ĐƯỜNG ĐI, CHU TRÌNH

Đường đi

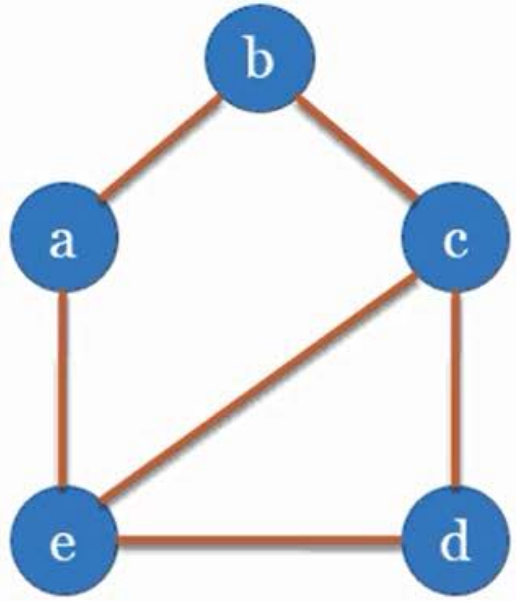
- Đường đi (path) (độ dài n) từ u tới v trong một đồ thị vô hướng là một dãy các cạnh e_1, e_2 … e_n của đồ thị sao cho $f(e_1) = \{x_0, x_1\}$, $f(e_2) = \{x_1, x_2\}$… $f(e_n) = \{x_{n-1}, x_n\}$, với $x_0 = u$ và $x_n = v$.
- Khi đồ thị là đơn ta ký hiệu đường đi này bằng dãy các đỉnh x_0, x_1, … x_n.

Chu trình

- Đường đi được gọi là chu trình (cycle/circuit) nếu nó bắt đầu và kết thúc tại một đỉnh (nghĩa là u = v).
- Đường đi hay chu trình gọi là đơn nếu nó không đi qua cùng một cạnh quá một lần.

Ví dụ về đường đi trên đồ thị

- {a, b, c, e, d} là một đường đi độ dài 4.
- {a, b, e, d} không là đường đi.
- {a, b, c, e, a} là một chu trình.
- {c, e, d, e, c} không phải là một đường đi đơn.

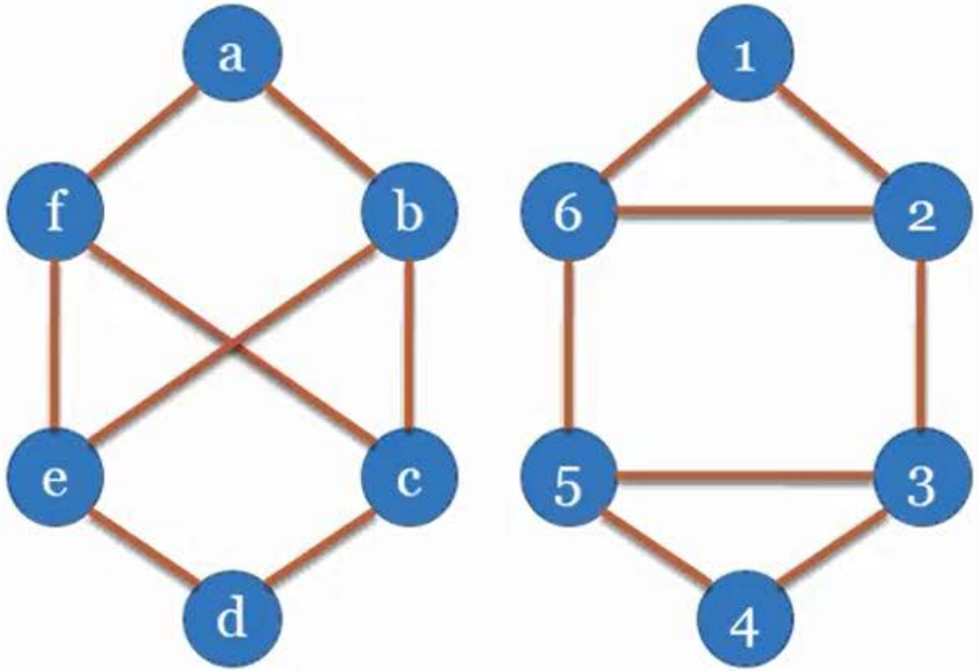


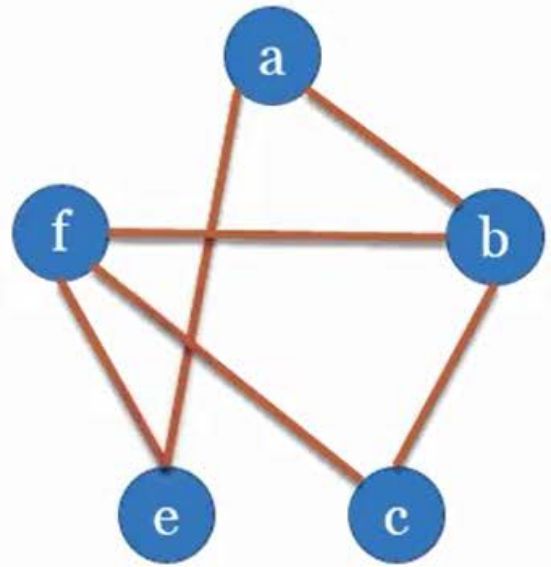
Chu trình và sự đẳng cấu

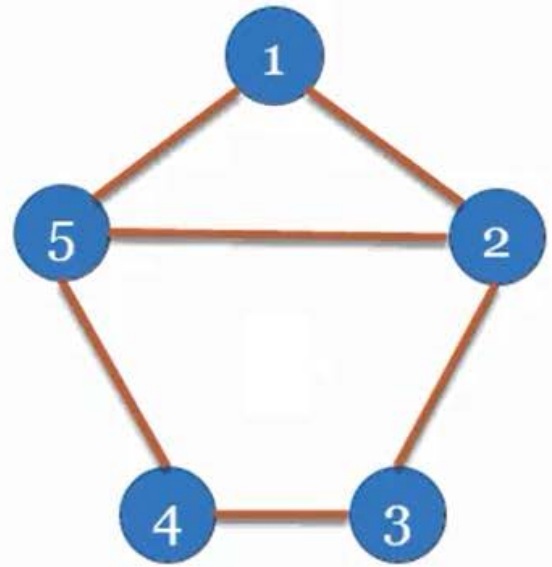
- Nếu 2 đồ thị đẳng cấu với nhau, nó sẽ có các chu trình có cùng độ dài k với nhau, trong đó k > 2.

(Điều này suy ra *Nếu điều kiện trên không thỏa, nghĩa là 2 đồ thị không đẳng cấu với nhau*)

Chu trình và sự đẳng cấu



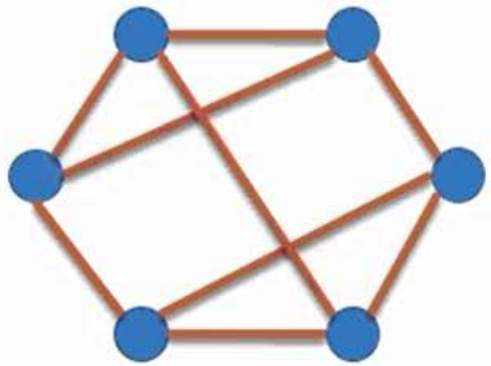


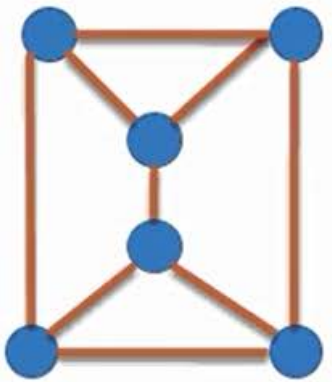


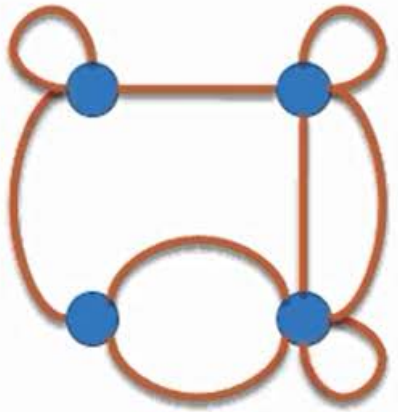
Liên thông

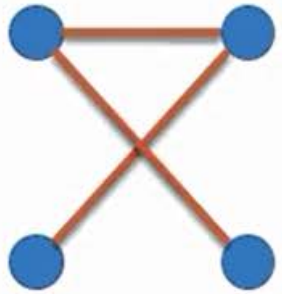
- Một đồ thị vô hướng được gọi là liên thông (connected) nếu có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị.
- Ngược lại, đồ thị này được gọi là không liên thông (disconnected).

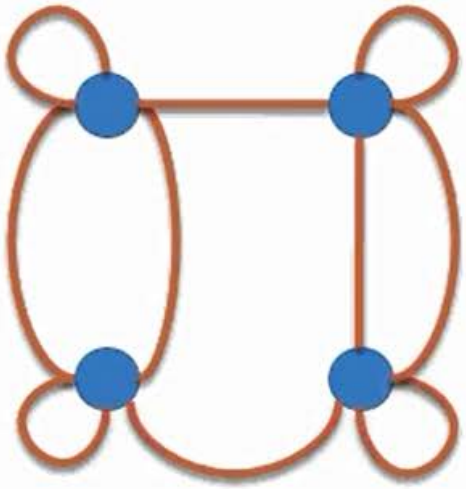
Ví dụ về đồ thị liên thông







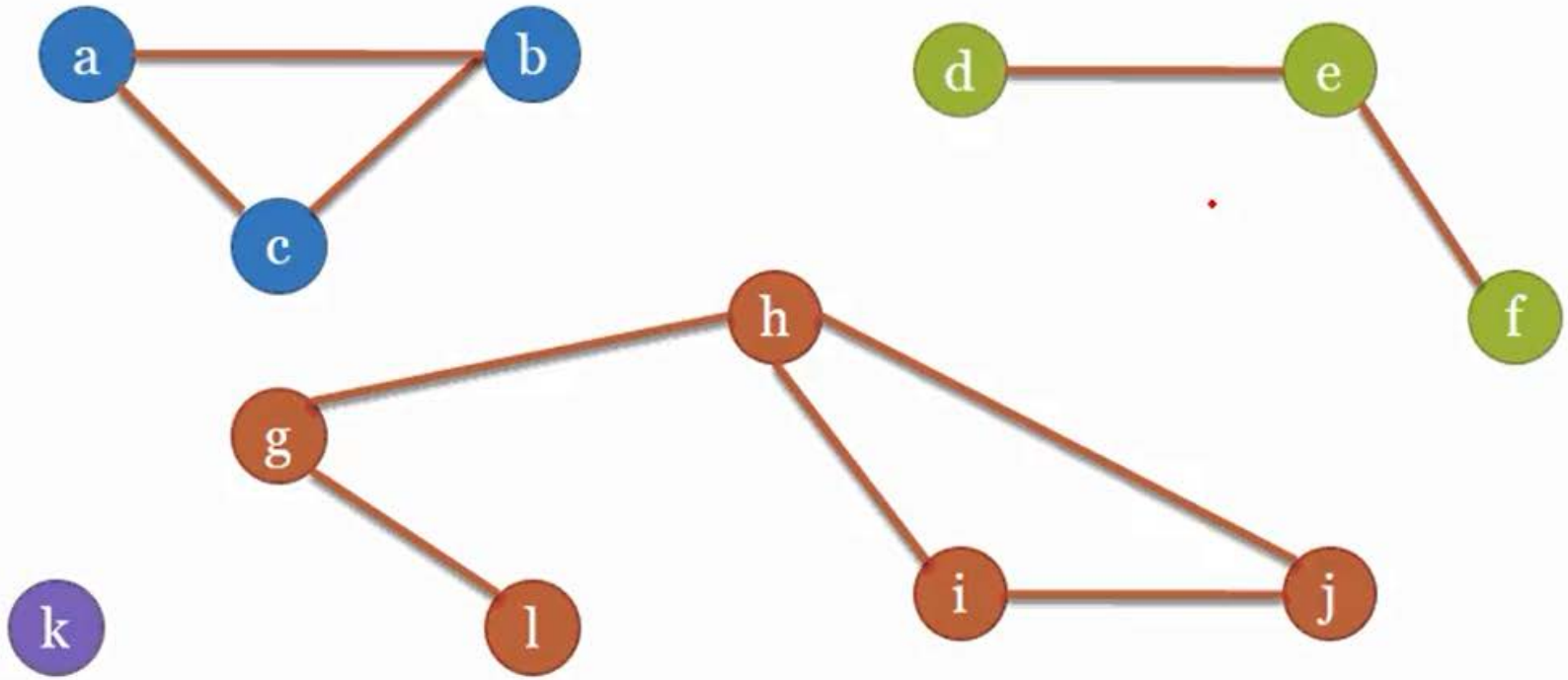




Thành phần liên thông

- Một đồ thị không liên thông sẽ bao gồm nhiều đồ thị con liên thông, các đồ thị con này được gọi là các thành phần liên thông (connected component).

Thành phần liên thông



Định lý về đường đi giữa 2 đỉnh bậc lẻ

- Định lý: Nếu một đồ thị G (không quan tâm liên thông hay không) có đúng 2 đỉnh bậc lẻ, chắc chắn sẽ có một đường đi nối 2 đỉnh này.

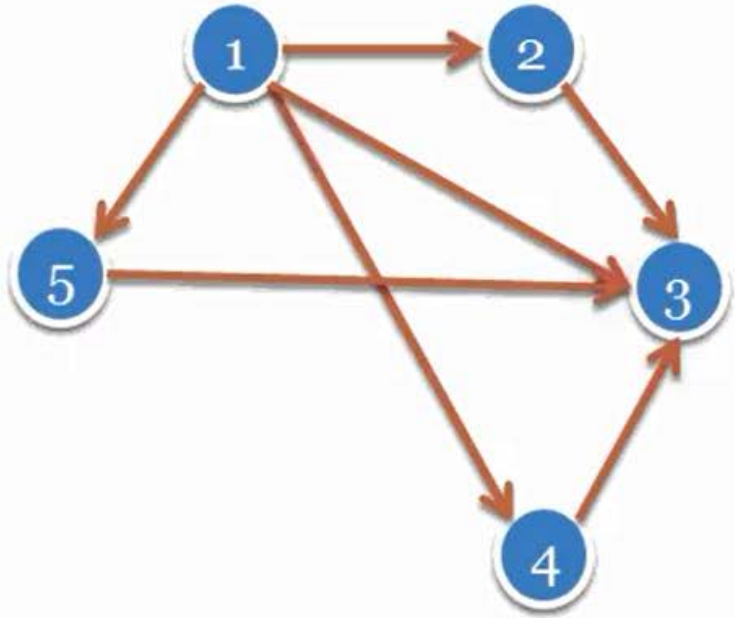
Chứng minh:

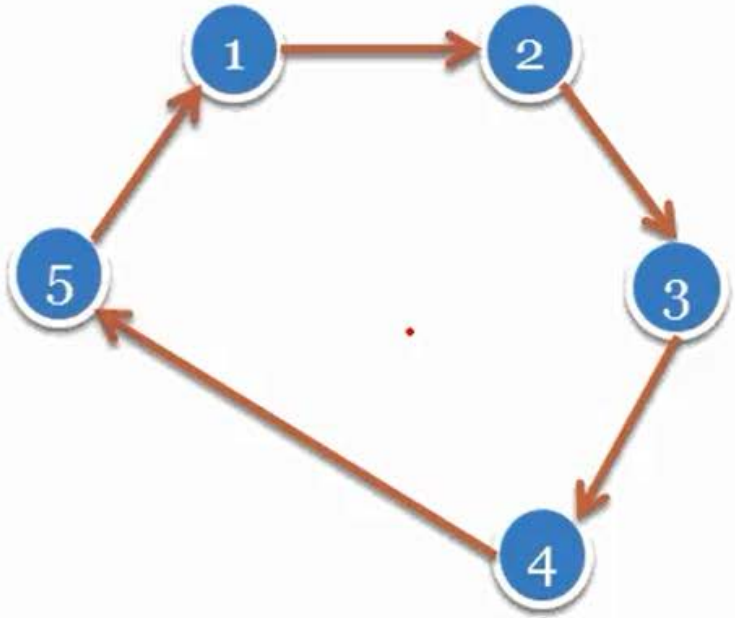
- TH1: G liên thông: rõ ràng có đường nối 2 đỉnh bậc lẻ này (do định nghĩa của đồ thị liên thông).
- TH2: G không liên thông: các thành phần liên thông của G là một đồ thị. Do đó, theo định lý về số đỉnh bậc lẻ, 2 đỉnh này phải thuộc về cùng một thành phần liên thông. Do vậy, phải có đường nối.

Đồ thị liên thông có hướng

- Đồ thị có hướng gọi là *liên thông mạnh* (strongly connected) nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị.
- Đồ thị có hướng gọi là *liên thông yếu* (weakly connected) nếu có đường đi giữa 2 đỉnh bất kỳ của đồ thị vô hướng nền.
- Đồ thị có hướng được gọi là liên thông một phần (unilaterally connected) nếu với mọi cặp đỉnh a, b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại.

Ví dụ đồ thị liên thông có hướng

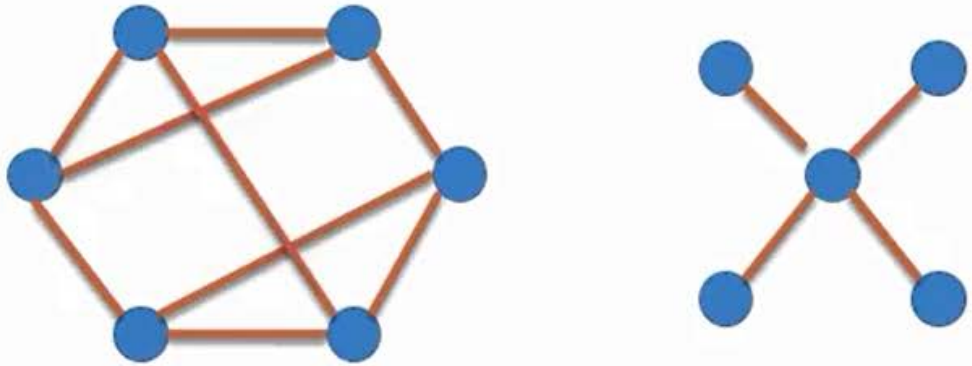




Đỉnh khớp, đồ thị song liên thông

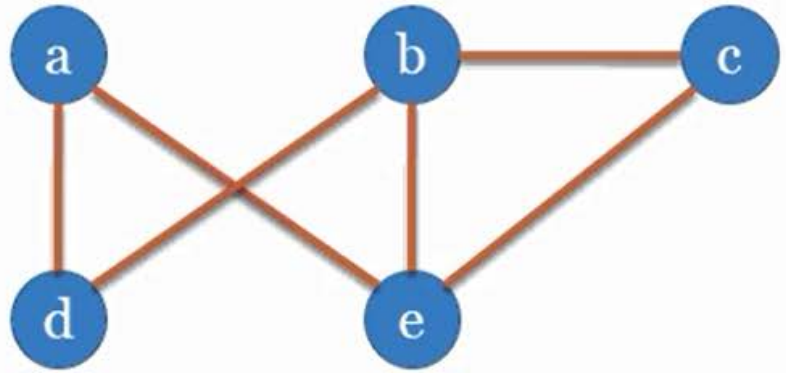
- Đỉnh khớp (cut vertex/ articulation point) của một đồ thị vô hướng là đỉnh mà nếu xóa đỉnh này khỏi đồ thị và các cạnh nối đến nó thì số thành phần liên thông của đồ thị sẽ tăng thêm.
- Cạnh cầu (bridge) của một đồ thị vô hướng là cạnh mà nếu xóa đi khỏi đồ thị thì số thành phần liên thông của đồ thị sẽ tăng thêm.
- Đồ thị song liên thông (biconnectivity) là đồ thị không chứa đỉnh khớp.

Ví dụ về đồ thị song liên thông



Bài tập

1. Mỗi danh sách các đỉnh sau đây có tạo nên đường đi trong đồ thị đã cho hay không? Đường đi nào là đường đi đơn? Đường đi nào là chu trình? Độ dài của các đường đi?
   a) a, e, b, c, d
   b) a, e, a, d, b, c, a
   c) e, b, a, d, b, e
   d) c, b, d, a, e, c



Bài tập

3. Các đồ thị sau đây có phải là đồ thị song liên thông?

a) K_3

b) K_4

c) $K_{2,3}$

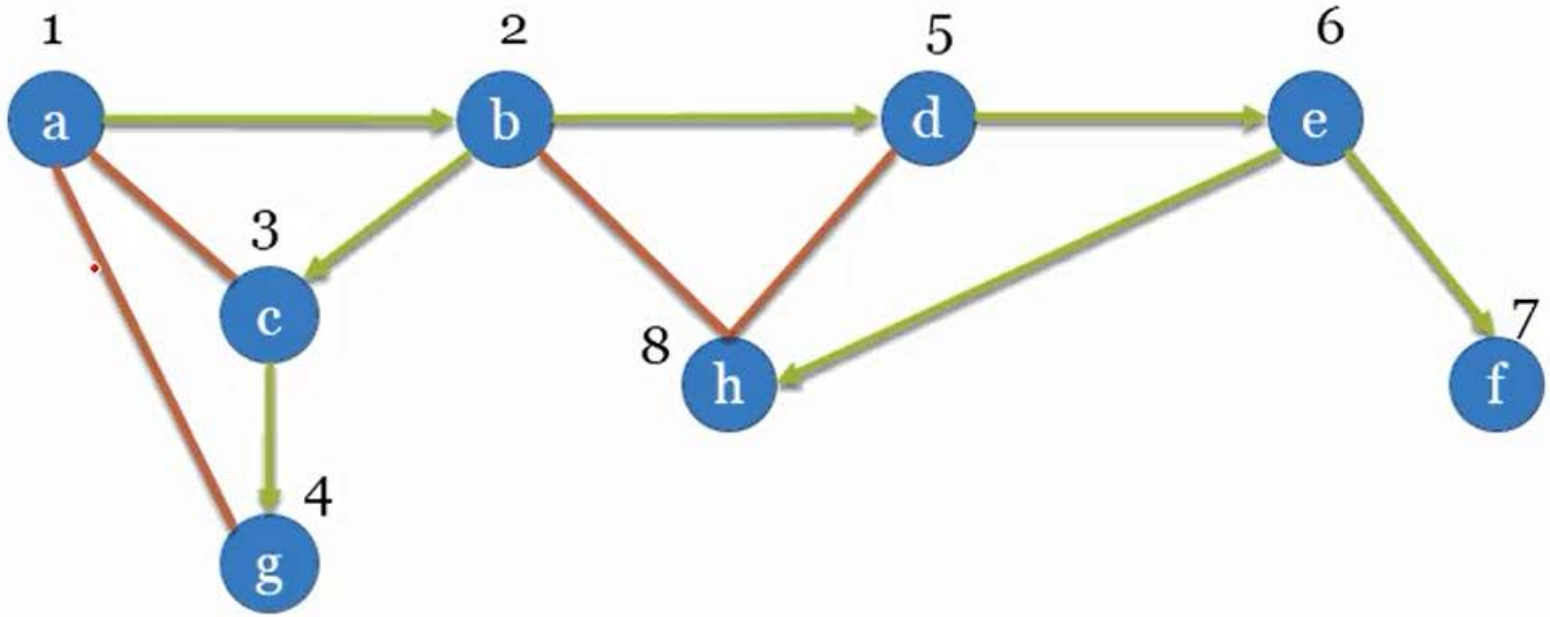
d) C_5

CÁC PHÉP DUYỆT ĐỒ THỊ

Duyệt theo chiều sâu (DFS)

- Depth First Search
- ***Nguyên lý:*** Khởi từ một đỉnh, đi theo các cạnh (cung) xa nhất có thể. Trở lại đỉnh sau của cạnh xa nhất, tiếp tục duyệt như trước cho đến đỉnh cuối cùng.

Ví dụ duyệt theo chiều sâu



Thuật toán duyệt theo chiều sâu

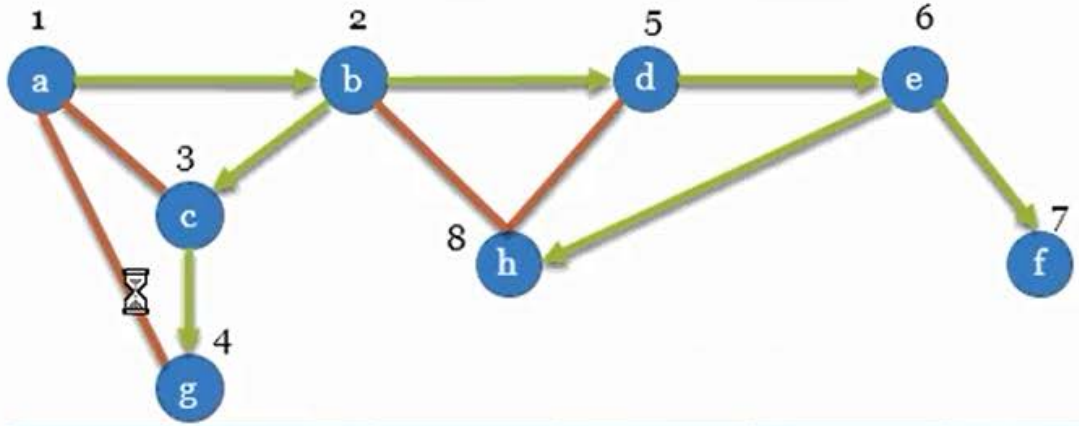
```
procedure DFS(v)
begin
     ThămĐỉnh(v);
     chưaXét[v] = false;
     for u ∈ Kề(V) do
          if chưaXét[u] then
               DFS(u);
end
```

Thuật toán duyệt đồ thị

- Chương trình chính

```
BEGIN
        for v ∈ V do
                chưaXét[v] = true;
        for v ∈ V do
                if chưaXét[v] then
                        DFS(v);//BFS(v)
END.
```

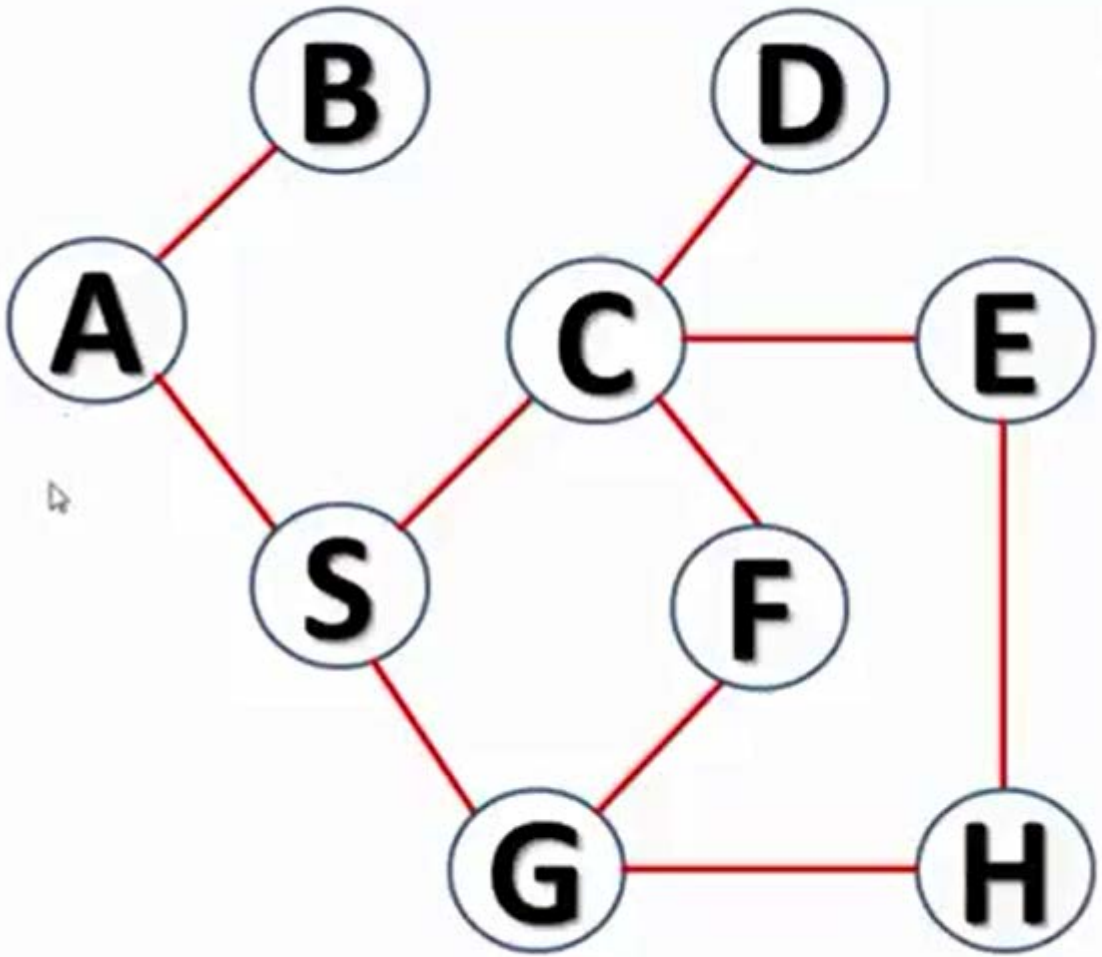
Ví dụ duyệt theo chiều sâu



F: Đỉnh hiện tại
F: Đỉnh chọn được ở bước kế tiếp.

ChưaXét[ ]	a	b	c	d	e	f	g	H
Khởi tạo	T	T	T	T	T	T	T	T
Thăm lần 1	F	T	T	T	T	T	T	T
Thăm lần 2	F	F	T	T	T	T	T	T
Thăm lần 3	F	F	F	T	T	T	T	T
Thăm lần 4	F	F	F	T	T	T	F	T
Thăm lần 5	F	F	F	F	T	T	F	T
Thăm lần 6	F	F	F	F	F	T	F	T
Thăm lần 7	F	F	F	F	F	F	F	T
Thăm lần 8	F	F	F	F	F	F	F	F

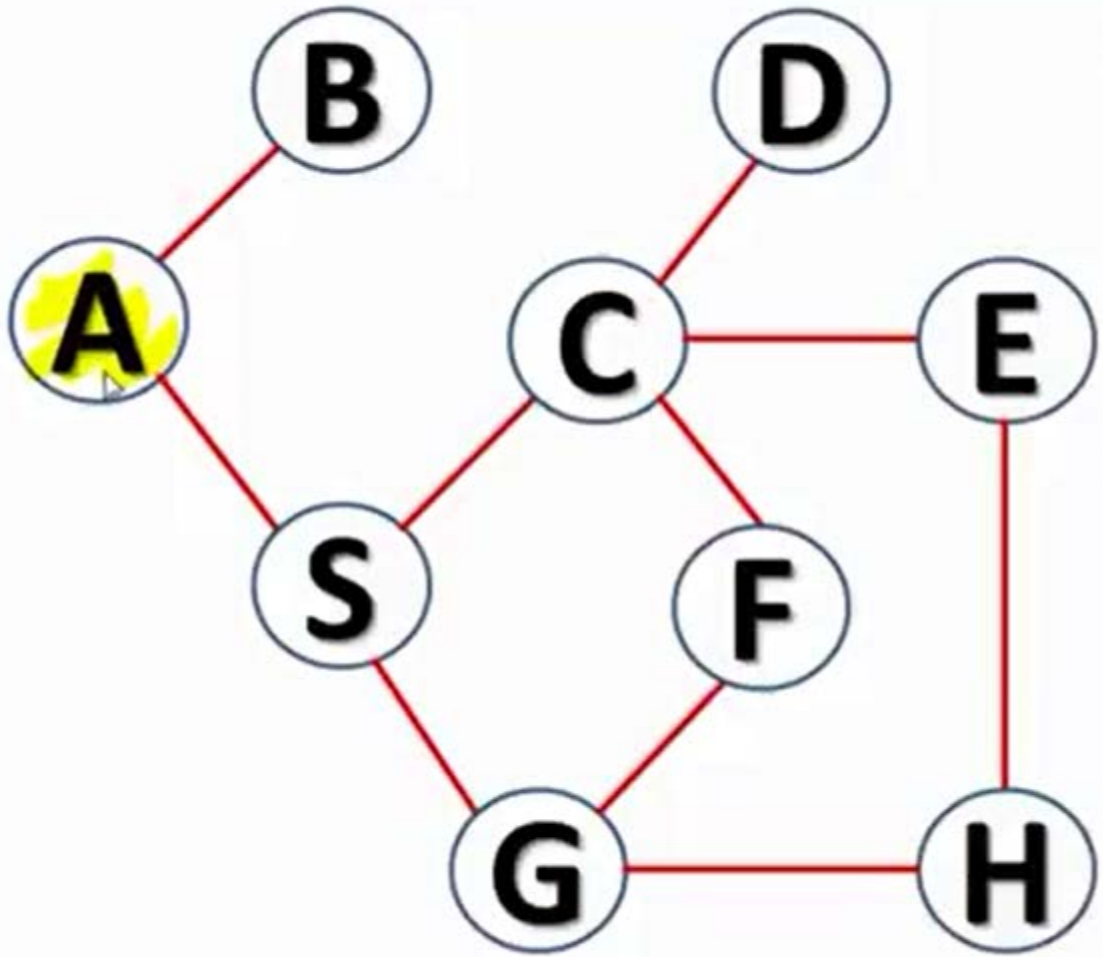
DEPTH FIRST SEARCH



Stack Status

OUTPUT :

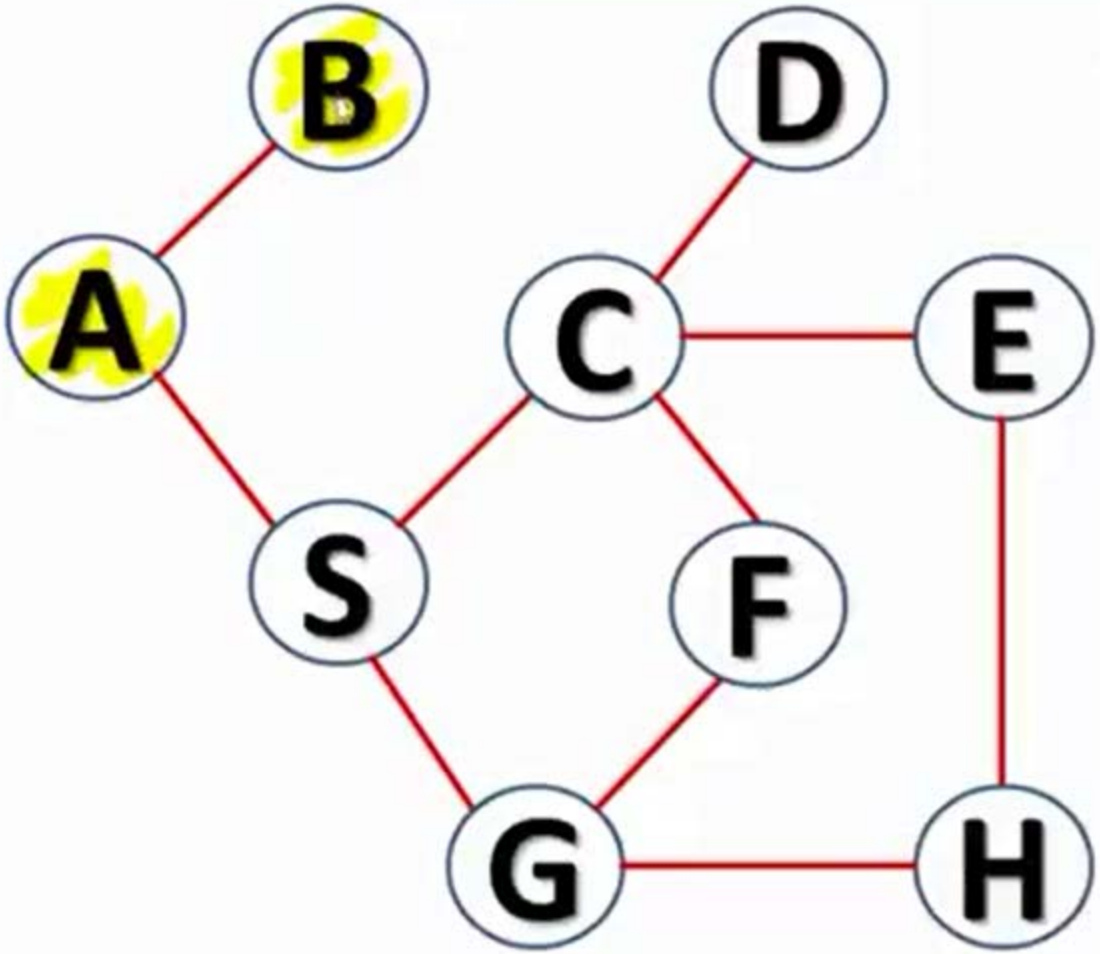
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A

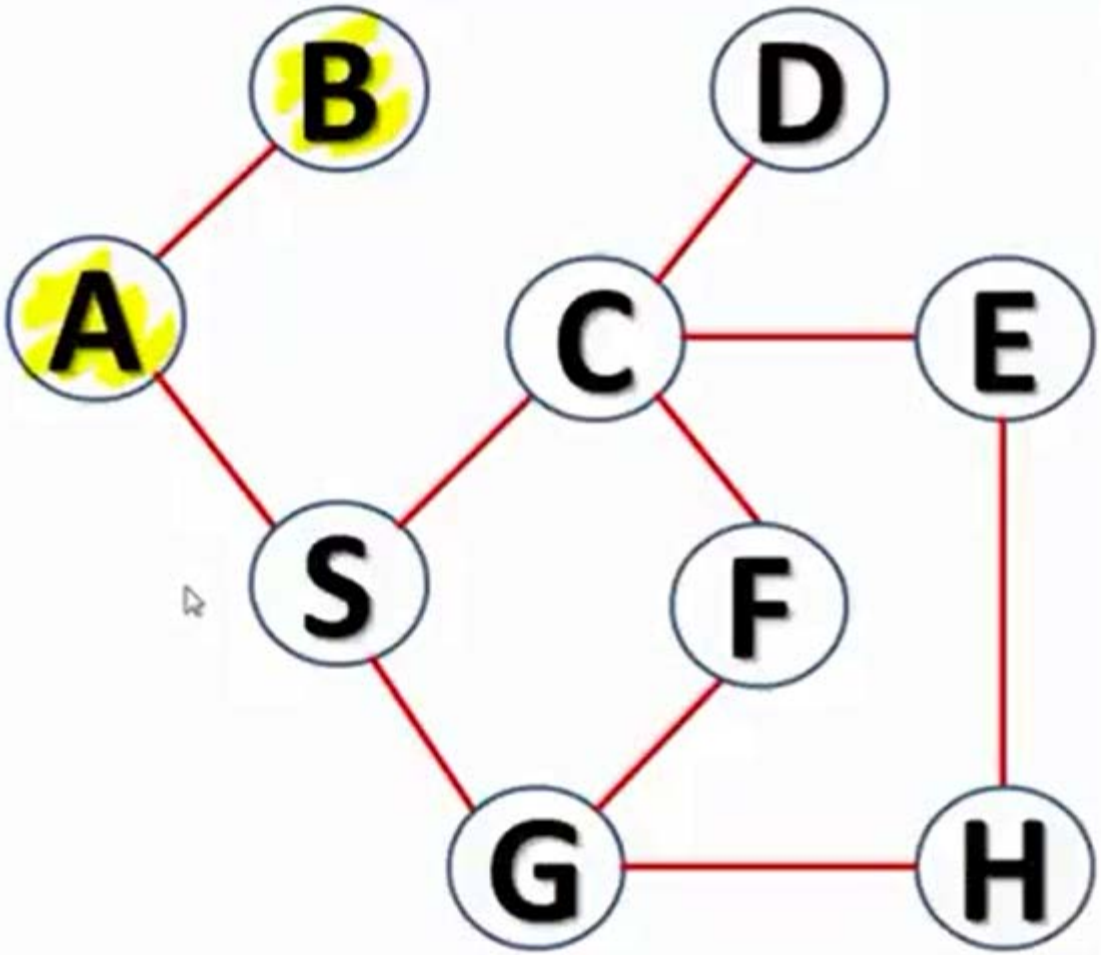
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A B

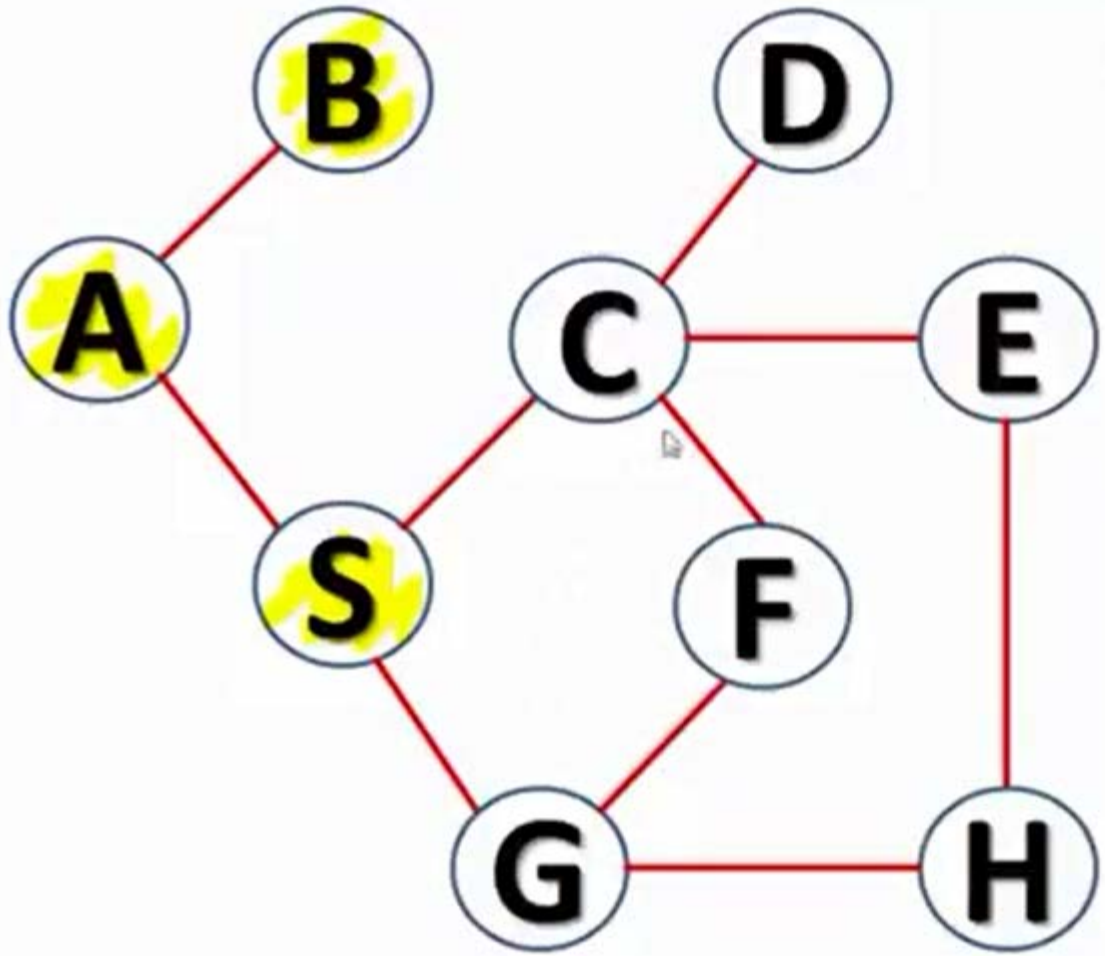
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A B

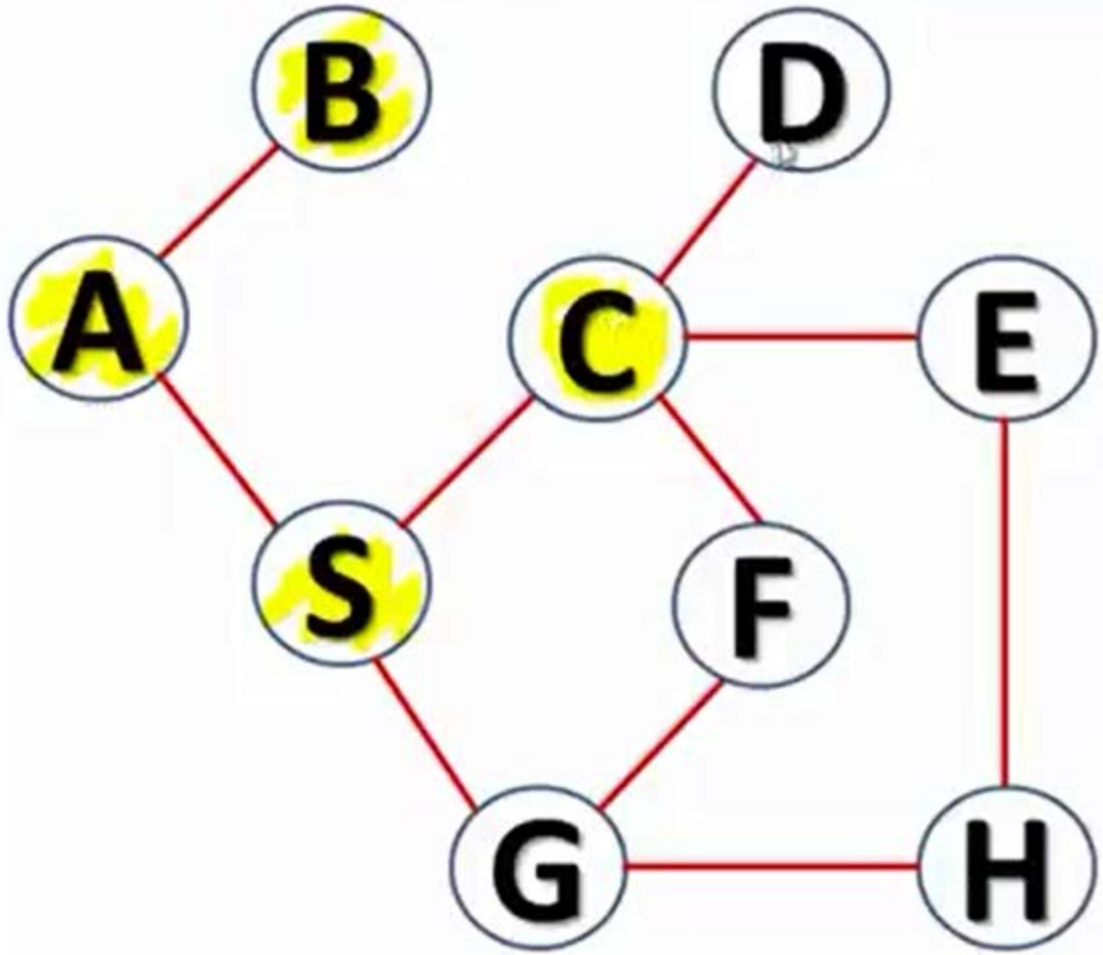
DEPTH FIRST SEARCH



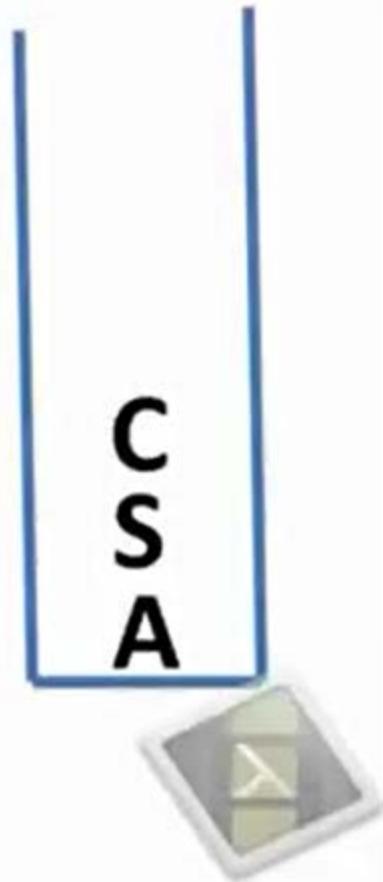
Stack Status

OUTPUT : A B S

DEPTH FIRST SEARCH

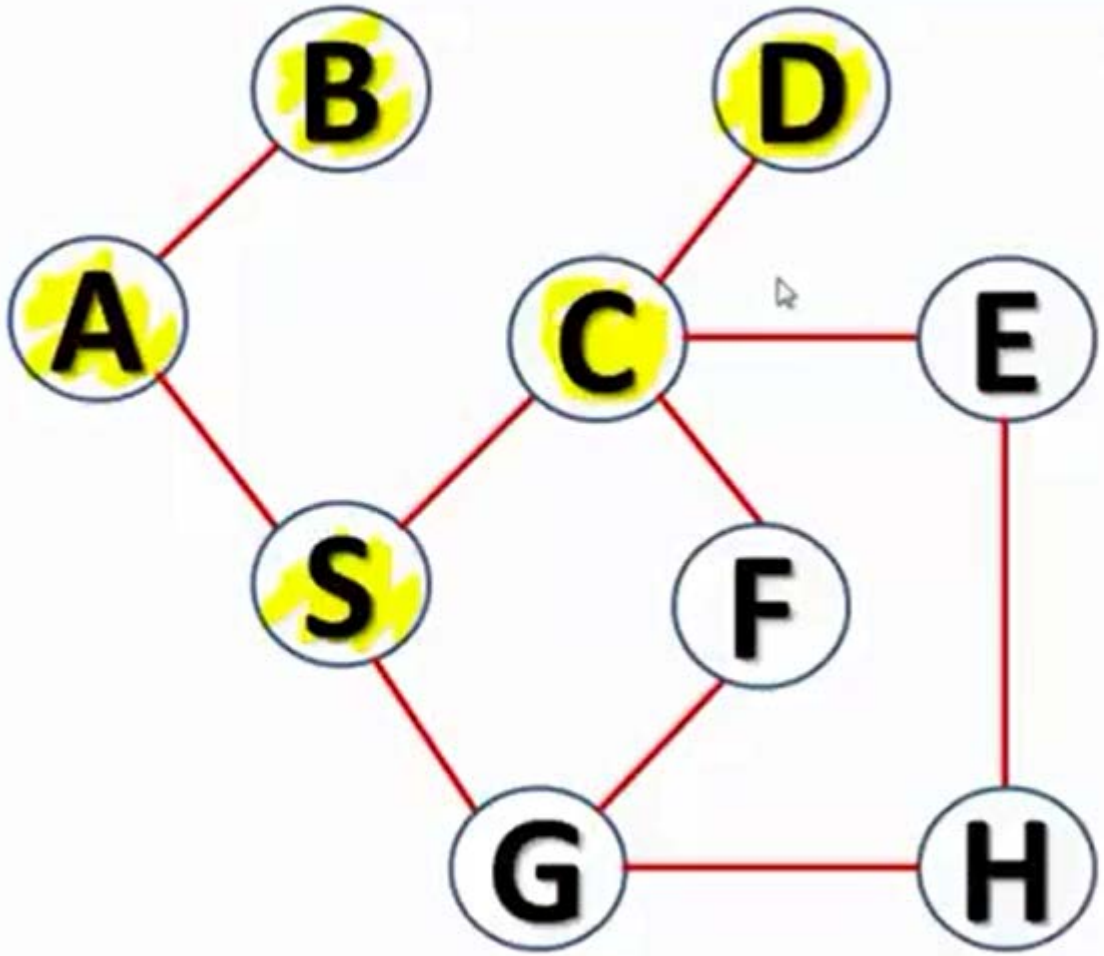


Stack Status



OUTPUT : A B S C

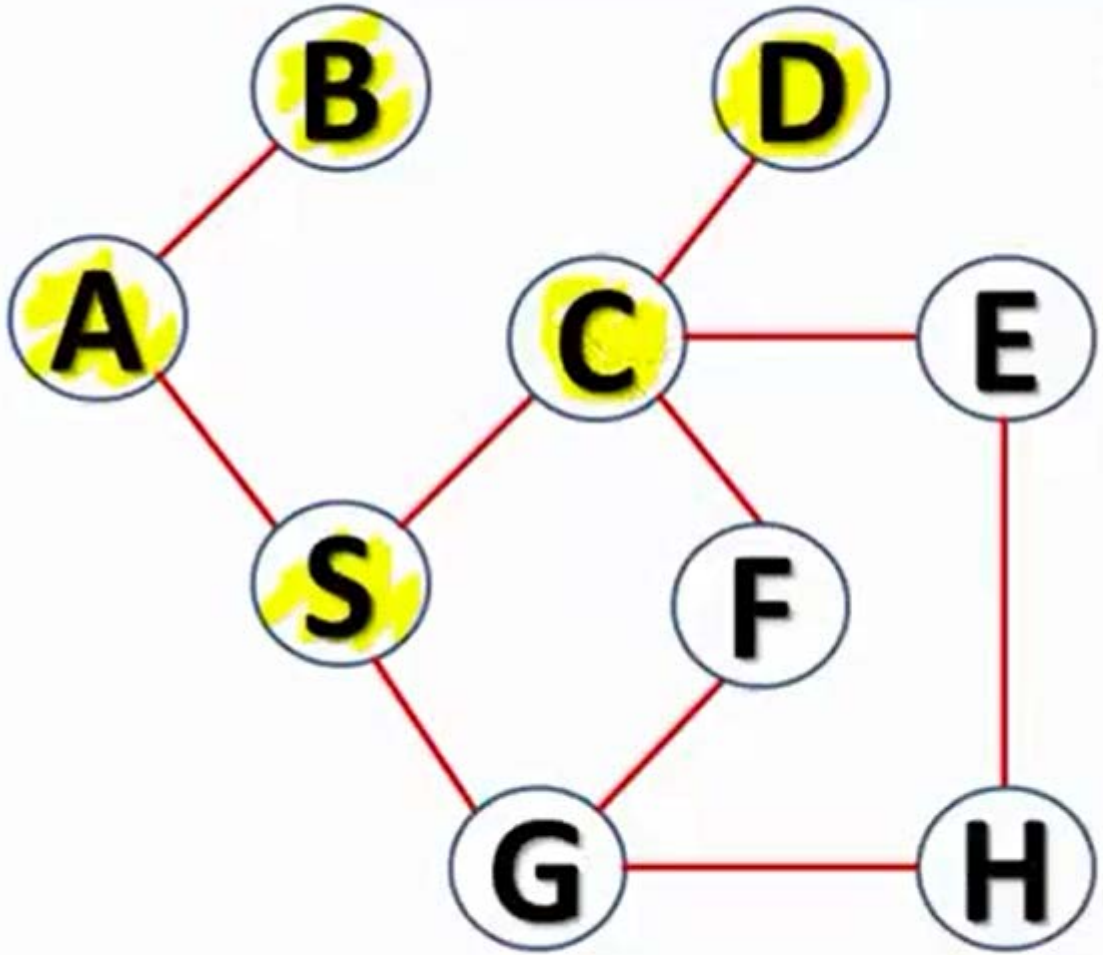
DEPTH FIRST SEARCH



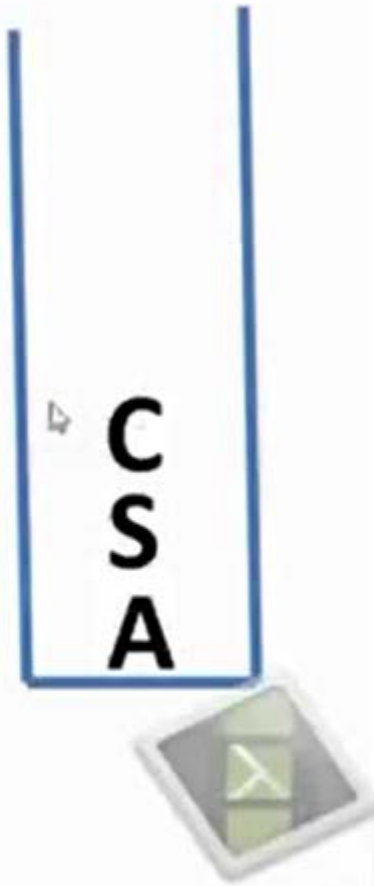
Stack Status

OUTPUT : A B S C D

DEPTH FIRST SEARCH

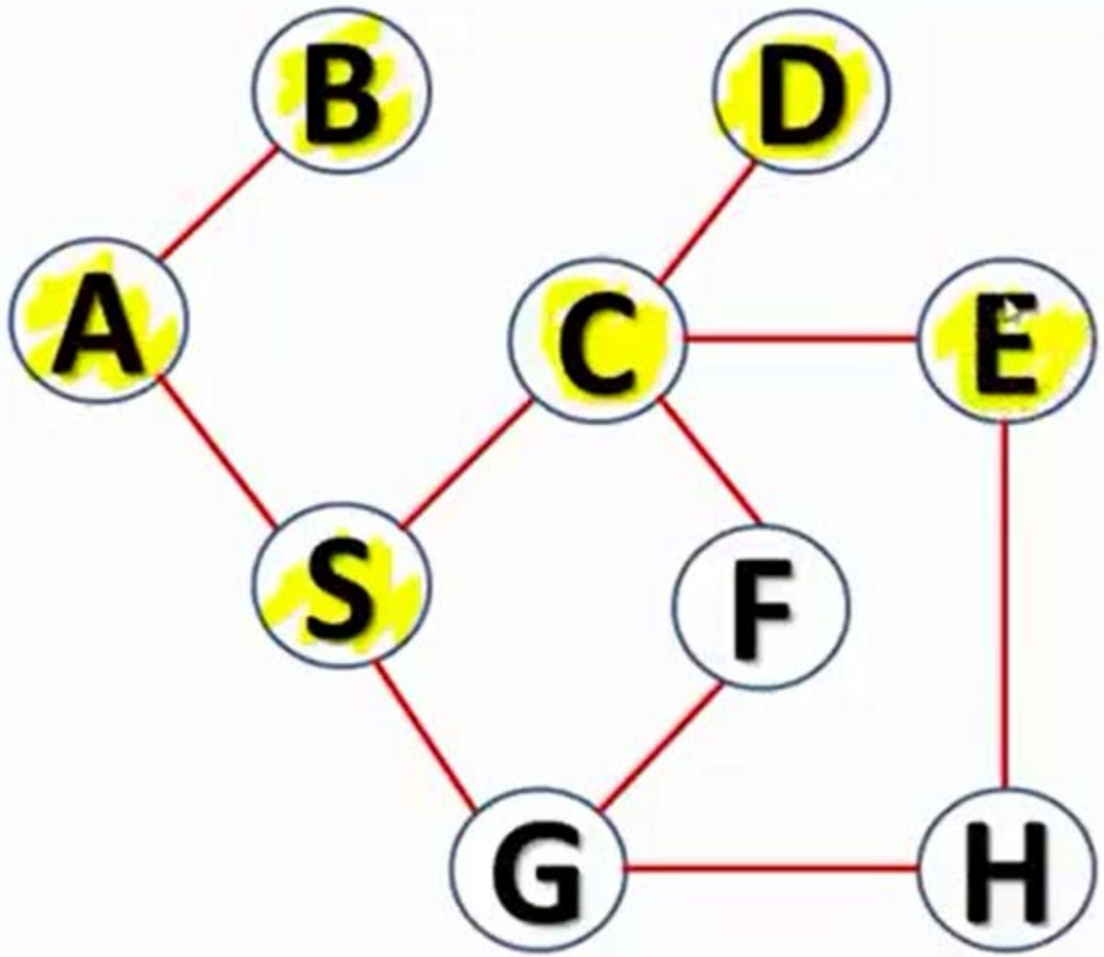


Stack Status



OUTPUT : A B S C D

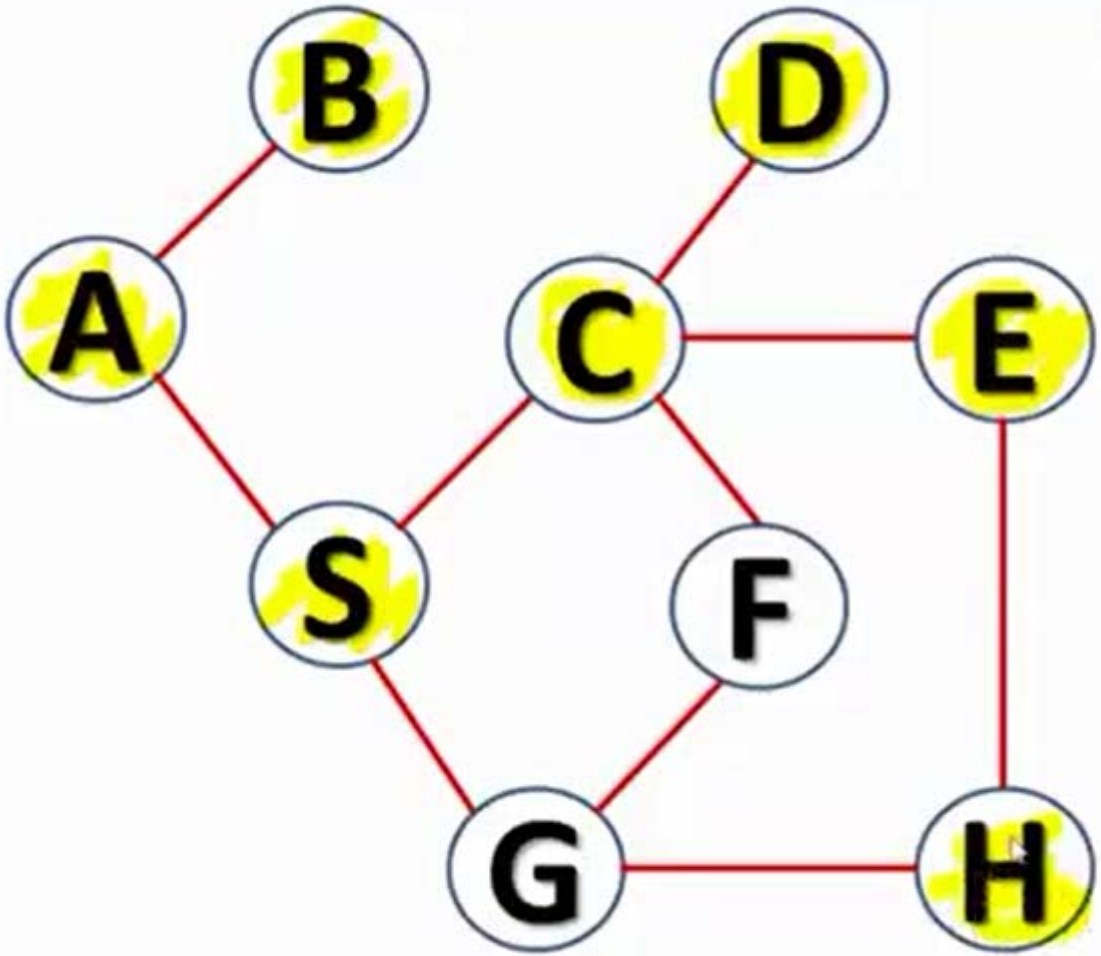
DEPTH FIRST SEARCH



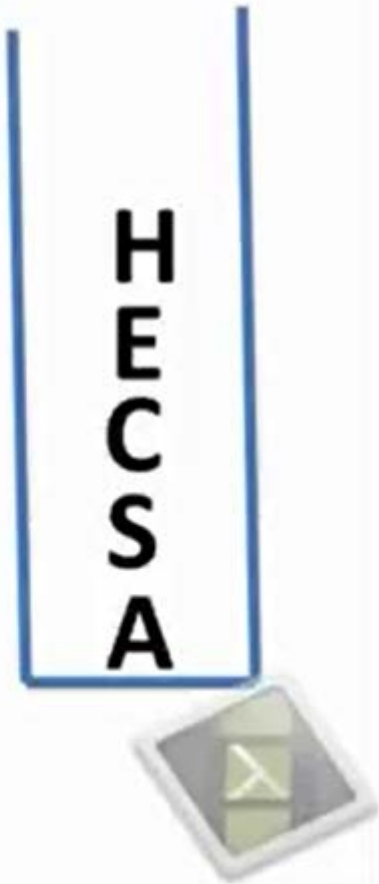
Stack Status

OUTPUT : A B S C D E

DEPTH FIRST SEARCH

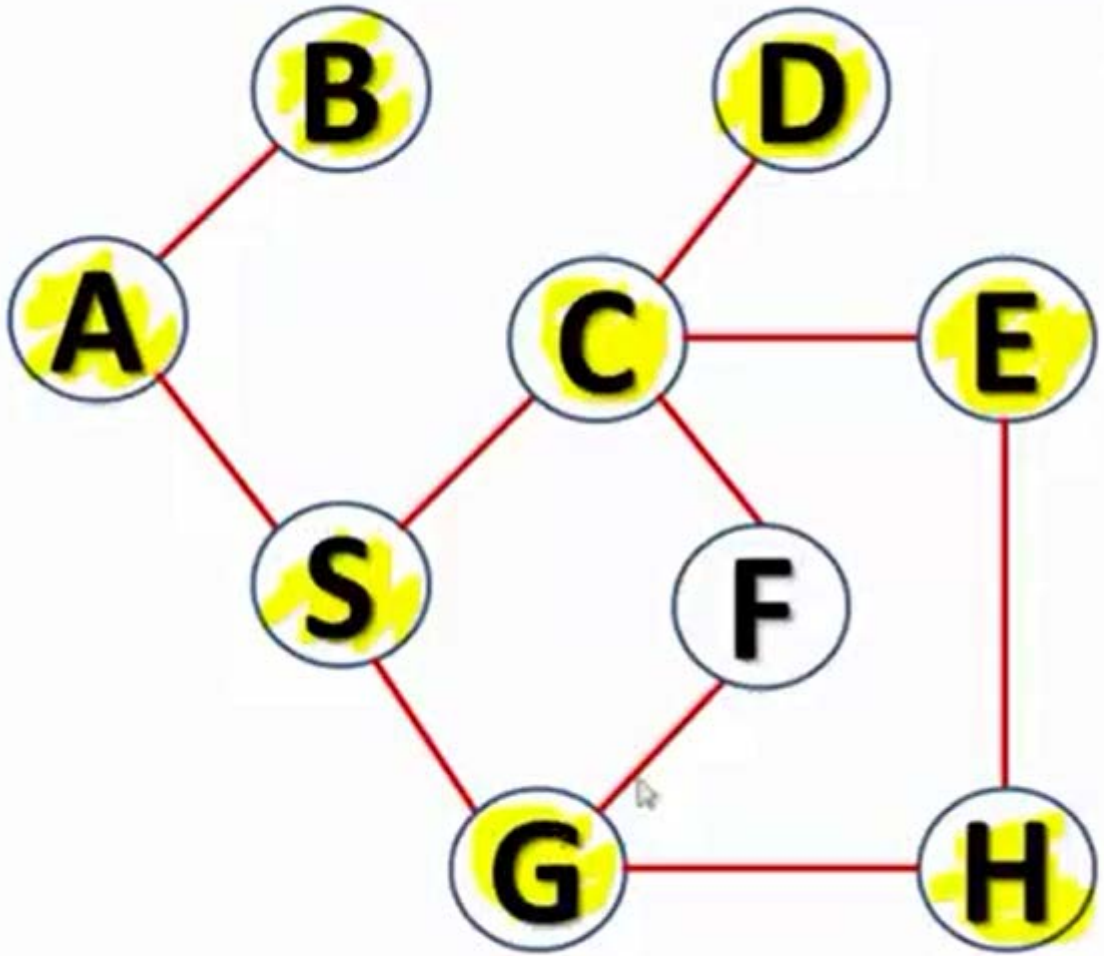


Stack Status



OUTPUT : A B S C D E H

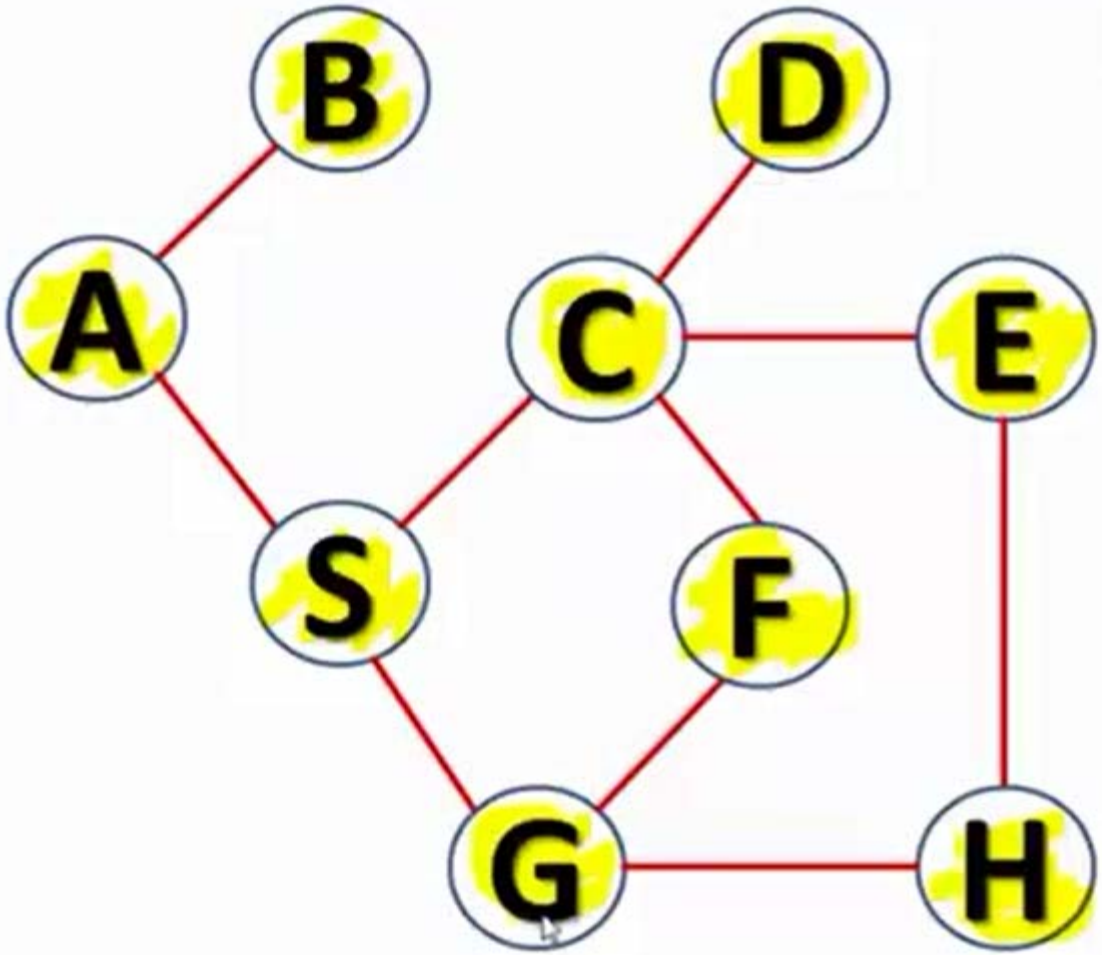
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A B S C D E H G

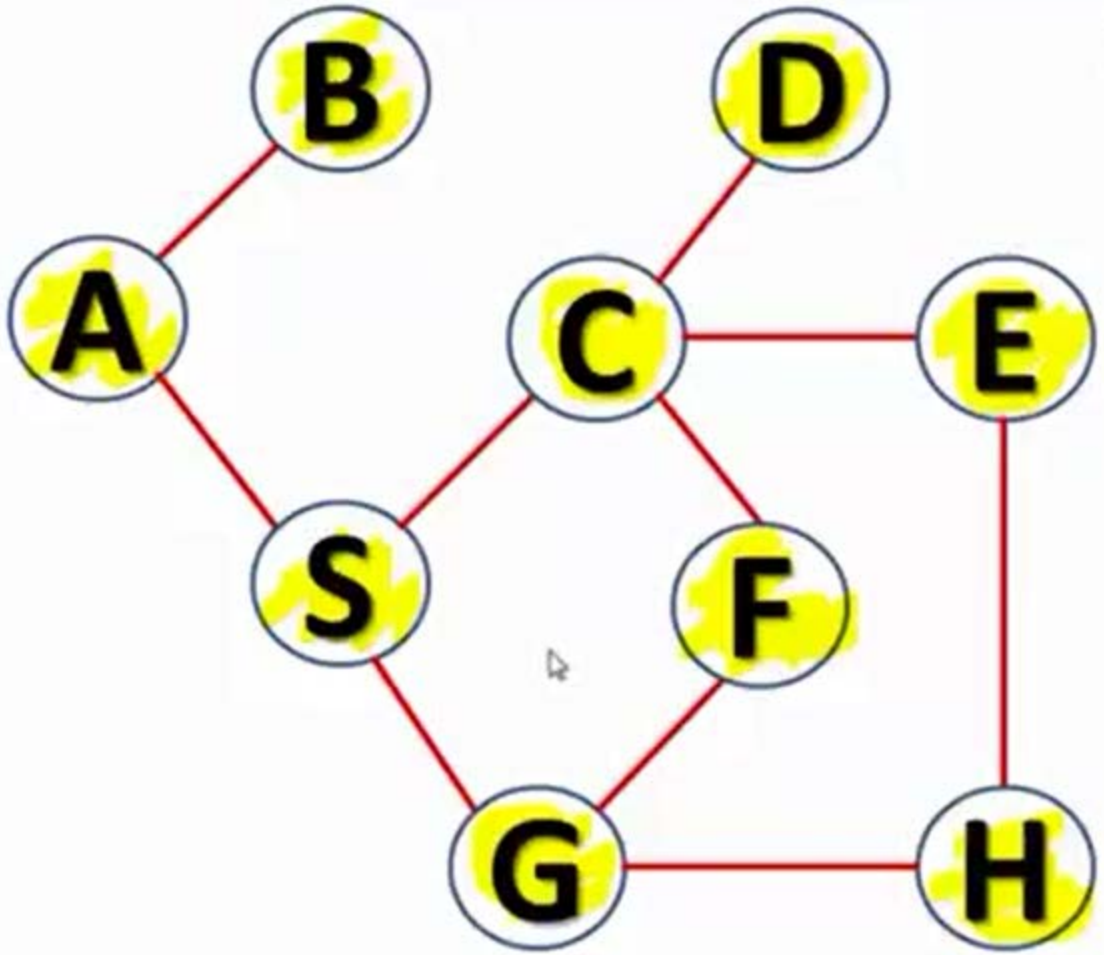
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A B S C D E H G F

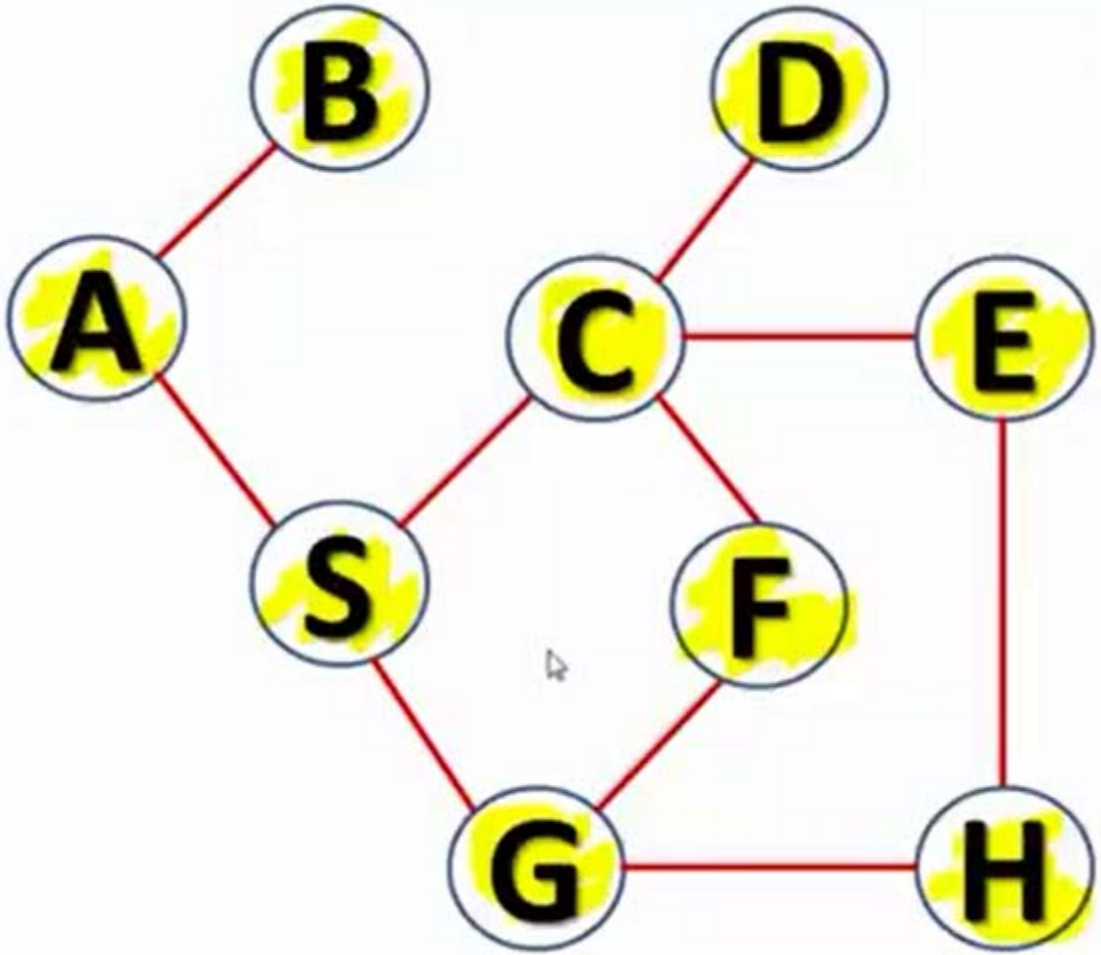
DEPTH FIRST SEARCH



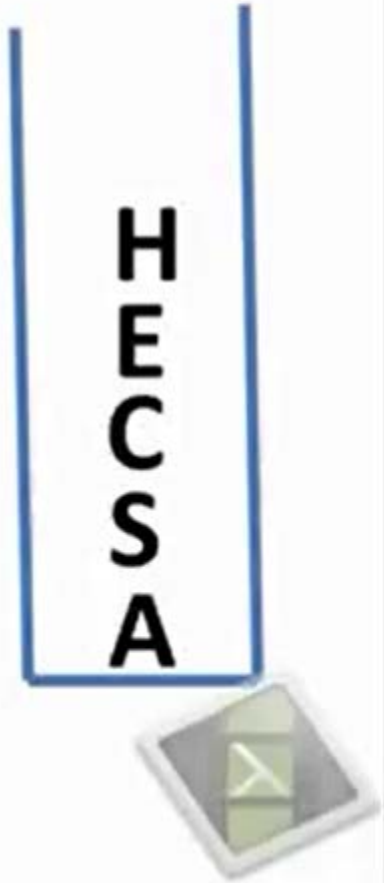
Stack Status

OUTPUT : A B S C D E H G F

DEPTH FIRST SEARCH

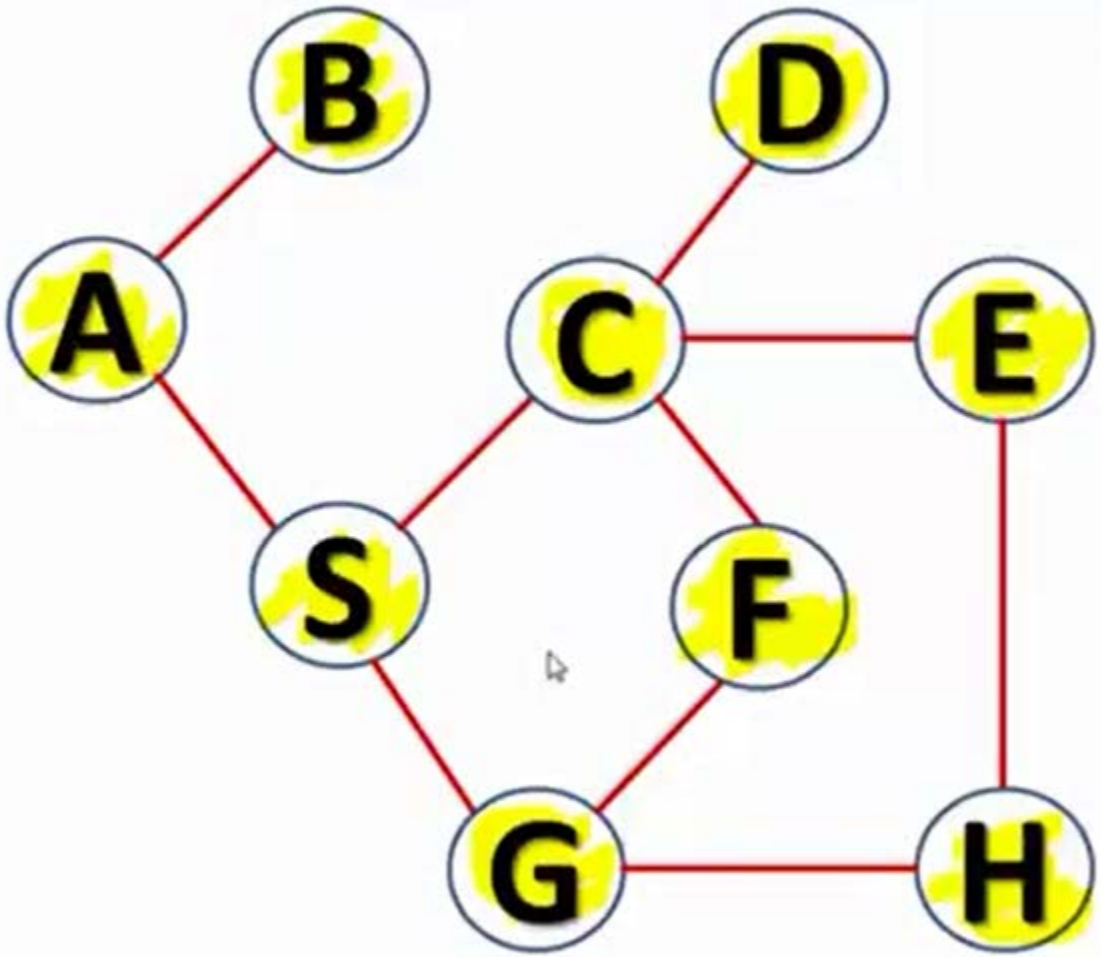


Stack Status



OUTPUT : A B S C D E H G F

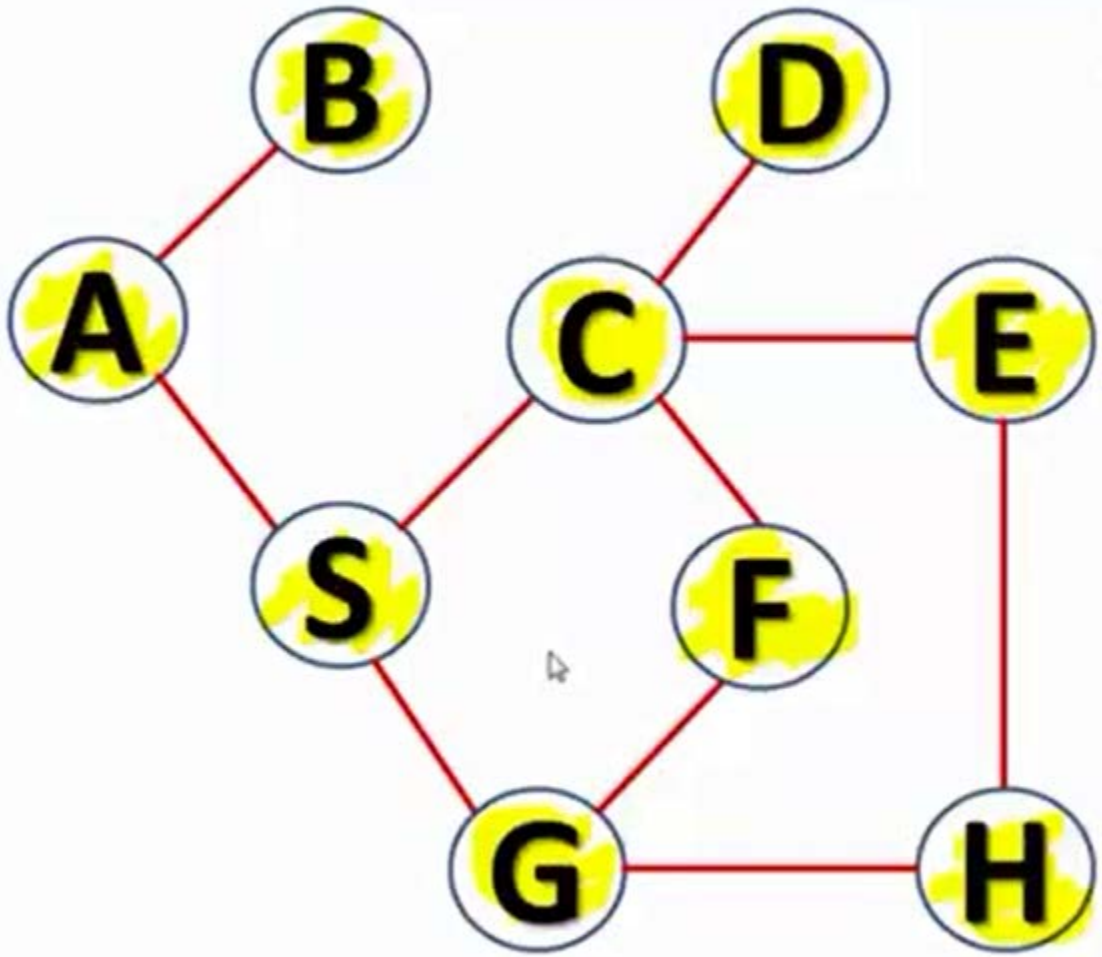
DEPTH FIRST SEARCH



Stack Status

OUTPUT : A B S C D E H G F

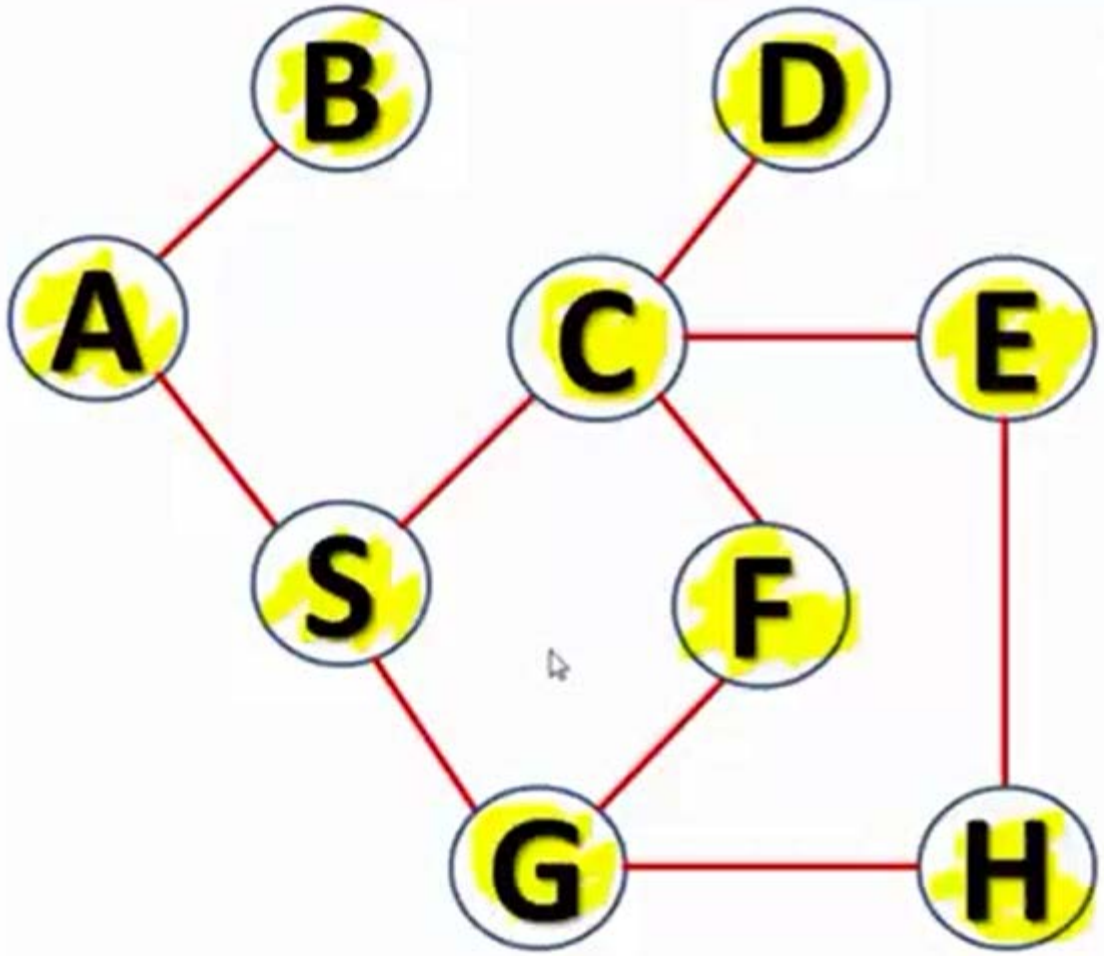
DEPTH FIRST SEARCH



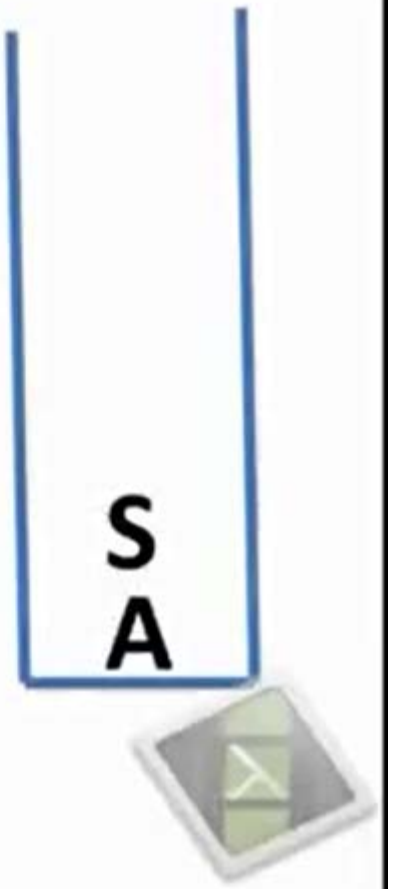
Stack Status

OUTPUT : A B S C D E H G F

DEPTH FIRST SEARCH

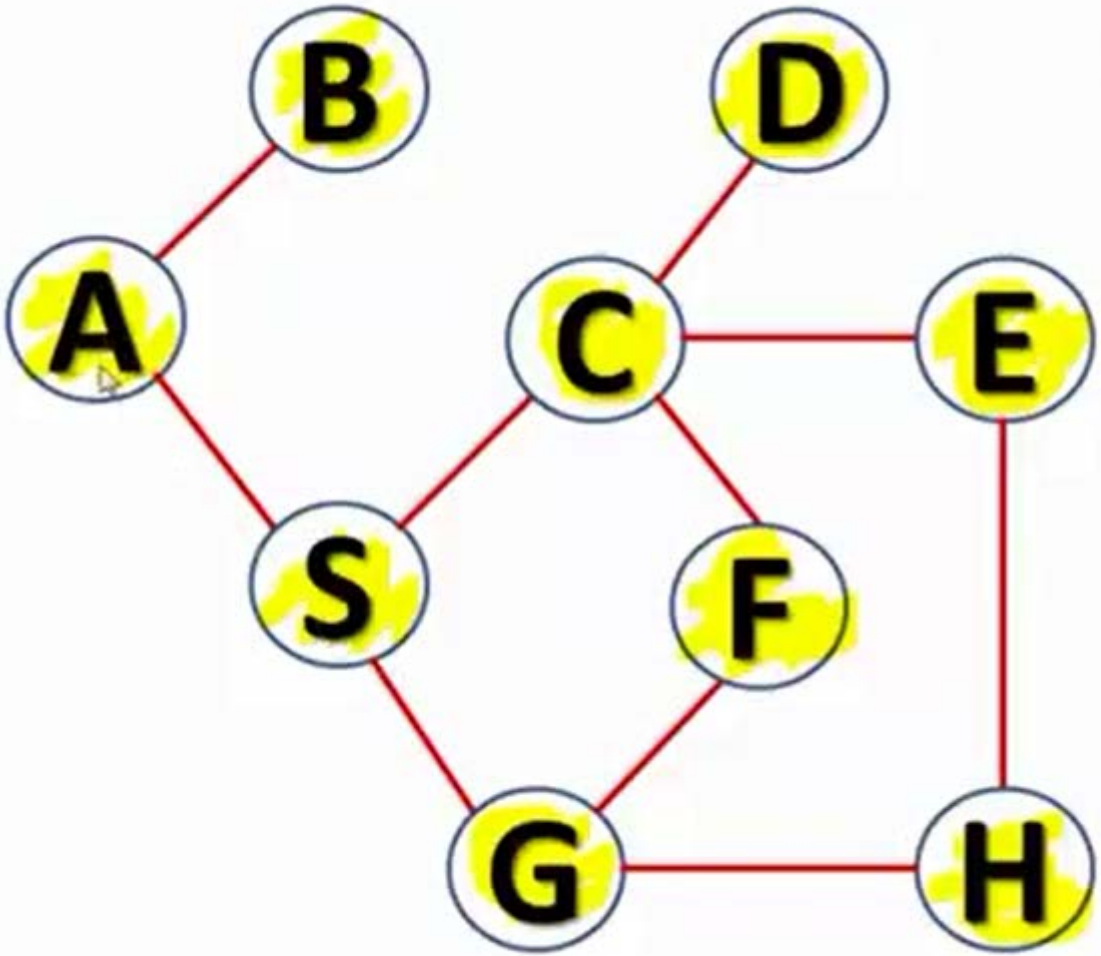


Stack Status



OUTPUT : A B S C D E H G F

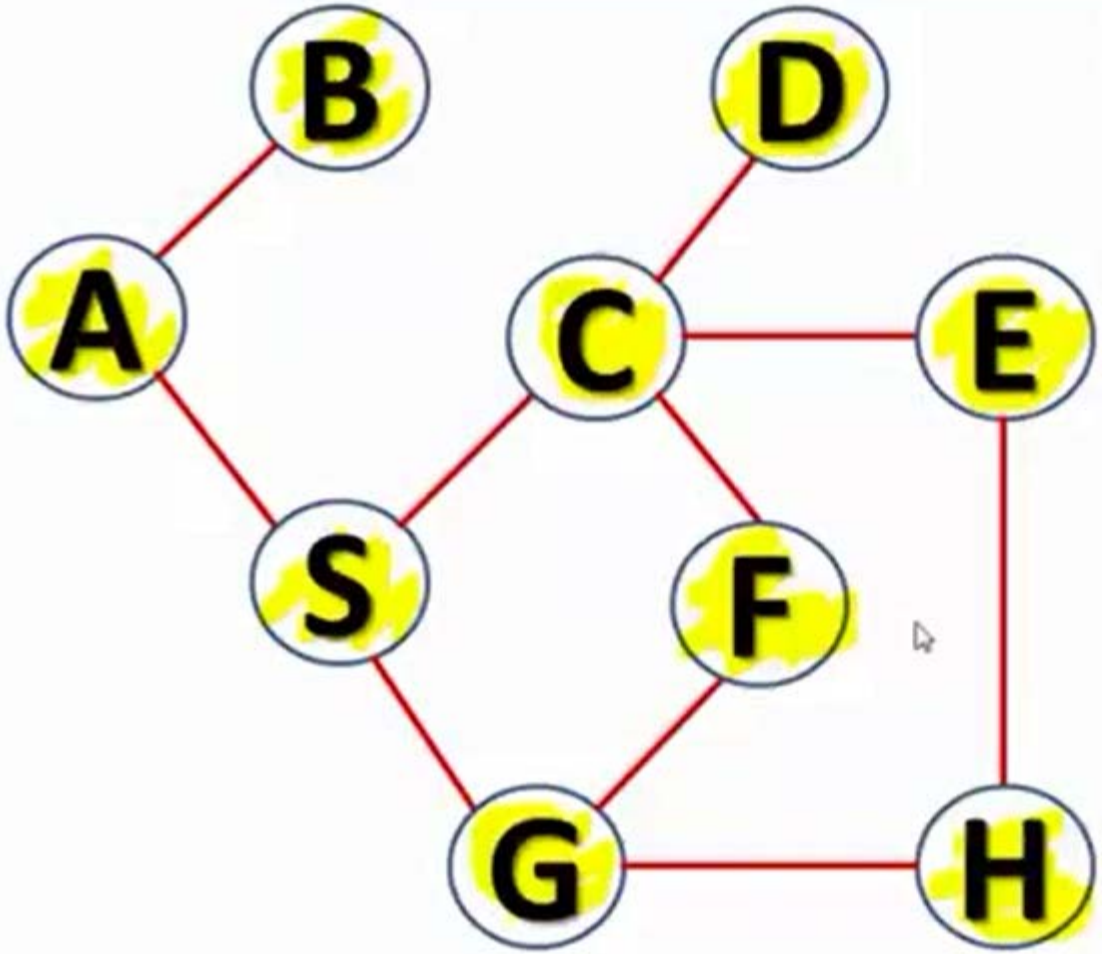
DEPTH FIRST SEARCH



Stack Status

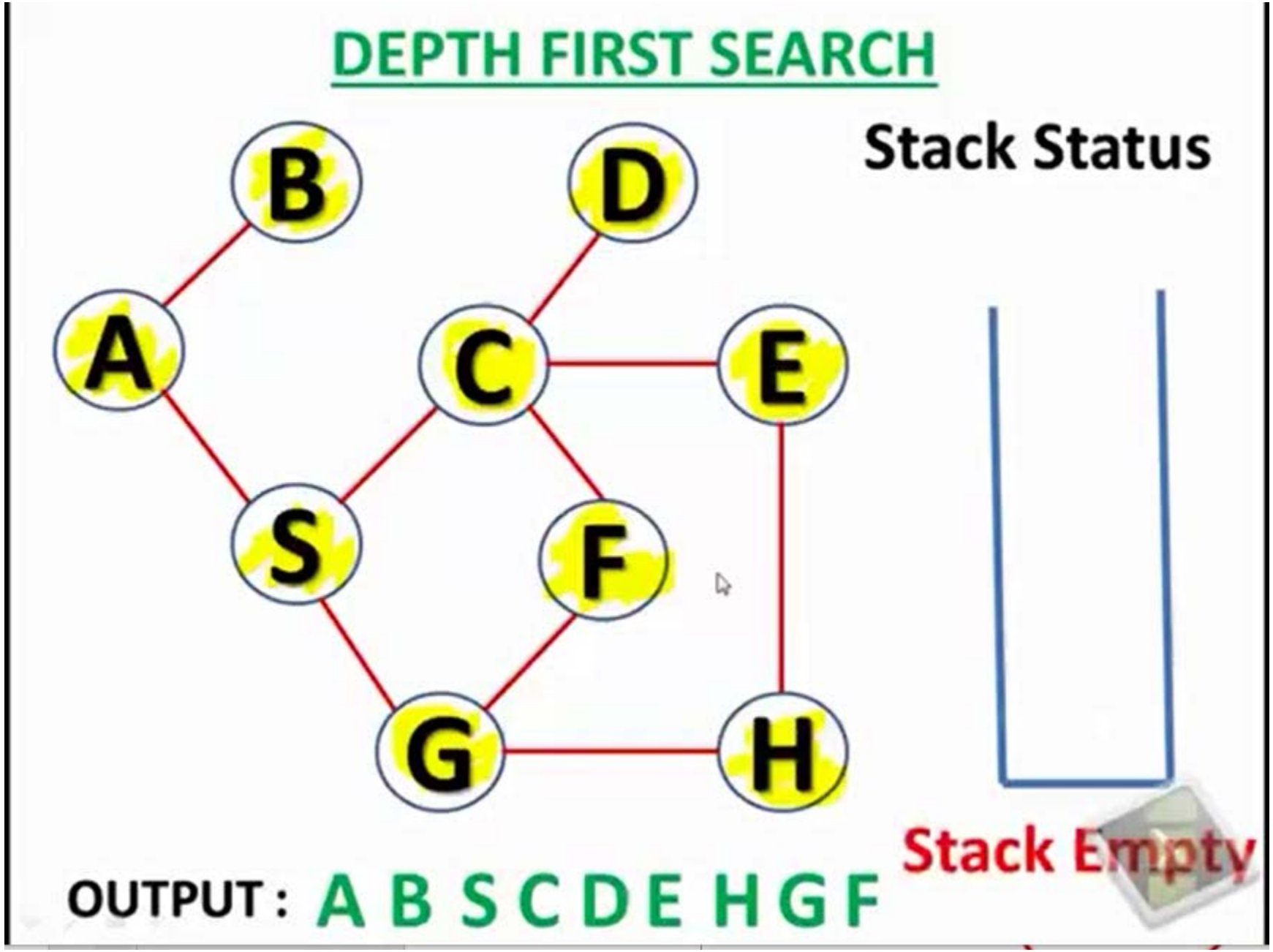
OUTPUT : A B S C D E H G F

DEPTH FIRST SEARCH



Stack Status

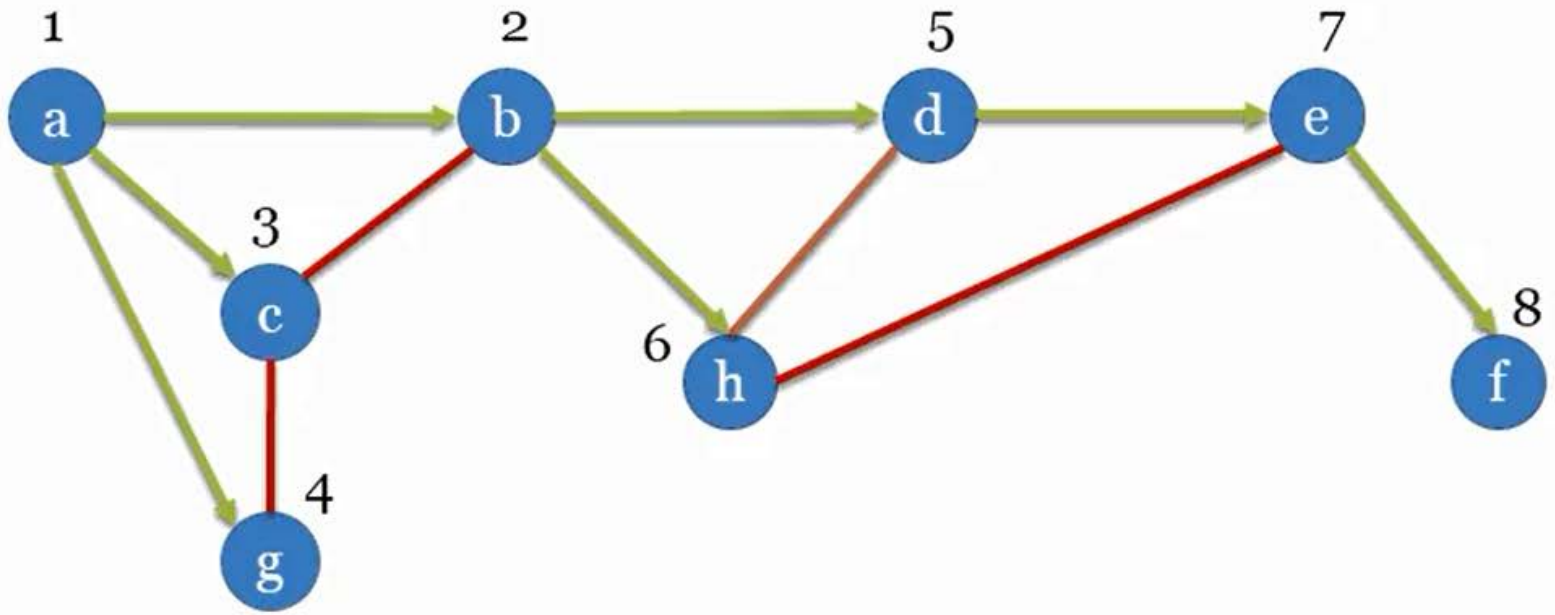
OUTPUT : A B S C D E H G F


DEPTH FIRST SEARCH
Stack Status
B
D
A
C
E
S
F
G
H
Stack Empty
OUTPUT : A B S C D E H G F

Duyệt theo chiều rộng

- Breadth First Search
- **Ý tưởng:** Đỉnh "gần" ưu tiên thăm trước

Ví dụ duyệt theo chiều rộng



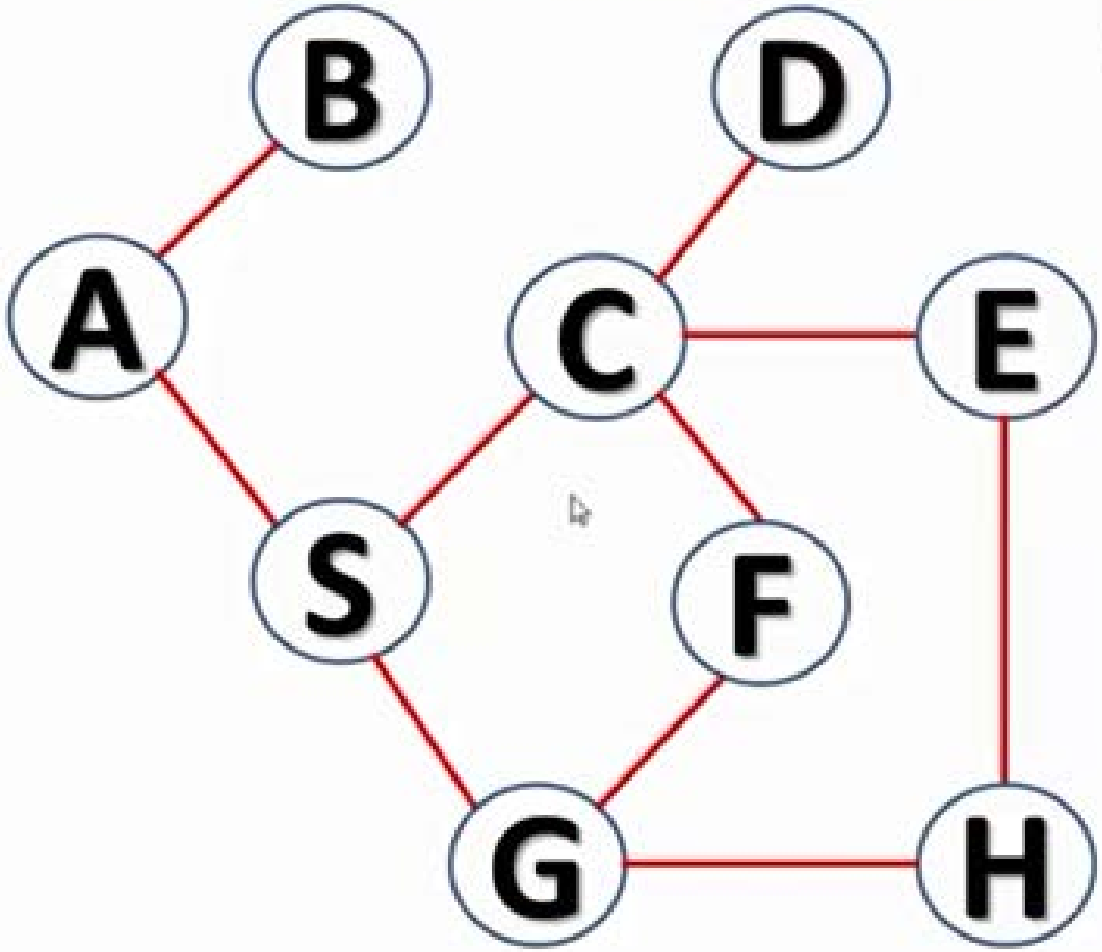
Thuật toán duyệt theo chiều rộng

```
procedure BFS(v)
begin
  QUEUE = ∅;
  QUEUE ← v; // Kết nạp v vào QUEUE
  chưaXét[v] = false;
  while QUEUE ≠ ∅ do
      p ← QUEUE;
      ThămĐỉnh(p);
      for u ∈ Kề(p) do
          if chưaXét[u] then
              QUEUE ← u;
              chưaXét[u] = false;
end;
```

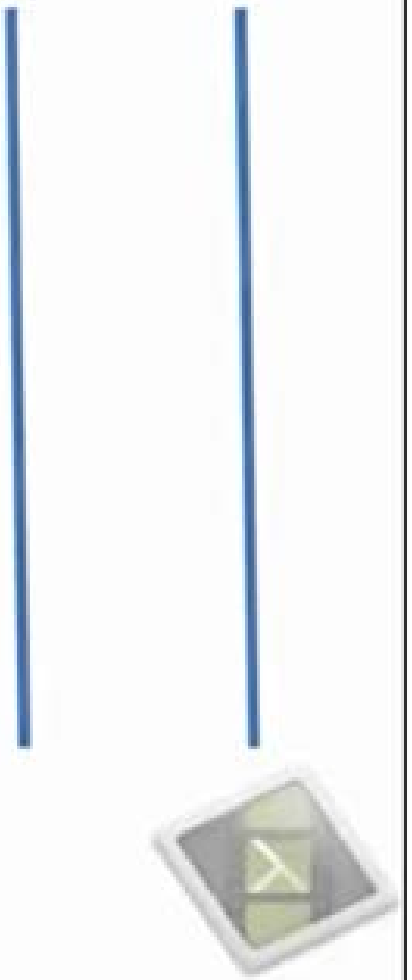
```
void DFS(int DinhV, bool ChuaXet[100], GRAPH G)
{
    int DinhU;
    ThamDinh(DinhV);
    ChuaXet[DinhV] = false;
    for(DinhU=0; DinhU<G.SoDinh; DinhU++)
    {
        //Kiem tra DinhU co ke DinhV
        if(G.MTKe[DinhU][DinhV] != 0)
        {
            if(ChuaXet[DinhU] == true)
            {
                DFS(DinhU, ChuaXet, G);
            }
        }
    }
}
```

```
void main()
{
    int Dinh;
    bool ChuaXet[MAX];
    GRAPH G;
    //Doc du lieu do thi tu tap tin van ban
    for(Dinh=0; Dinh<G.SoDinh; Dinh++)
    {
        ChuaXet[Dinh] = true;
    }
    for(Dinh=0; Dinh<G.SoDinh; Dinh++)
    {
        if(ChuaXet[Dinh] == true)
        {
            DFS(Dinh, ChuaXet, G);
        }
}
```

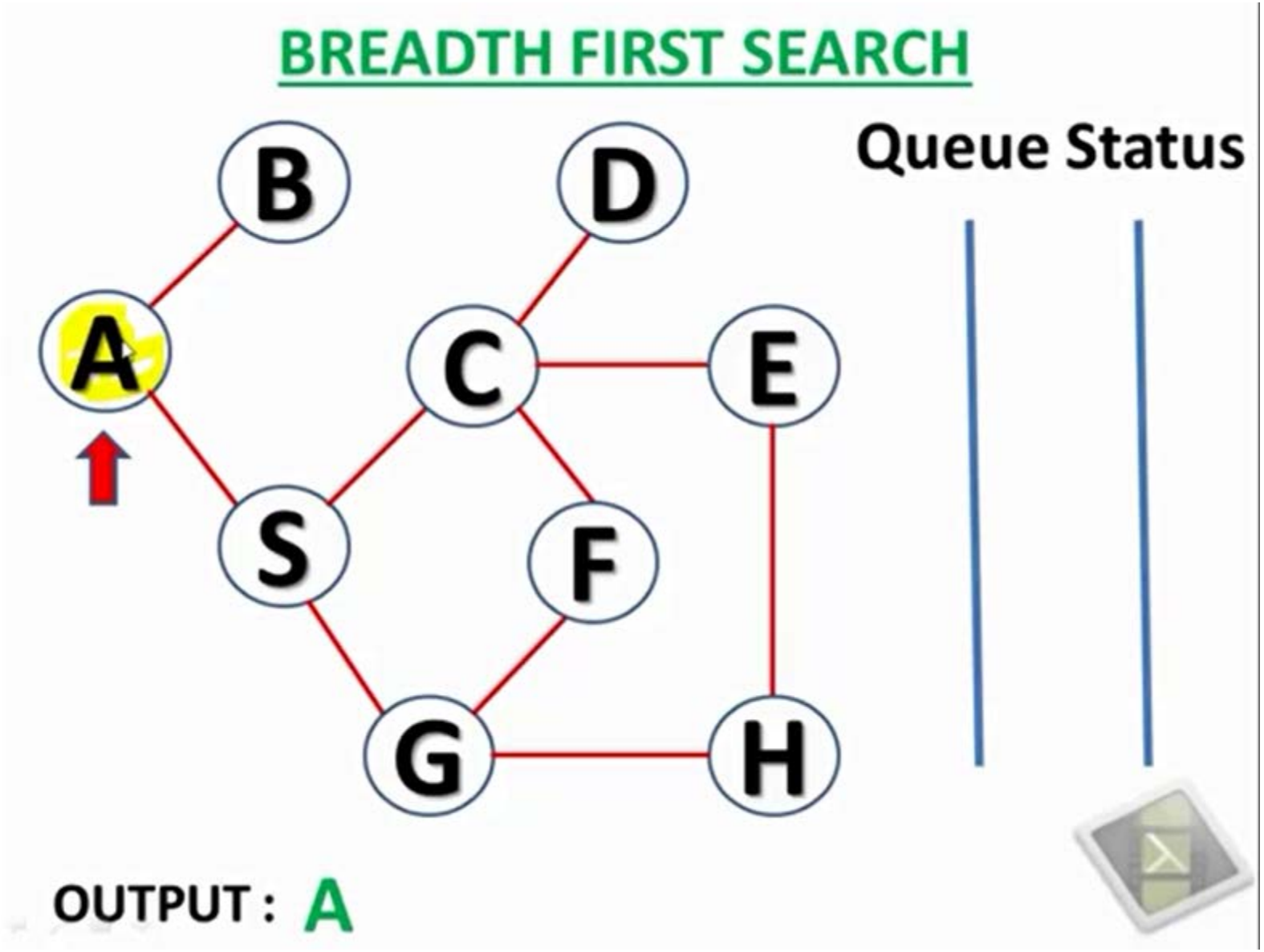
BREADTH FIRST SEARCH

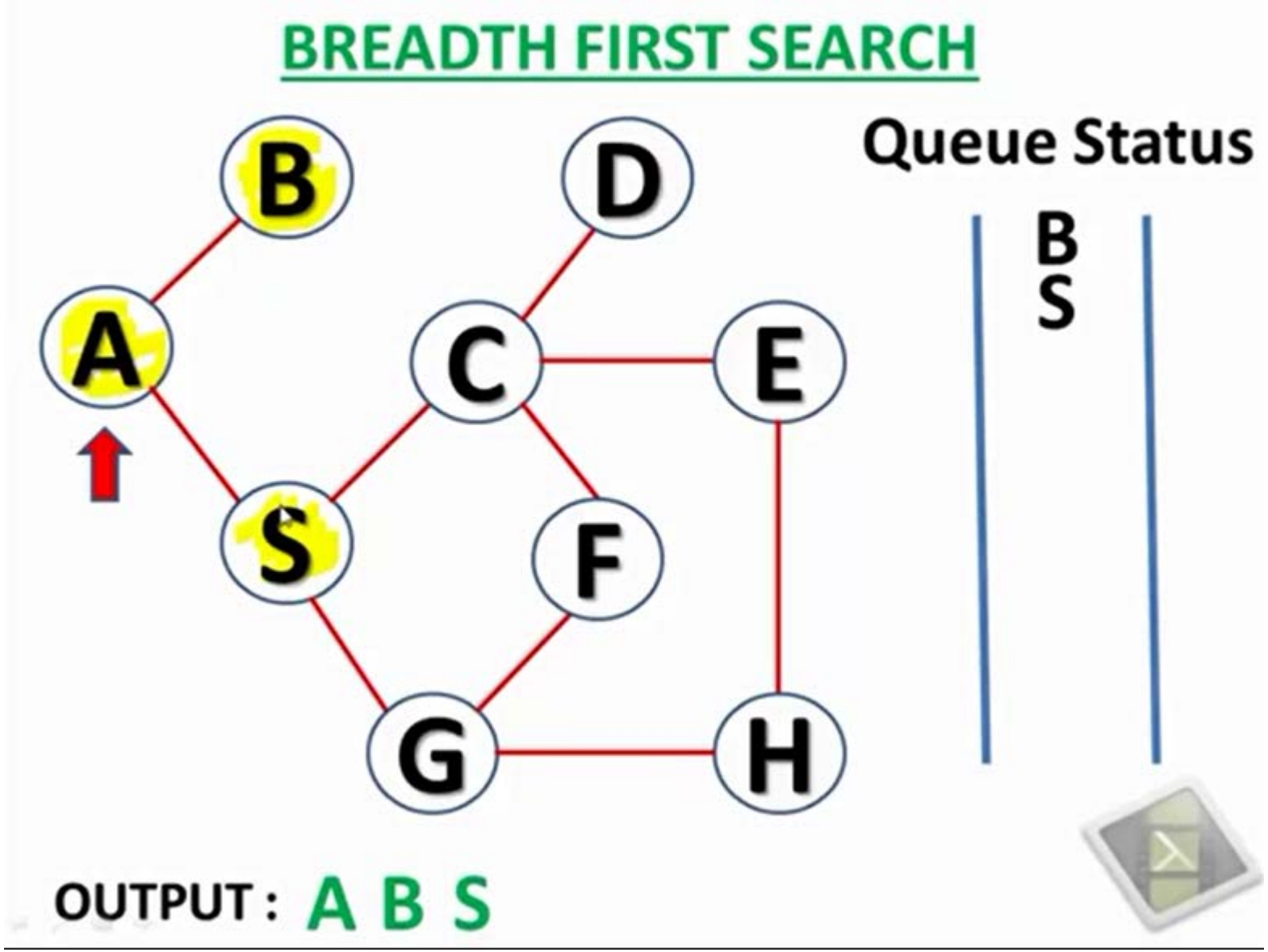


Queue Status

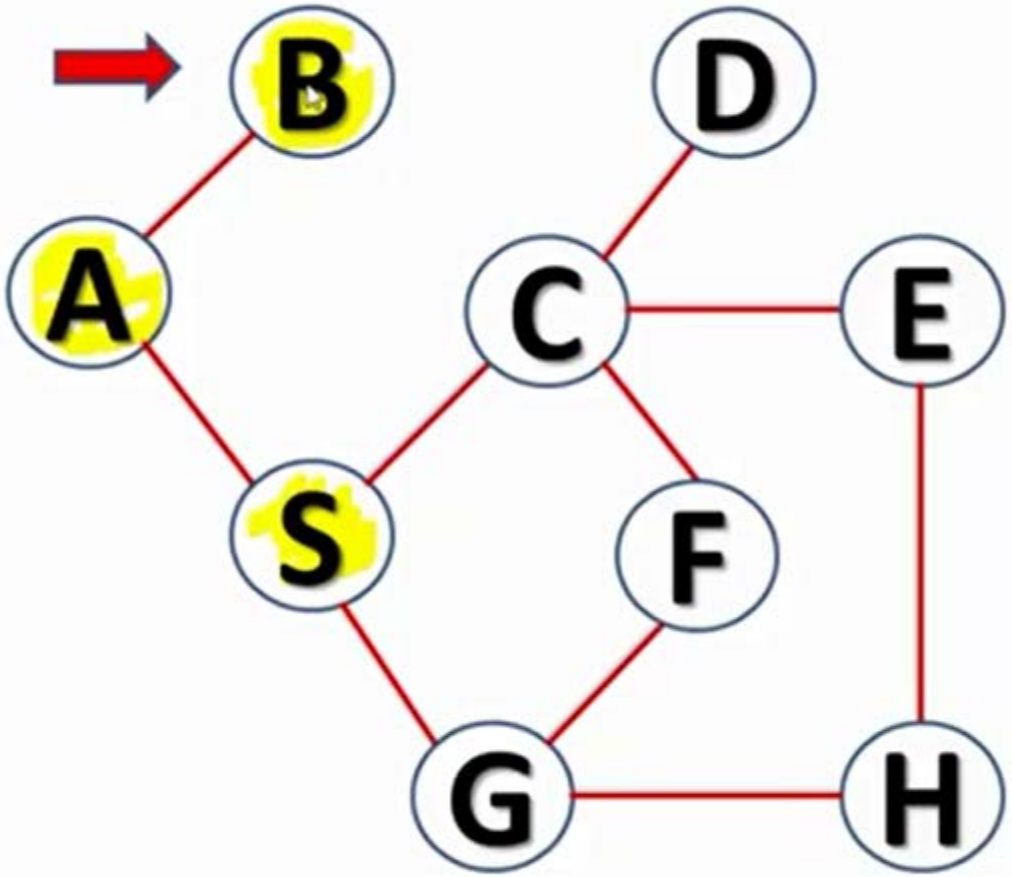


OUTPUT :


BREADTH FIRST SEARCH
B
D
A
C
E
S
F
G
H
Queue Status
OUTPUT : A


BREADTH FIRST SEARCH
B
D
A
C
E
S
F
G
H
Queue Status
B
S
OUTPUT : A B S

BREADTH FIRST SEARCH

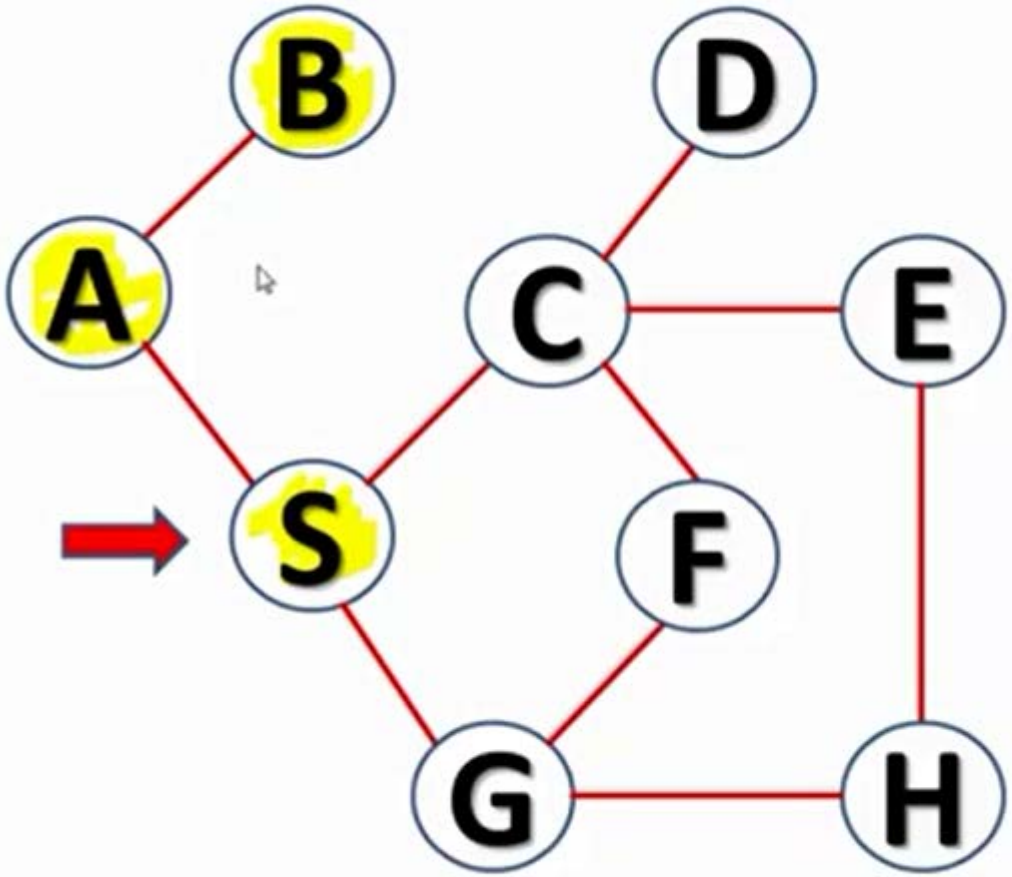


Queue Status

S

OUTPUT : A B S

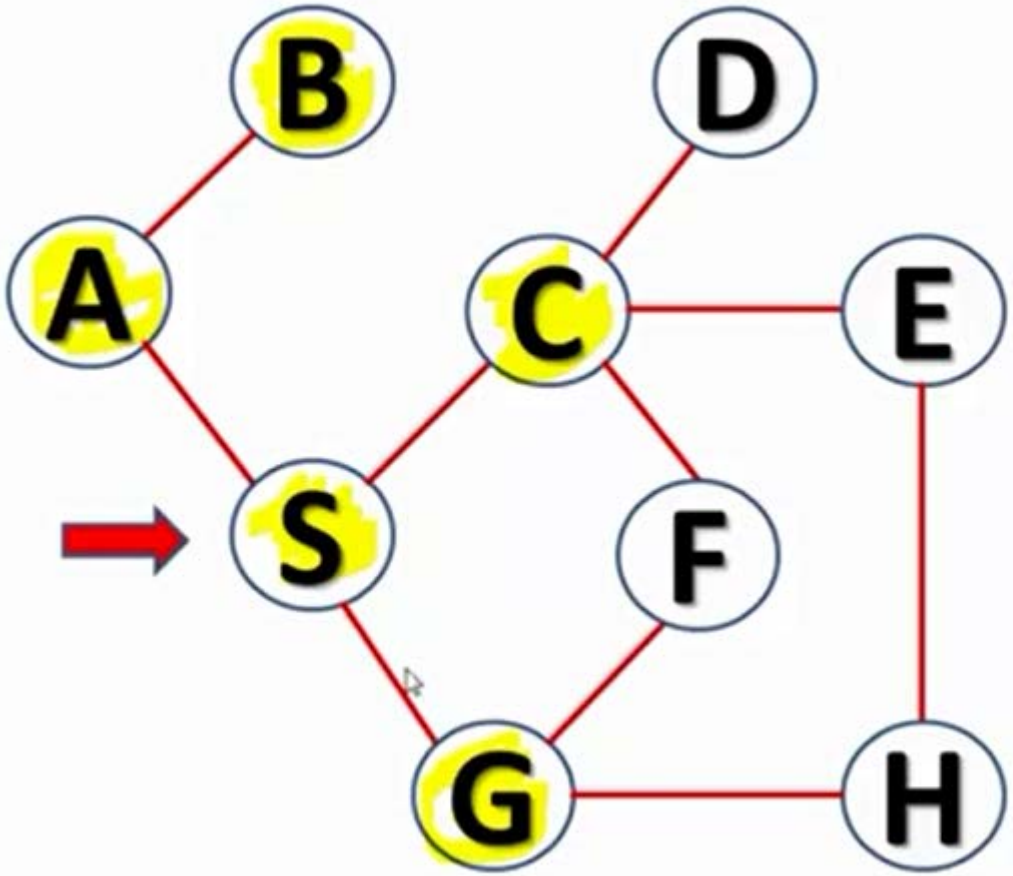
BREADTH FIRST SEARCH



Queue Status

OUTPUT : A B S

BREADTH FIRST SEARCH

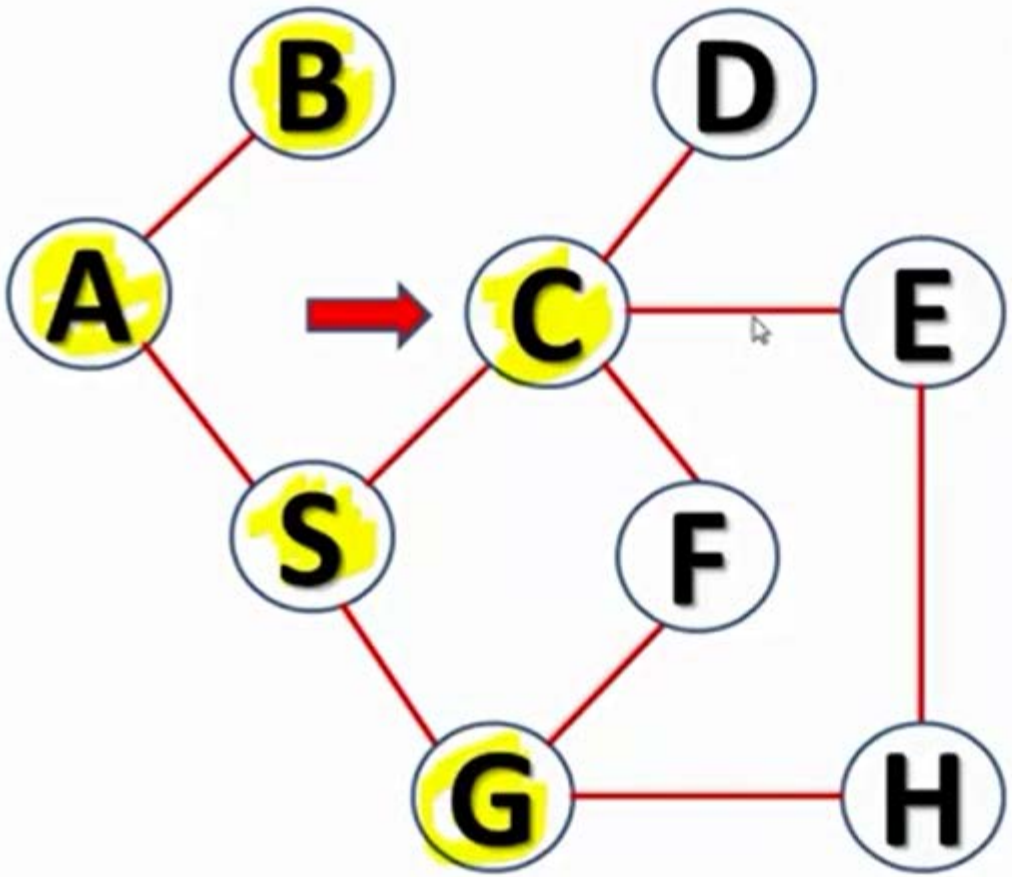


Queue Status

C
G

OUTPUT : A B S C G

BREADTH FIRST SEARCH

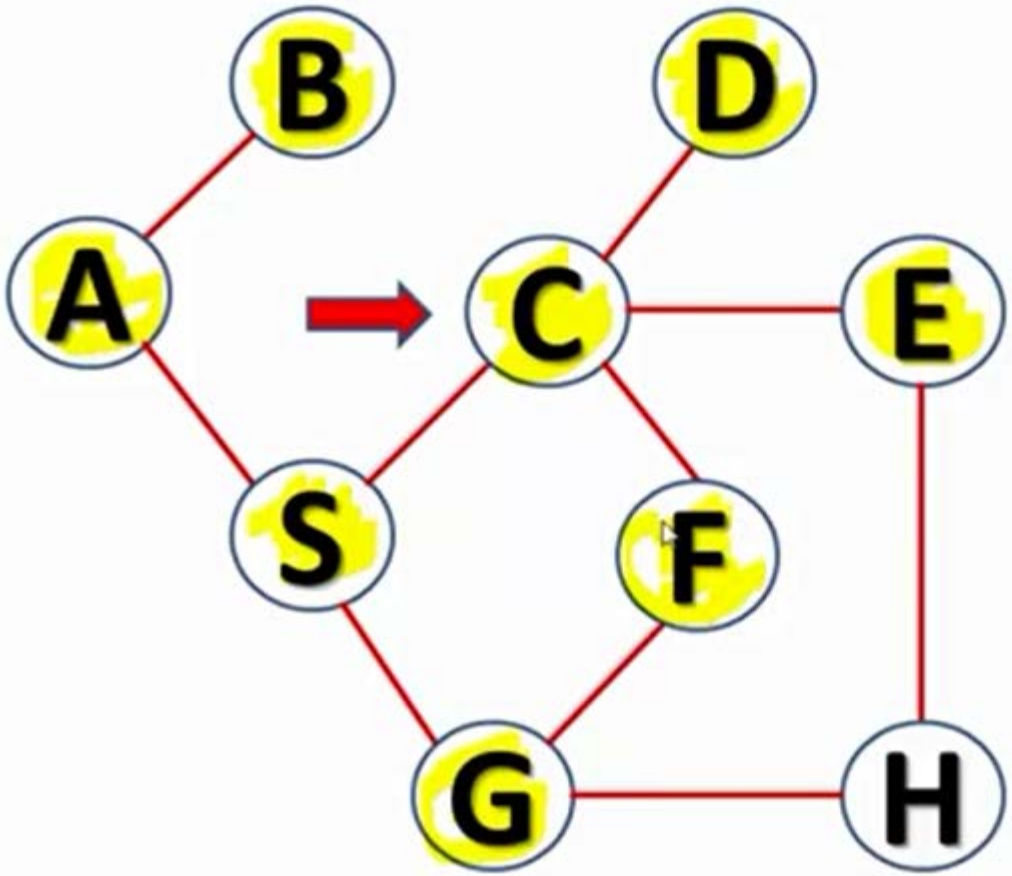


Queue Status

G

OUTPUT : A B S C G

BREADTH FIRST SEARCH

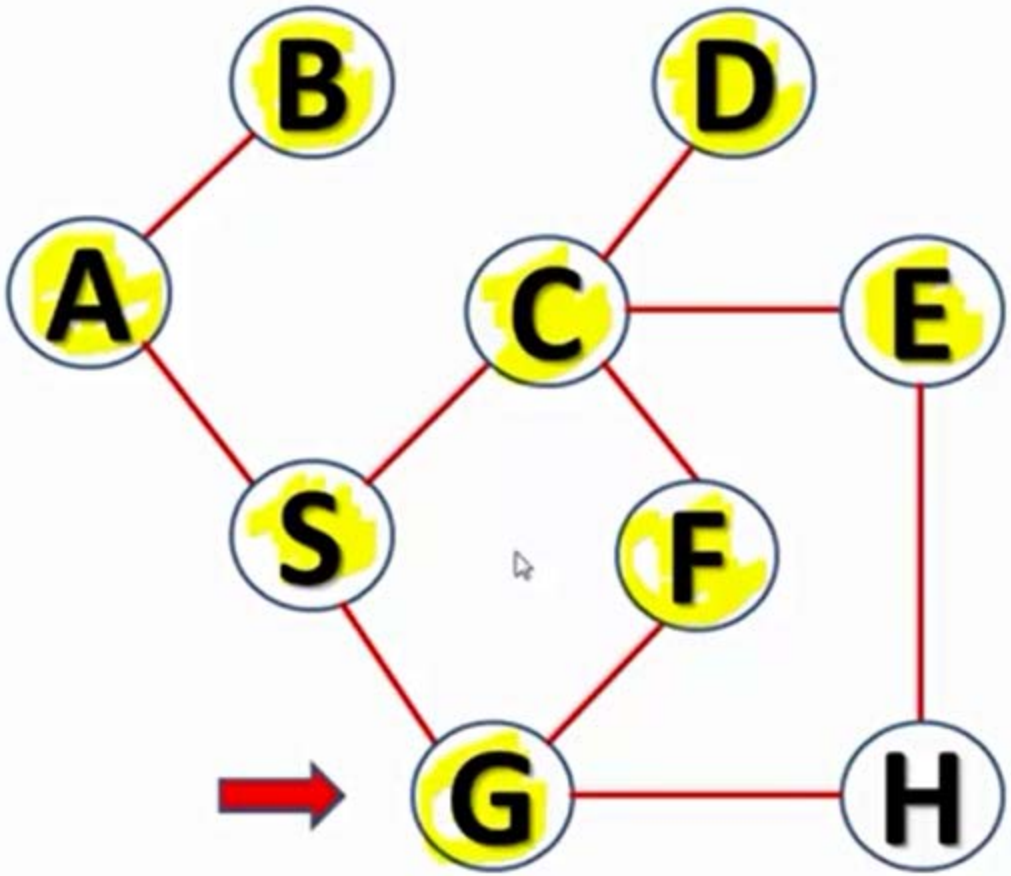


Queue Status

G
D
E
F

OUTPUT : A B S C G D E

BREADTH FIRST SEARCH

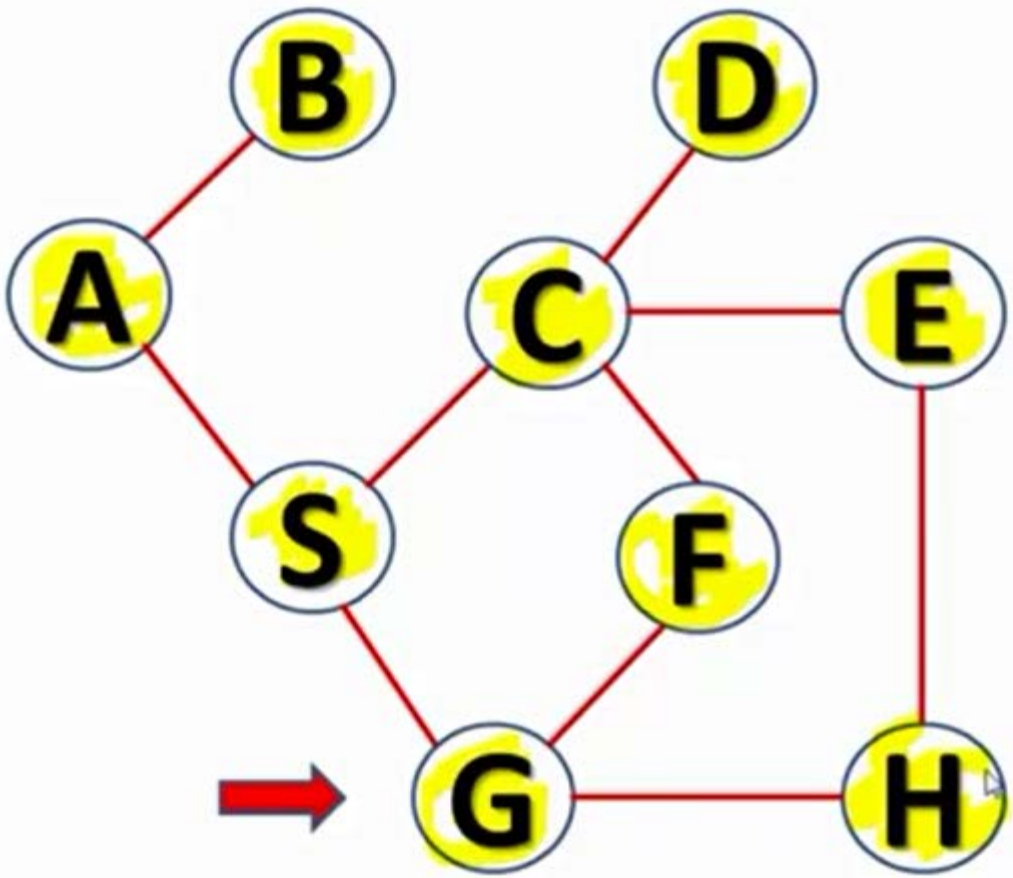


Queue Status

D
E
F

OUTPUT : A B S C G D E F

BREADTH FIRST SEARCH

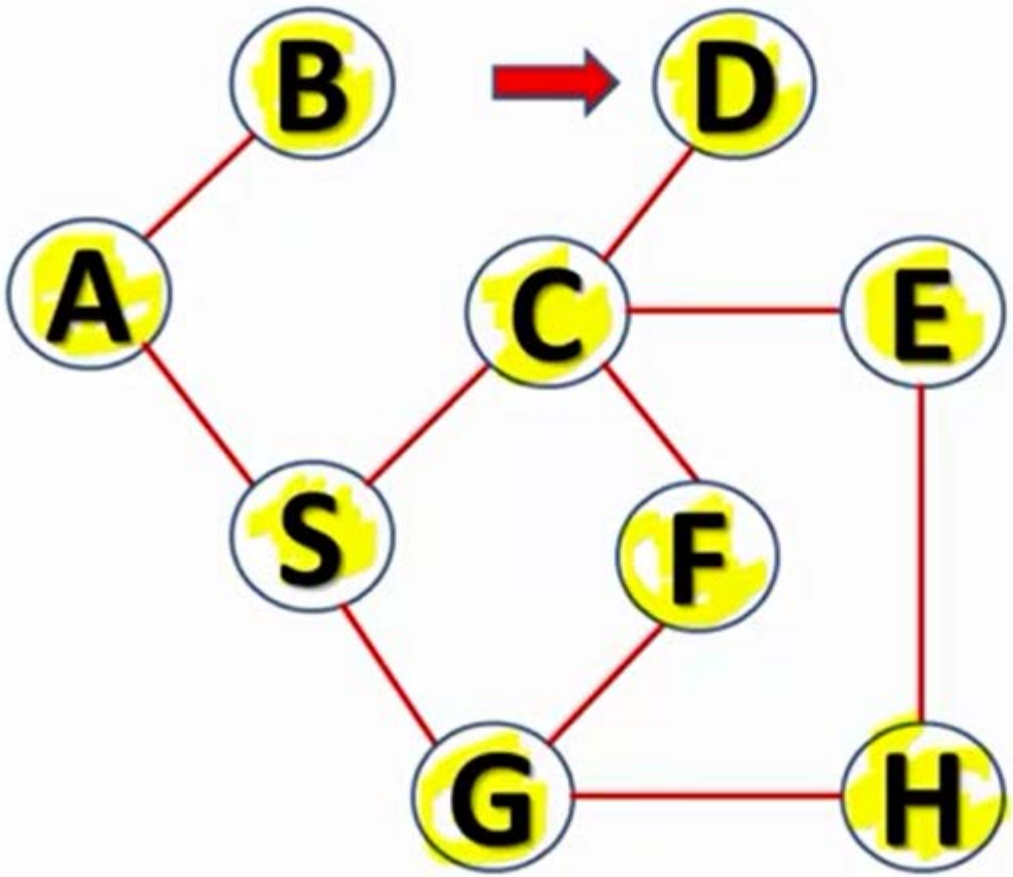


Queue Status

D
E
F
H

OUTPUT : A B S C G D E F H

BREADTH FIRST SEARCH

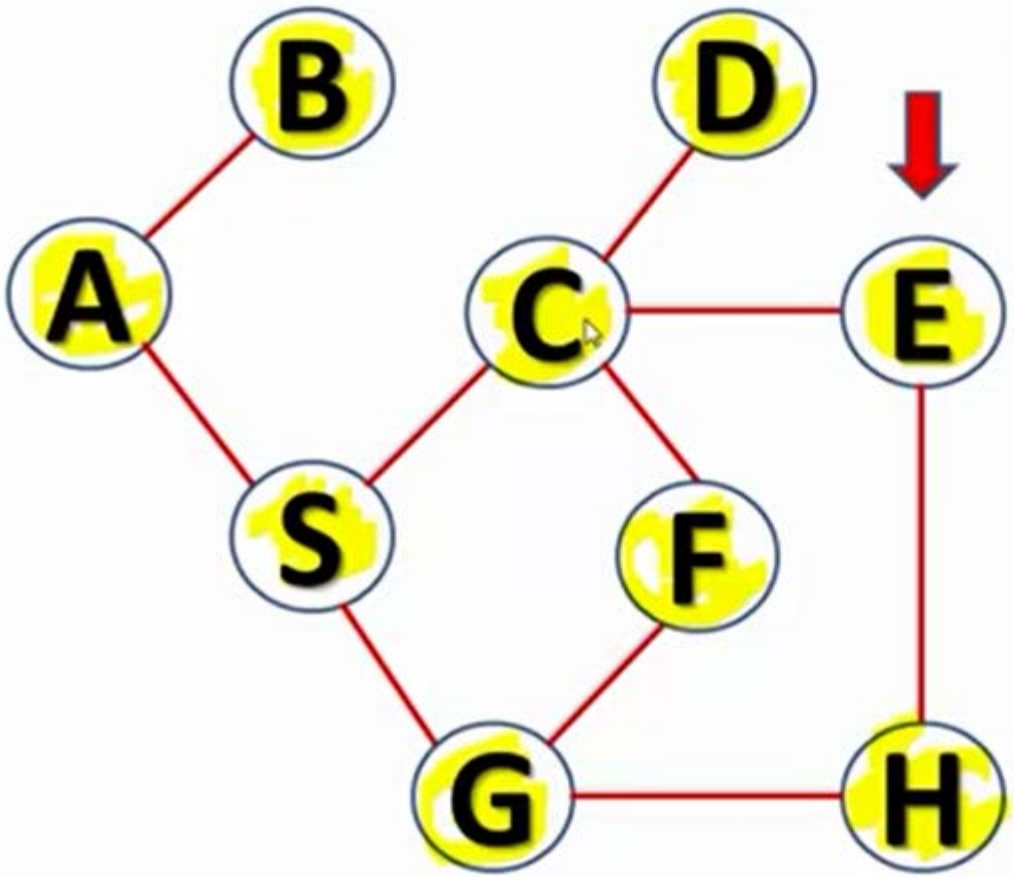


Queue Status

E
F
H

OUTPUT : A B S C G D E F H

BREADTH FIRST SEARCH

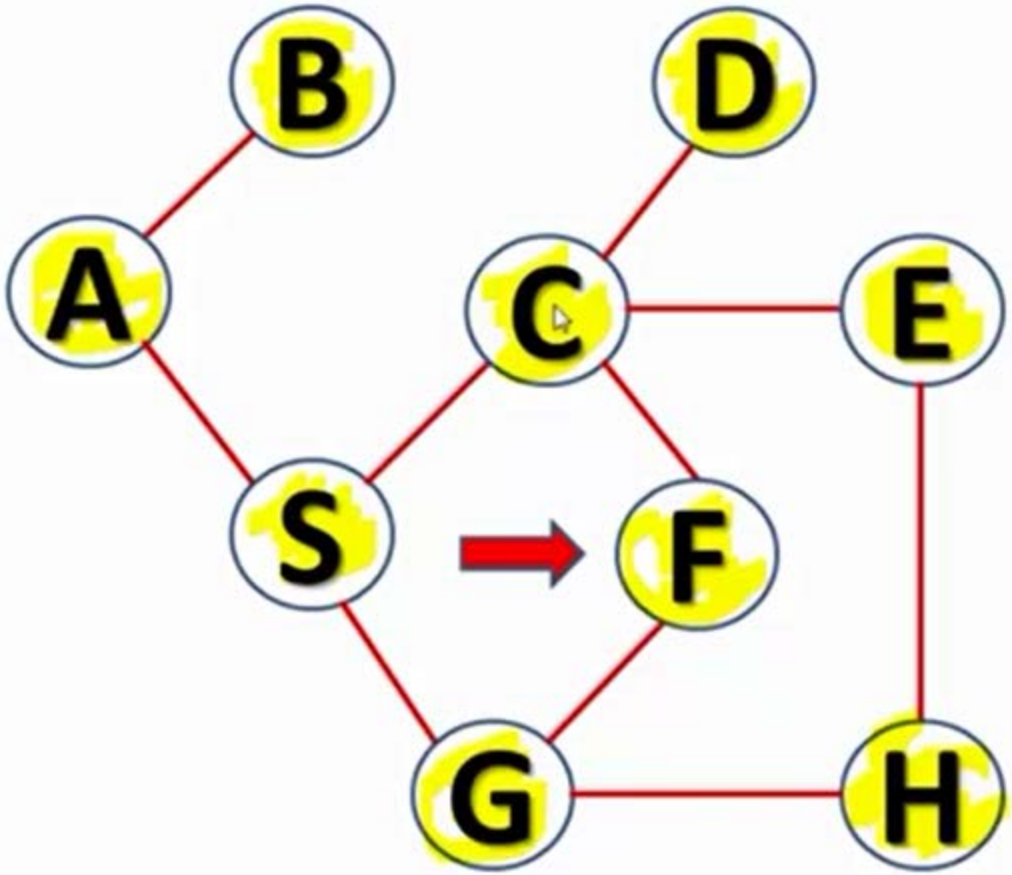


Queue Status

F
H

OUTPUT : A B S C G D E F H

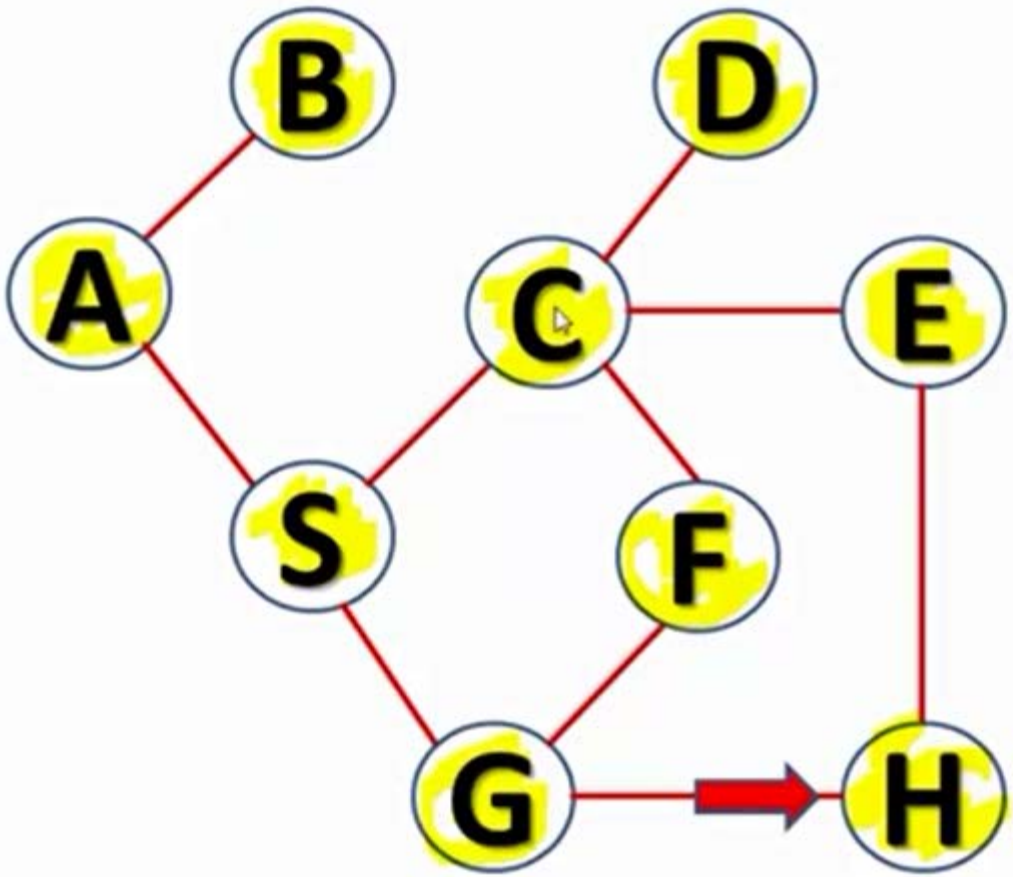
BREADTH FIRST SEARCH



Queue Status

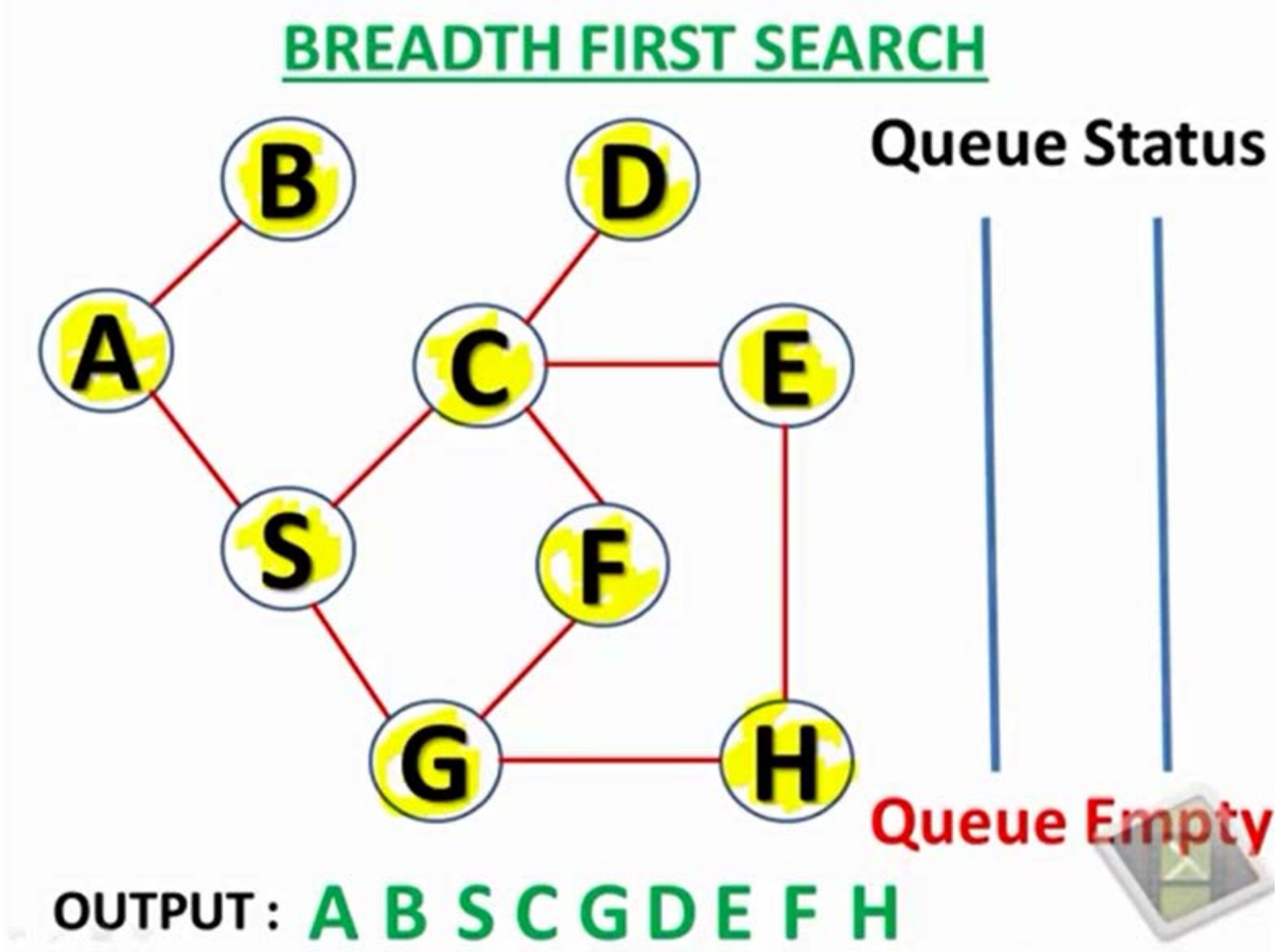
OUTPUT : A B S C G D E F H

BREADTH FIRST SEARCH



Queue Status

OUTPUT : A B S C G D E F H


BREADTH FIRST SEARCH
B
D
A
C
E
S
F
G
H
Queue Status
Queue Empty
OUTPUT : A B S C G D E F H