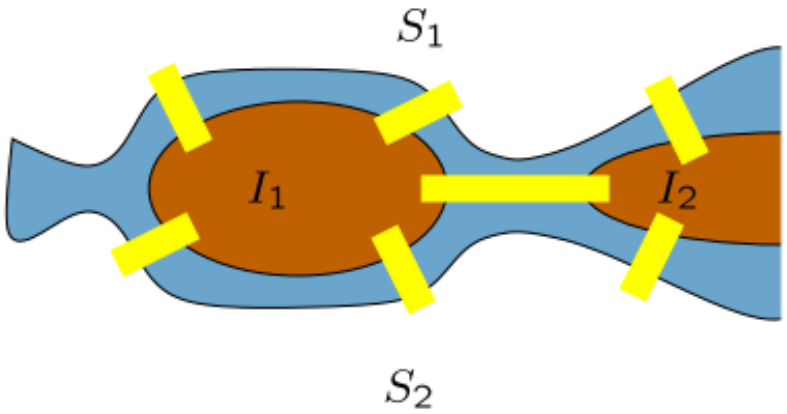
# Nội dung

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

# Giới thiệu

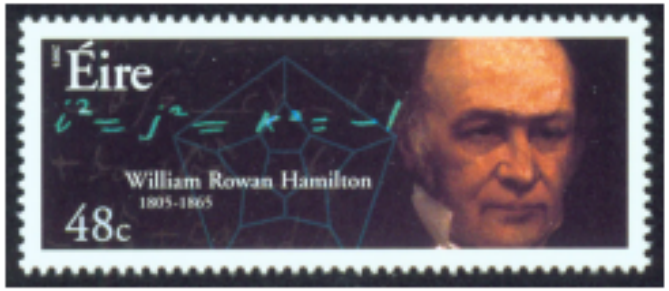
- Lý thuyết đồ thị (LTĐT, Graph Theory)
  - Ngành khoa học phát triển từ rất lâu
  - Có nhiều ứng dụng hiện đại
  - Ý tưởng xuất phát từ nhà toán học Thụy Sĩ Leonhoard Euler (1707 – 1783) vào thế kỷ XVIII
  - Bài toán 7 cây cầu ở Königsberg



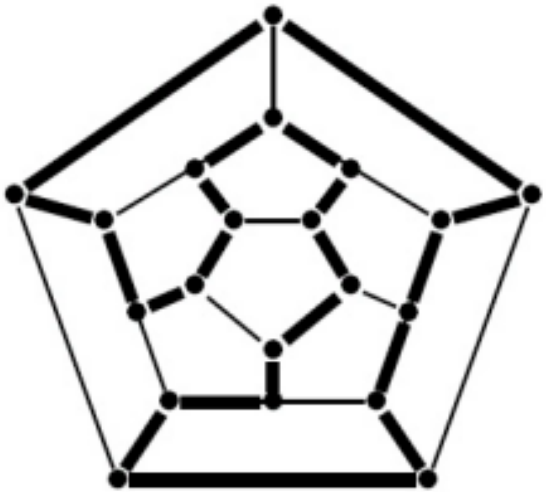



# Giới thiệu

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





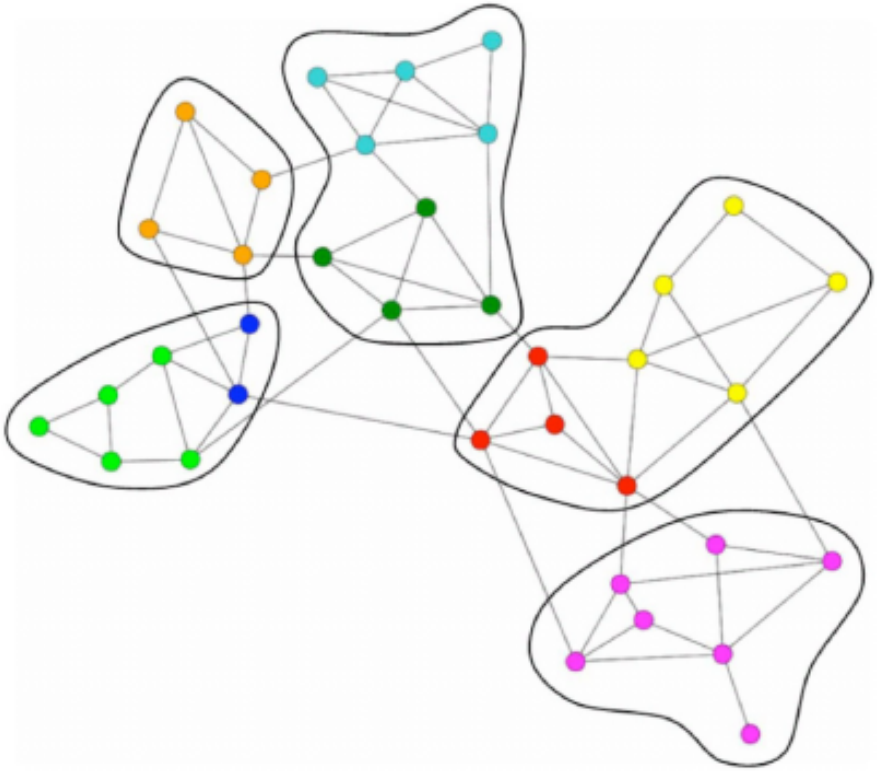


# Giới thiệu

- 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

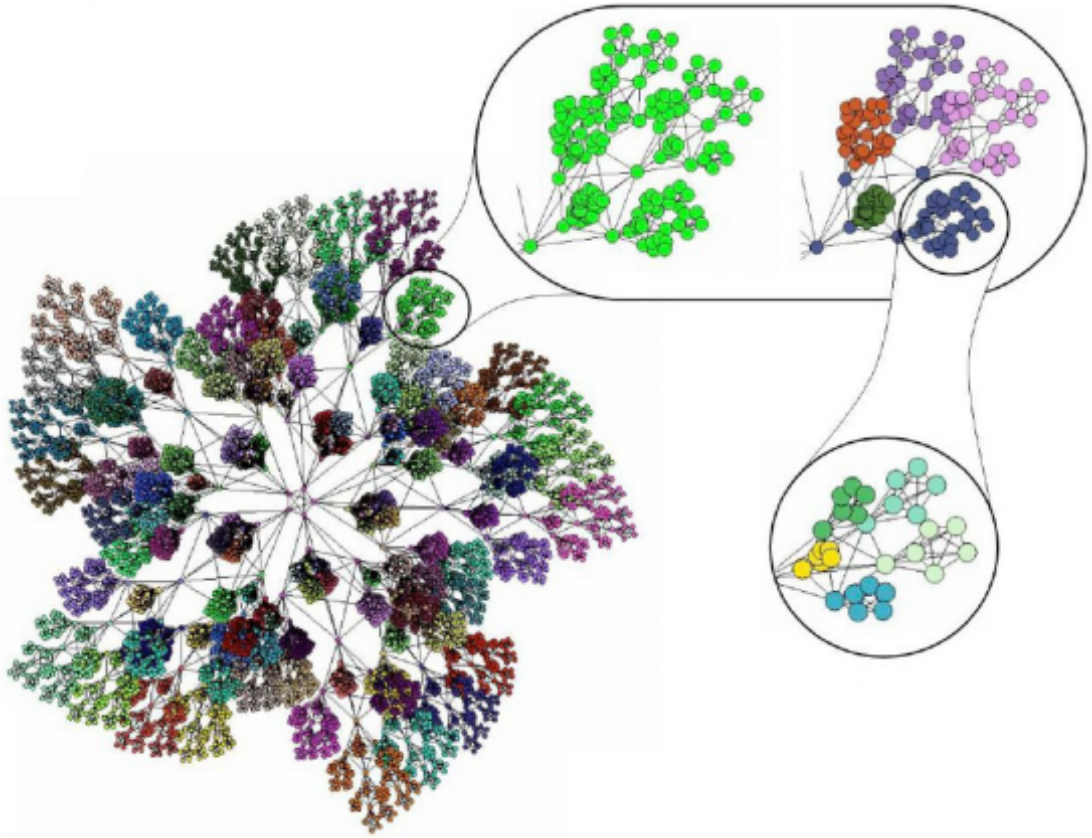
# Giới thiệu

- Tìm cộng đồng trên mạng



# Giới thiệu

- Kích thước có thể rất lớn



# Giới thiệu

graph theory

Search

About 8,640,000 results (0.15 seconds)

Web

Images

Maps

Videos

News

Shopping

Books

More

Show search tools

**Graph theory** - Wikipedia, the free encyclopedia
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

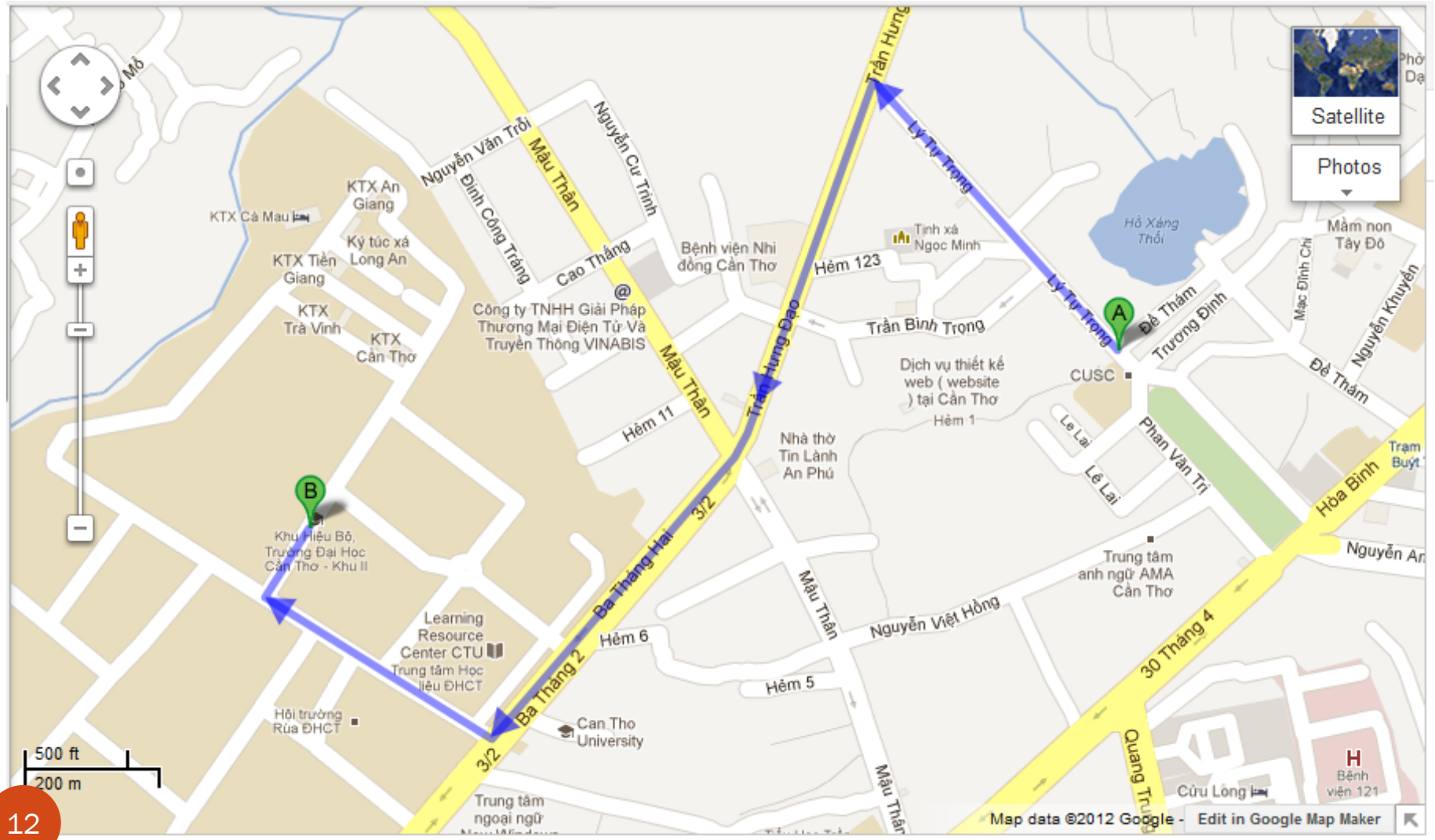
**Graph Theory** Tutorials
www.utm.edu/departments/math/**graph**/
This is the home page for a series of short interactive tutorials introducing the basic concepts of **graph theory**. There is not a great deal of theory here, we will just ...

**graph theory** -- **graph theory** textbooks and resources
www.**graphtheory**.com/
The website www.**graphtheory**.com is sponsored by the mathematical textbooks of Professor Jonathan Gross of Columbia University. It provides comprehensive ...

# Giới thiệu


Calculating
LIBERTY
JACKSON
94
FIND
MENU
MAP
SPEAK
GARMIN StreetPilot 2720

# Giới thiệu


Satellite
Photos
A
B
Lý Tự Trọng
Trần Hưng Đạo
Mậu Thân
Nguyễn Văn Trỗi
Nguyễn Cư Trinh
Đinh Công Tráng
Cao Thắng
Hẻm 123
Trần Bình Trọng
Hẻm 11
Hẻm 1
Hẻm 6
Hẻm 5
Ba Tháng Hai
3/2
Nguyễn Việt Hồng
30 Tháng 4
Hòa Bình
Phan Văn Trị
Lê Lai
Đề Thám
Trương Định
Nguyễn Khuyến
Mạc Đĩnh Chi
Quang Trung
KTX An Giang
KTX Cà Mau
KTX Tiền Giang
Ký túc xá Long An
KTX Trà Vinh
KTX Cần Thơ
Bệnh viện Nhi đồng Cần Thơ
Tịnh xá Ngọc Minh
Hồ Xáng Thổi
Công ty TNHH Giải Pháp Thương Mại Điện Tử Và Truyền Thông VINABIS
Dịch vụ thiết kế web ( website ) tại Cần Thơ
CUSC
Nhà thờ Tin Lành An Phú
Trung tâm anh ngữ AMA Cần Thơ
Mầm non Tây Đô
Trạm Buýt
Khu Hiệu Bộ, Trường Đại Học Cần Thơ - Khu II
Learning Resource Center CTU
Trung tâm Học liệu ĐHCT
Hội trường Rùa ĐHCT
Can Tho University
Trung tâm ngoại ngữ
Cửu Long
Bệnh viện 121
500 ft
200 m
Map data ©2012 Google -
Edit in Google Map Maker

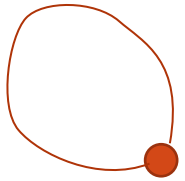
# Định nghĩa – Đồ thị

- **Đồ thị** (Graph) G là một bộ đôi <V, E>, trong đó:
  - V: **tập các đỉnh (**vertex set**)**, và
  - E: **tập các cung** (edge set), mỗi cung nối 2 đỉnh trong V

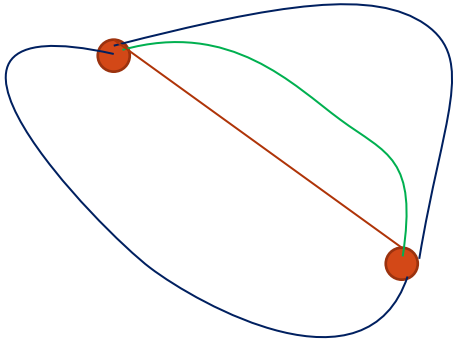
- Cho cung e nối 2 đỉnh x và y
  - Kí hiệu e = (x, y)
  - x, y được gọi là **đầu mút** (endpoints) của cung e
  - x và y được gọi là **kề** nhau (adjacent) hoặc **lân cận** của nhau (neighbours)
  - e được gọi là **liên thuộc** (incident) với x và y

# Định nghĩa – Khuyên + Đa cung

- Khuyên (loop): cung có hai đầu mút trùng nhau

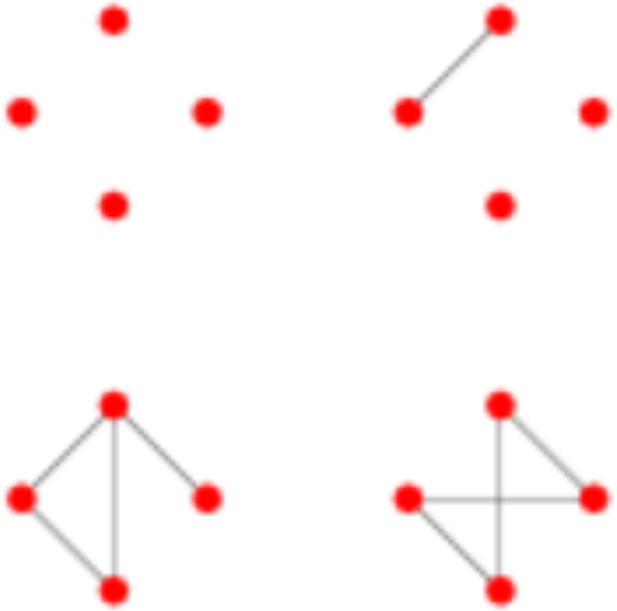


- Đa cung (multi edges): các cung có cùng chung đầu mút



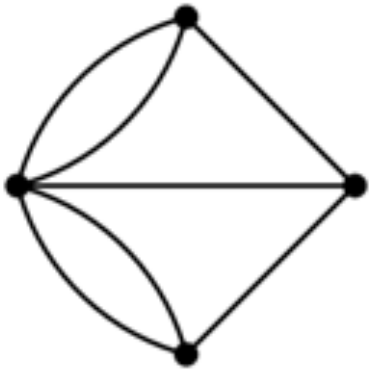
# Định nghĩa – Đơn đồ thị

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

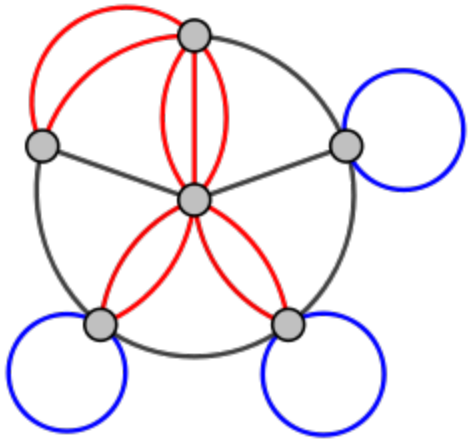


# Định nghĩa – Đa đồ thị + Giả đồ thị

- Đa đồ thị (Multigraphs)
  - 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

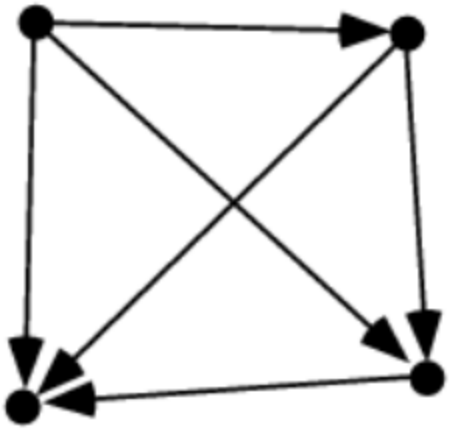
$$\sum_{v \in V} \deg(v) = 2|E| = 2e$$

# 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

# Định nghĩa – Đồ thị có hướng

- Đơn đồ thị có hướng
  - Cung có hướng (x, y) ≠ (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) hoặc (y, x).



- 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

# Bậc của đồ thị

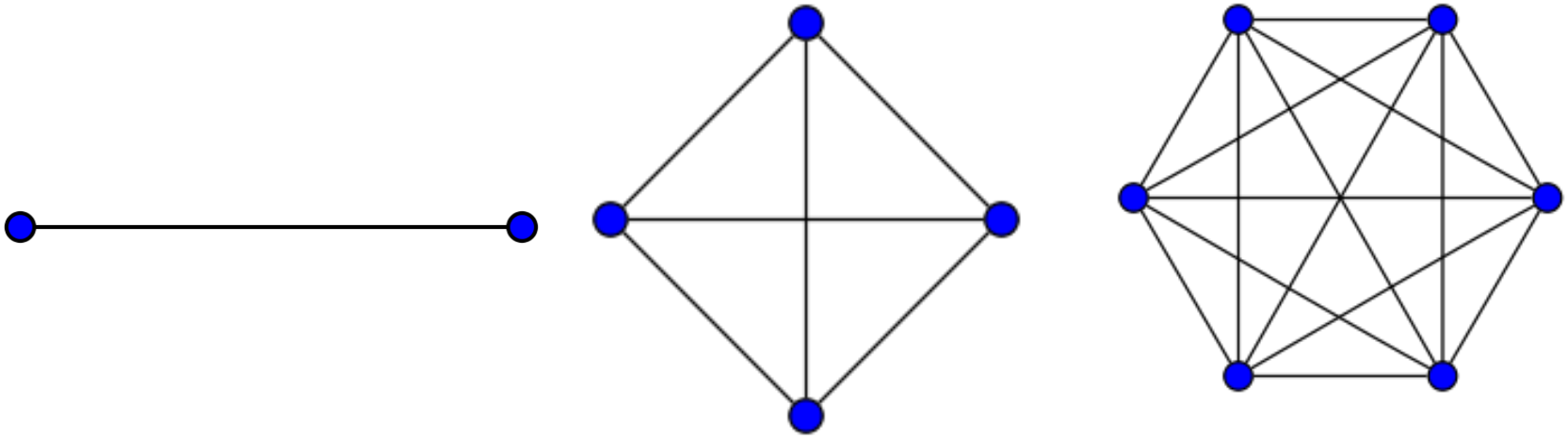
- Bậc vào của đỉnh (đồ thị có hướng)
  - Kí hiệu: $\deg^-(v)$ = số cung đi đến v
- Bậc ra của đỉnh (đồ thị có hướng)
  - Kí hiệu: $\deg^+(v)$ = số cung đi ra từ v
- Tính chất:
  - $\deg(v) = \deg^-(v) + \deg^+(v)$

- Định lý 3:
  - Cho G = <V, E> làm một đồ thị có hướng

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

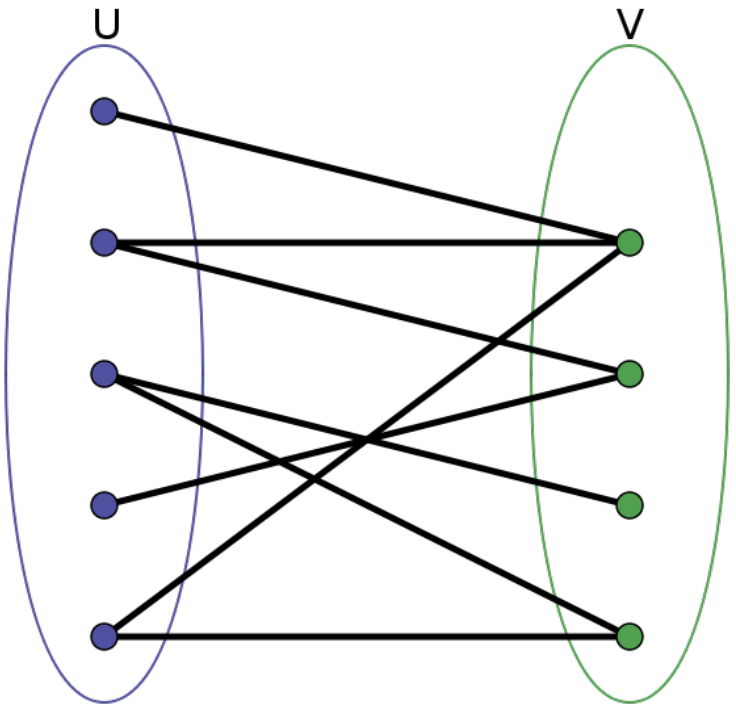
# Đồ thị đặc biệt

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



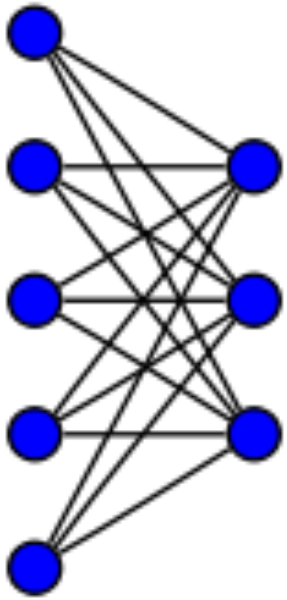
# Đồ thị đặc biệt

- 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

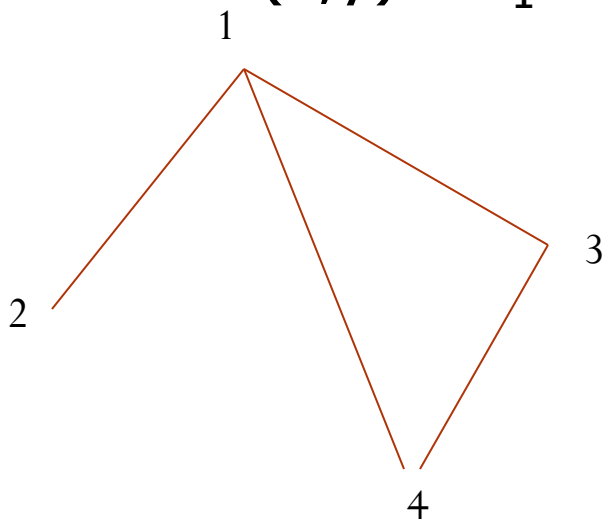
- Cho đồ thị G = <V, E>
  - Đồ thị con $G_S$ = <V, $E_s$>, $E_s \subset E$ được xây dựng từ G là đồ thị có được sau khi loại bỏ các cung không thuộc $E_s$.
  - Đồ thị con $G_S$ = <$V_s$, $E_s$>, $V_s \subset V$ , $E_s \subset E$ được xây dựng từ G là đồ thị có được sau khi loại bỏ các đỉnh không nằm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

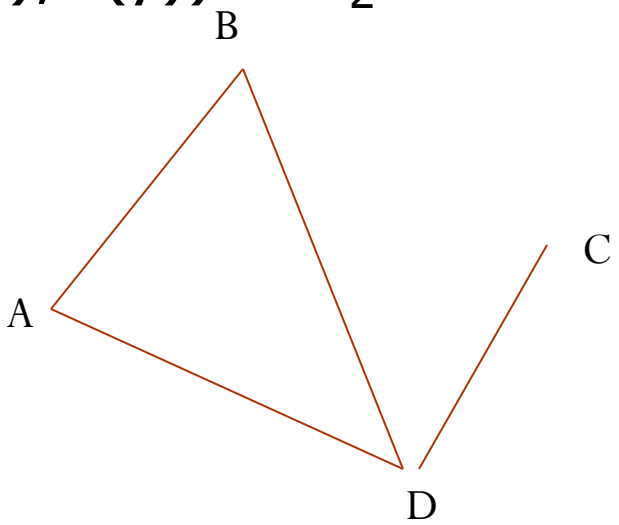
# Sự đẳng cấu của đồ thị

- 2 đồ thị $G_1 = <V_1, E_1>$ và $G_2 = <V_2, E_2>$ 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 $(x,y) \in E_1 \Leftrightarrow (f(x), f(y)) \in V_2$



# Bài tập

- Vẽ các đồ thị sau

a) $V = \{u, v, w, x\}, E = \{uv, vw, wx, vx\}$

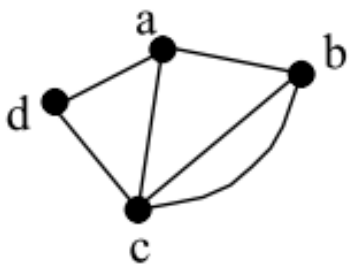
b) $V = \{1, 2, 3, 4, 5, 6, 7, 8\}, E = \{12, 22, 23, 34, 35, 67, 68, 78\}$

c) $V = \{n, p, q, r, s, t\}, E = \{np, nq, nt, rs, rt, st, pq\}$

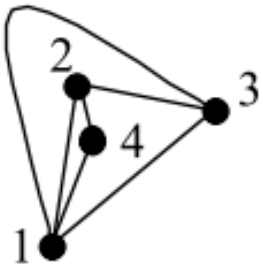
- Với mỗi đồ thị xét xem nó có phải là đơn đồ thị không ? Tại sao ?

# Bài tập

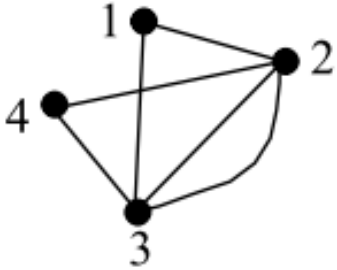
- Trong các đồ thị B, C, D (các) đồ thị nào đẳng cấu với đồ thị A. Nếu đẳng cấu tìm song ánh tương ứng



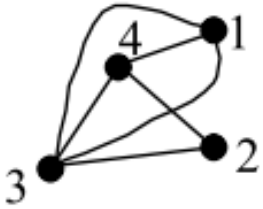
A



B



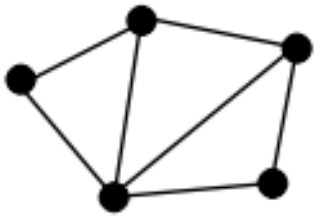
C



D

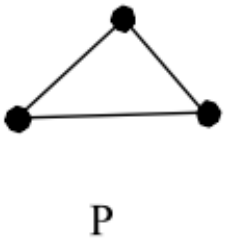
# Bài tập

- Cho đồ thị G như hình vẽ

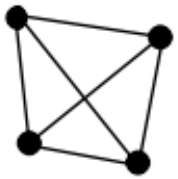


G

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

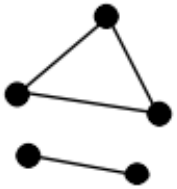


P

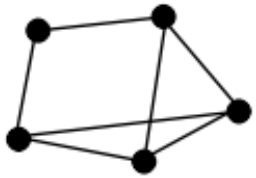


Q

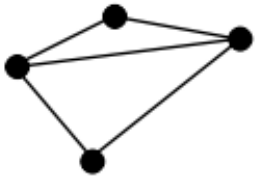
R



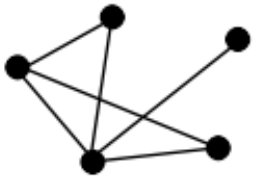
S



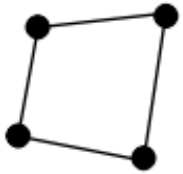
T



U



V



W

# Biểu diễn đồ thị trên máy tính

- Cấu trúc dữ liệu đồ thị
  - Danh sách cung
  - Ma trận kề (đỉnh – đỉnh)
  - Ma trận đỉnh cung
  - Danh sách kề (danh sách con/danh sách cha)
- 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
  - Sử dụng hàng đợi
  - Các đỉnh trong hàng đợi là các đỉnh *đã duyệt* nhưng *chưa xét các đỉnh kề của nó*

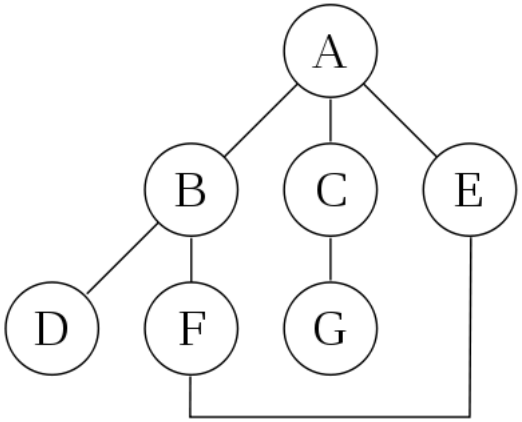
- Giải thuật:
  - Đưa 1 đỉnh **s** bất kỳ vào hàng đợi (vd: đỉnh 1)
  - **Duyệt s (vd: in s ra màn hình)**
  - *Đánh dấu **s** đã được duyệt*
  - **while** hàng đợi chưa rỗng **do**
    - Lấy đỉnh ở đầu hàng đợi ra => gọi là đỉnh u
    - **for** các đỉnh kề v của u **do**
      - **if** (v chưa được duyệt)
        - **Duyệt v (vd: in v ra màn hình)**
        - *Đánh dấu v đã duyệt*
        - Đưa v vào hàng đợi

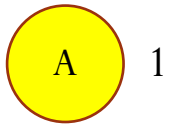
# Duyệt đồ thị

- Duyệt theo chiều rộng
  - Sử dụng hàng đợi
  - Các đỉnh trong hàng đợi là các đỉnh *đã duyệt* nhưng *chưa xét các đỉnh kề của nó*

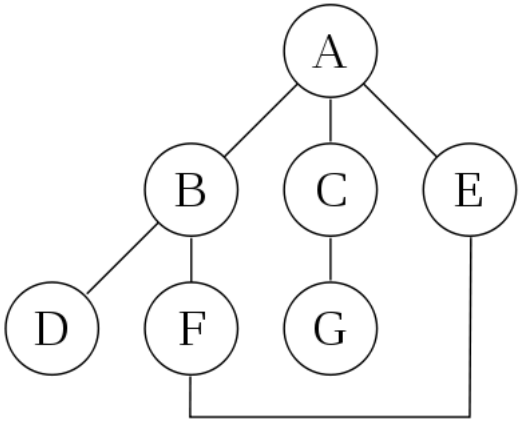
- Giải thuật: (PHIÊN BẢN KHÁC)
  - Đưa 1 đỉnh **s** bất kỳ vào hàng đợi (vd: đỉnh 1)

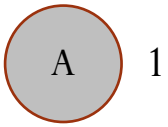
  - *Đánh dấu **s** đã được duyệt*
  - **while** hàng đợi chưa rỗng **do**
    - Lấy đỉnh ở đầu hàng đợi ra => gọi là đỉnh u
    - **Duyệt u (vd: in u ra màn hình)**
    - **for** các đỉnh kề v của u **do**
      - **if** (v chưa được duyệt)
        - *Đánh dấu v đã xét*
        - Đưa v vào hàng đợi



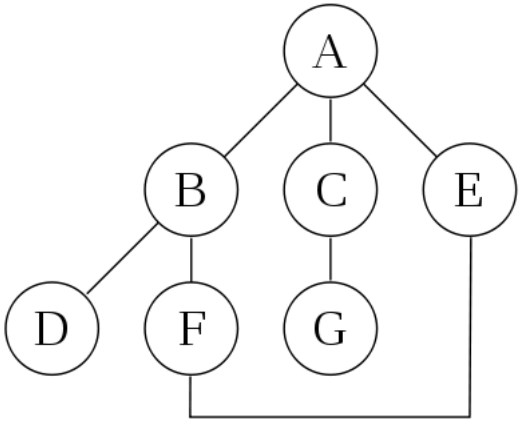

1

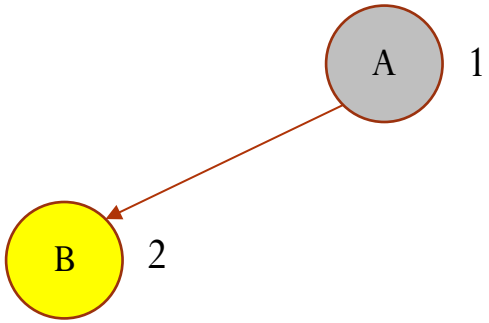


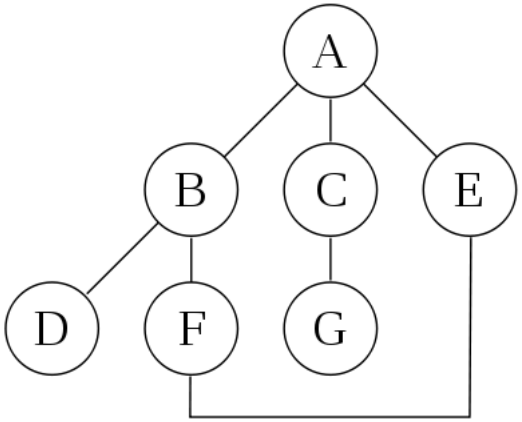



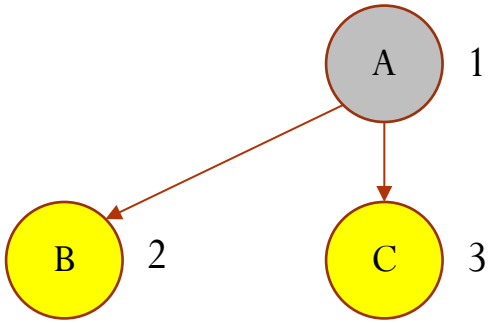

1

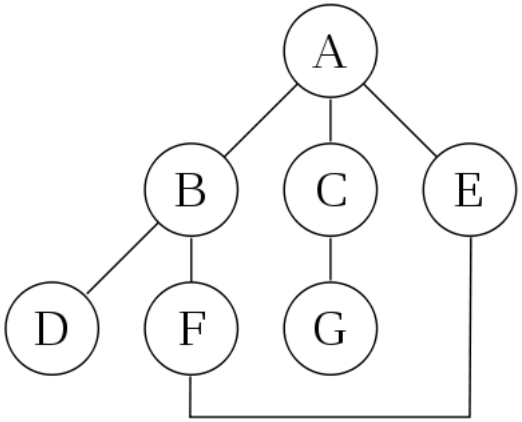
Lấy A ra khỏi hàng đợi

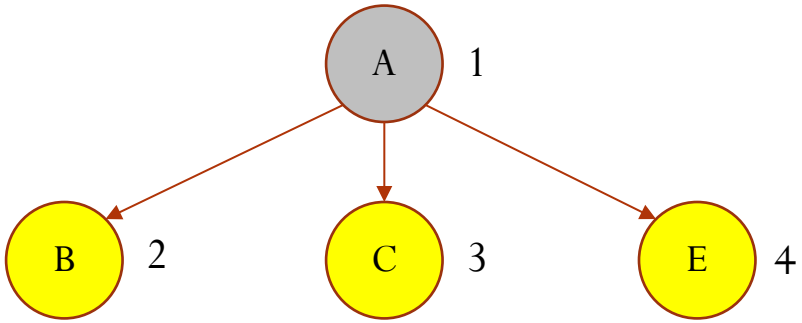

A
B
C
E
D
F
G

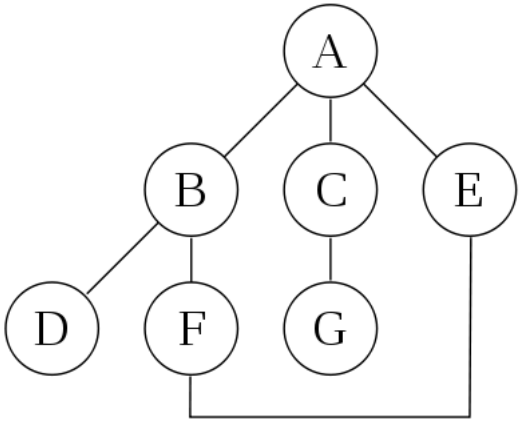

A
1
B
2


A
B
C
E
D
F
G

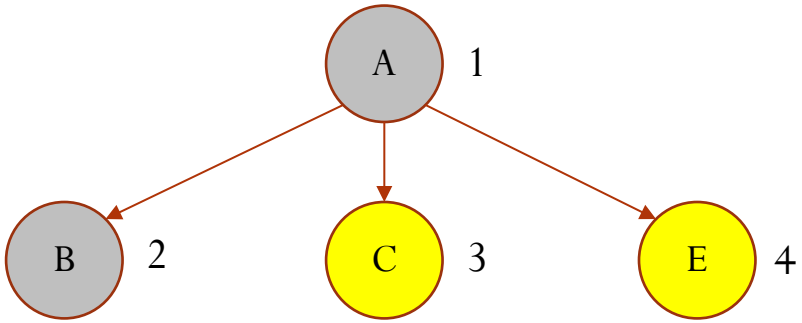

A
1
B
2
C
3

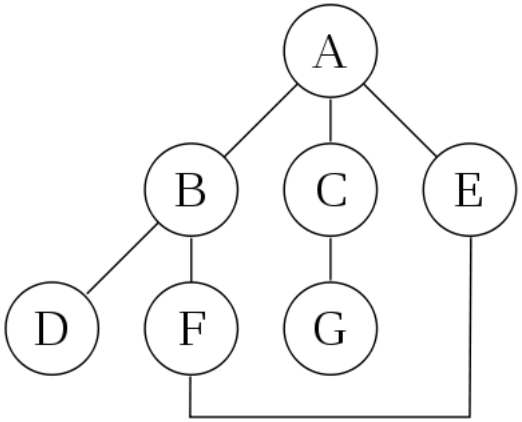

A
B
C
E
D
F
G


A
1
B
2
C
3
E
4

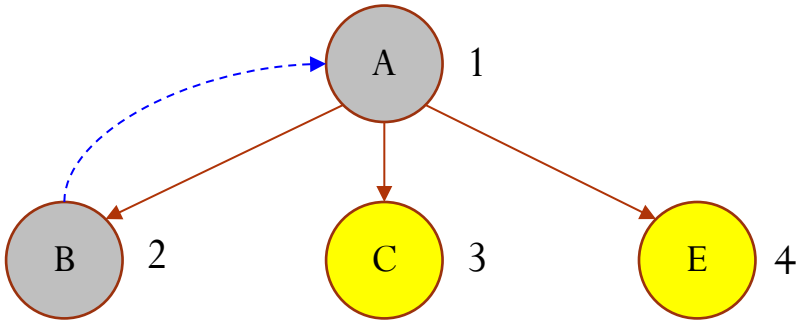


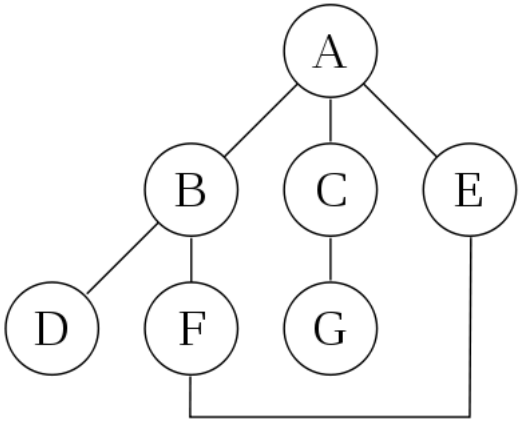

Lấy B ra khỏi hàng đợi

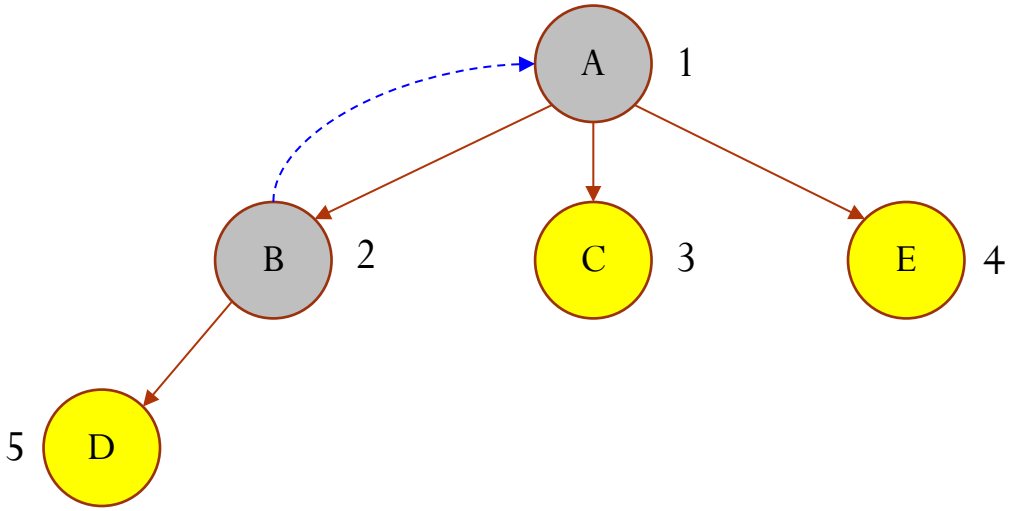


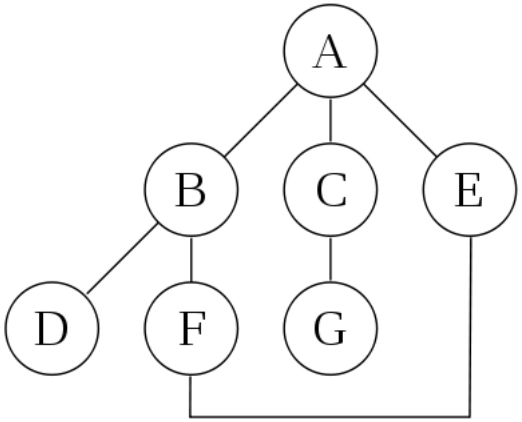


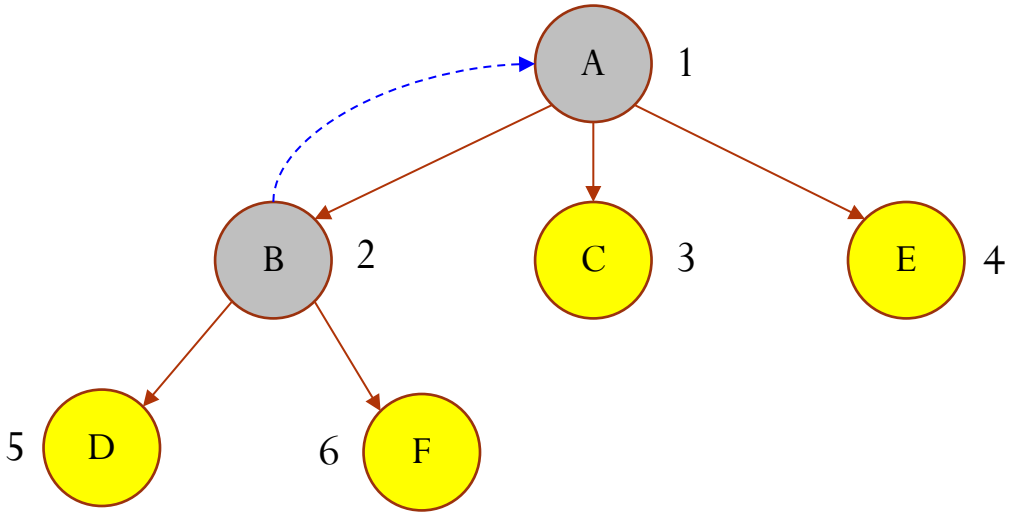

A đã được duyệt

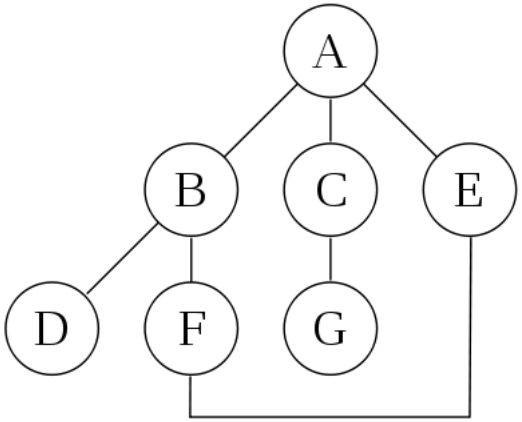



A
B
C
E
D
F
G

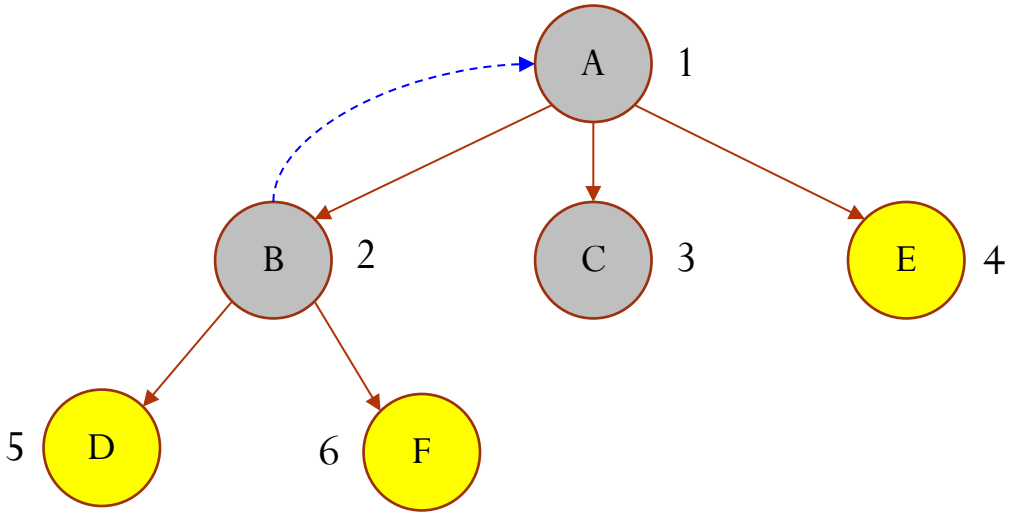

A
1
B
2
C
3
E
4
5
D

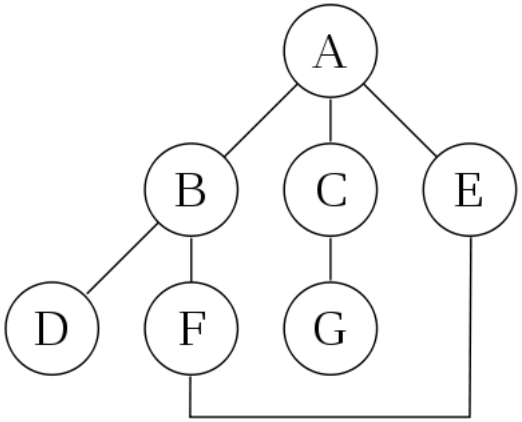

A
B
C
E
D
F
G


A
1
B
2
C
3
E
4
5
D
6
F

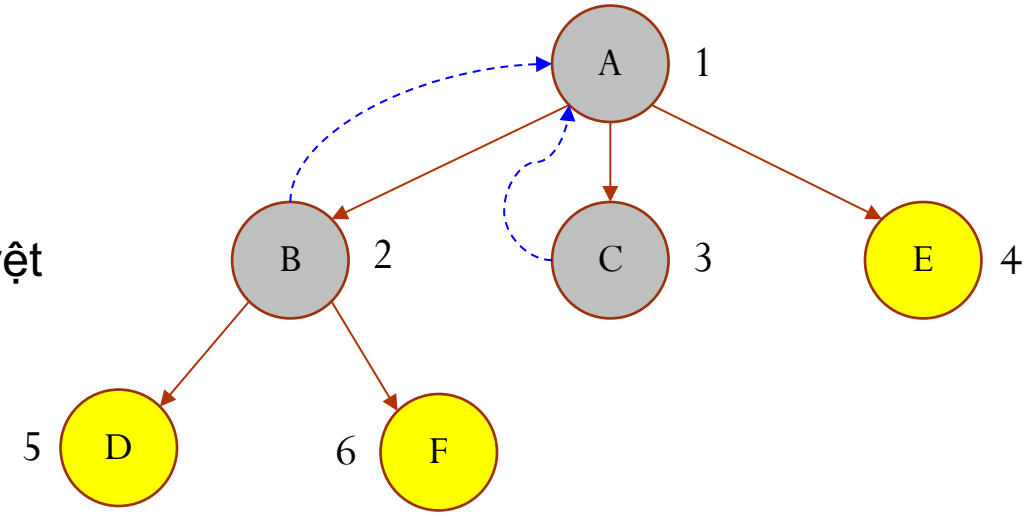


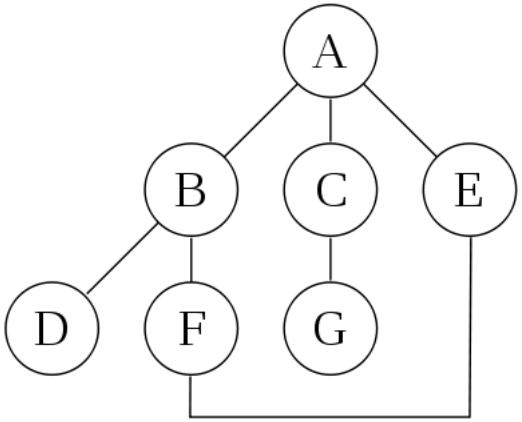

Lấy C ra khỏi hàng đợi

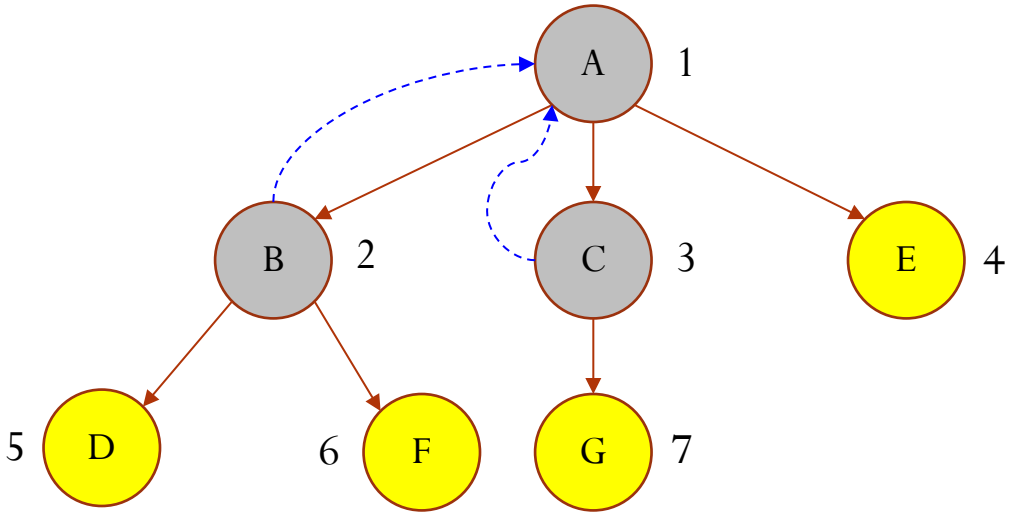


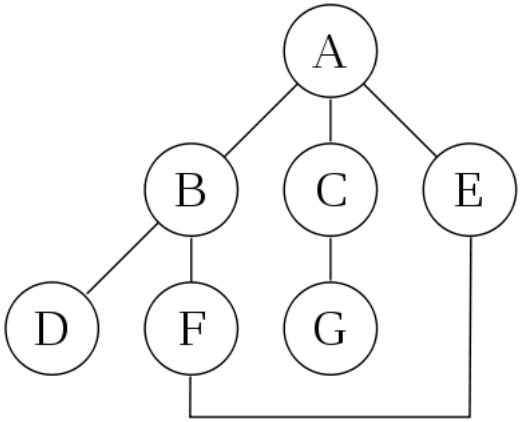



A đã được duyệt

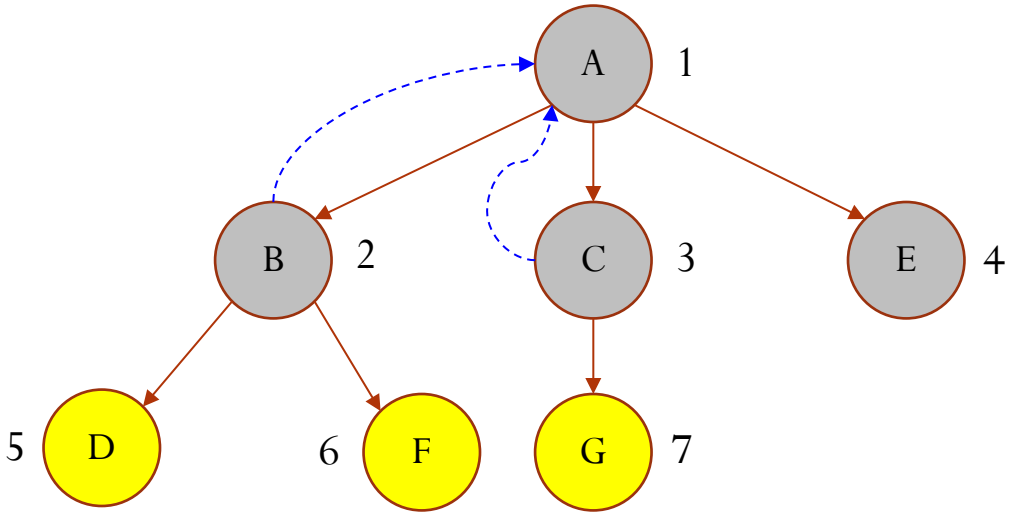


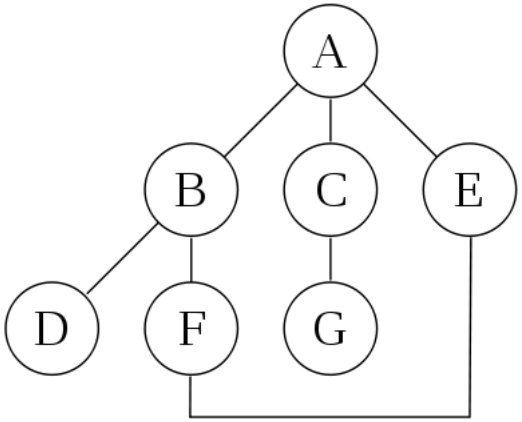

A
B
C
E
D
F
G


A
1
B
2
C
3
E
4
5
D
6
F
G
7

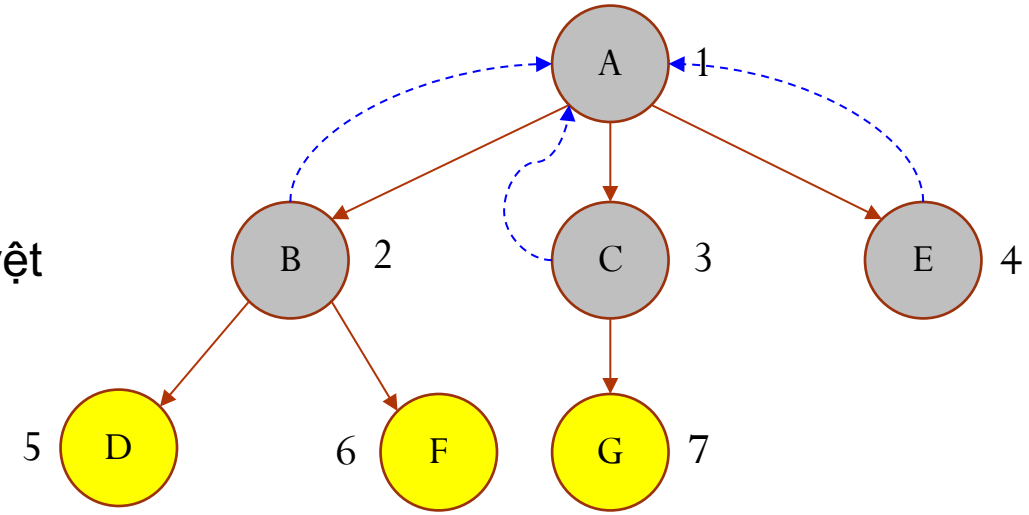


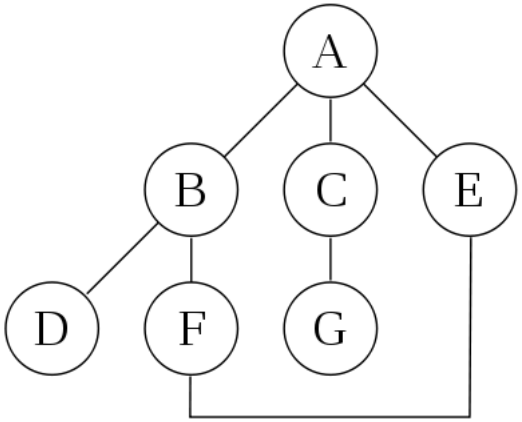

Lấy E ra khỏi hàng đợi



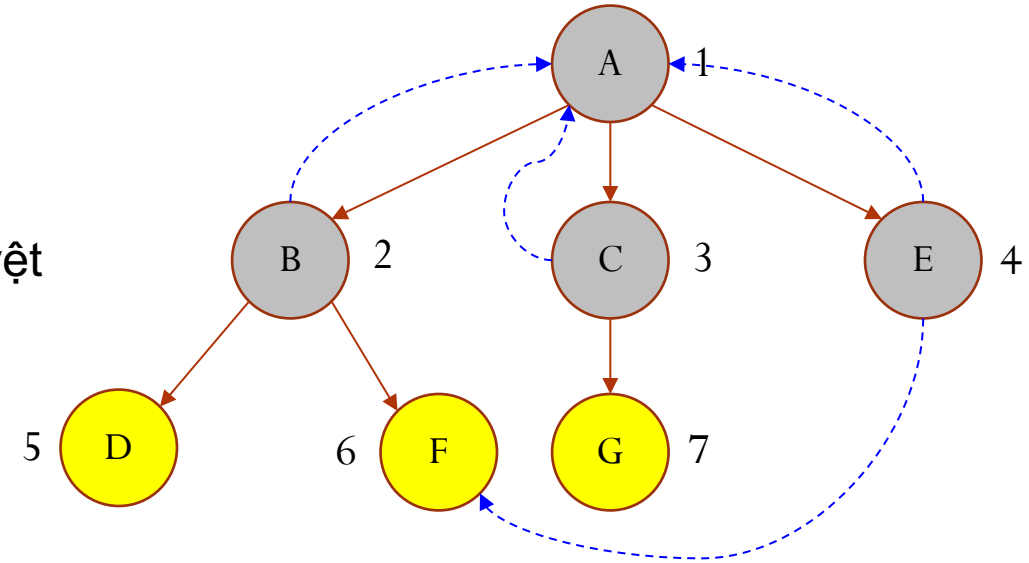


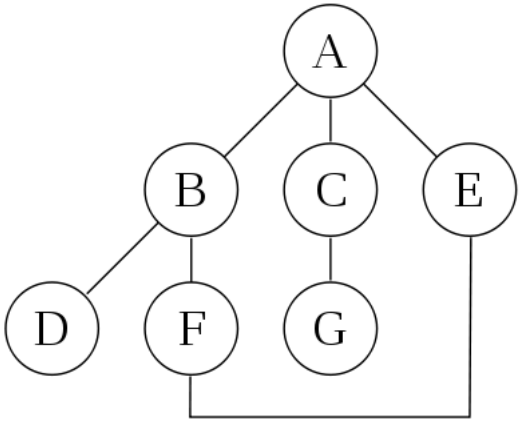

A đã được duyệt



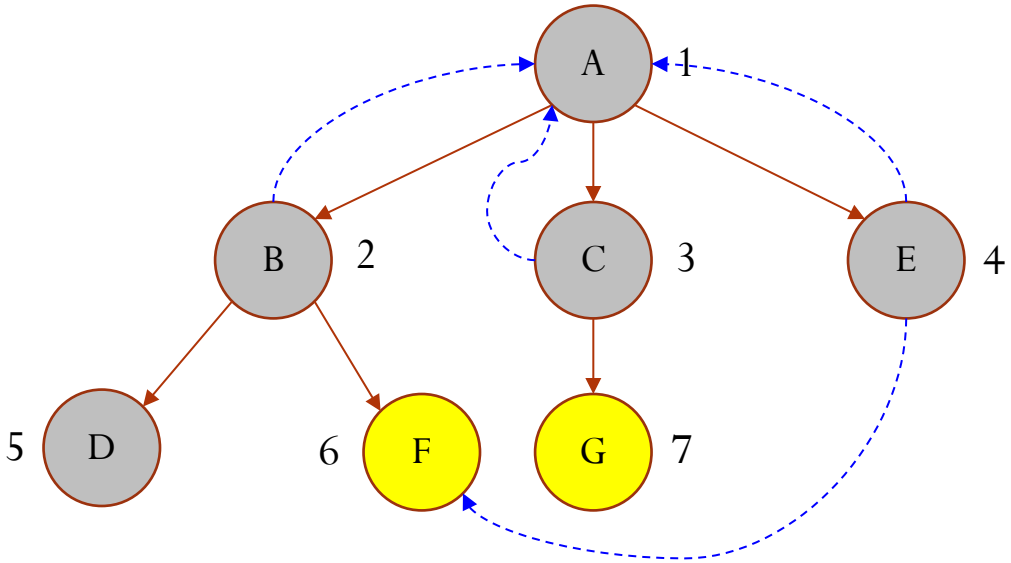


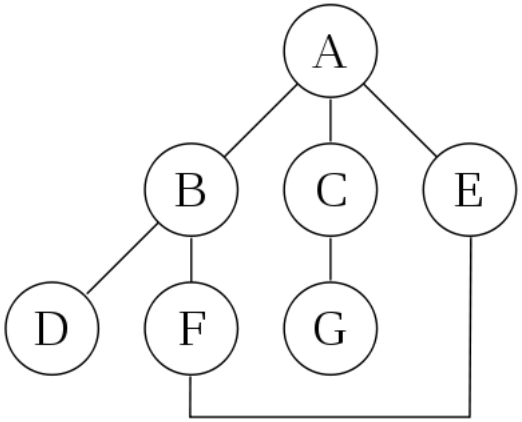

F đã được duyệt



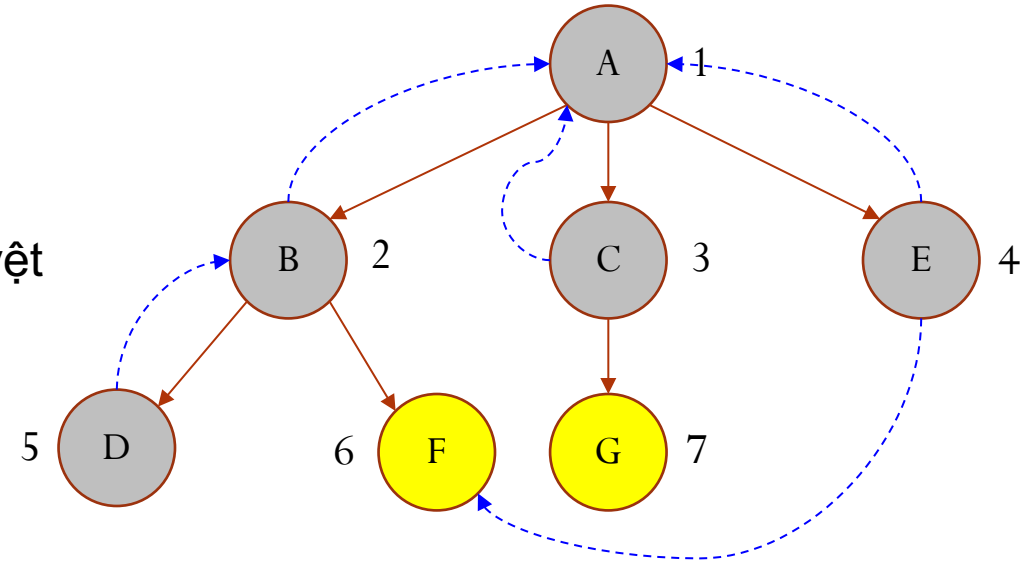


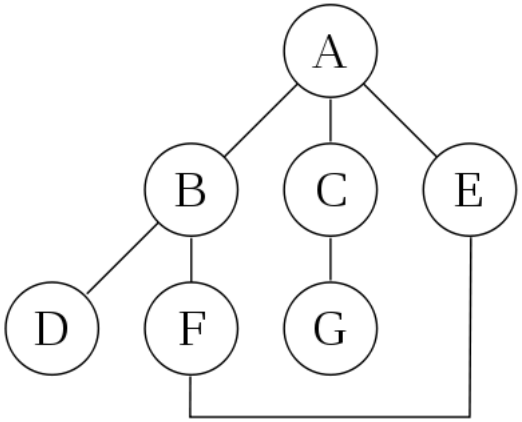

Lấy D ra khỏi hàng đợi



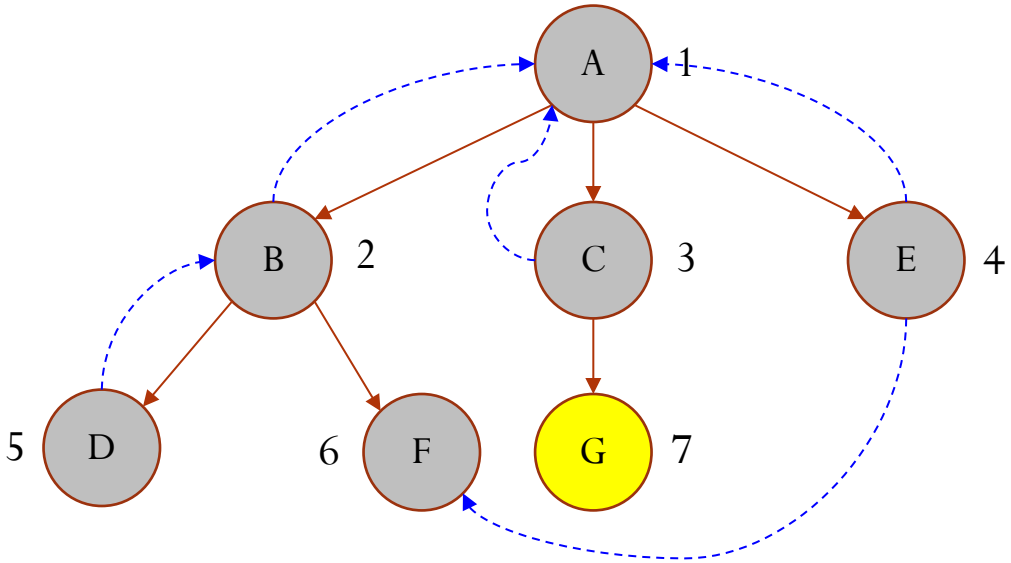


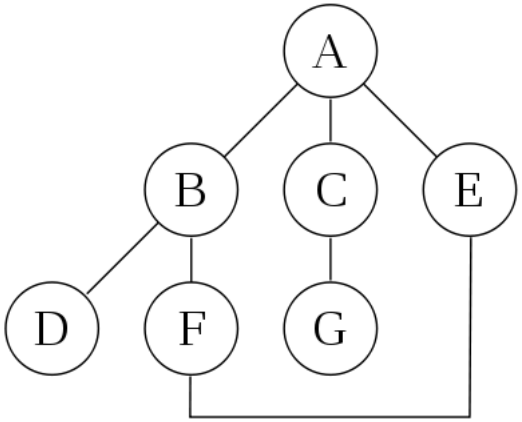

B đã được duyệt



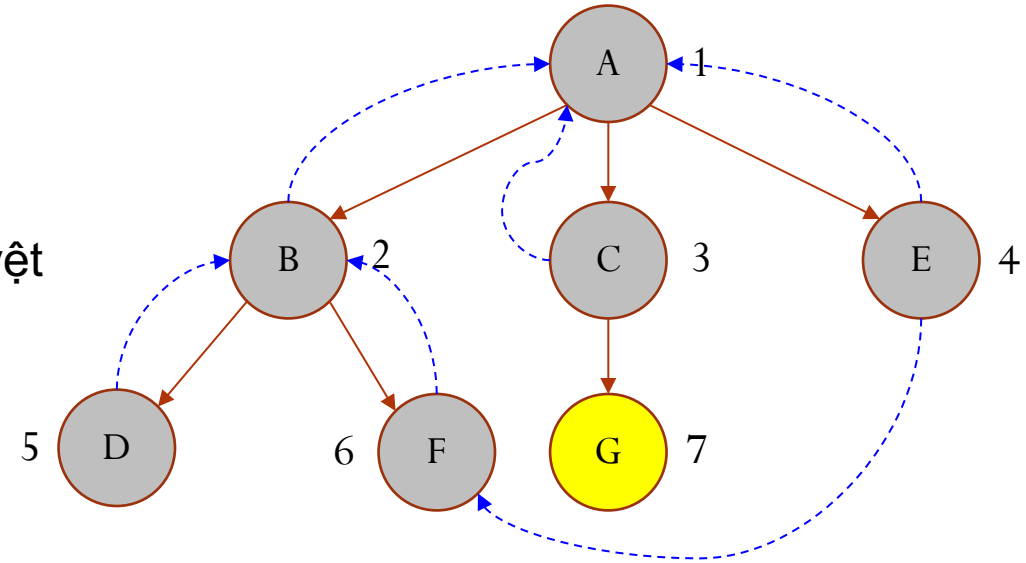


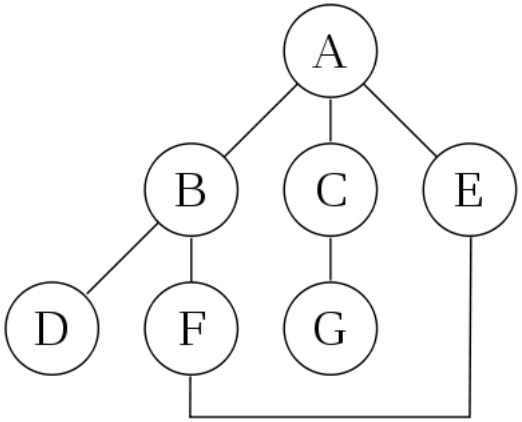

Lấy F ra khỏi hàng đợi



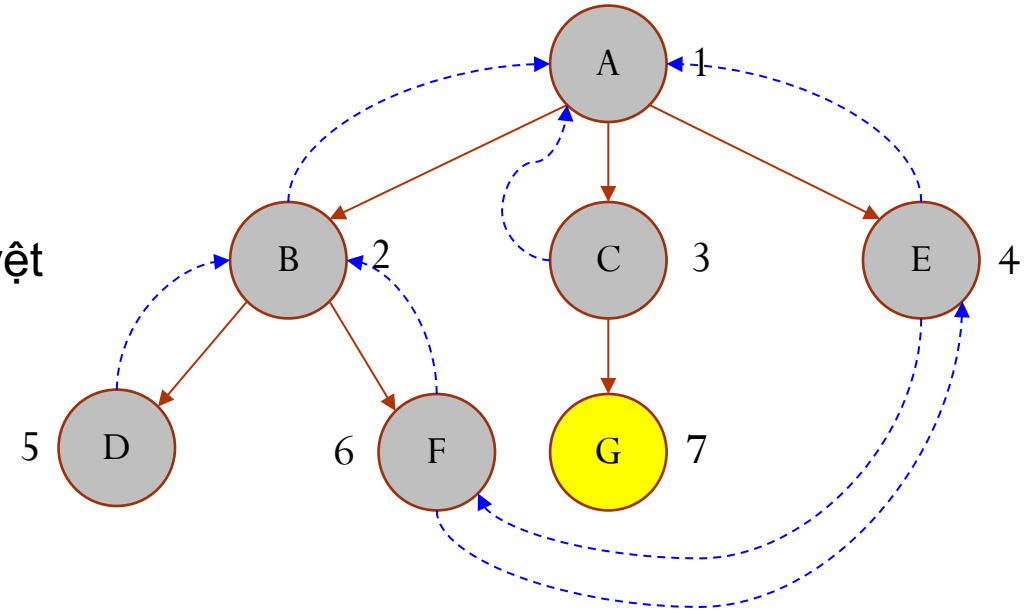


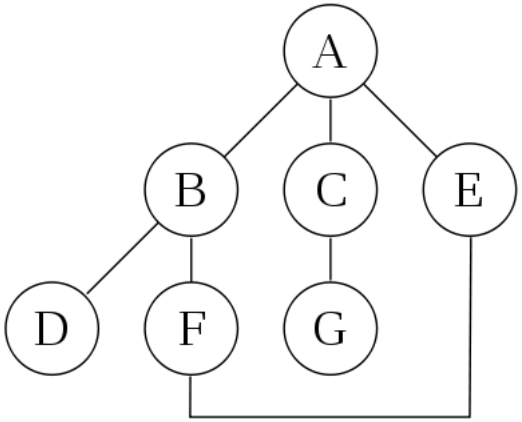

B đã được duyệt



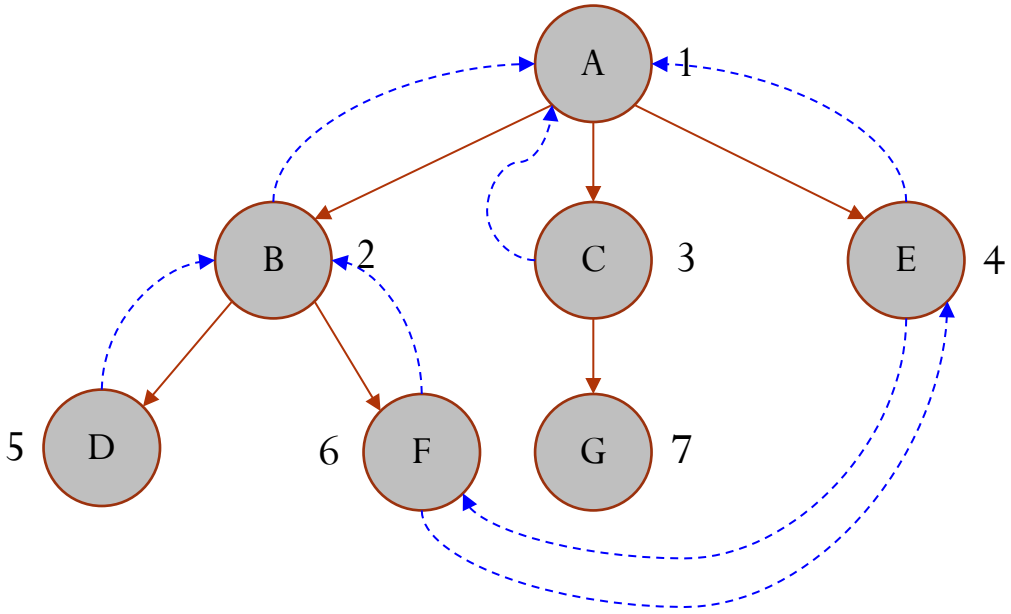


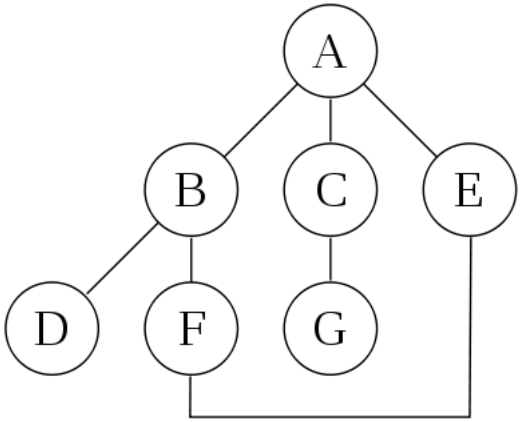

E đã được duyệt



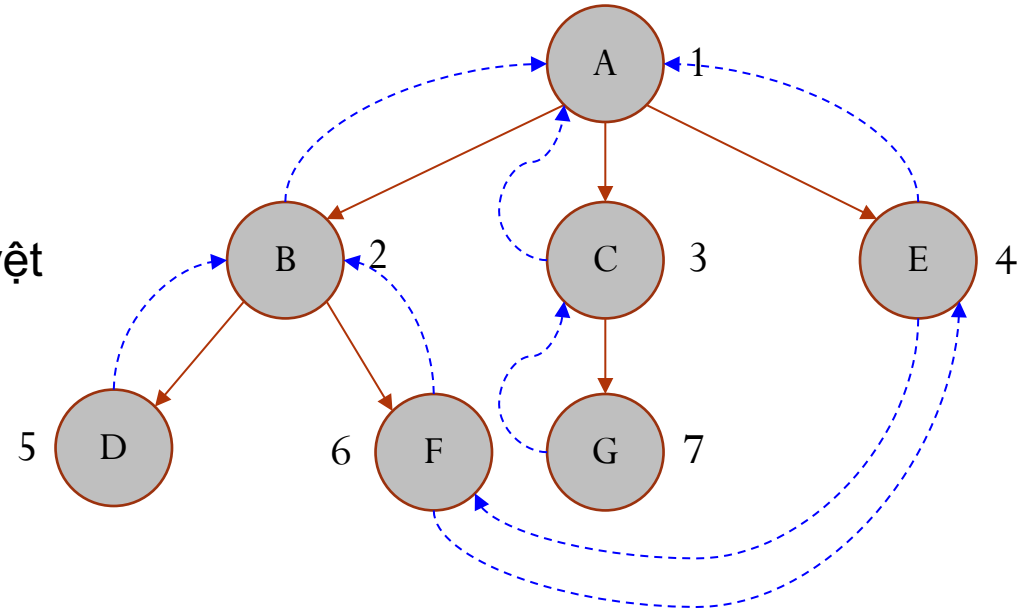


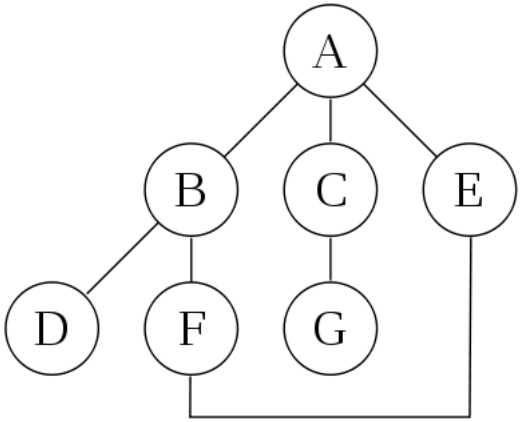

Lấy G ra khỏi hàng đợi

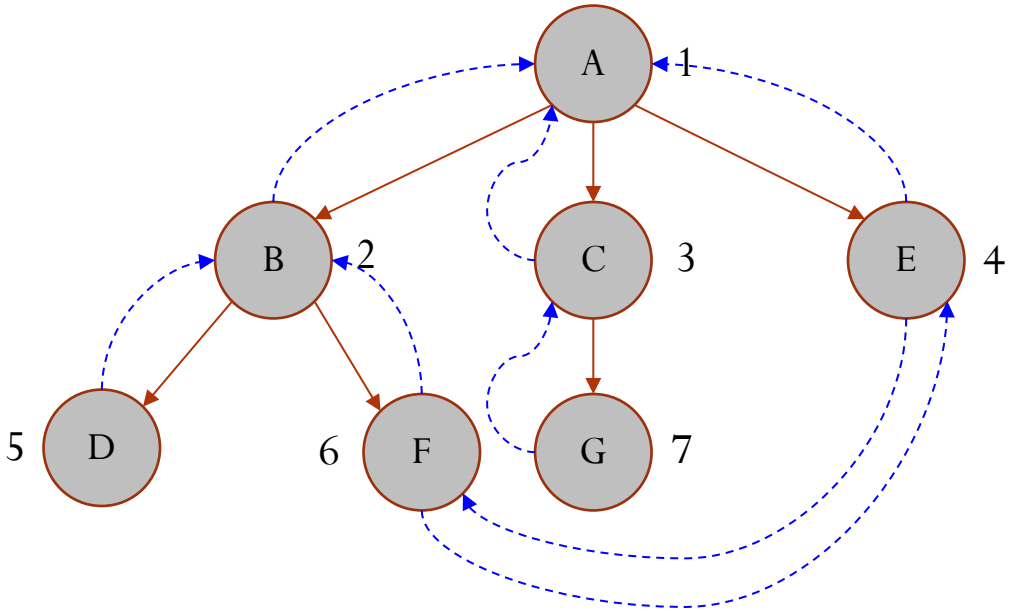





C đã được duyệt





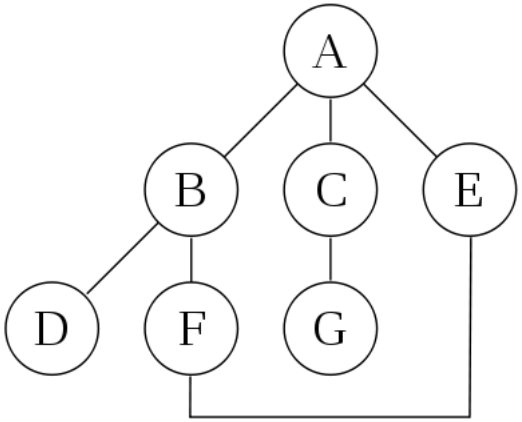

Thứ tự duyệt:
A, B, C, E, D, F, G

Hàng đợi rỗng
=> Kết thúc

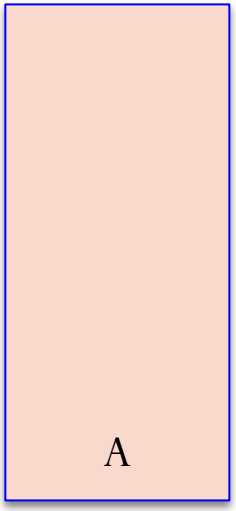
# Duyệt đồ thị

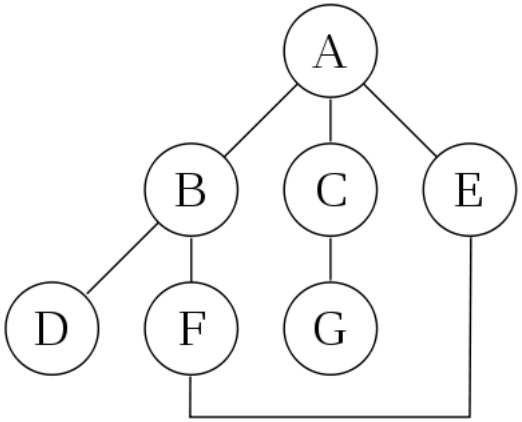
- Duyệt theo chiều sâu
  - Sử dụng ngăn xếp
  - Các đỉnh trong ngăn xếp các đỉnh *sẽ được duyệt.* Một đỉnh có thể có mặt *nhiều lần* trong ngăn xếp.

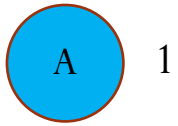
- Giải thuật:
  - Đưa 1 đỉnh s bất kỳ vào ngăn xếp (vd: đỉnh 1)
  - **while** ngăn xếp chưa rỗng **do**
    - Lấy đỉnh ở đầu ngăn xếp ra => gọi là đỉnh u
    - **if** (u đã duyệt) **continue**; *// bỏ qua*
    - **Duyệt u (vd: in u ra màn hình)**
    - *Đánh dấu u đã duyệt*
    - **for** các đỉnh kề v của u **do**
      - Đưa v vào ngăn xếp



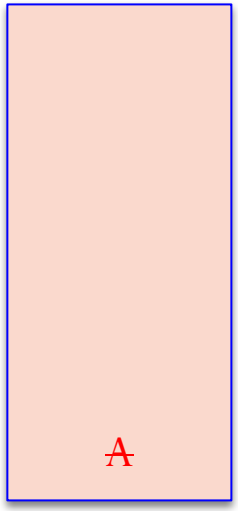

Push A vào stack



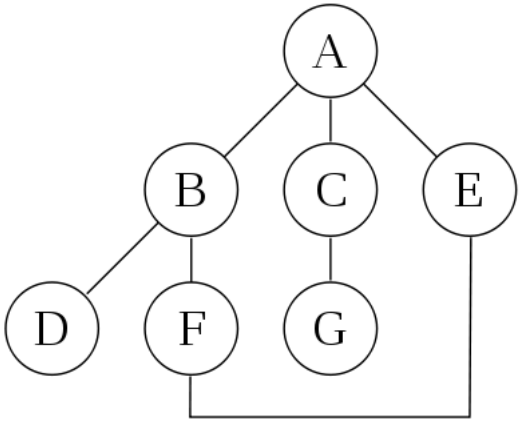


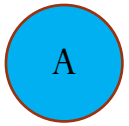




1

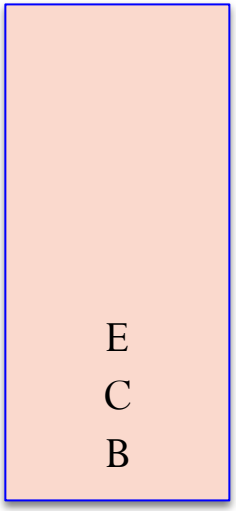



Lấy A ra khỏi stack
Đánh dấu A đã duyệt

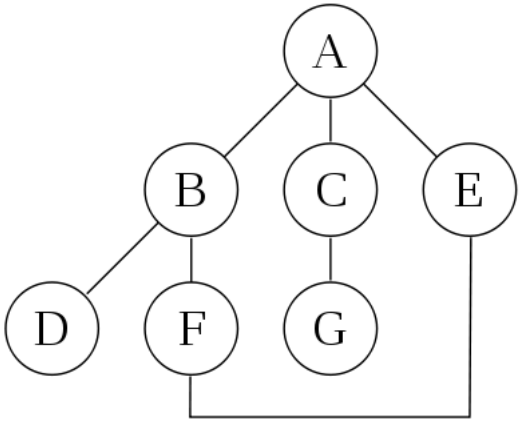


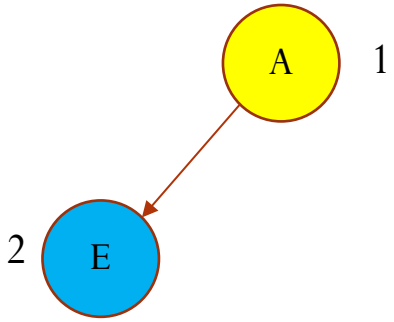



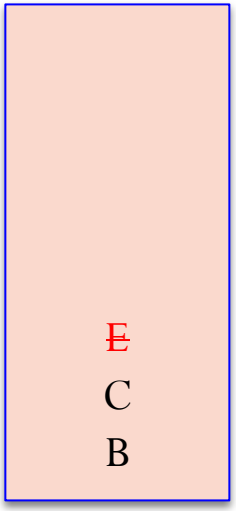

1



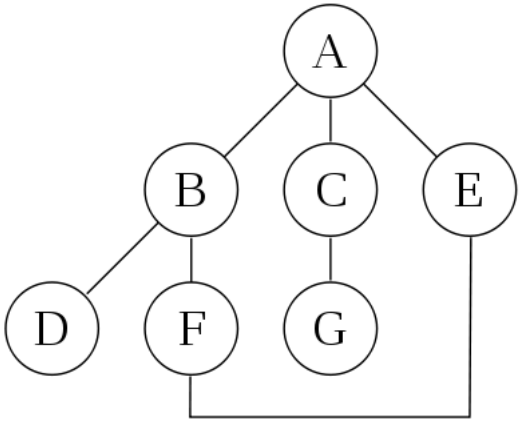

Lần lượt push B,
C, E vào stack
(chú ý thứ tự)



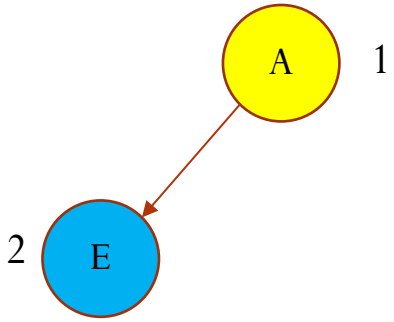



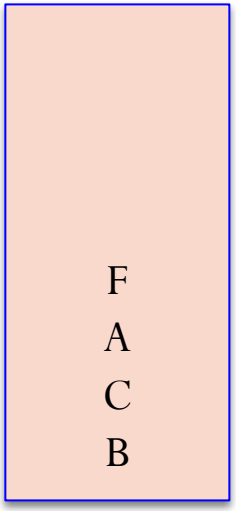



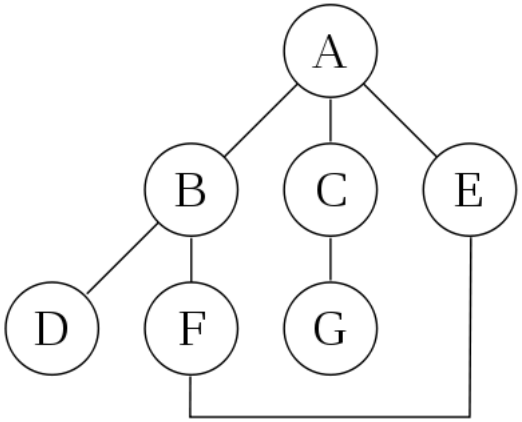

Lấy E ra khỏi stack
Đánh dấu E đã duyệt



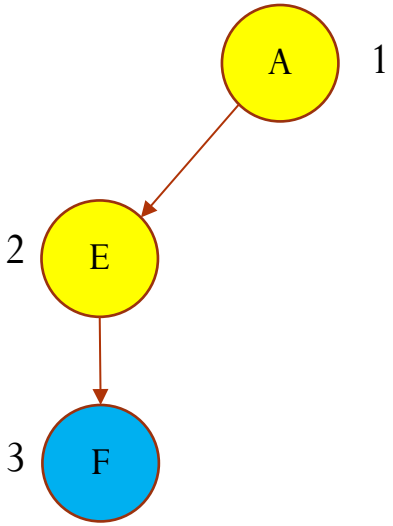

Push A, F vào stack



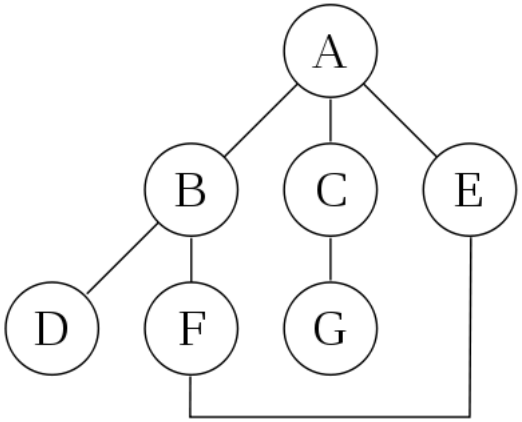





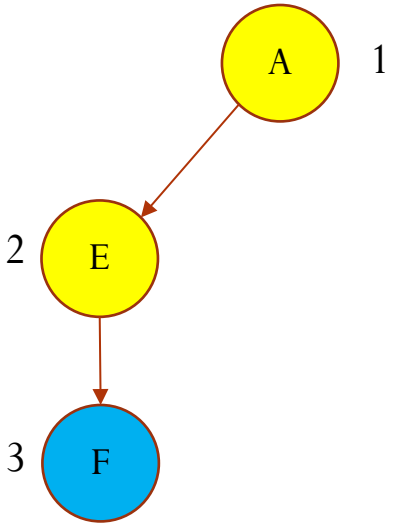

Lấy F ra khỏi stack
Đánh dấu F đã duyệt

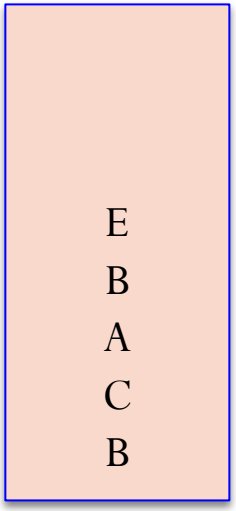


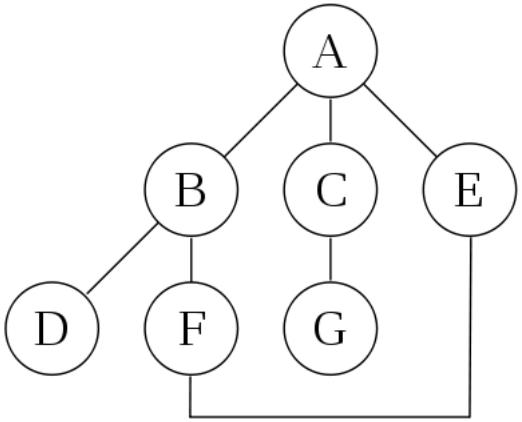

~~F~~
A
C
B



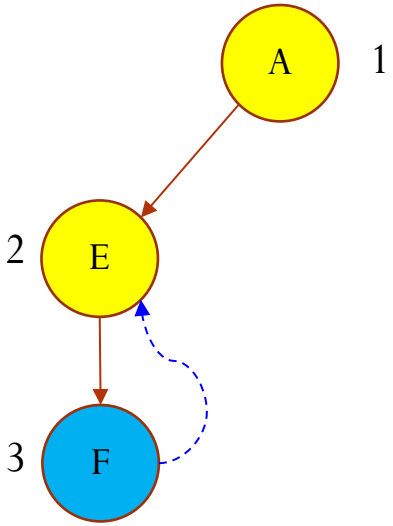

Push B, E vào stack



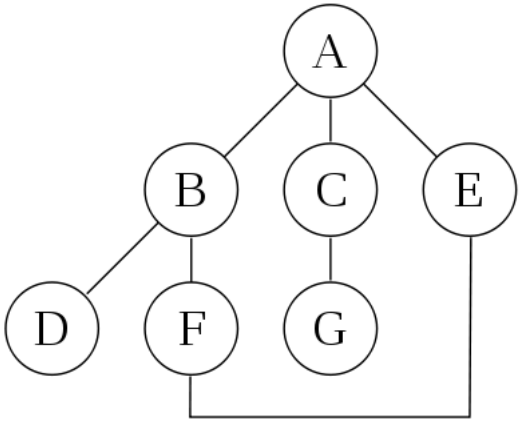





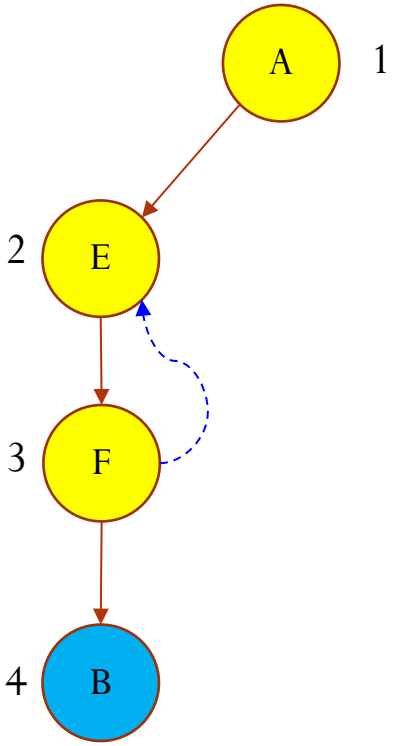

Lấy E ra khỏi stack
E đã được duyệt =>
bỏ qua



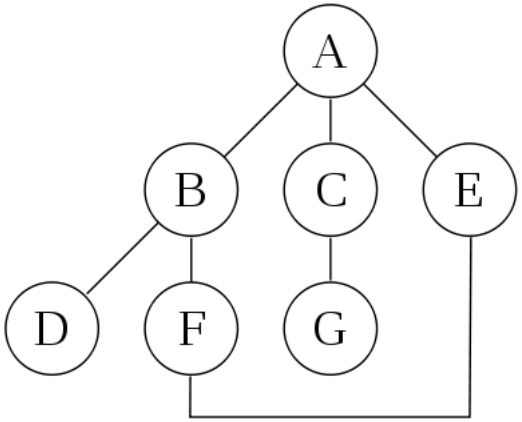

~~E~~
B
A
C
B



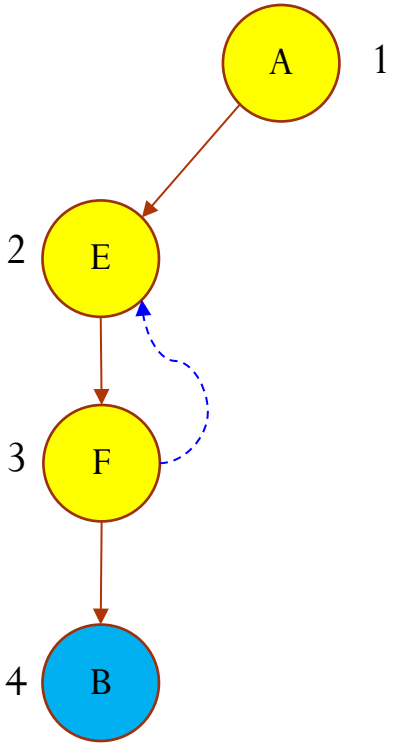

Lấy B ra khỏi stack
Đánh dấu B đã duyệt



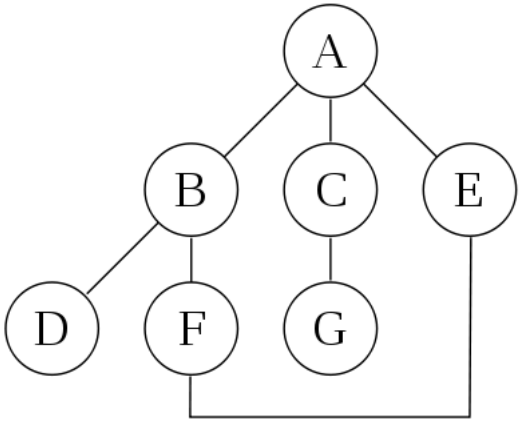

~~B~~
A
C
B



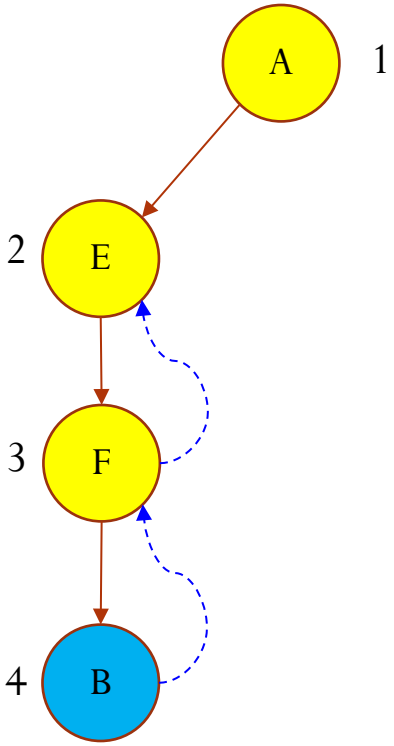

Push A, D, F vào stack



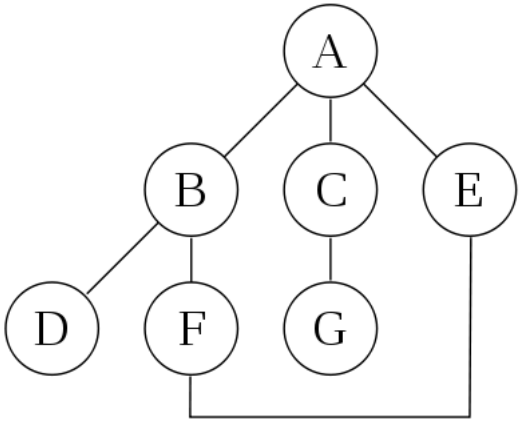

F
D
A
A
C
B



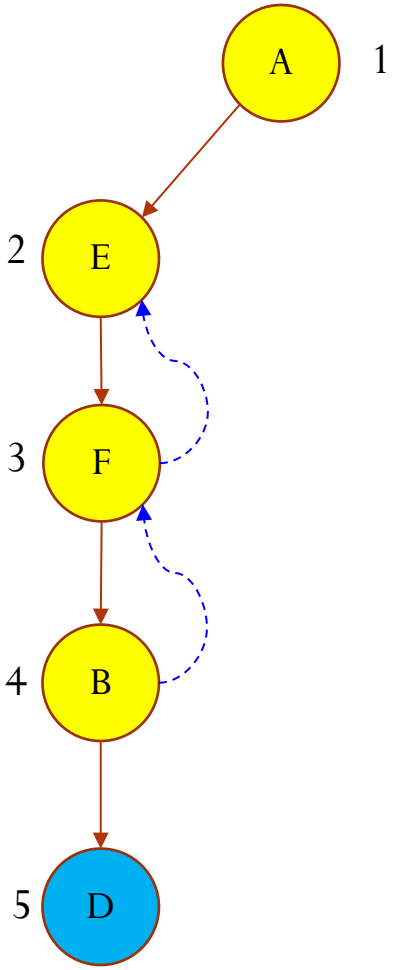

Lấy F ra khỏi stack
F đã được duyệt =>
bỏ qua



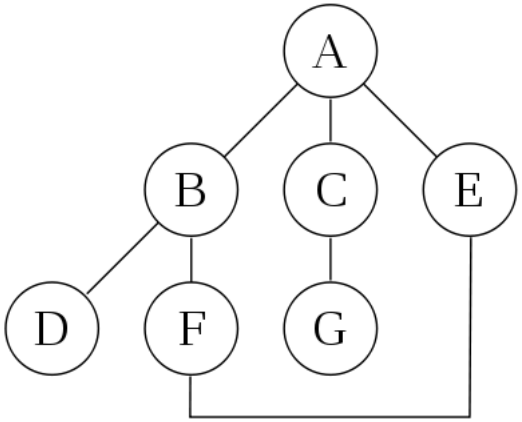

~~F~~
D
A
A
C
B



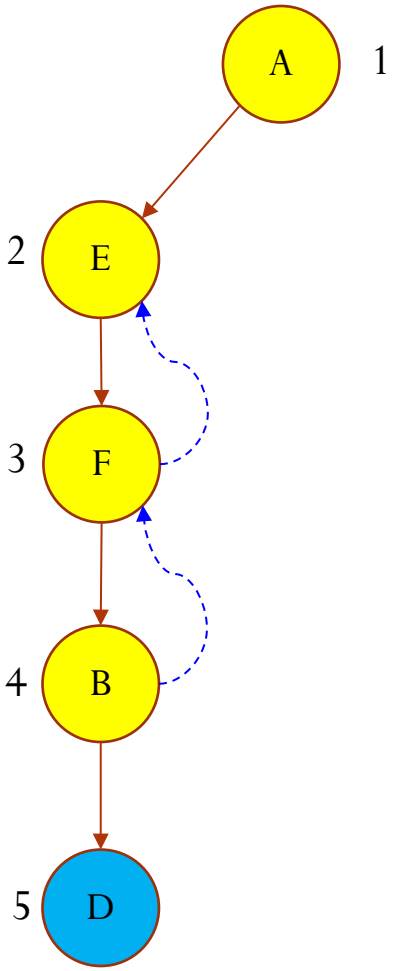

Lấy D ra khỏi stack
Đánh dấu D đã duyệt



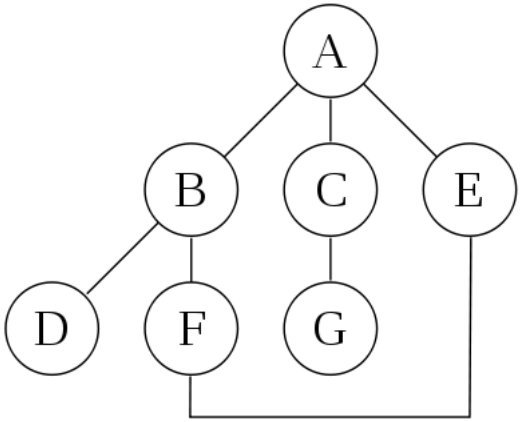

~~D~~
A
A
C
B



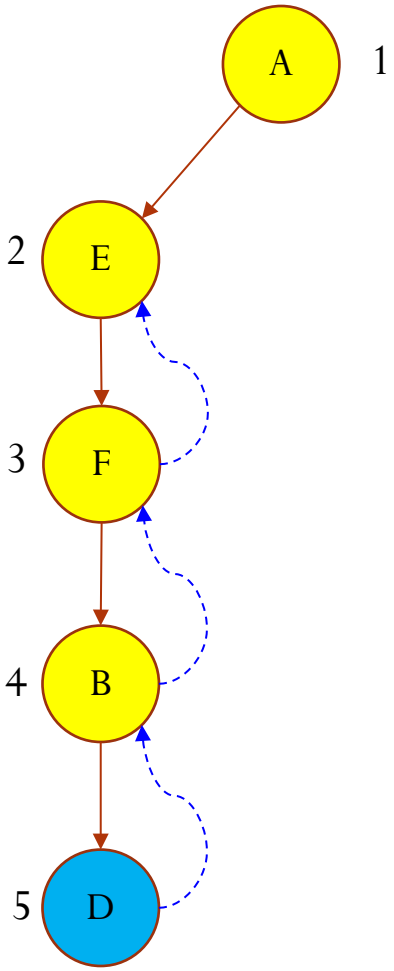

Push B vào stack



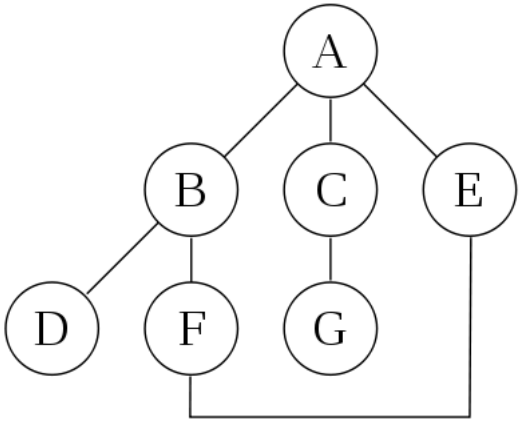

B
A
A
C
B



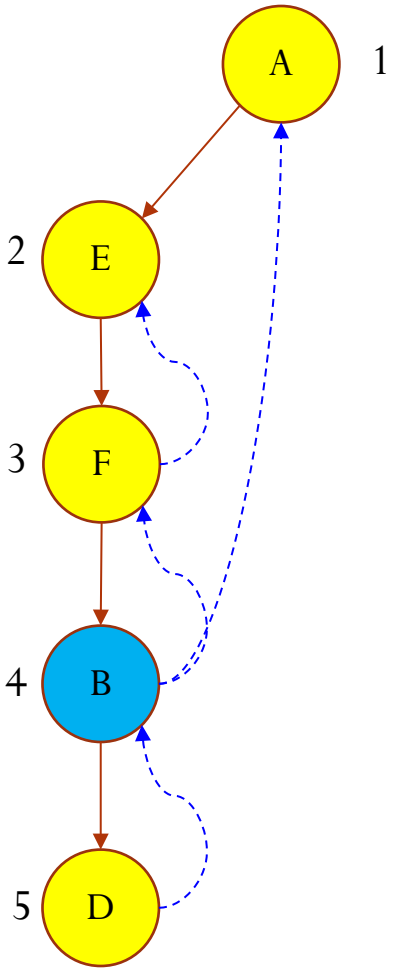

Lấy B ra khỏi stack
B đã được duyệt =>
bỏ qua



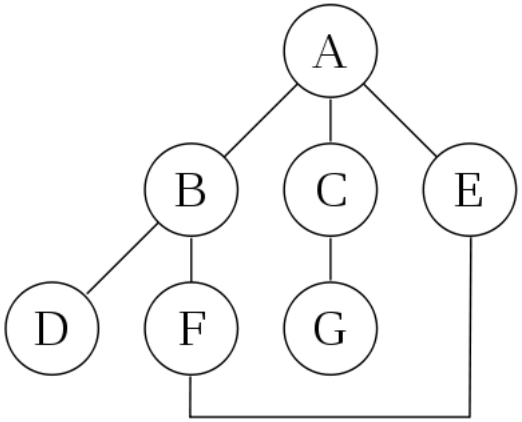

~~B~~
A
A
C
B



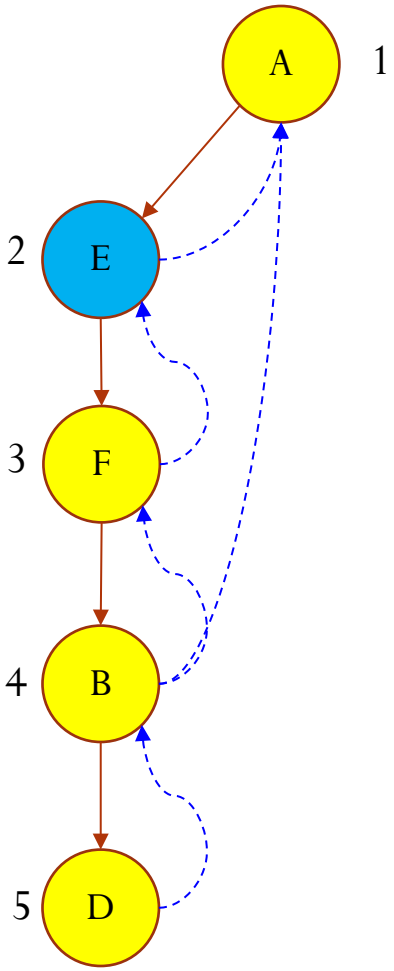

Lấy A ra khỏi stack
A đã được duyệt =>
bỏ qua



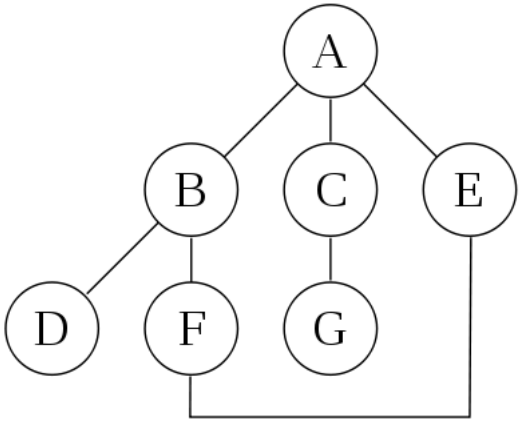

~~A~~
A
C
B



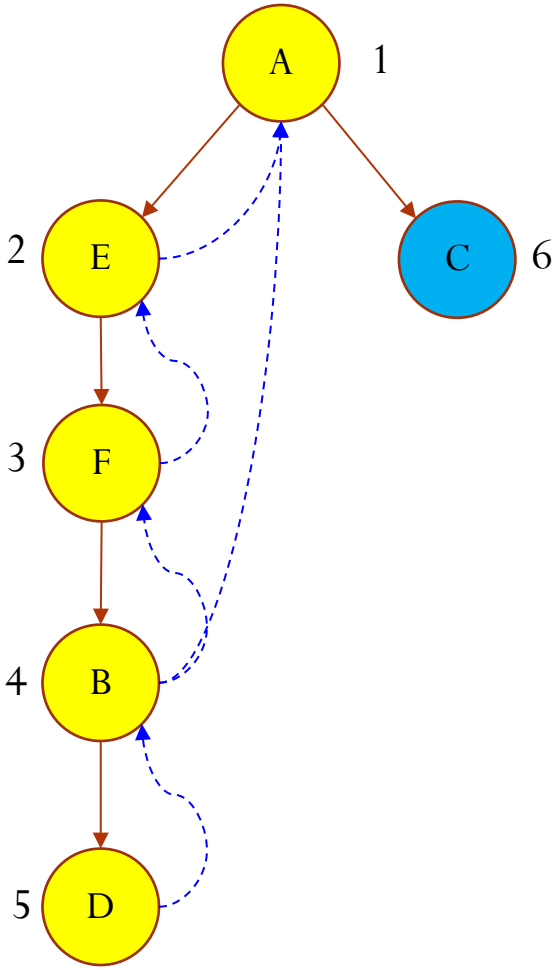

Lấy A ra khỏi stack
A đã được duyệt =>
bỏ qua

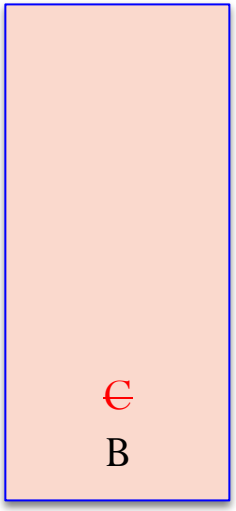


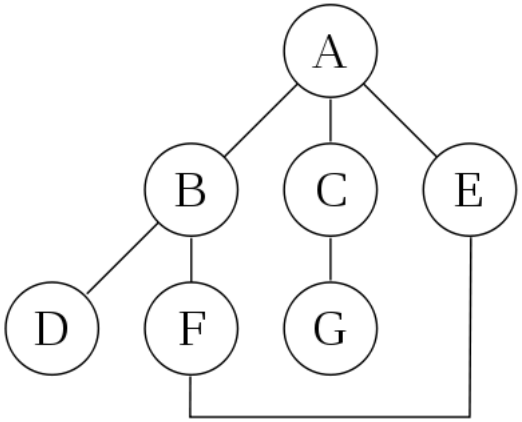

~~A~~
C
B



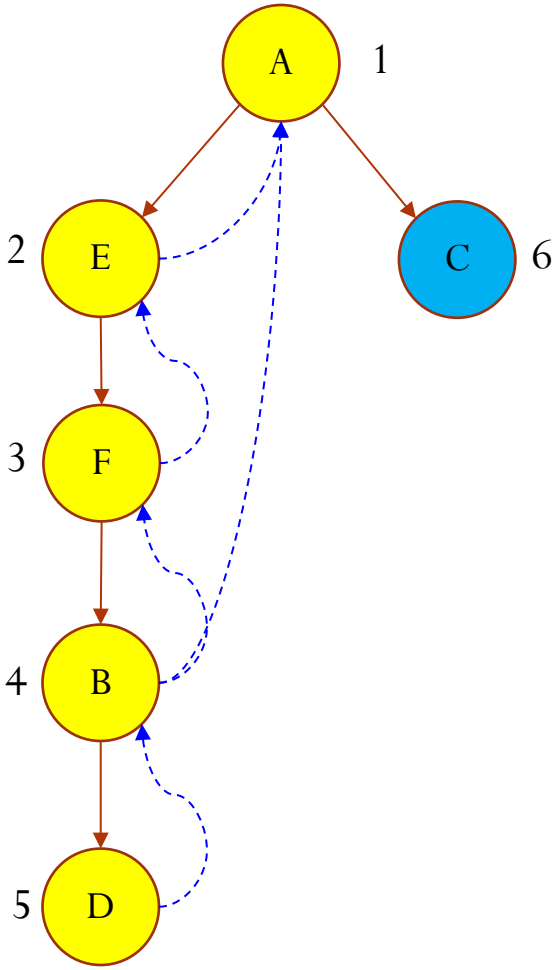

Lấy C ra khỏi stack
Đánh dấu C đã duyệt

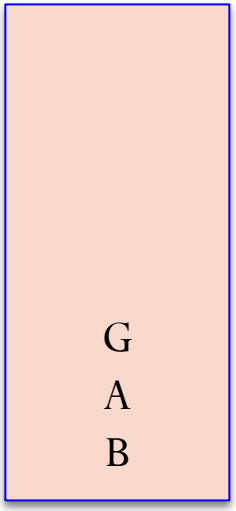


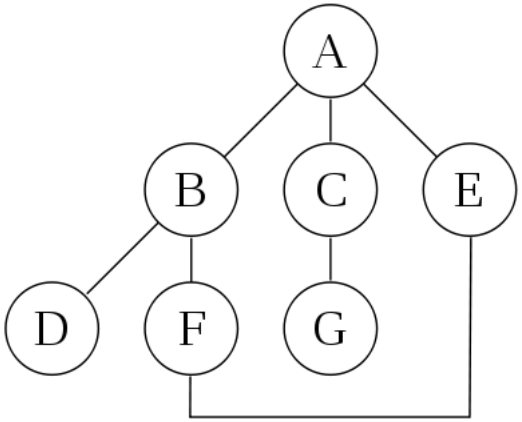





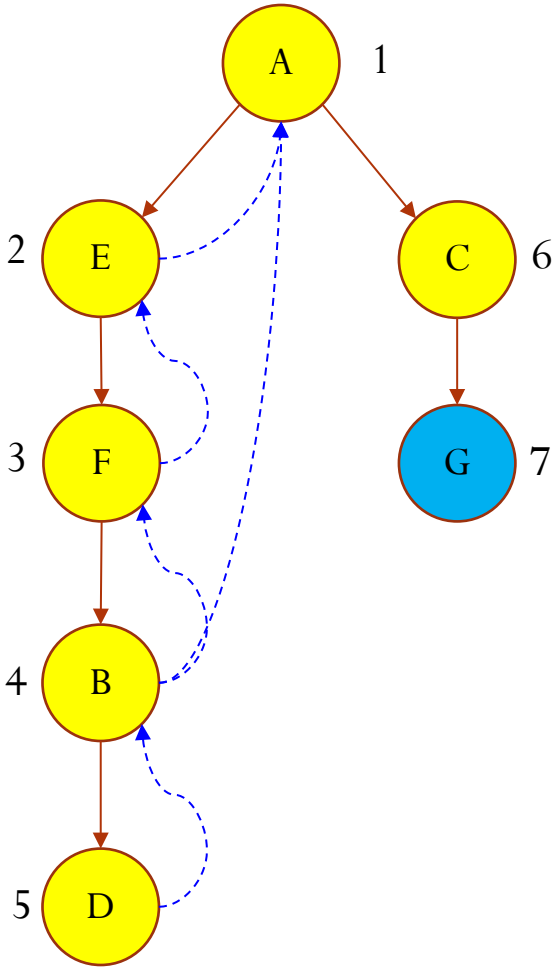

Lấy C ra khỏi stack
Push A, G vào stack



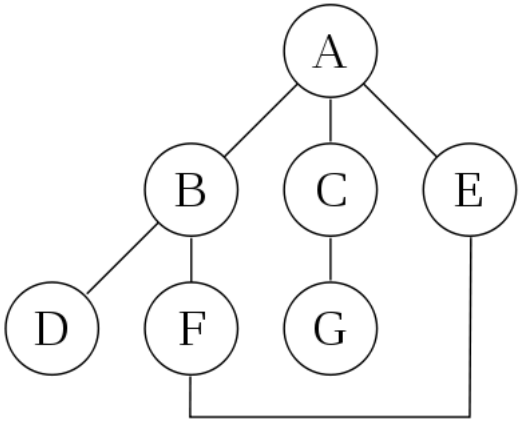





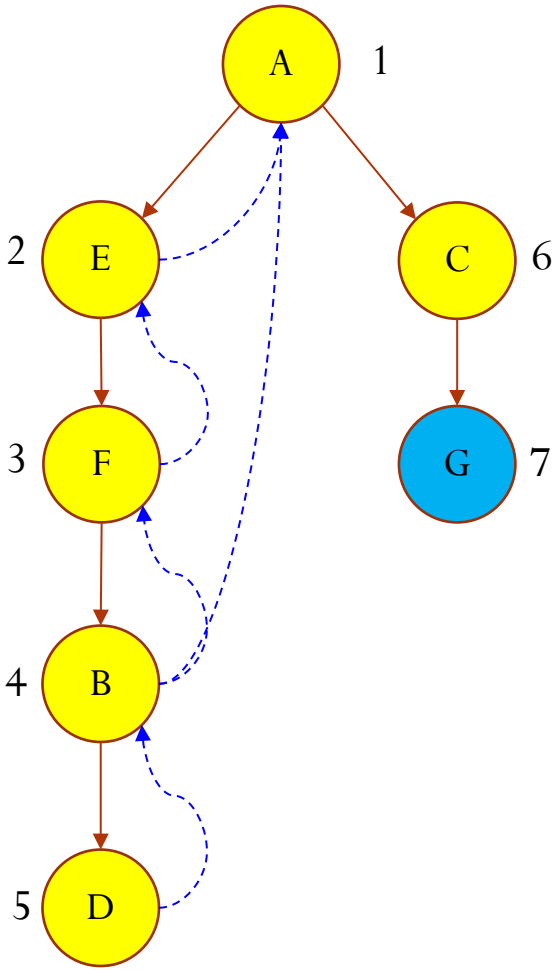

Lấy G ra khỏi stack
Đánh dấu G đã duyệt

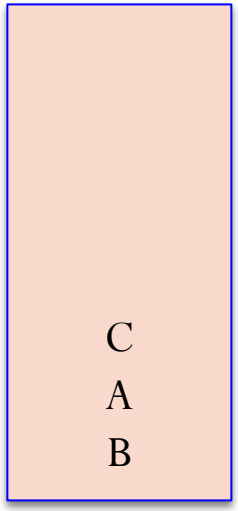


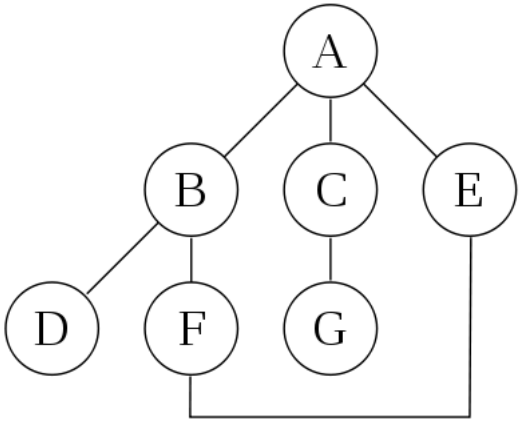

~~G~~
A
B



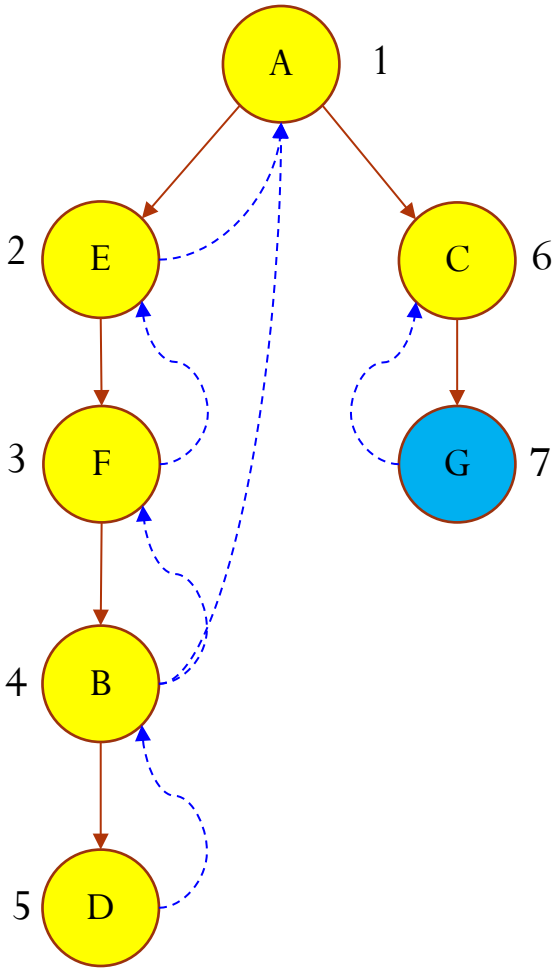

Push C vào stack



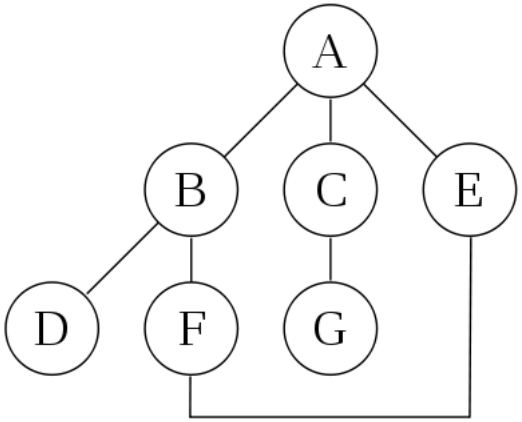





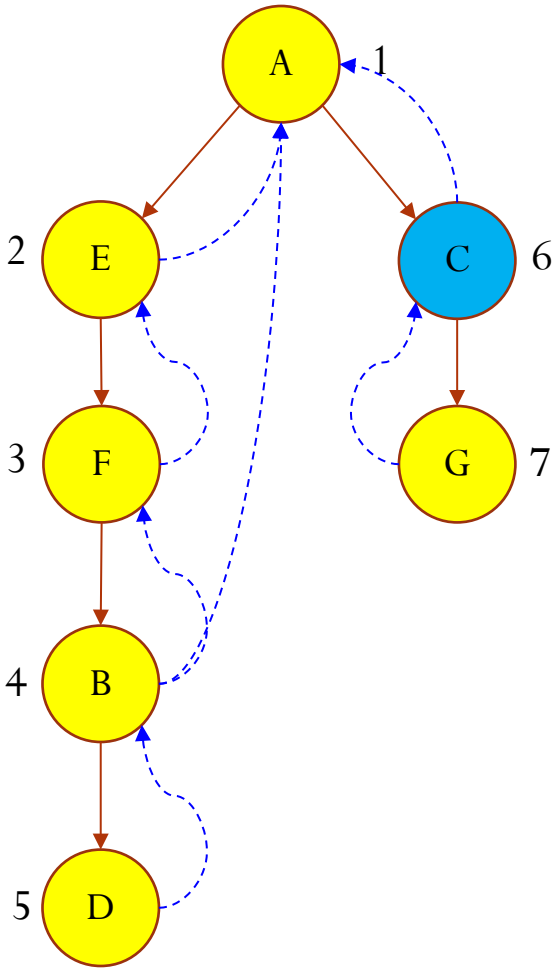

Lấy C ra khỏi stack
C đã được duyệt =>
bỏ qua

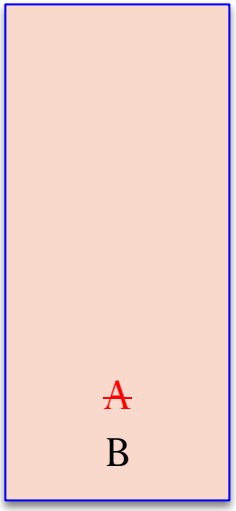


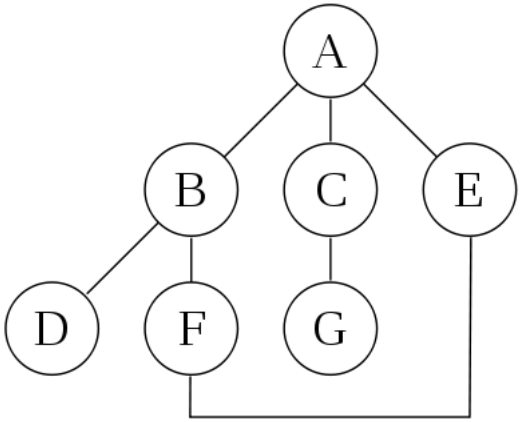

~~C~~
A
B



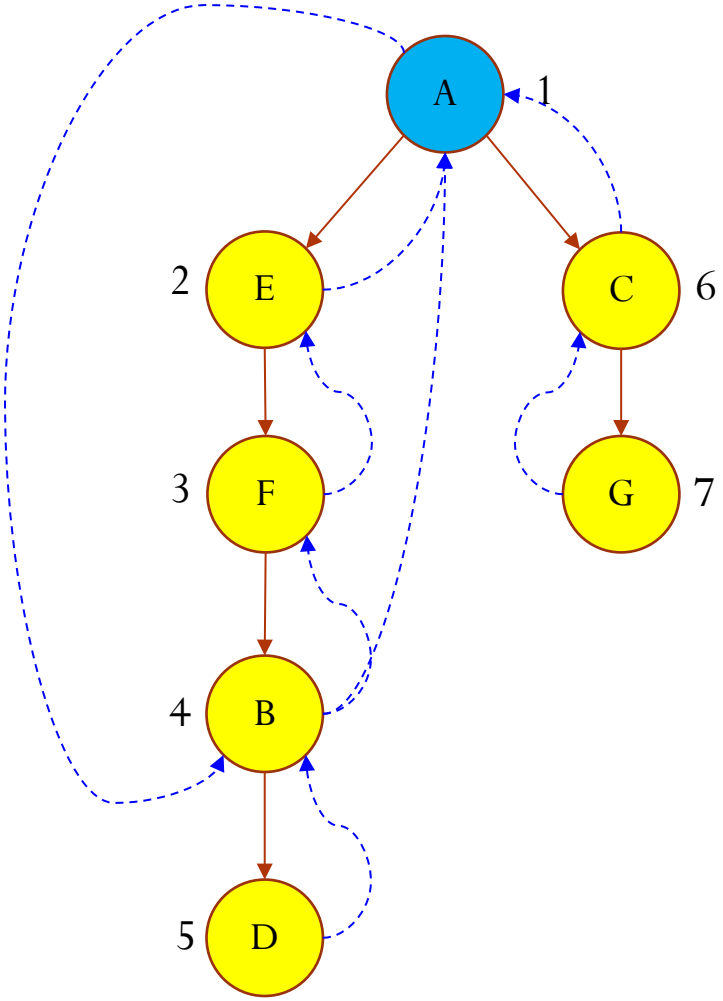

Lấy A ra khỏi stack
A đã được duyệt =>
bỏ qua

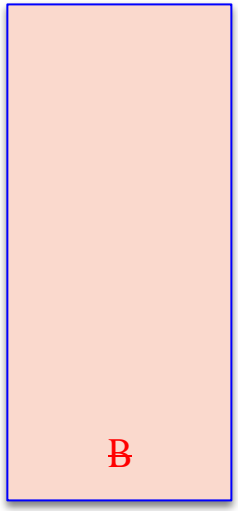


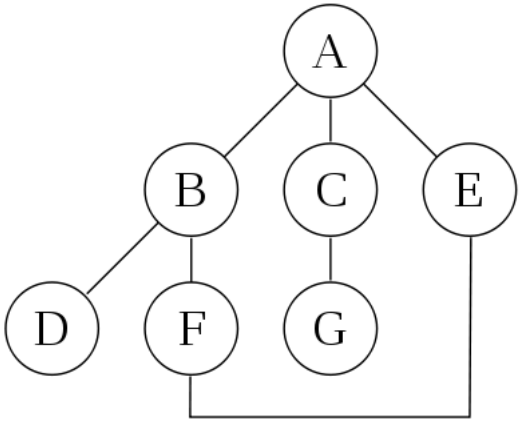


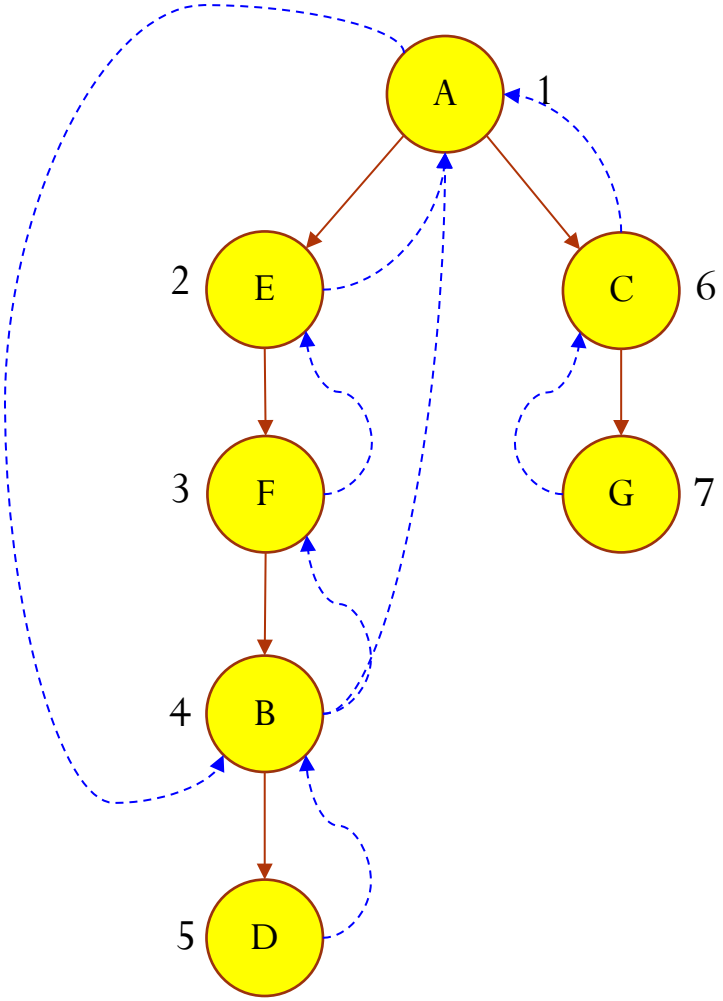

Lấy B ra khỏi stack
B đã được duyệt =>
bỏ qua



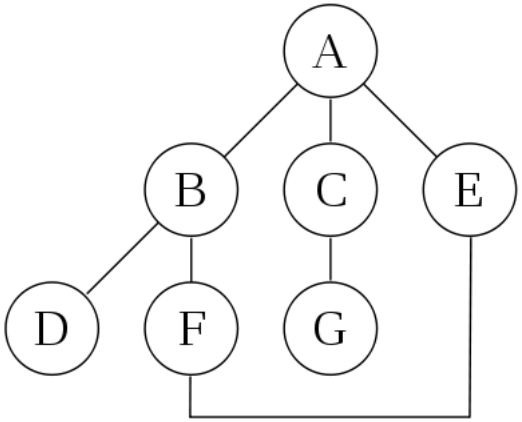



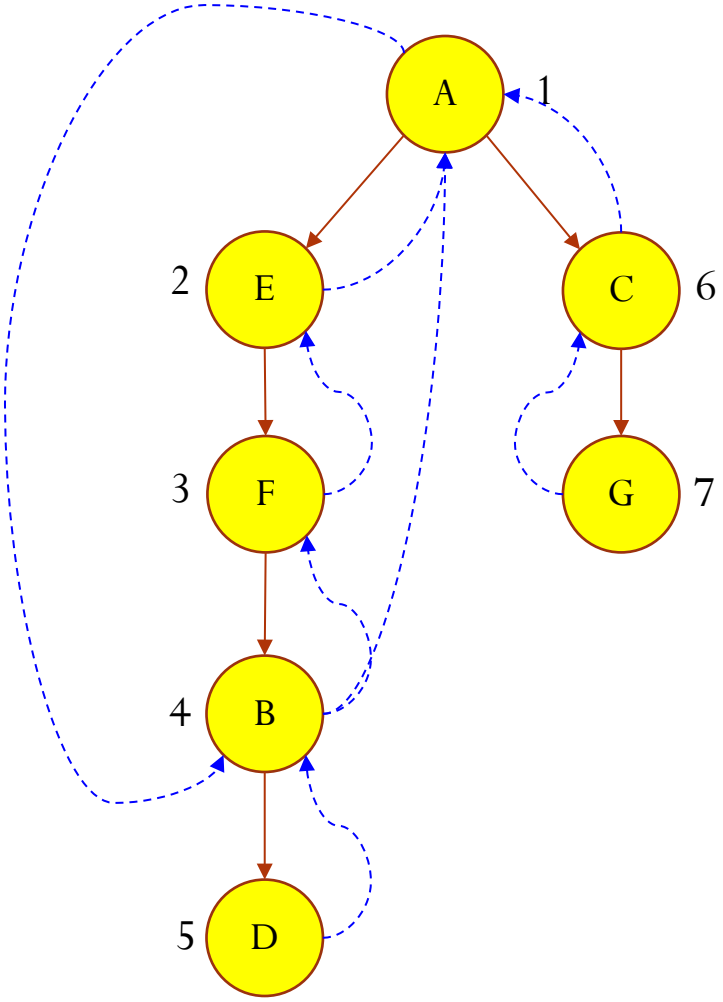

Stack rỗng =>
Kết thúc

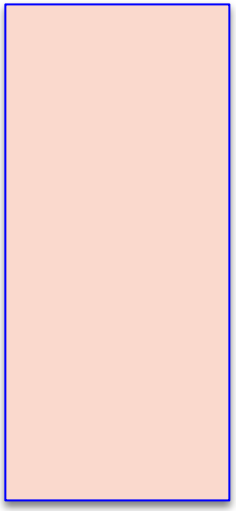



Stack rỗng =>
Kết thúc



Thứ tự duyệt:
A, E, F, B, D, C, G

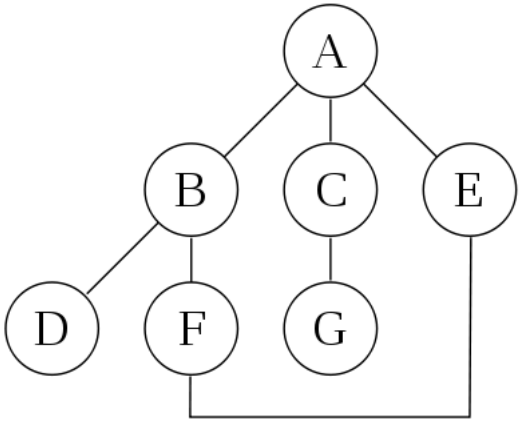
Stack rỗng =>
Kết thúc

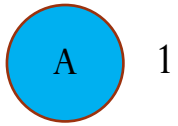
# Duyệt đồ thị

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

- Giải thuật:

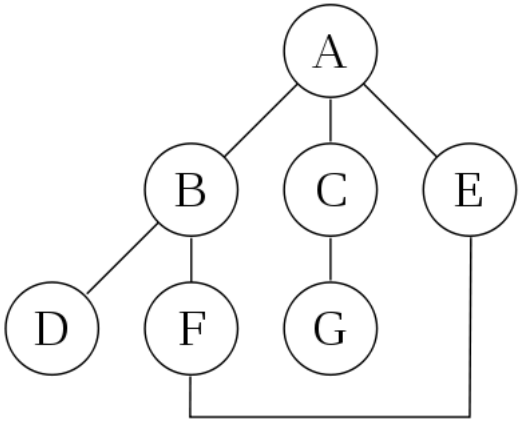
```
void visit(u) {
    if (u đã duyệt) return;
    //Duyệt u: in u ra màn hình hoặc đánh số đỉnh u
    Đánh dấu u đã duyệt
    for (các đỉnh kề v của u) do
        visit(v); //gọi đệ quy duyệt các đỉnh kề v của u
}
```

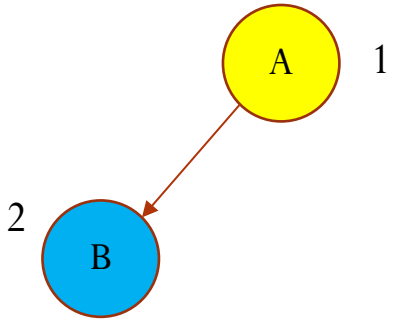


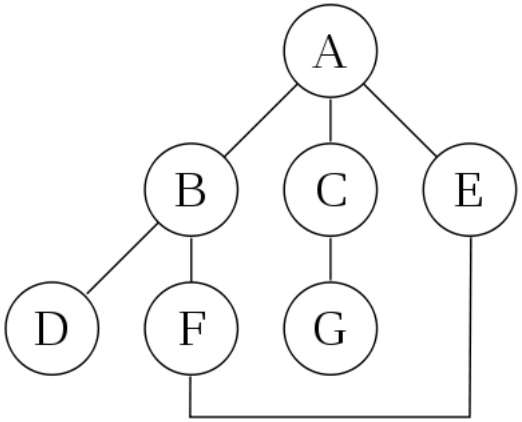

1

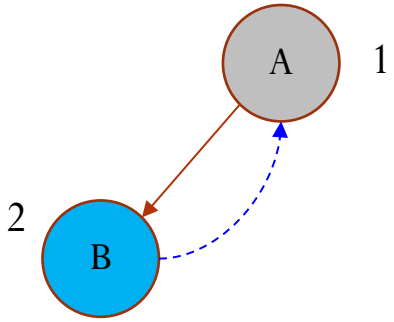
Đánh dấu A đã duyệt



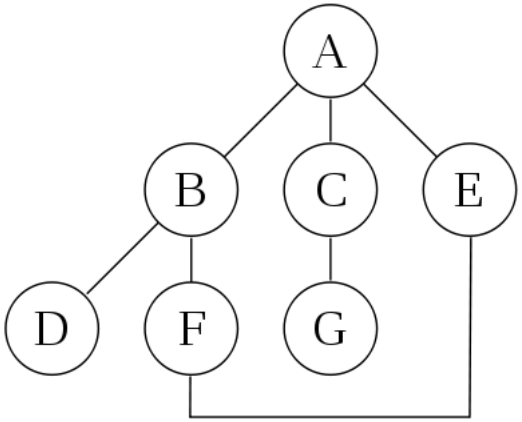




Đánh dấu B đã duyệt

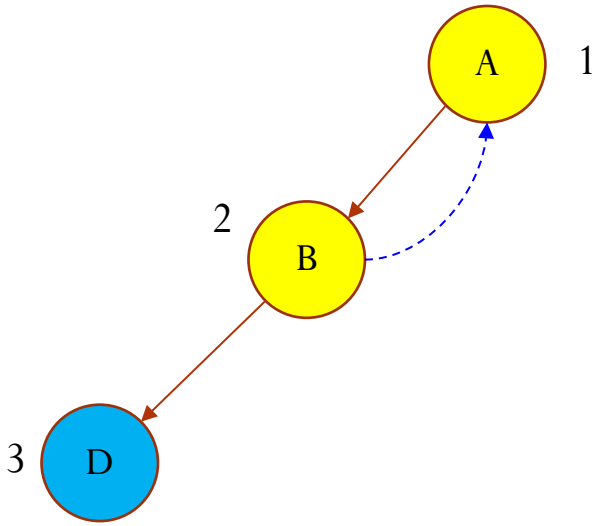


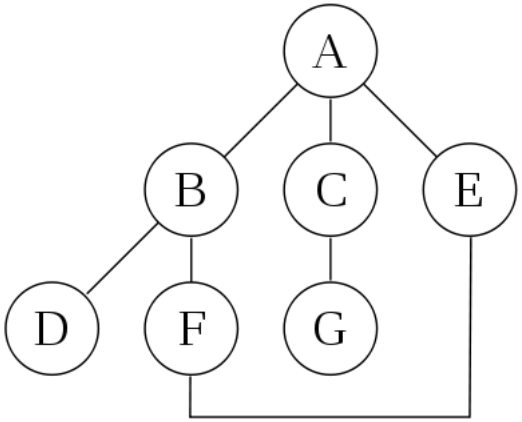




A đã được duyệt =>
bỏ qua

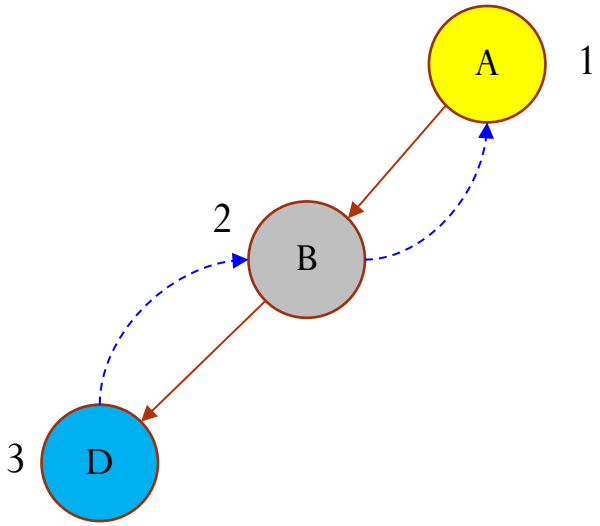


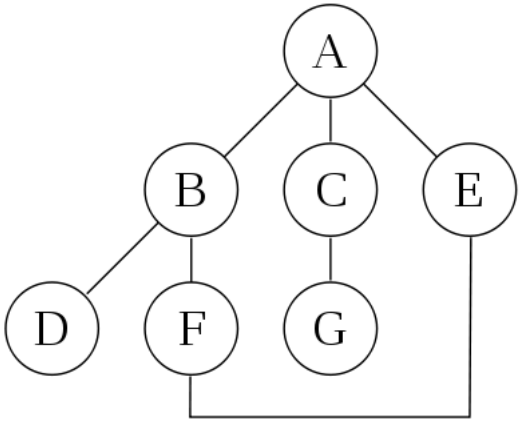

Đánh dấu D đã duyệt



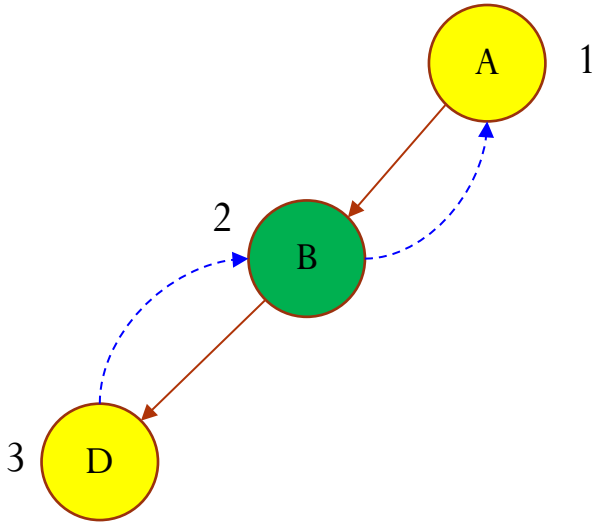


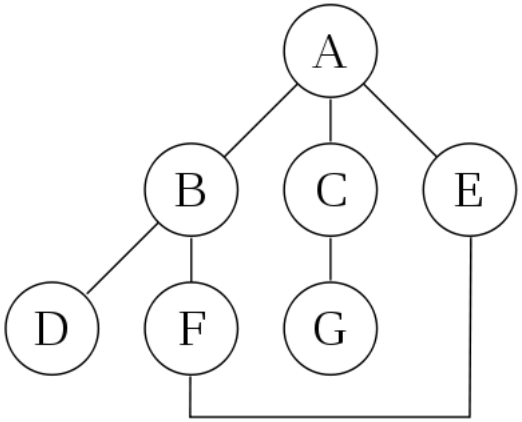

B đã được duyệt =>
bỏ qua



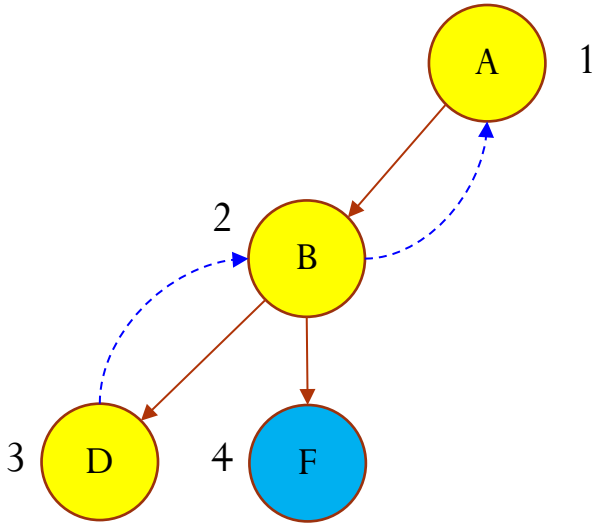



D đã xét xong =>
quay về B

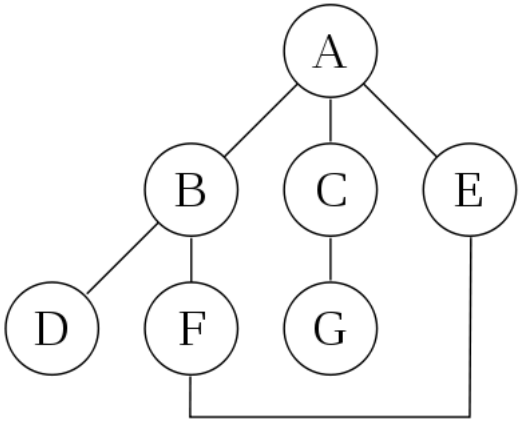




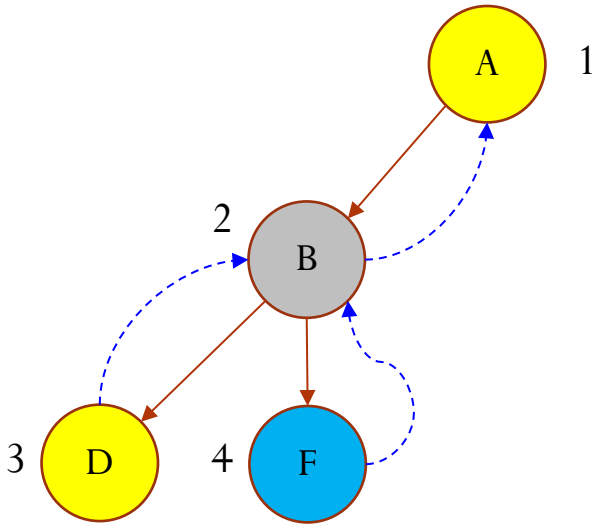

Đánh dấu F được duyệt

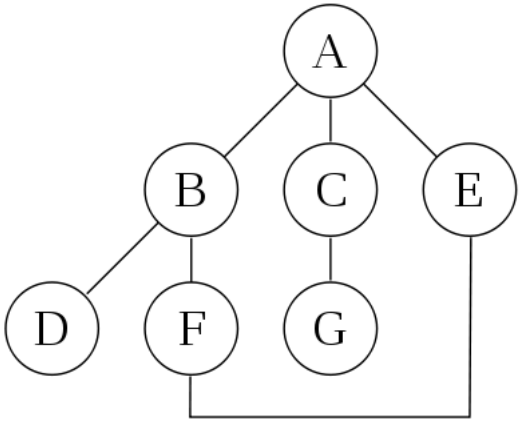



5

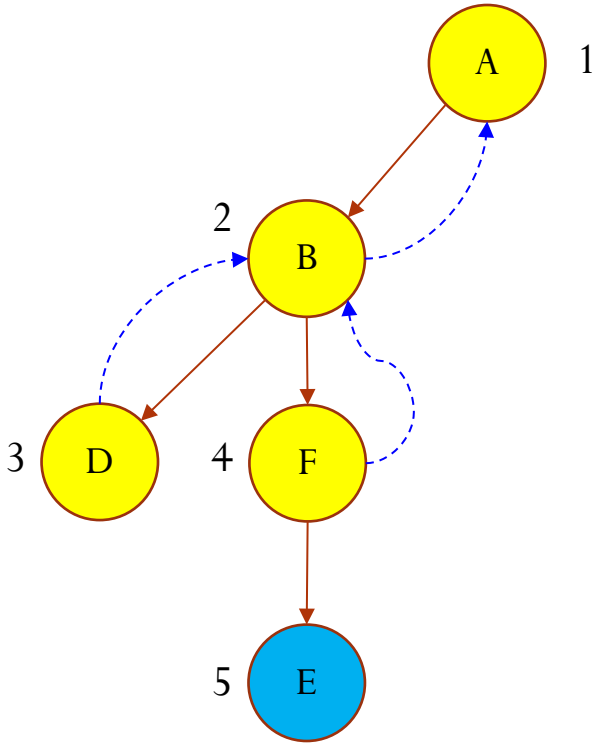


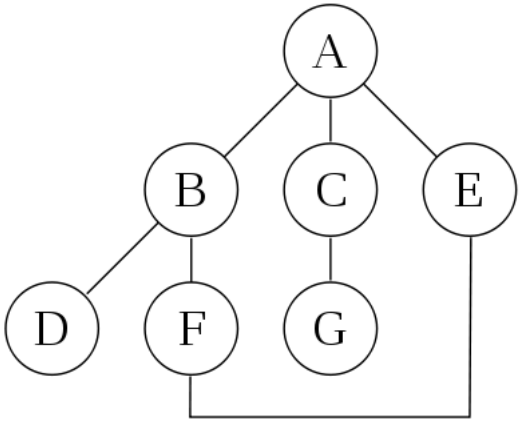

B đã được duyệt =>
bỏ qua



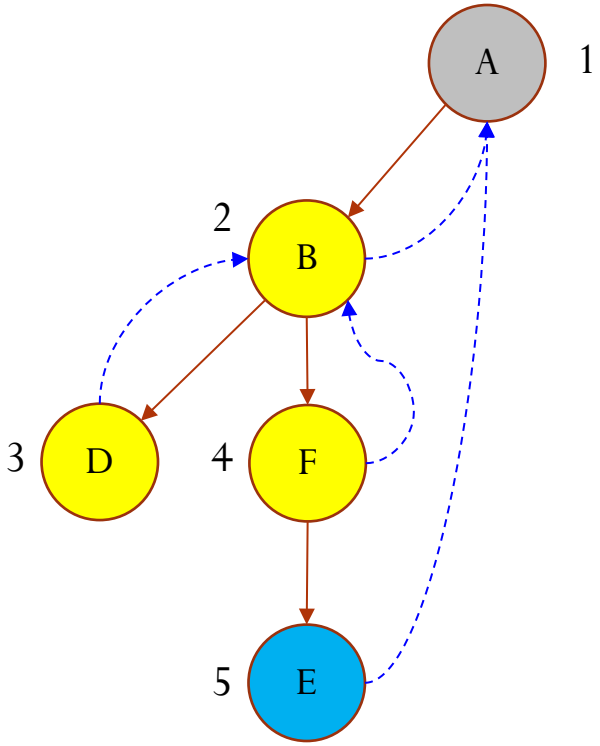


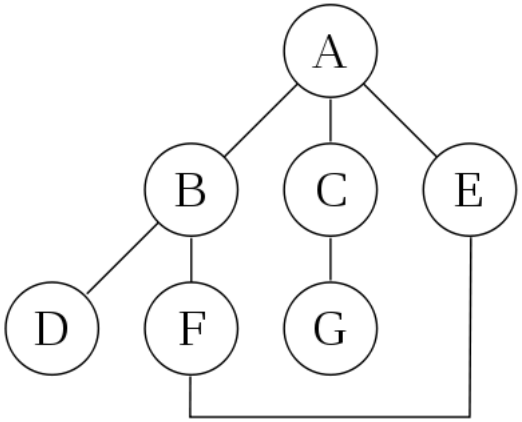

Đánh dấu E đã duyệt



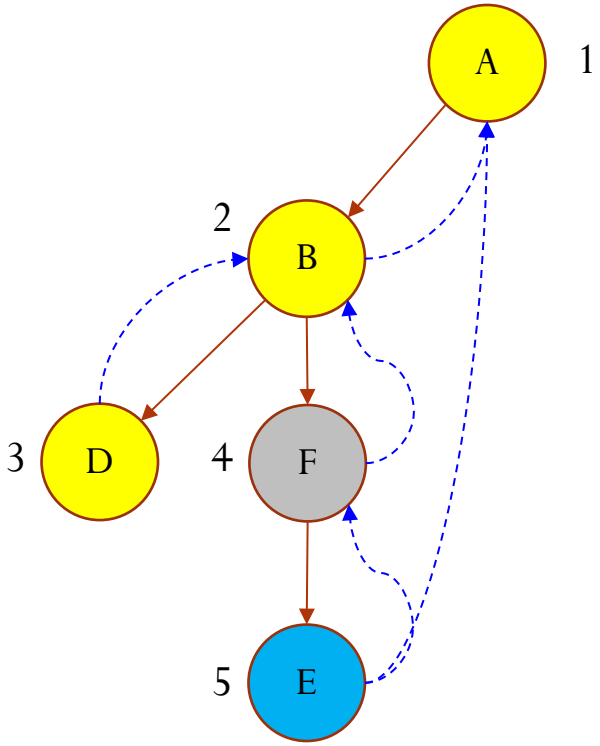


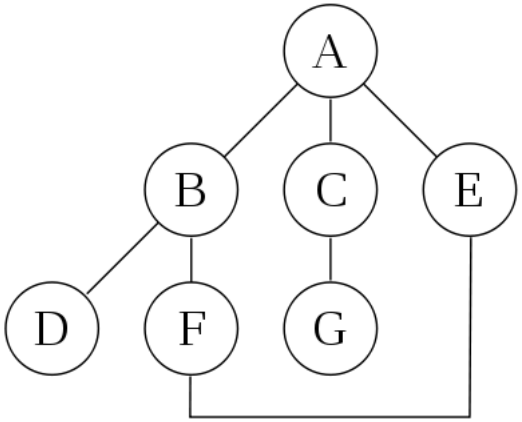

A đã được duyệt =>
bỏ qua



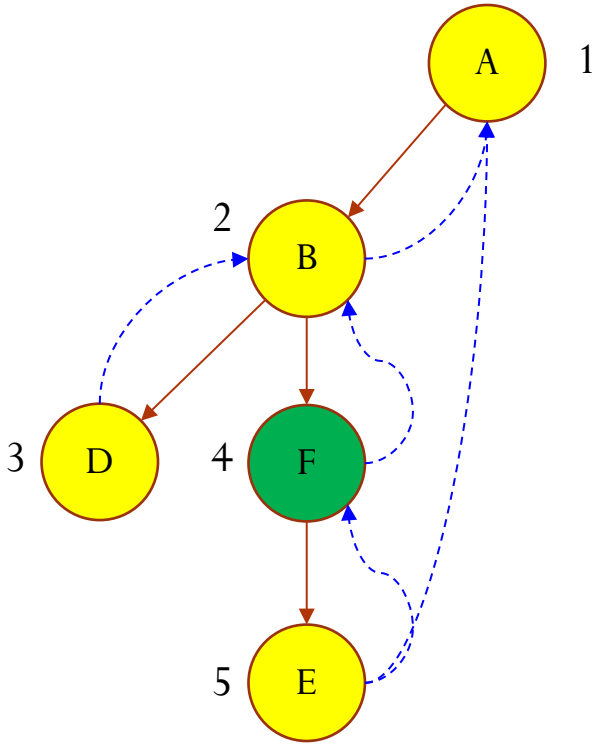


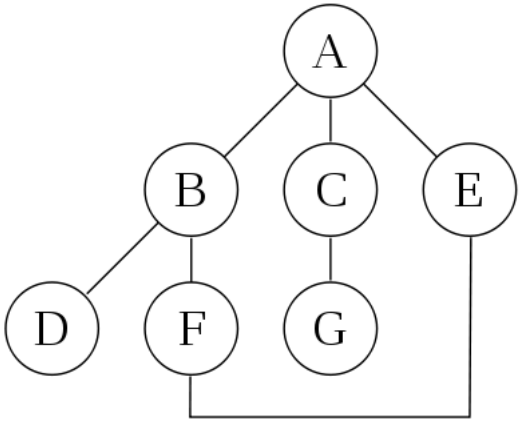

F đã được duyệt =>
bỏ qua



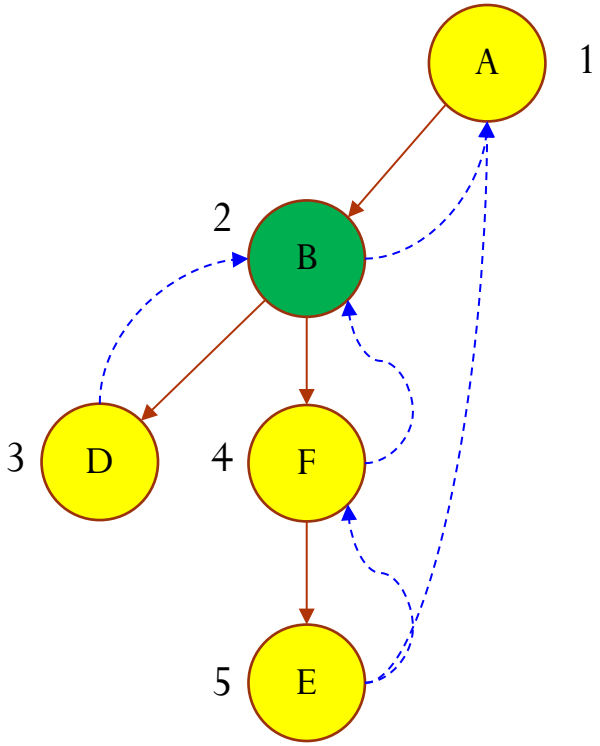


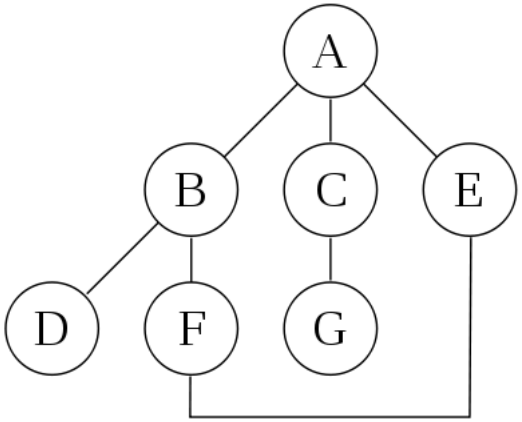

E đã xét xong =>
quay về F



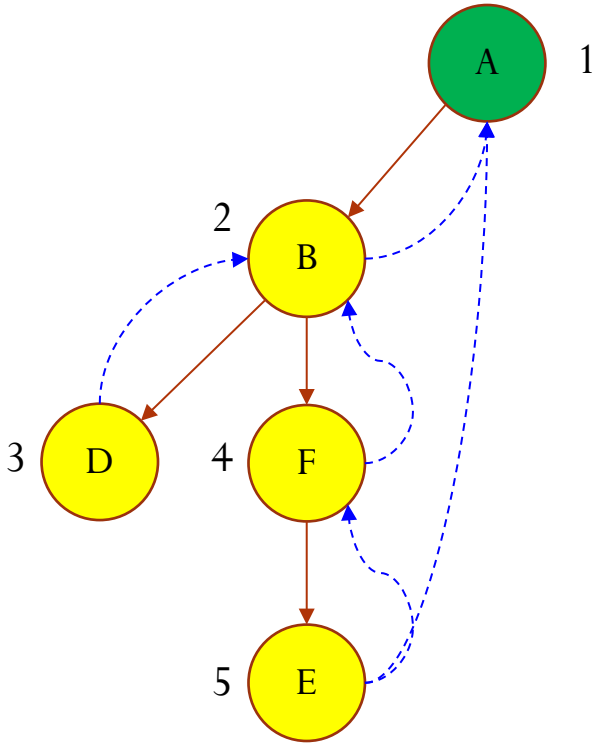


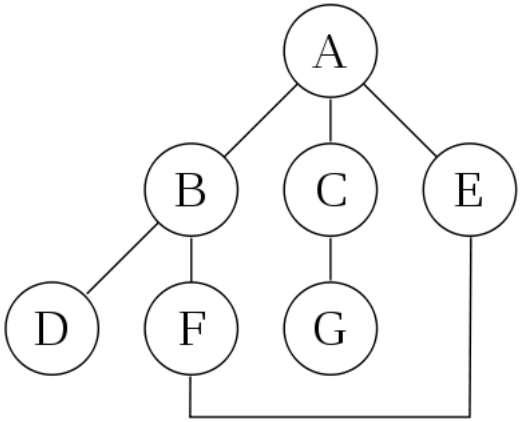

F đã xét xong =>
quay về B



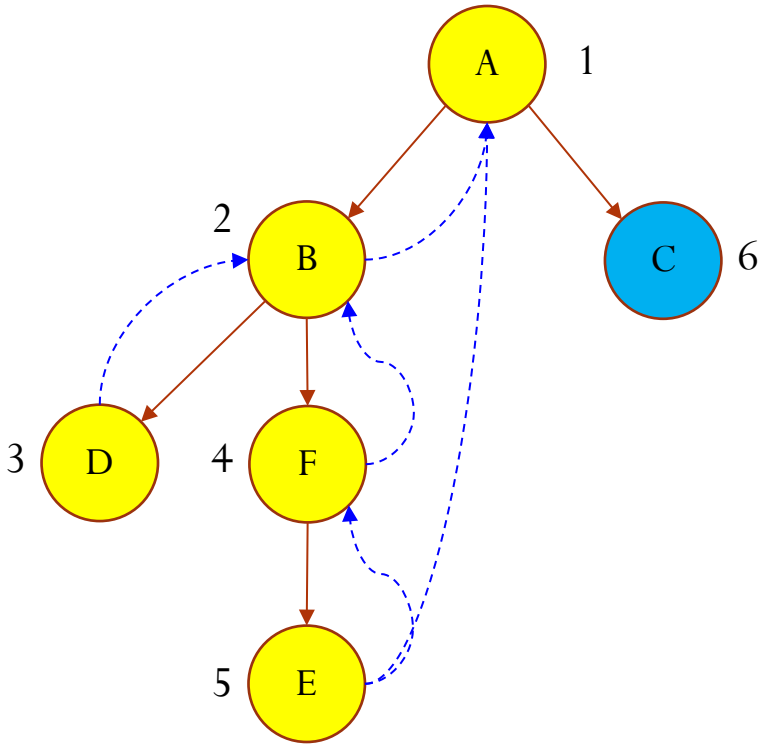


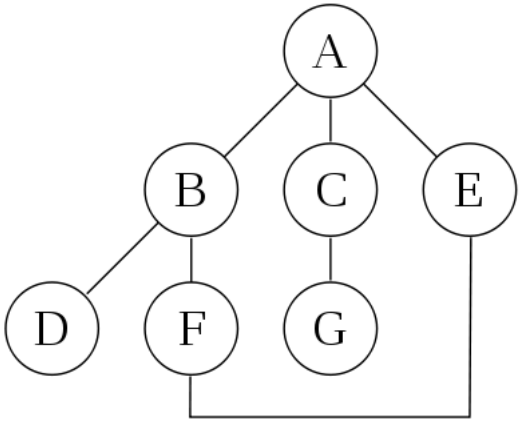

B đã xét xong =>
quay về A



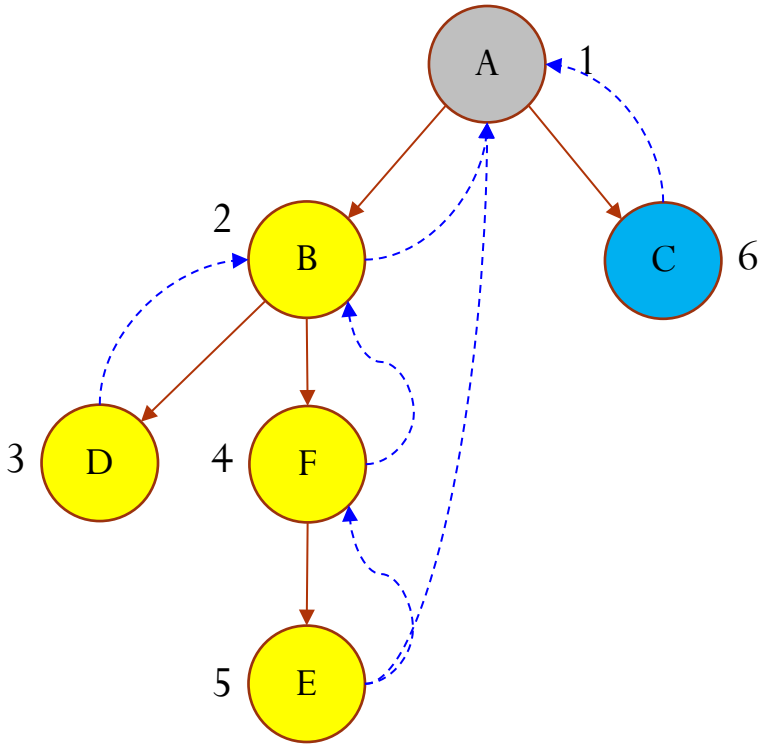


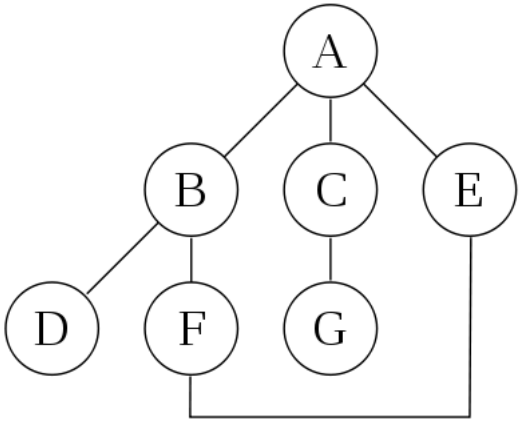

Đánh dấu C đã duyệt



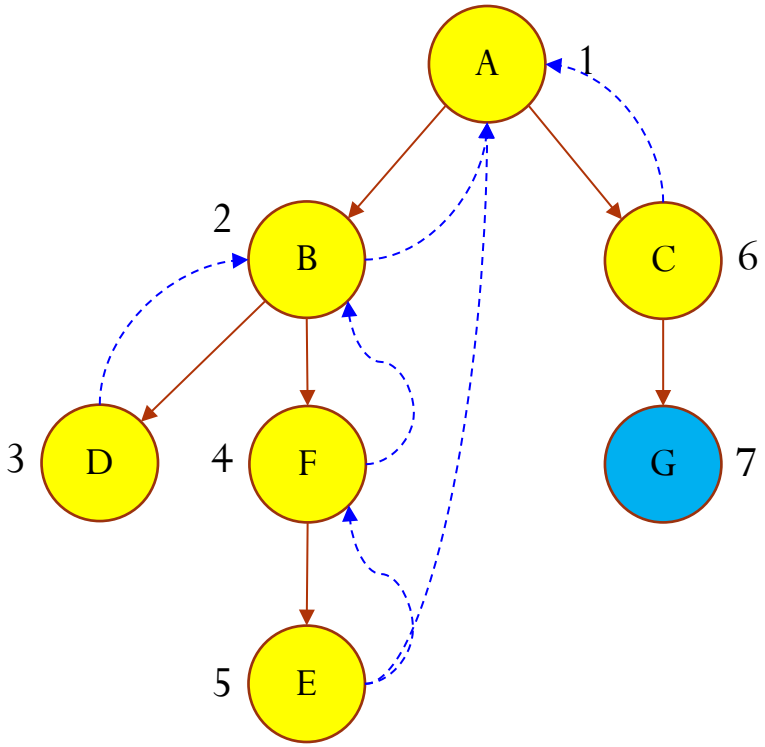


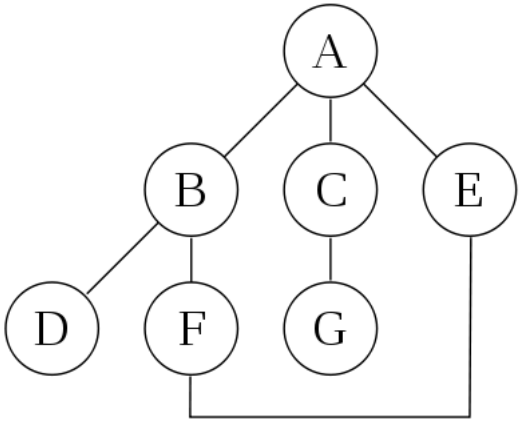

A đã được duyệt =>
bỏ qua



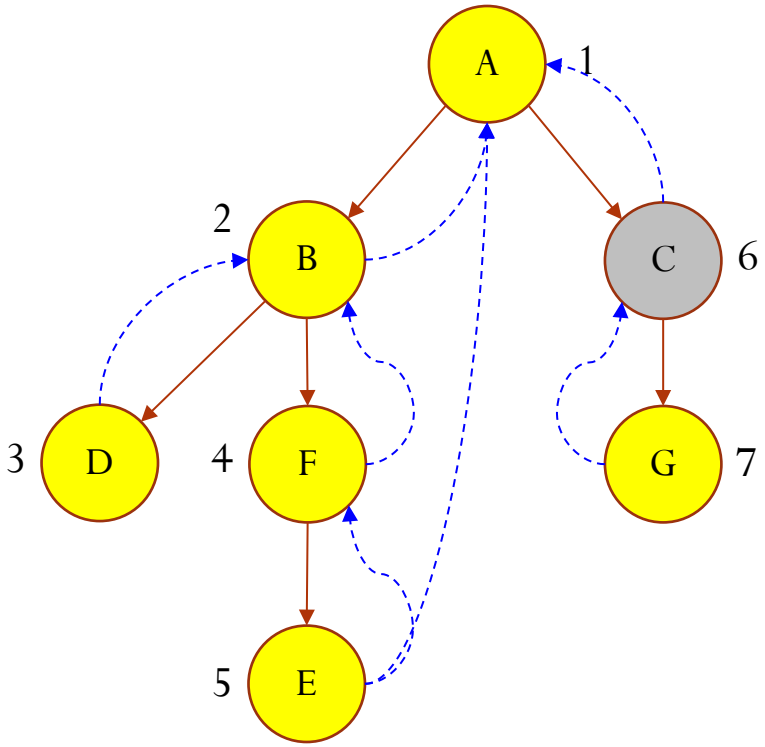


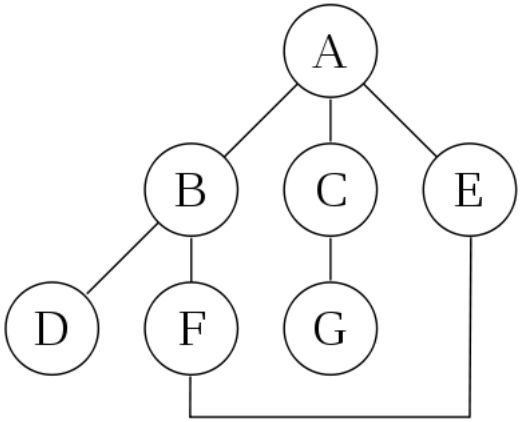

Đánh dấu G đã duyệt



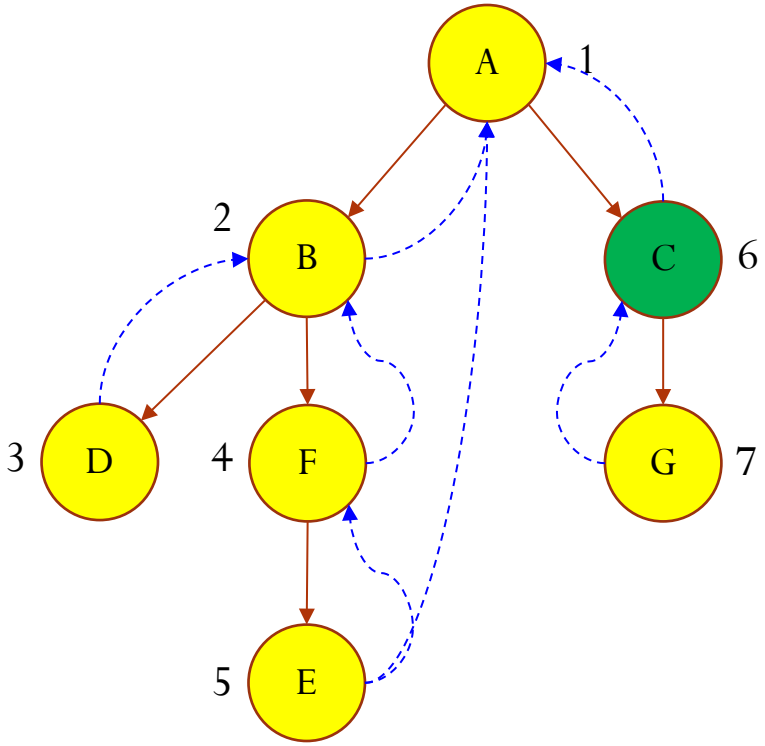


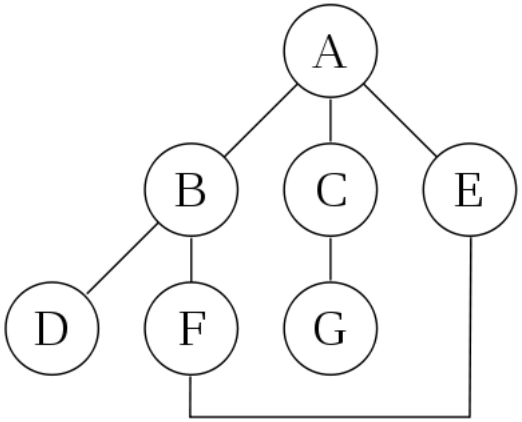

C đã được duyệt =>
bỏ qua



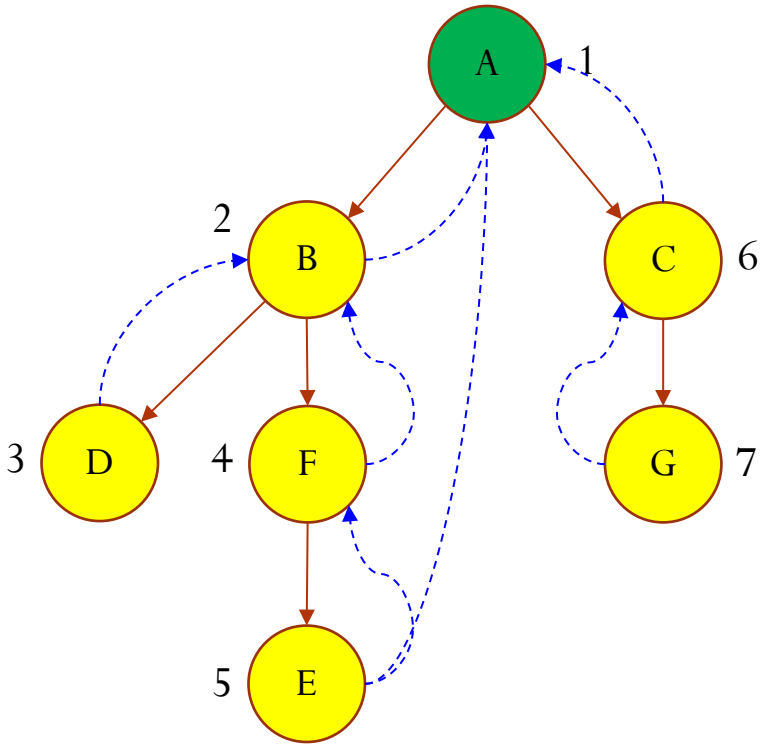


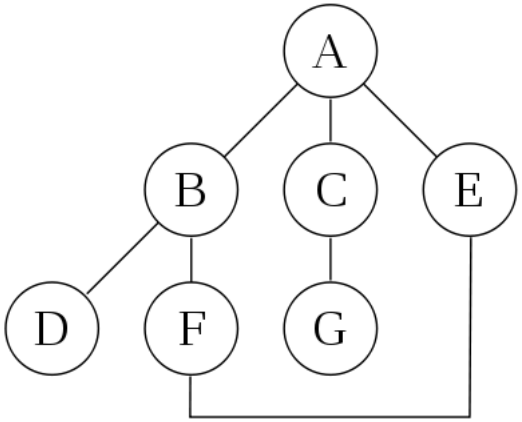

G đã xét xong =>
quay về C



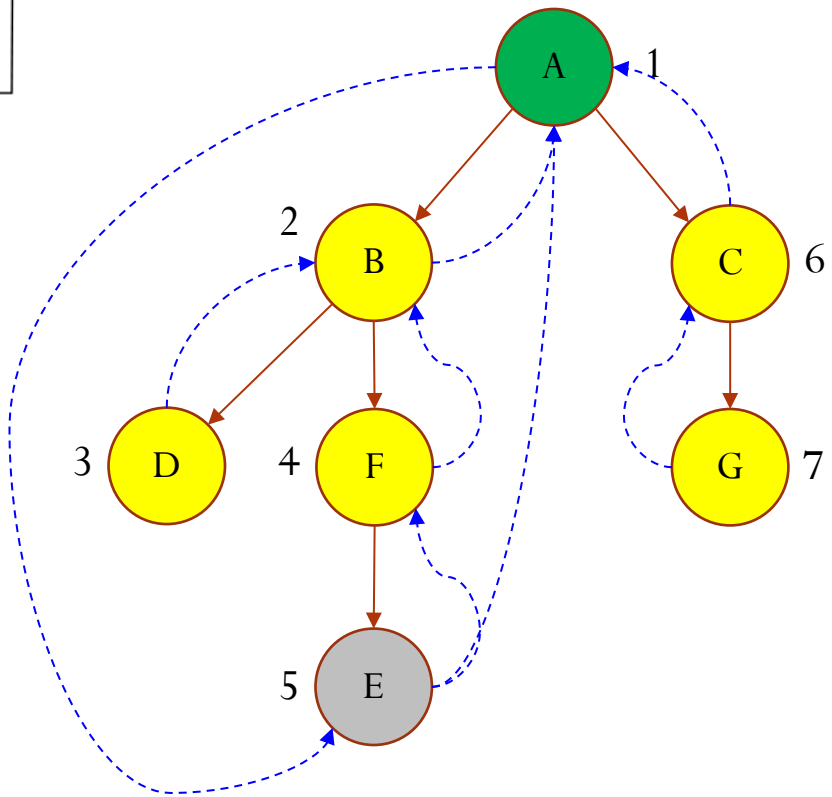


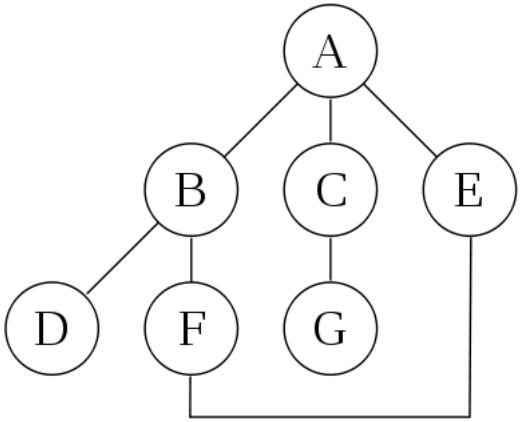

C đã xét xong =>
quay về A



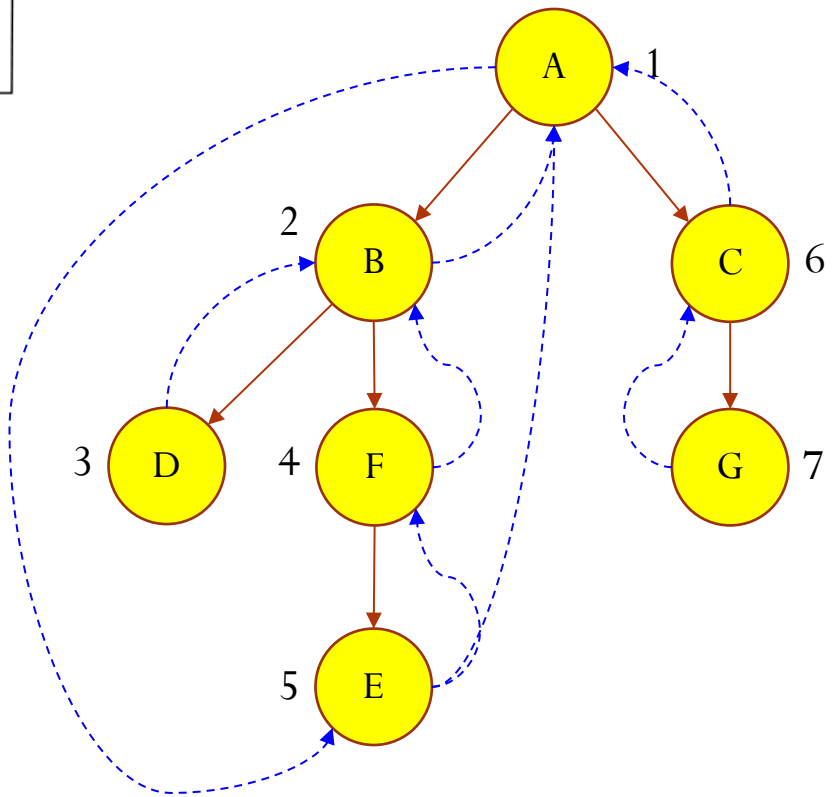



E đã được duyệt => bỏ qua






A đã xét xong
=> Kết thúc




Thứ tự duyệt:
A, B, D, F, E, C, G

# 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 giải thuật đệ quy