GRAPH SEARCH BFS & DFS

By: Parminder Benipal

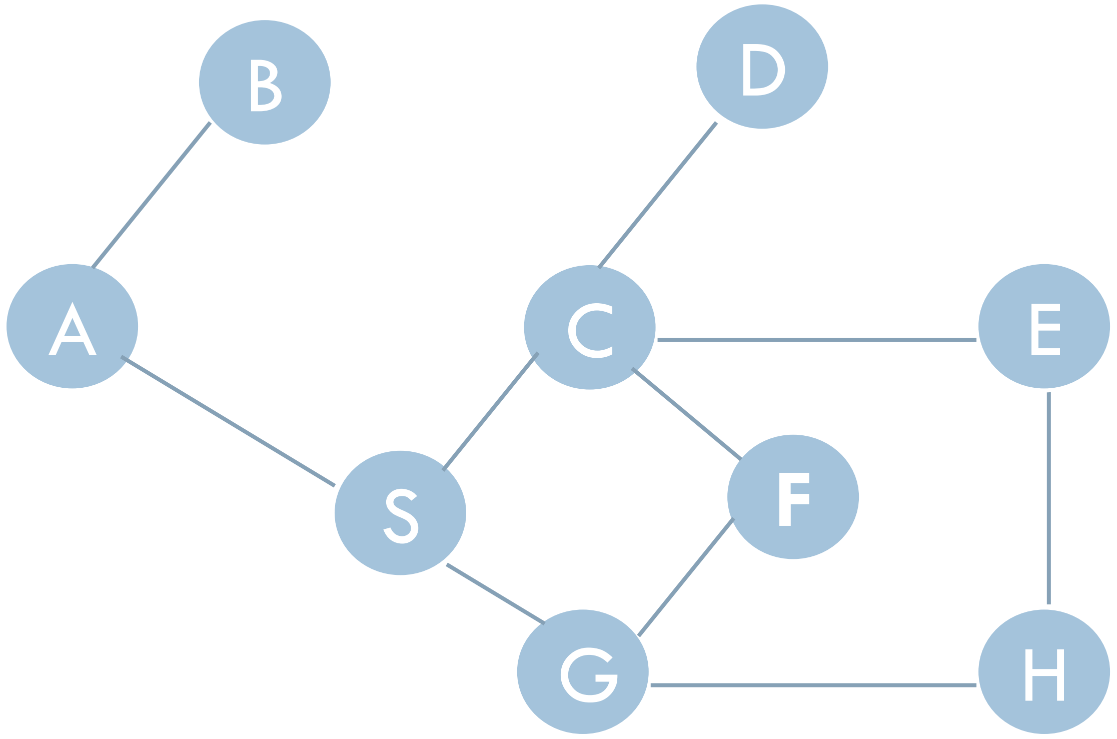
Usage

- Transportation networks (airline carrier, airports as node and direct flights as edges (direct edge).
- Communication networks (a collection of computers as nodes and the physical link between them as edges).
- Information networks (World Wide Web can be viewed as directed graph, the Web pages are nodes and the hyperlink between the pages are directed edges).
- Social Network (People are nodes and friendship is an edge).

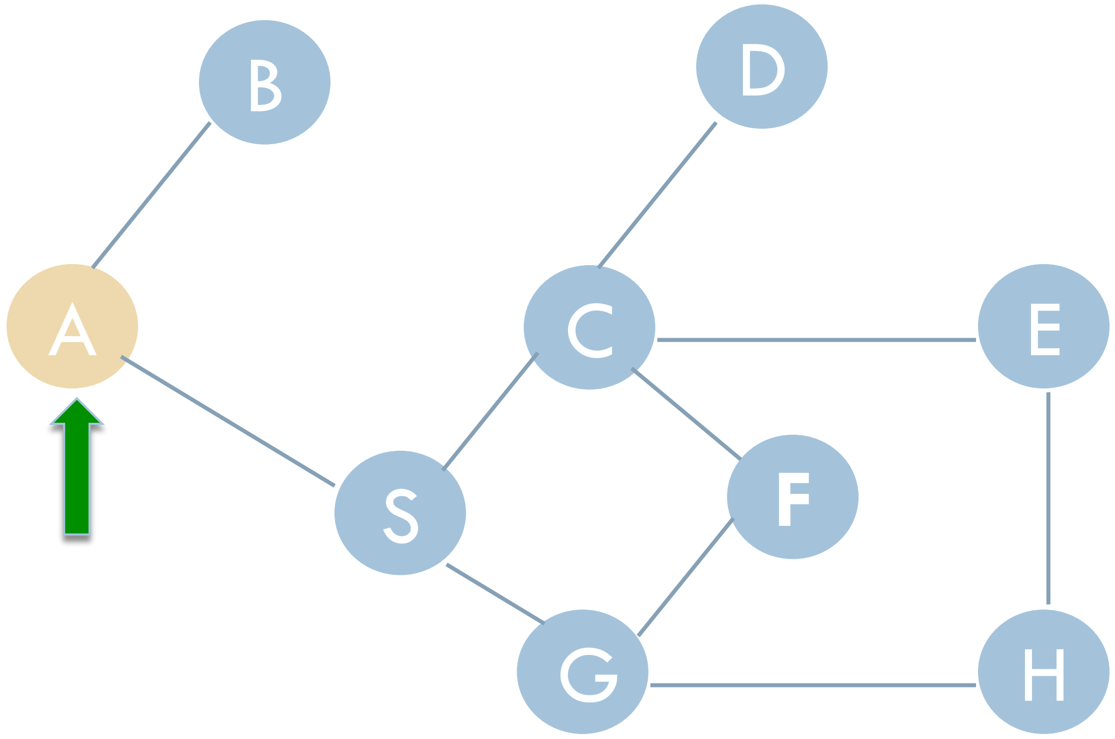
Breadth-First Search (BFS)

- BFS begins at a root node and inspects all the neighboring nodes. Then for each of those neighbor nodes in turn, it inspects their neighbor nodes which were unvisited, and so on

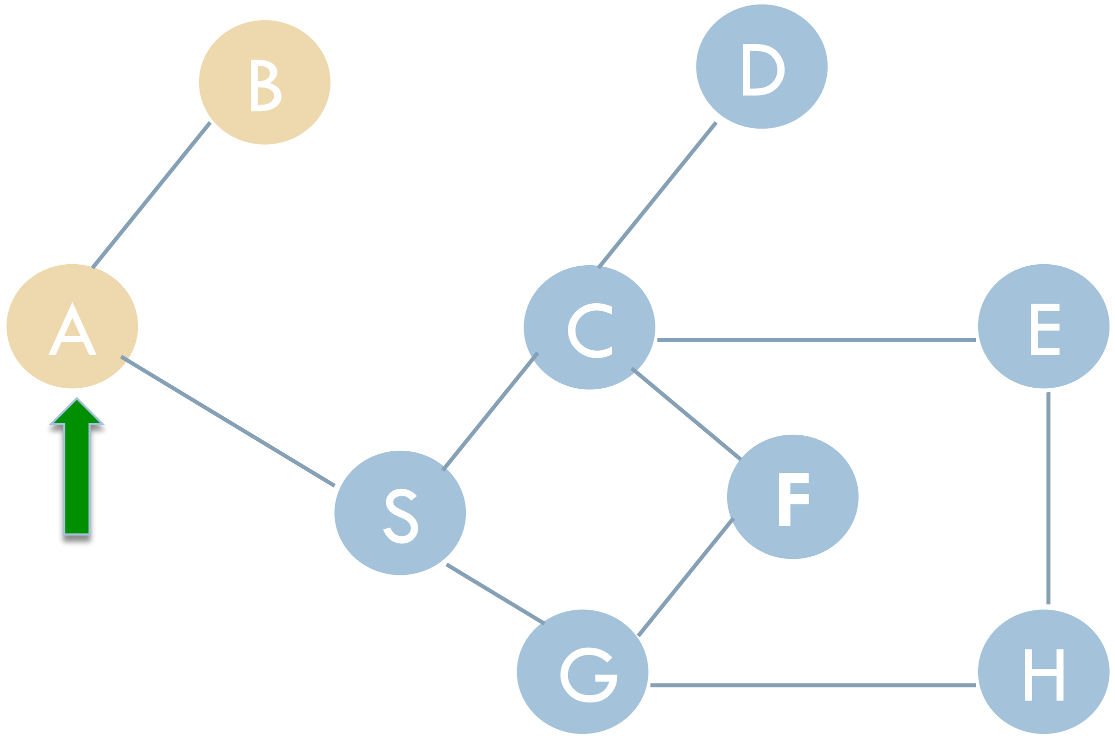
Breadth-First Search (BFS)



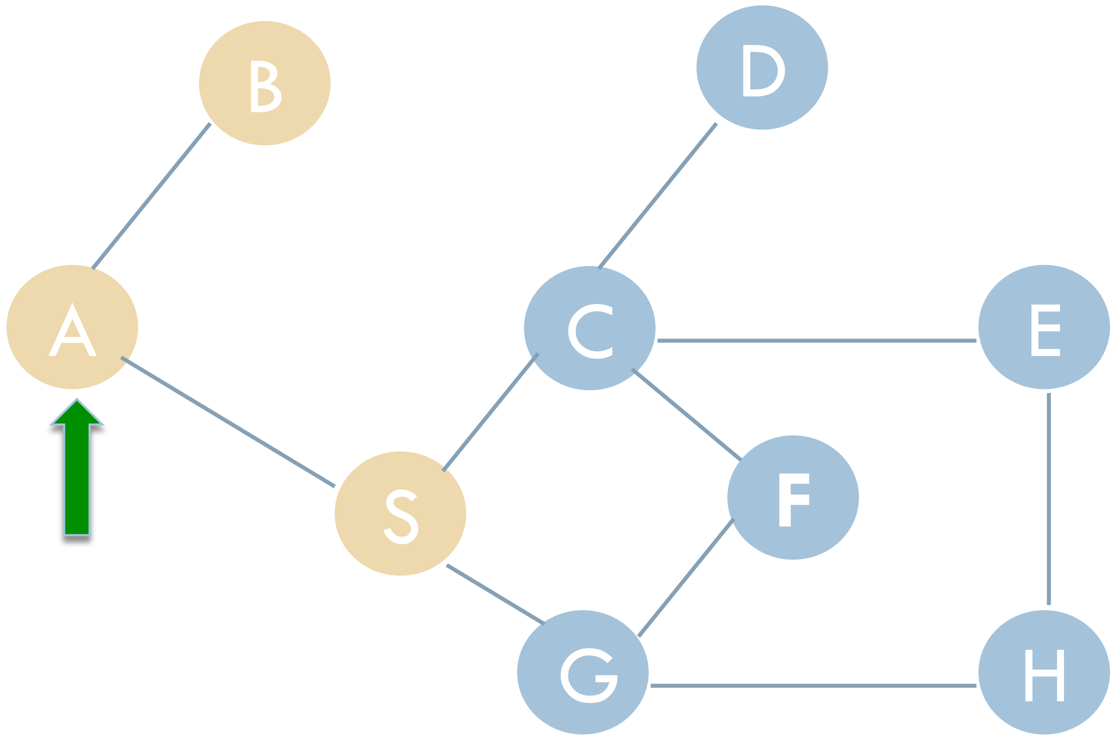
Breadth-First Search (BFS)



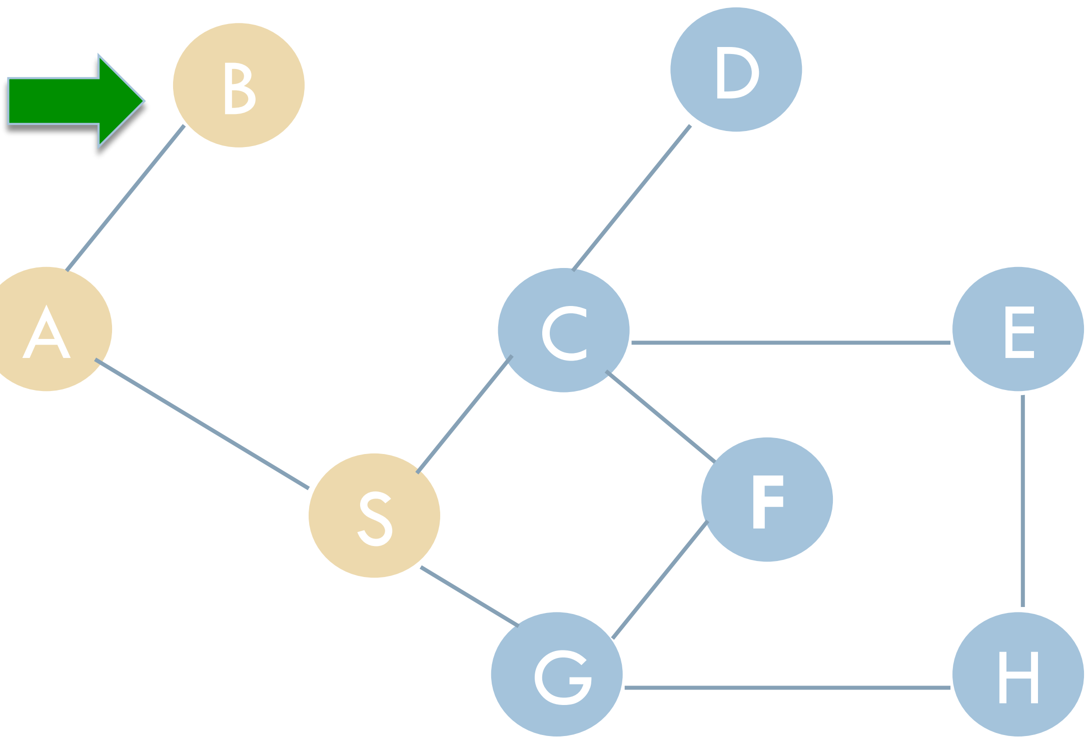
Breadth-First Search (BFS)



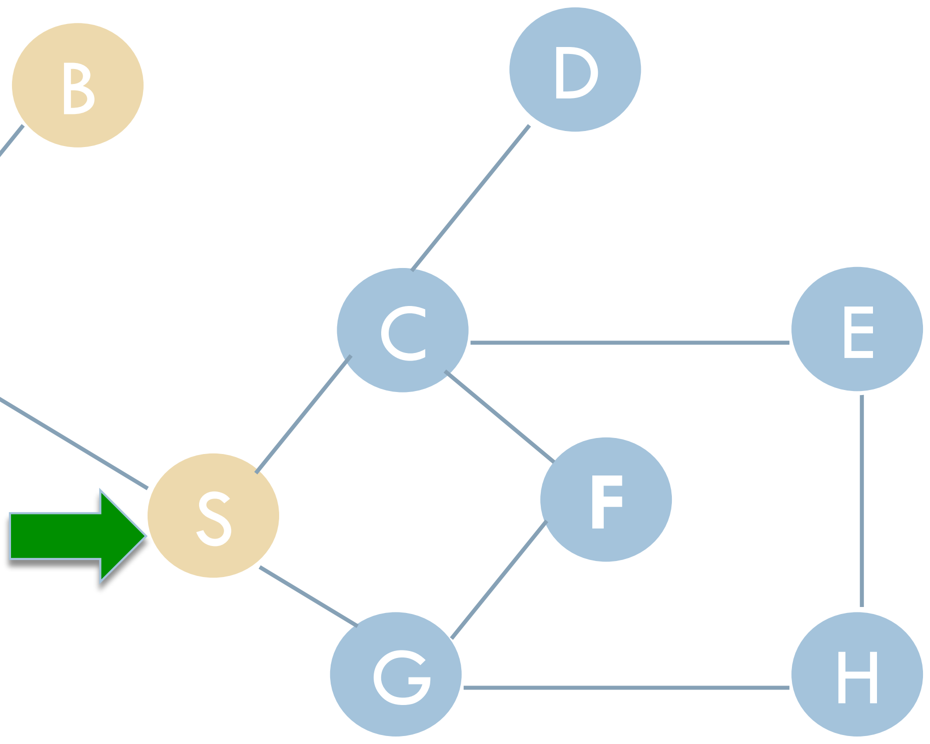
Breadth-First Search (BFS)



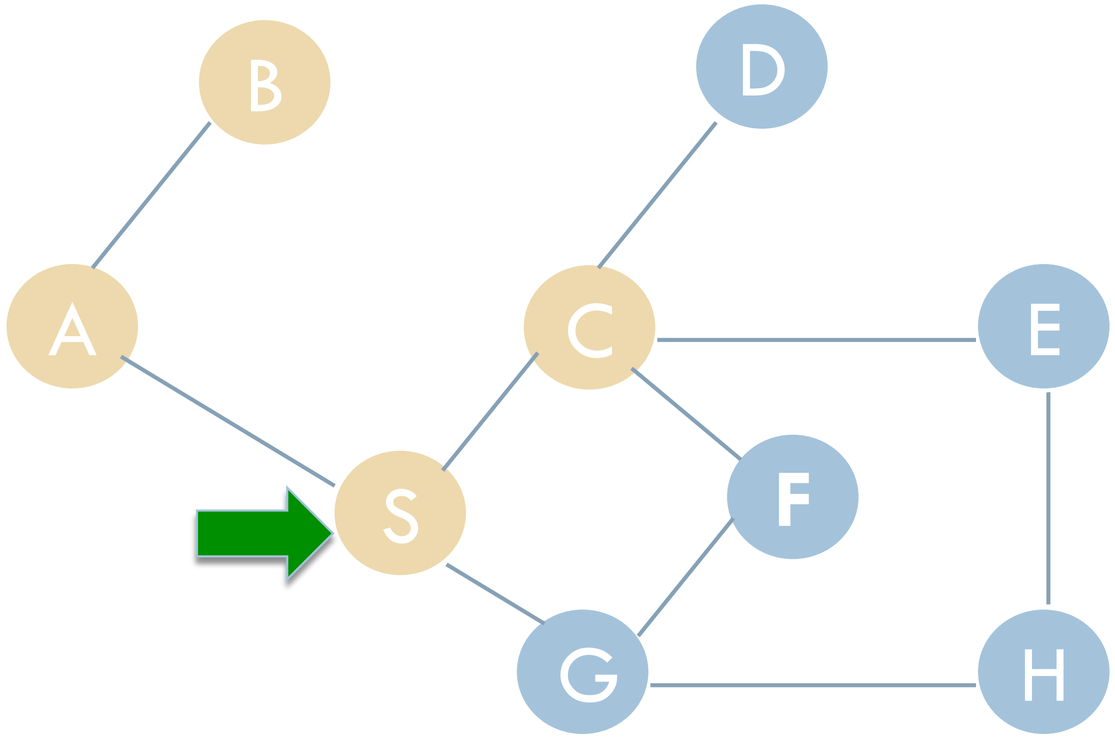
Breadth-First Search (BFS)



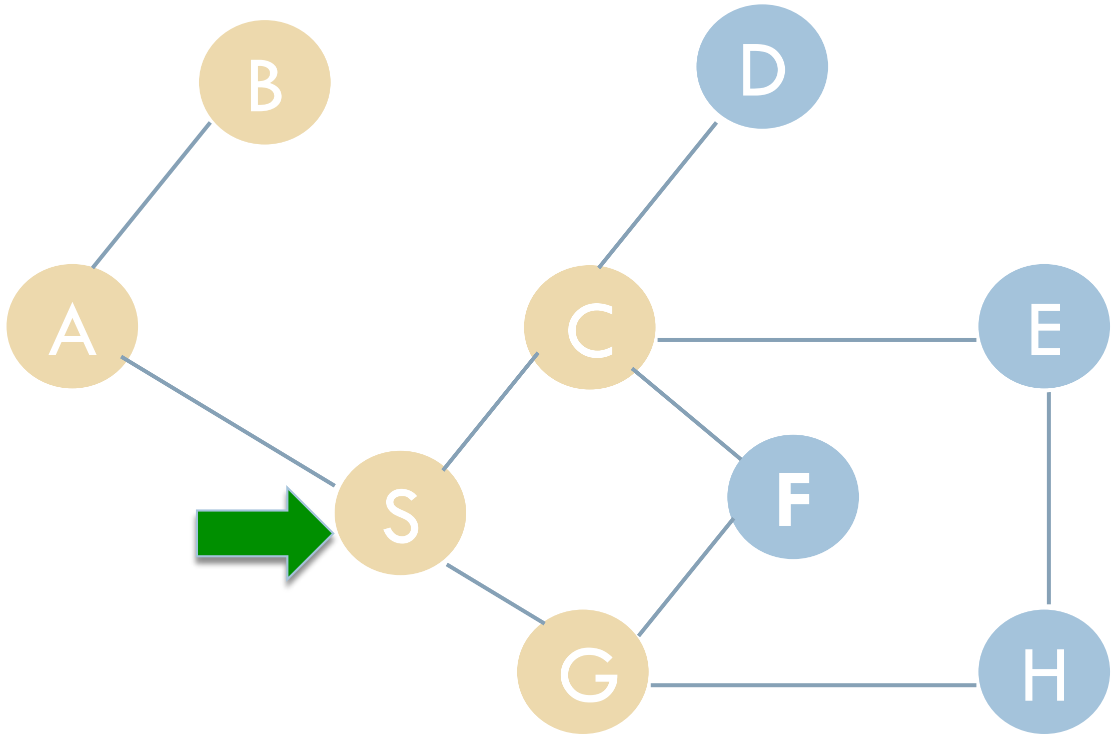
Breadth-First Search (BFS)



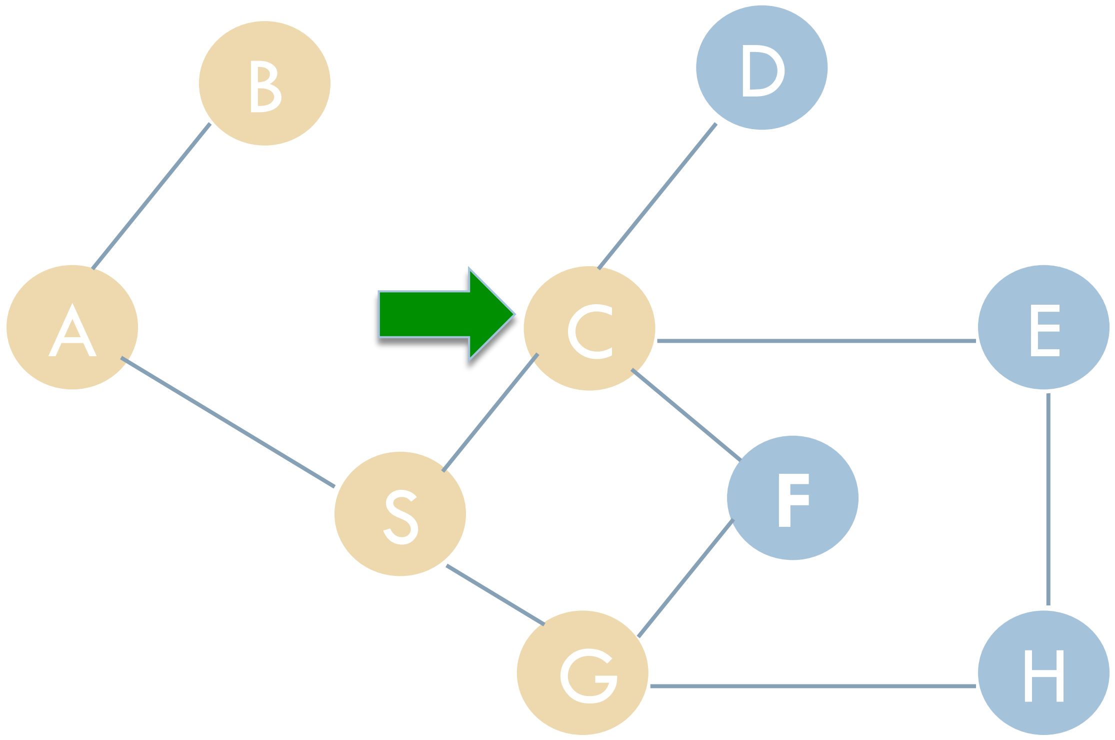
Breadth-First Search (BFS)



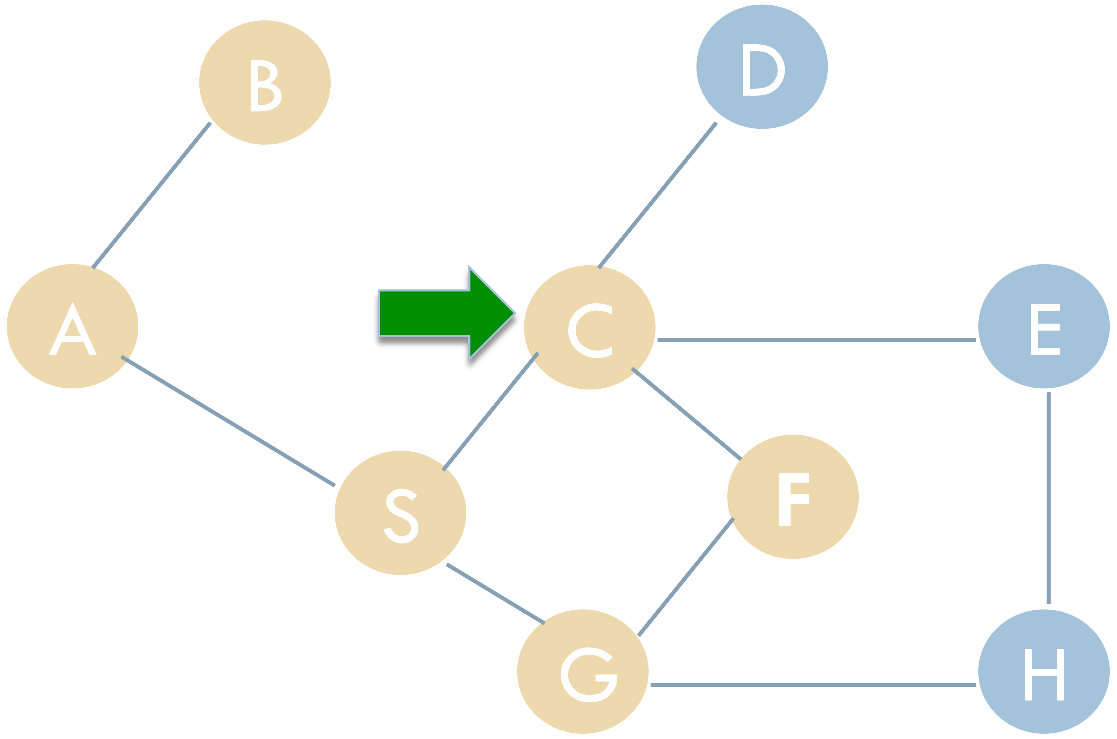
Breadth-First Search (BFS)



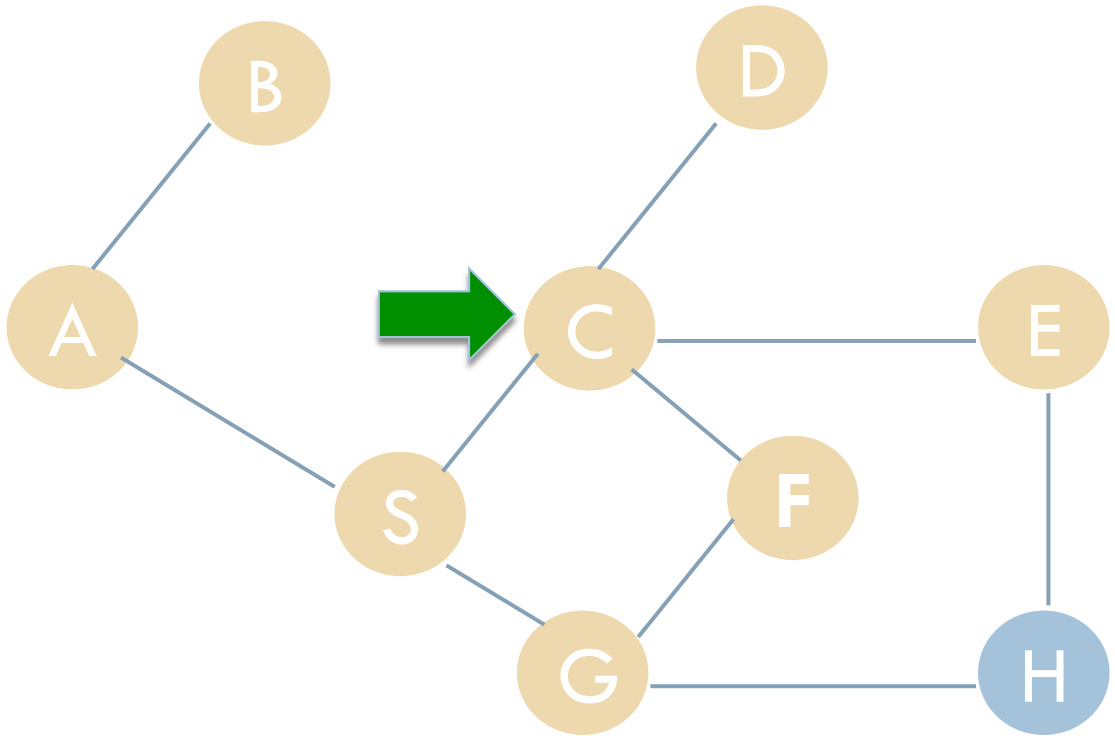
Breadth-First Search (BFS)



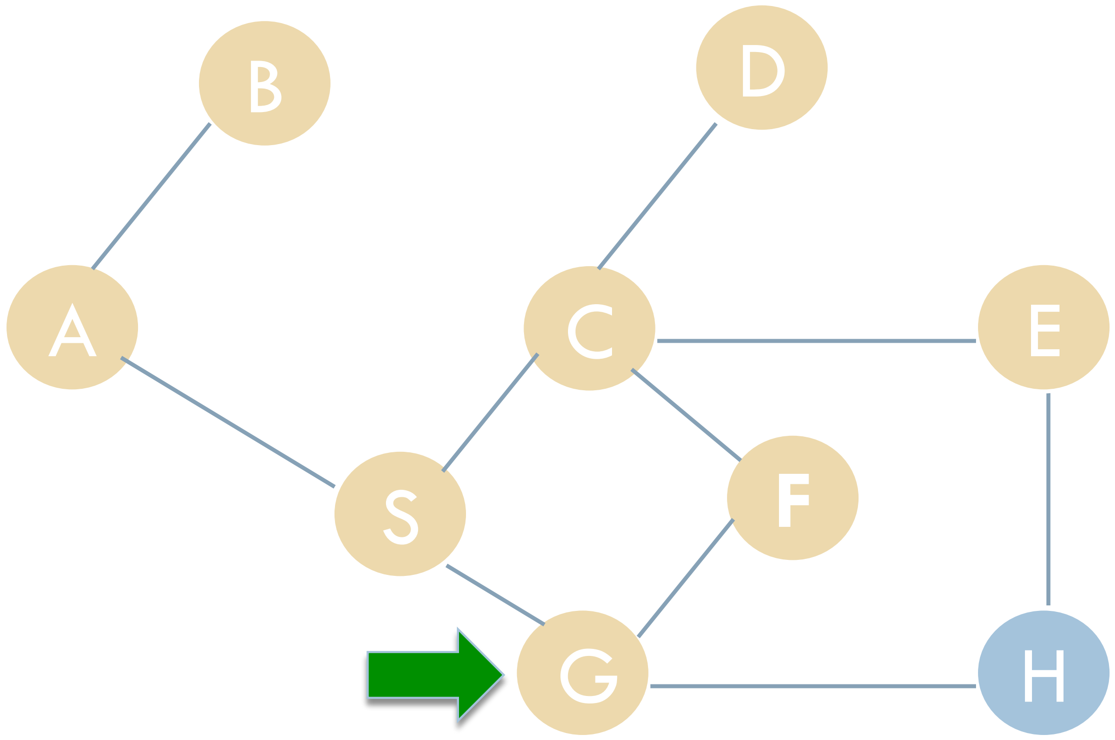
Breadth-First Search (BFS)



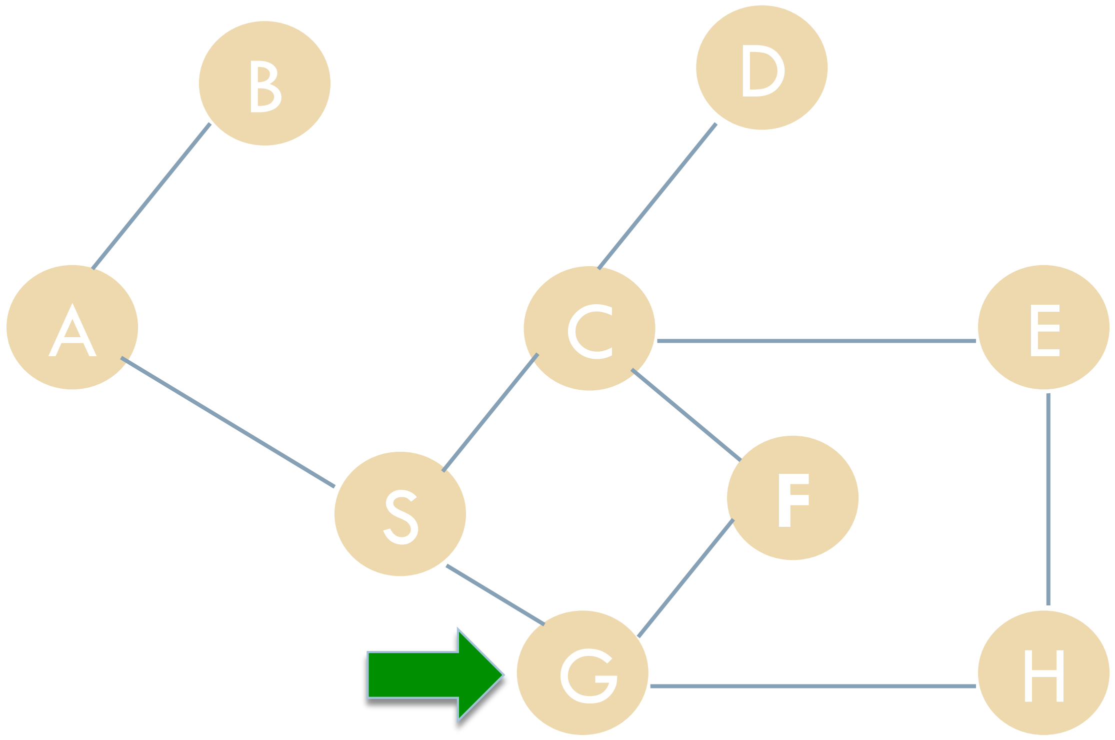
Breadth-First Search (BFS)



Breadth-First Search (BFS)

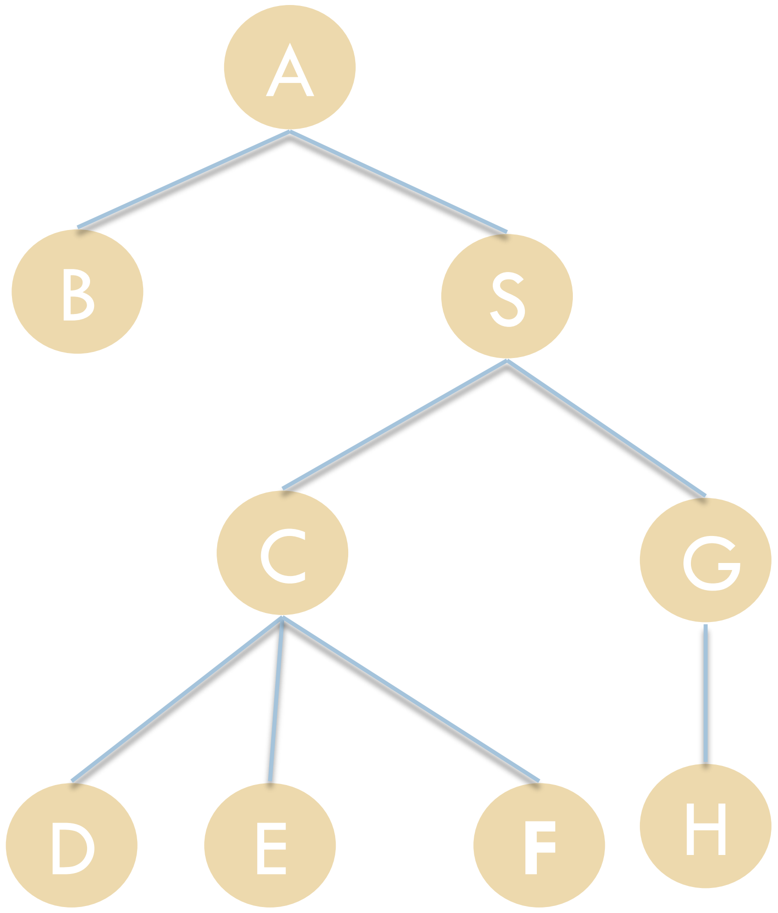


Breadth-First Search (BFS)



{A B S C G D E F H}

Tree after BFS run



BFS Algorithm

```
BFS ($s$)
1. Set Discover[s]=true and Discover[v]=false for all other $v$
2. Set $L[0] = \{s\}$
3. Set layer counter i=0
4. Set $T = \emptyset$
4. While $L[i]$ is not empty
5.     Initialize empty set $L[i+1]$
6.     For each node $u \in L[i]$
7.       Consider each edge $uv$
8.       If $Discover[v] =$ false then
9.           Set $Discover[v] =$ true
10.          Add edge $uv$ to $T$
11.          Add $v$ to the list $L[i+1]$
```

BFS running time

1) If we represent the graph G by adjacency matrix then the running time of BFS algorithm is $O(n^2)$, where n is the number of nodes.
2) If we represent the graph G by link lists then the running time of BFS algorithm is $O(m + n)$, where m is the number of edges and n is the number of nodes.

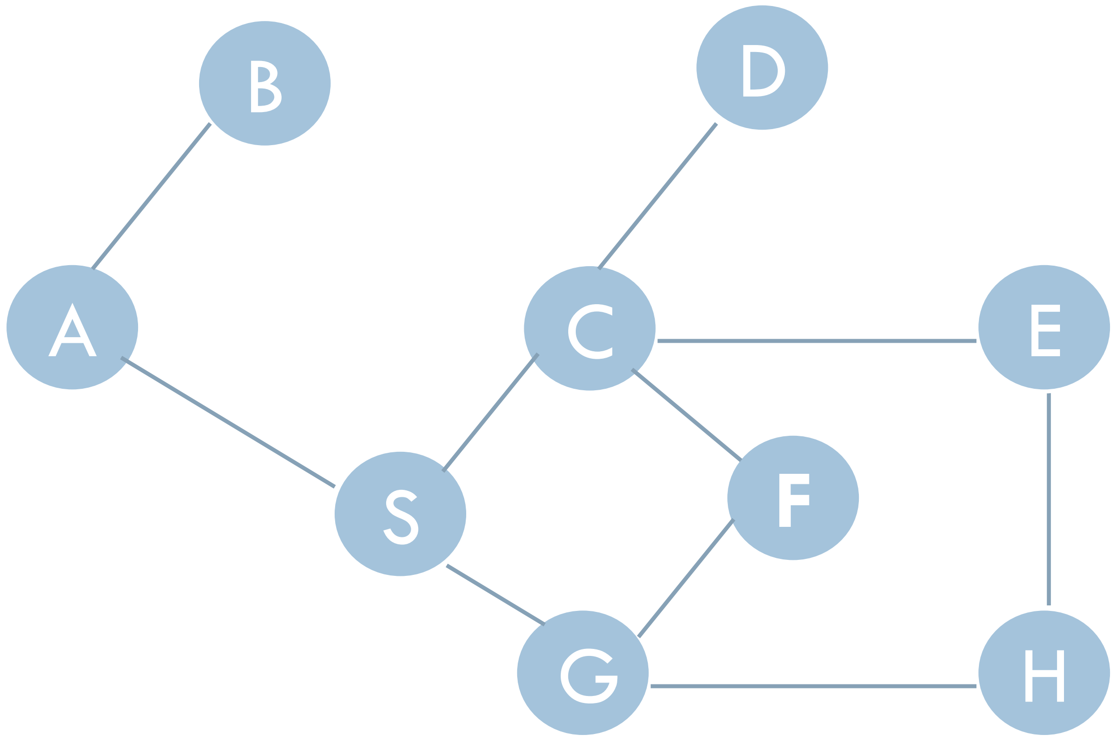
BFS Applications

- Breadth-first search can be used to solve many problems in graph theory, for example:
  - Finding all nodes within one connected component
  - Finding the shortest path between two nodes u and v
  - Finding the diameter of a graph (seen in assignment).
  - Testing a graph for bipartiteness (how ?). If there are two vertices x,y in the same level (layer) L_i that are adjacent then the graph is not bipartite.
  - More…

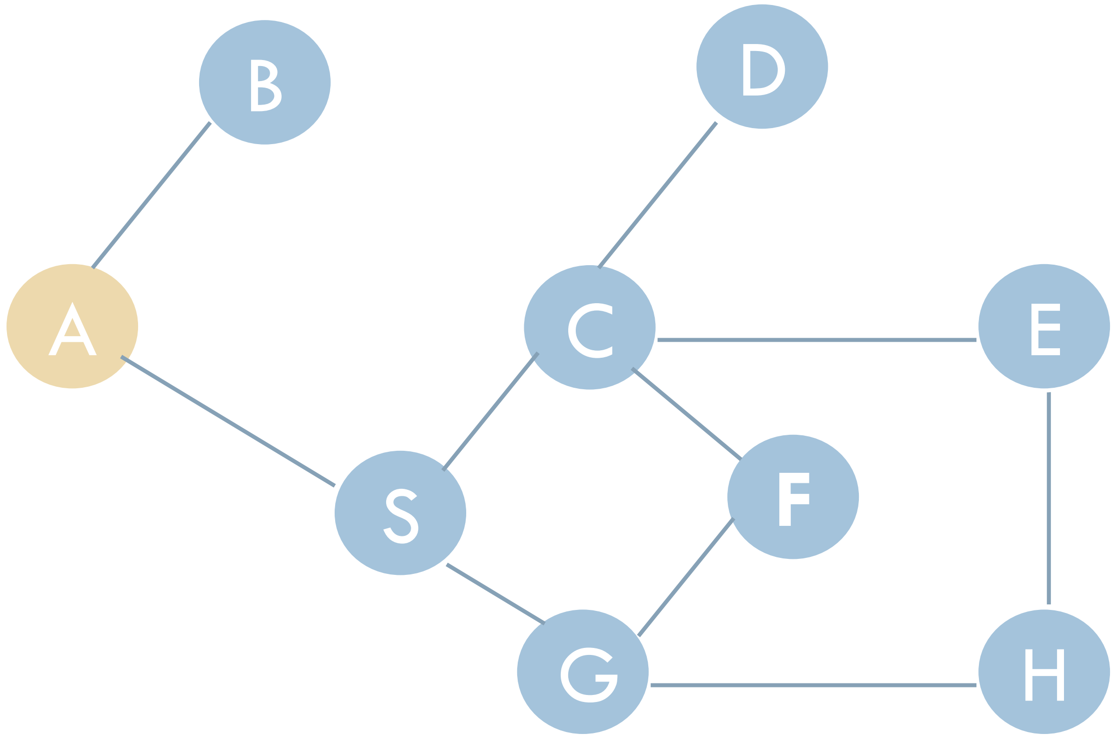
Depth-First Search (DFS)

- We don’t visit the nodes level by level! As long as there is an unvisited node adjacent to the current visited node we continue! Once we are stuck, trace back and go to a different branch!

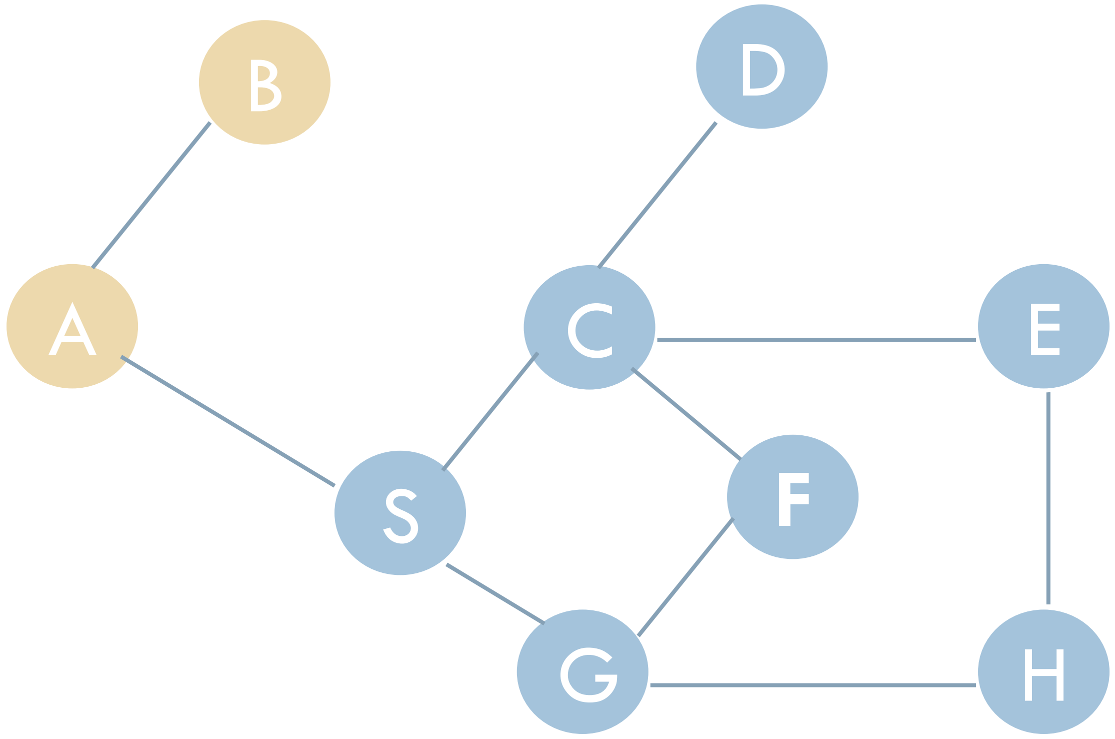
Depth-First Search (DFS)



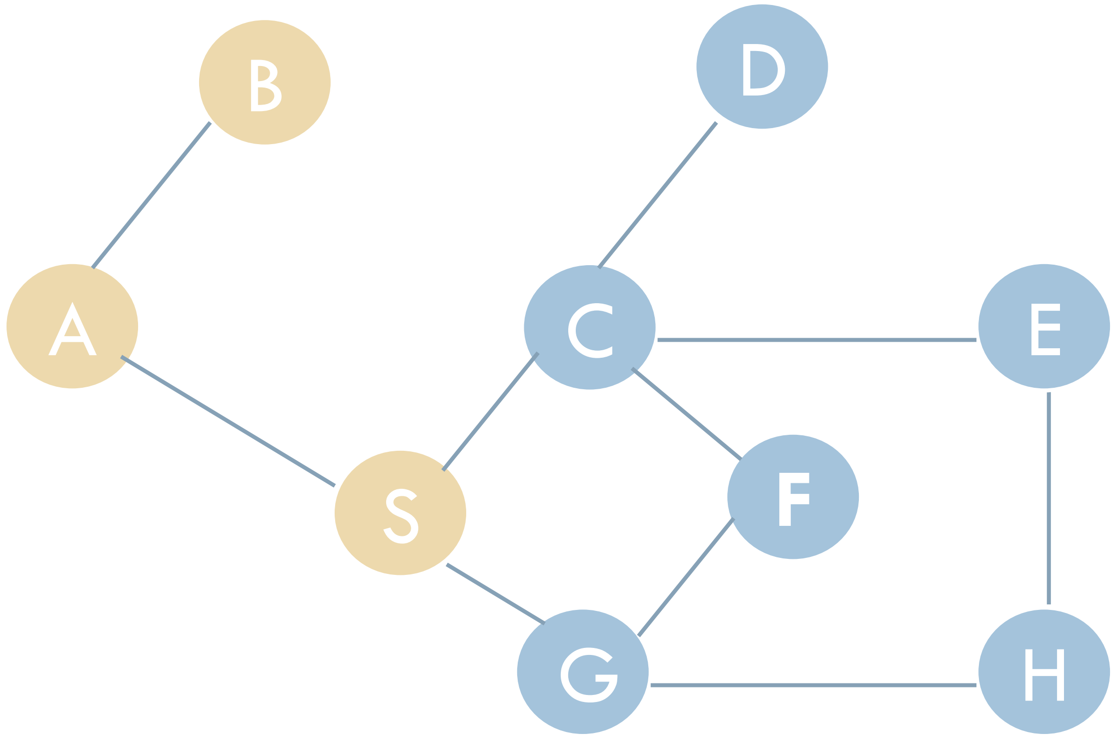
Depth-First Search (DFS)



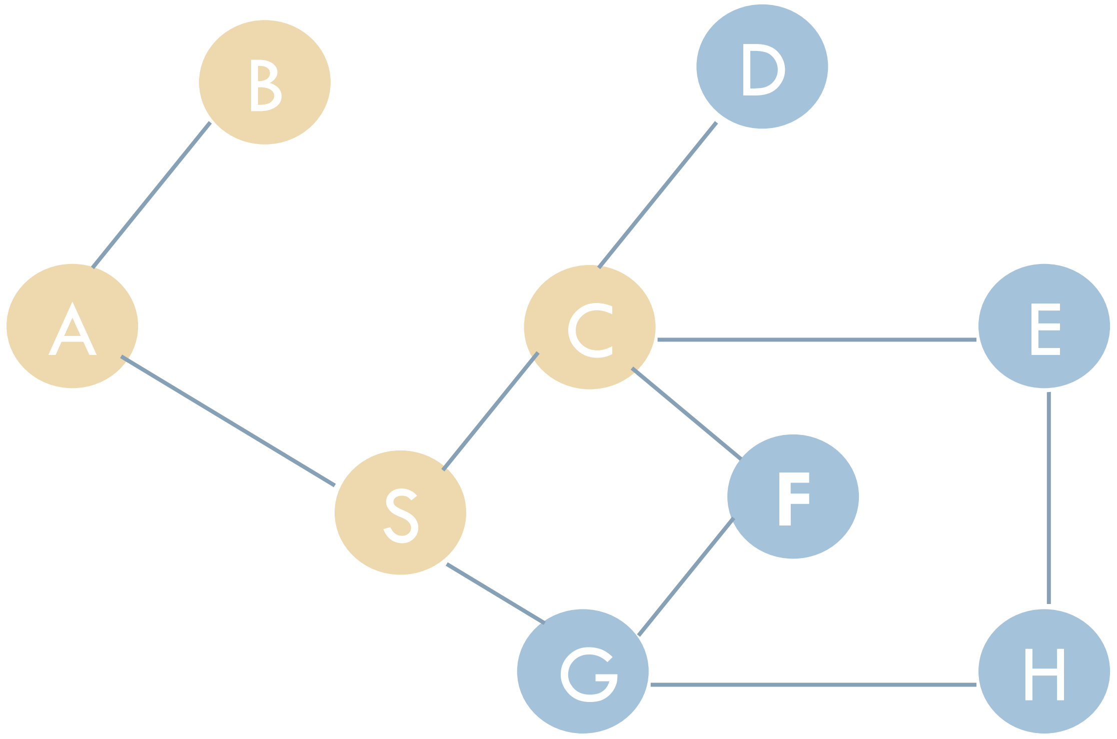
Depth-First Search (DFS)



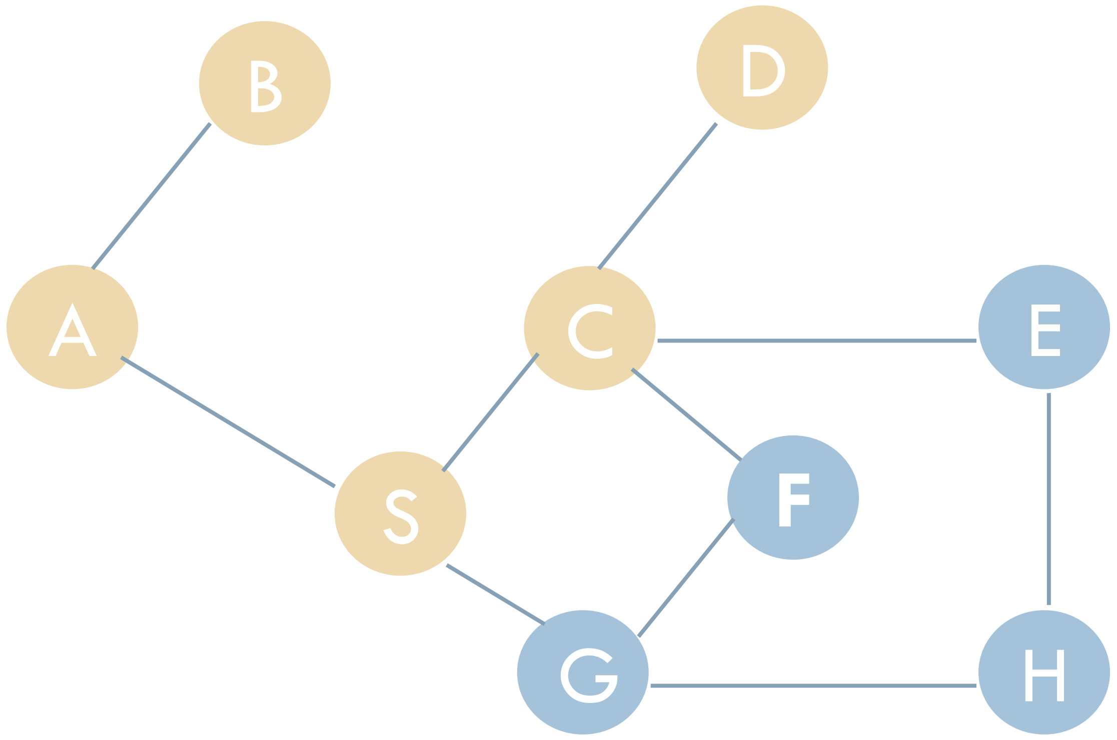
Depth-First Search (DFS)



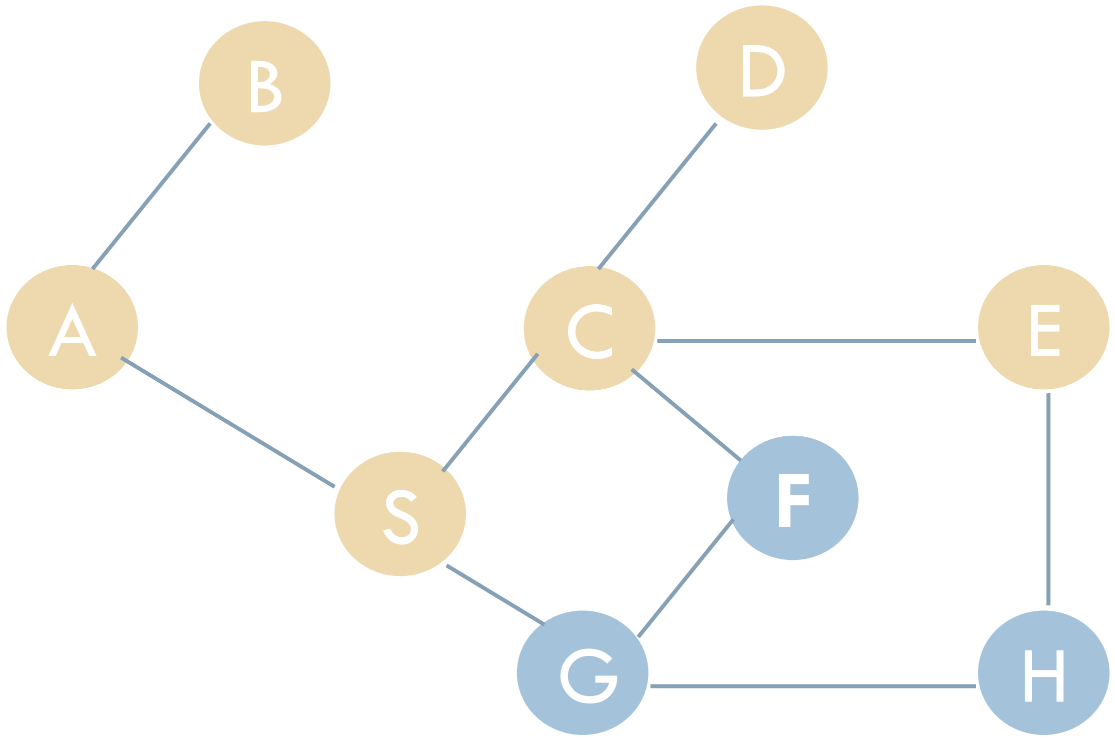
Depth-First Search (DFS)



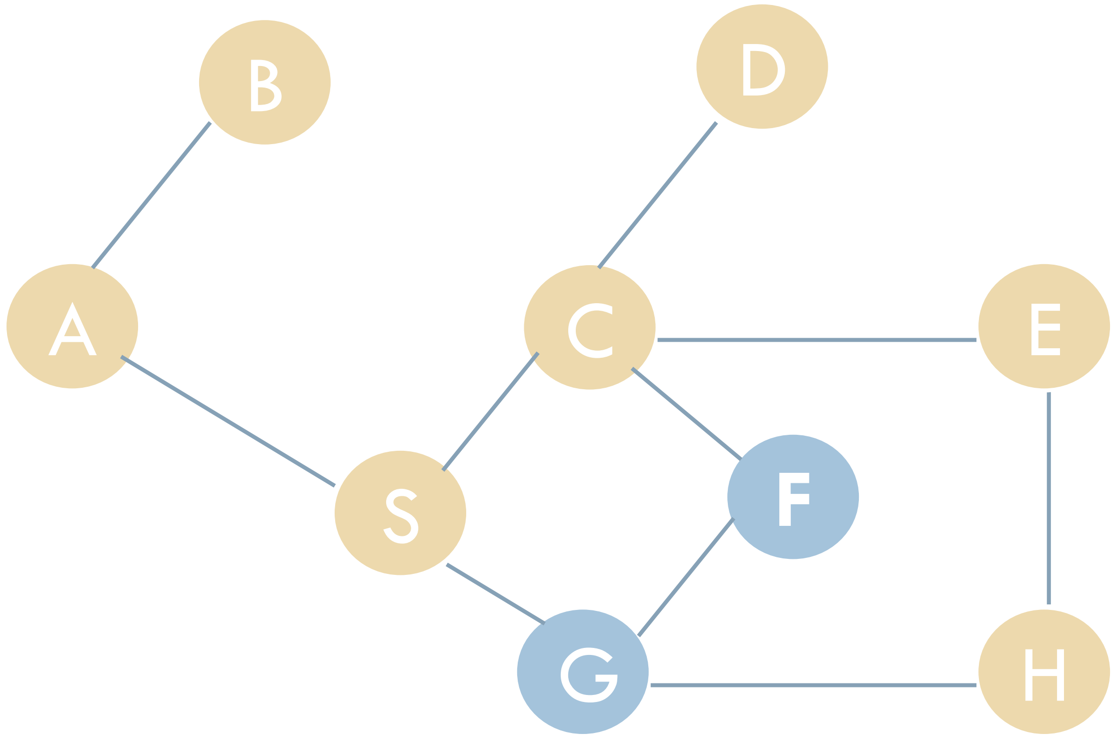
Depth-First Search (DFS)



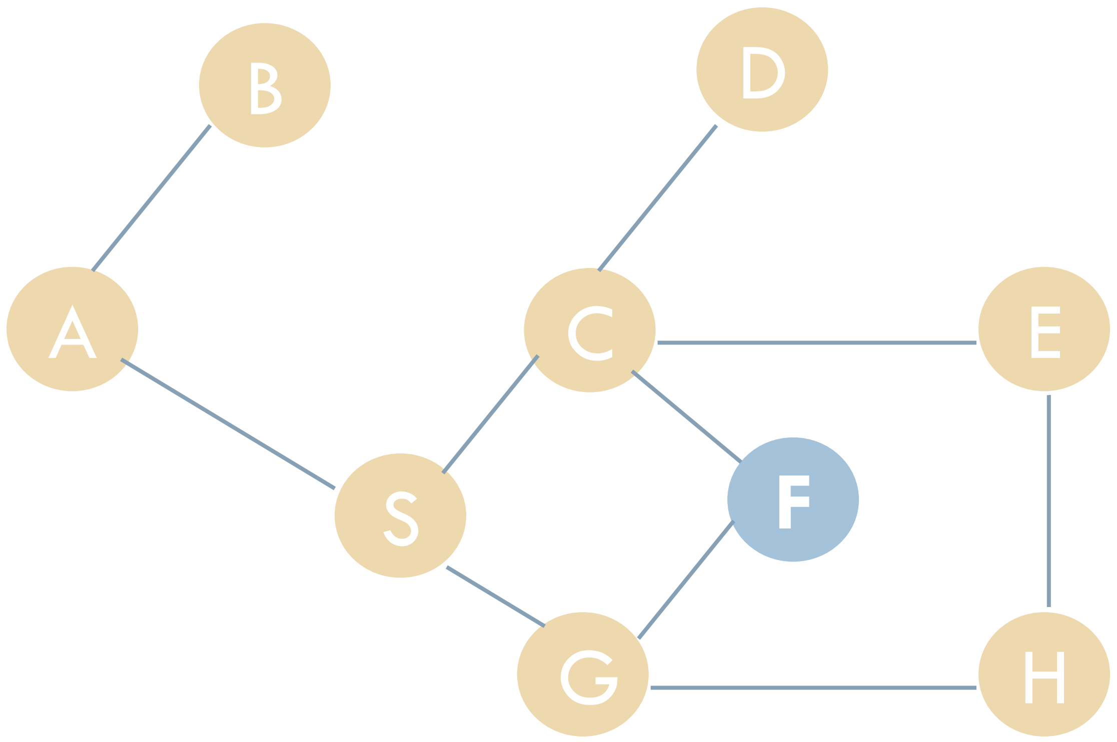
Depth-First Search (DFS)



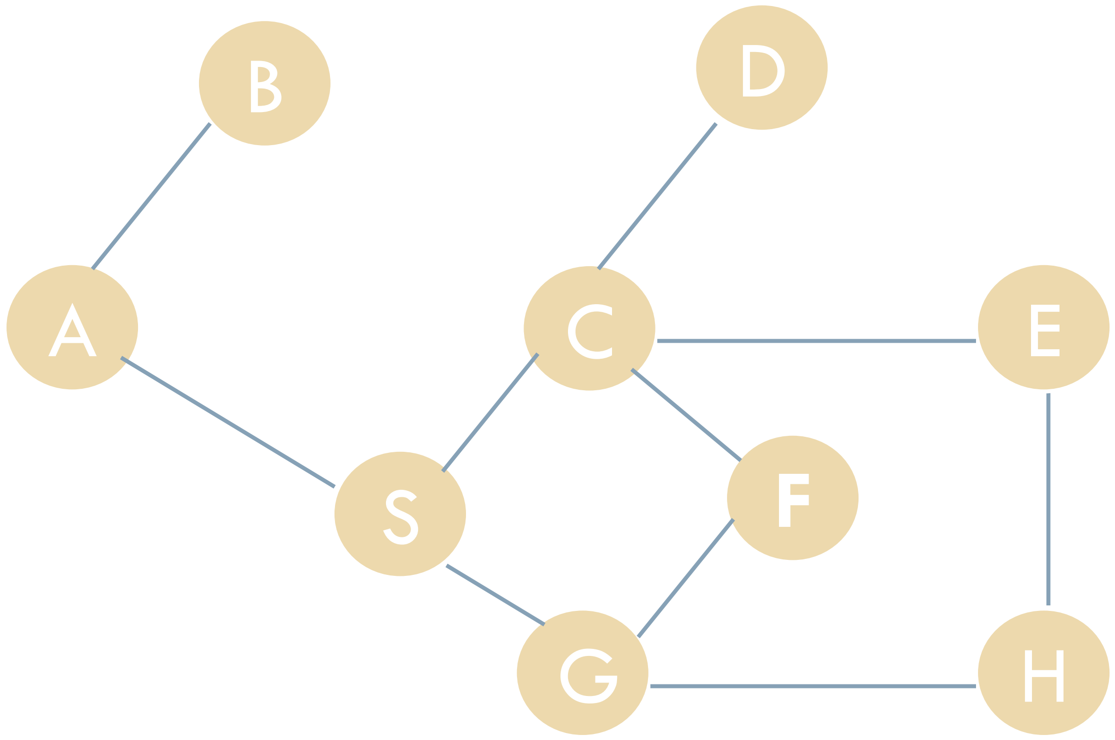
Depth-First Search (DFS)



Depth-First Search (DFS)

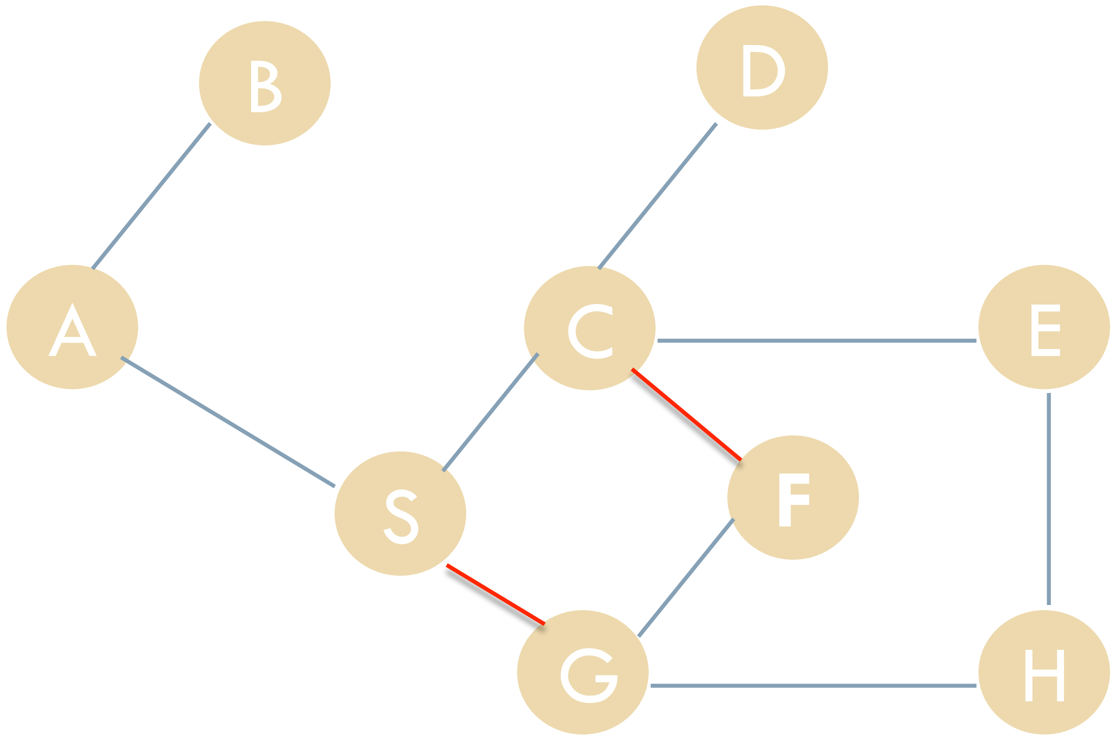


Depth-First Search (DFS)

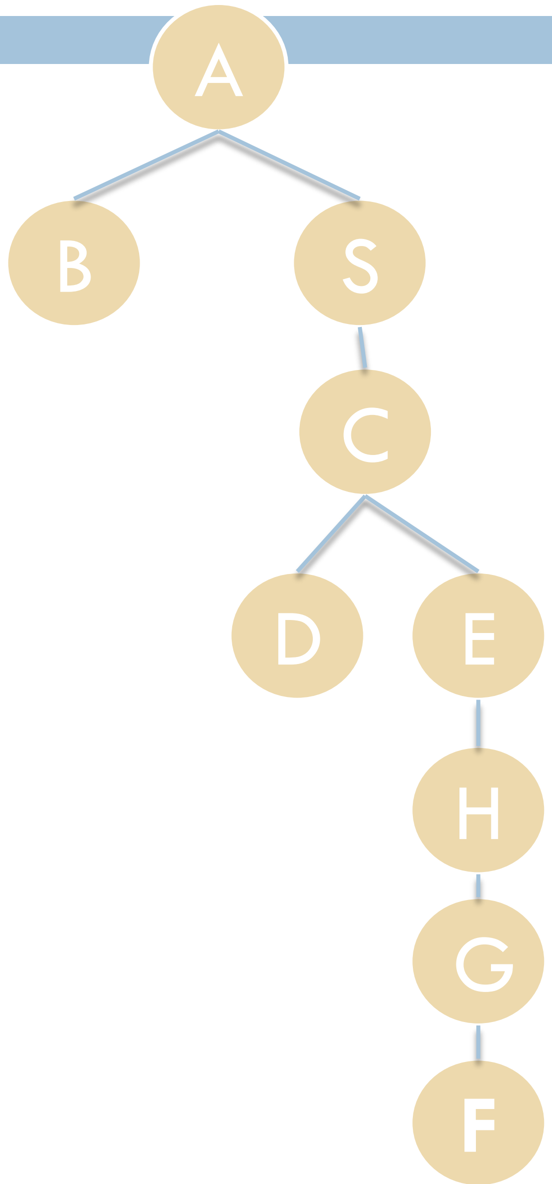


{A B S C D E H G F}

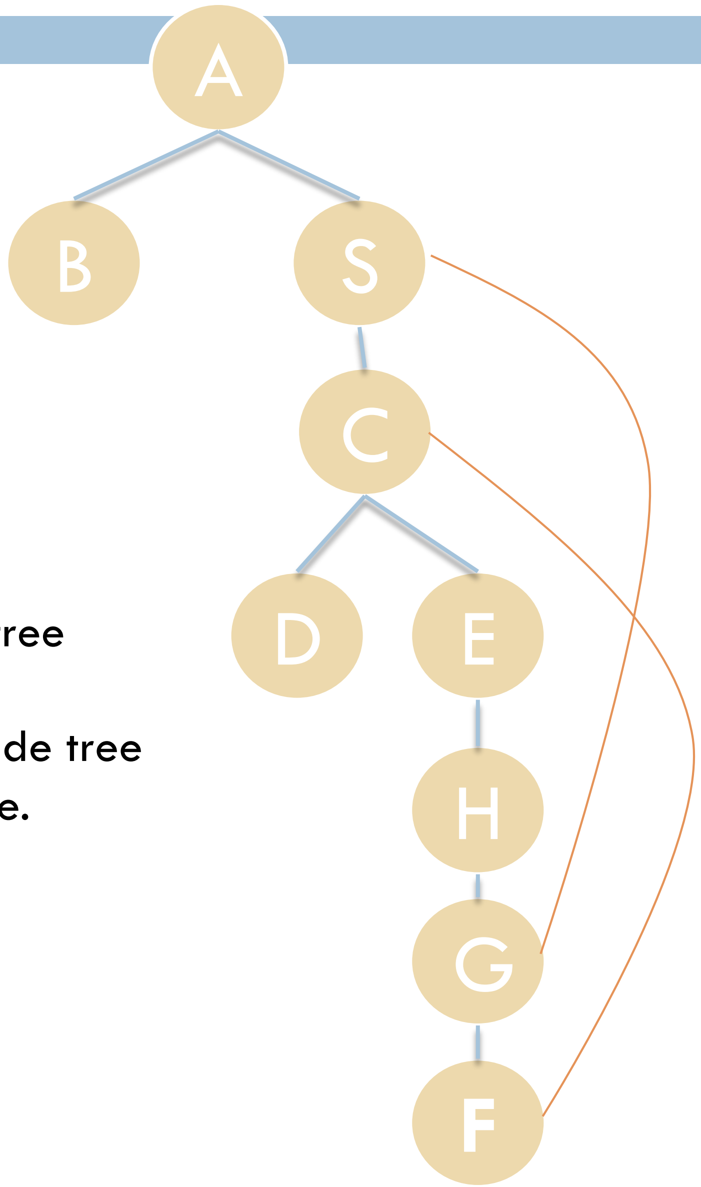
Edges of G that are not in DFS



Tree after DFS run and edges in G



Tree after DFS run



No edge of G
goes from a left subtree
to a right subtree.
The edges of G outside tree
are called back edge.
No cross edge !

Depth-First Algorithm using Stack

```
DFS($s$)
1.Initialize $S$ to be a stack with element $s$ only.
2. While $S$ is not empty
3.     Take a node $u$ from top of $S$.
4.     If Explored[$u$] =false then
5.         Set Explored[$u$]=true
6.         For every $uv$ edge add $v$ to $S$.
```

DFS running time

1) If we represent the graph G by adjacency matrix then the running time of DFS algorithm is $O(n^2)$, where n is the number of nodes.
2) If we represent the graph G by link lists then the running time of DFS algorithm is $O(m + n)$, where m is the number of edges and n is the number of nodes.

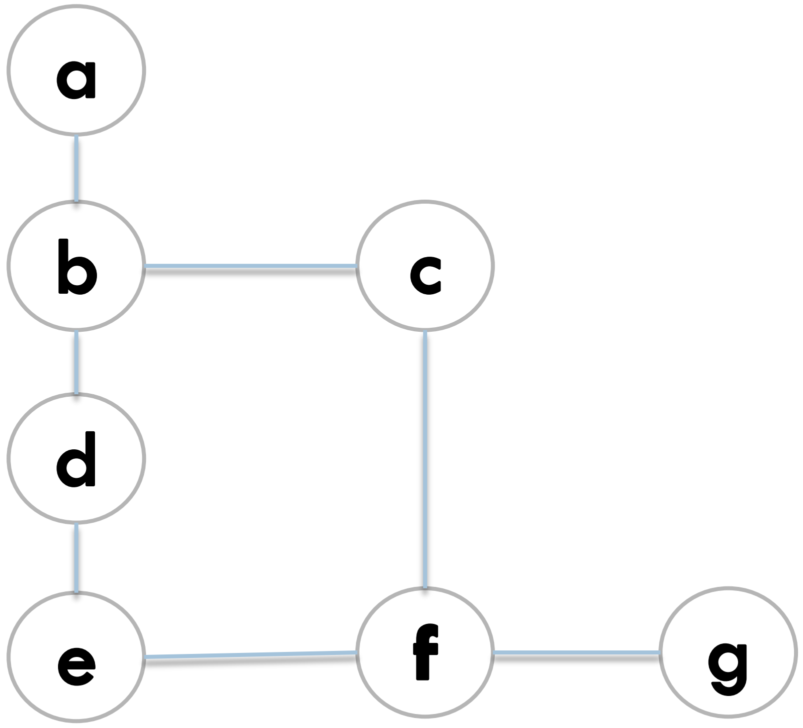
DFS Applications

- We use DFS to find a cycle or shortest cycle. We also use DFS to find strong components in digraph (used in Tarjan's algorithm)

BFS & DFS Example

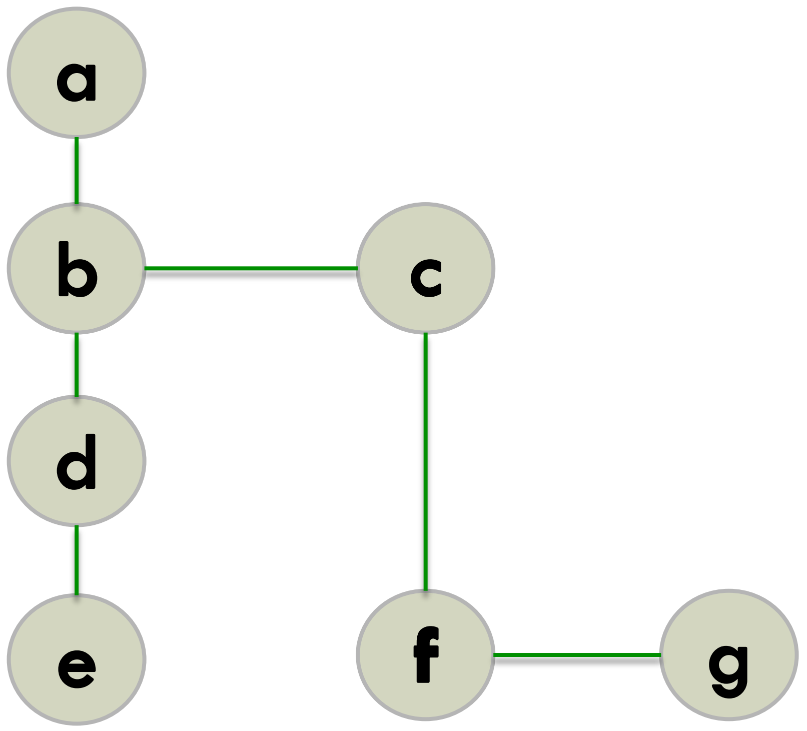
V(G) = {a,b,c,d,e,f,g}

E(G) = {ab,bc,bd,de,ef,fc,fg}

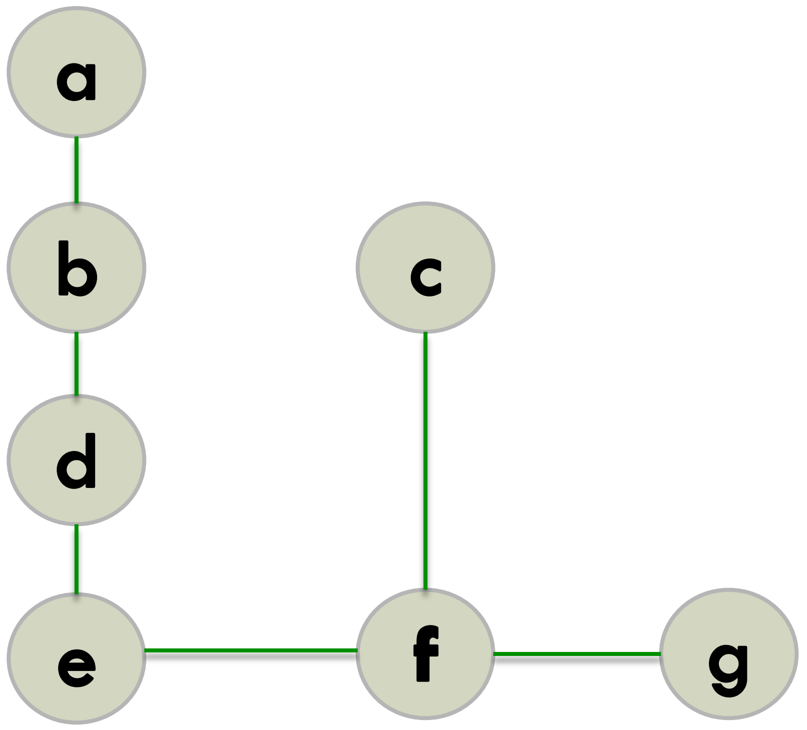


BFS & DFS Example

BFS: ab, bd, bc, de, cf, fg



DFS: ab, bd, de, ef, fg, fc



BFS & DFS Example

- Previous example shows that if there is a cycle in graph G then the BFS tree and DFS tree are different.

THANKS !