Design and Analysis of Algorithm

Lecture-20:
Basic Search and
Traversing Techniques

Contents

1. Techniques for binary trees.
2. Techniques for graph.
3. Connected Components and Spanning Trees

Objective Question

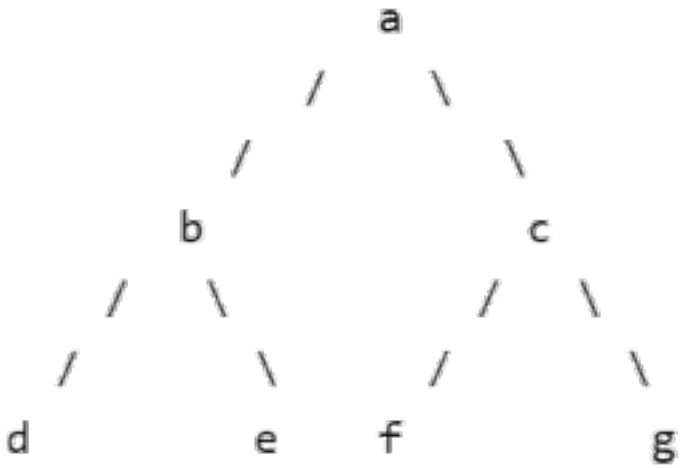
The pre-order traversal of a binary search tree is given by $12, 8, 6, 2, 7, 9, 10, 16, 15, 19, 17, 20$. Then the post-order traversal of this tree is:

(A) $2, 6, 7, 8, 9, 10, 12, 15, 16, 17, 19, 20$
(B) $2, 7, 6, 10, 9, 8, 15, 17, 20, 19, 16, 12$
(C) $7, 2, 6, 8, 9, 10, 20, 17, 19, 15, 16, 12$
(D) $7, 6, 2, 10, 9, 8, 15, 16, 17, 20, 19, 12$

Objective Question

The inorder and preorder traversal of a binary tree are $d\ b\ e\ a\ f\ c\ g$ and $a\ b\ d\ e\ c\ f\ g$, respectively. What is the postorder traversal of this binary tree?

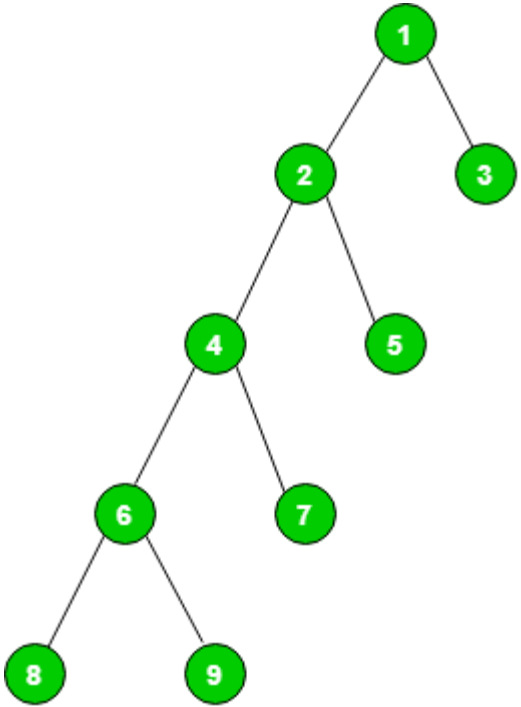
A. $d\ e\ b\ f\ g\ c\ a$
B. $e\ d\ b\ g\ f\ c\ a$
C. $e\ d\ b\ f\ g\ c\ a$
D. $d\ e\ f\ g\ b\ c\ a$



Objective Question

The postorder traversal of a binary tree is $8, 9, 6, 7, 4, 5, 2, 3, 1$. The inorder traversal of the same tree is $8, 6, 9, 4, 7, 2, 5, 1, 3$. The height of a tree is the length of the longest path from the root to any leaf. The height of the binary tree above is

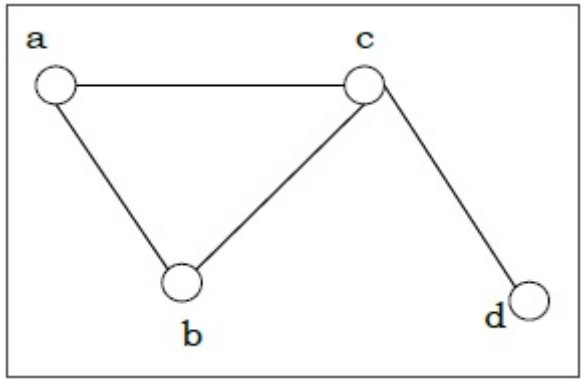
A. 2
B. 3
C. 4
D. 5



Graph

Definition:

A graph (denoted as $G = (V, E)$) consists of a non-empty set of vertices or nodes V and a set of edges E. A vertex a represents an endpoint of an edge. An edge joins two vertices a, b and is represented by set of vertices it connects.

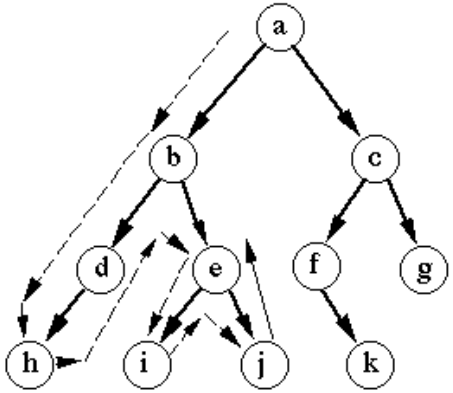


Traversal

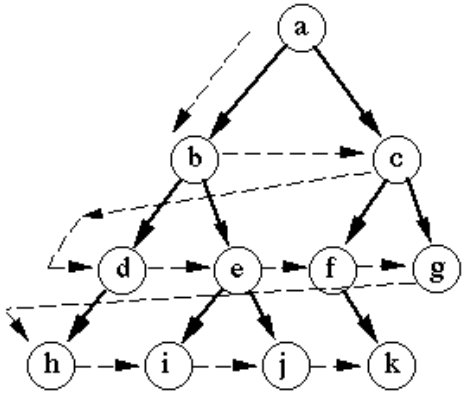
When the search necessarily involves the examination of every vertex in the object being searched, it is called a traversal.

Traversal Techniques

- Breadth First search
- Depth First Search



Depth-first search



Breadth-first search

Breadth First Search

Approach

Step 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.

Step 2 − If no adjacent vertex is found, remove the first vertex from the queue.

Step 3 − Repeat Step 1 and Step 2 until the queue is emp

Algorithm 9.4 BFS
Input: A directed or undirected graph $G = (V, E)$.
Output: Numbering of the vertices in breadth-first search order.

```
1.  bfn ← 0
2.  for each vertex v ∈ V
3.      mark v unvisited
4.  end for
5.  for each vertex v ∈ V
6.      if v is marked unvisited then bfs(v)
7.  end for
```

Procedure *bfs*(v)

```
1.  Q ← {v}
2.  mark v visited
3.  while Q ≠ {}
4.      v ← Pop(Q)
5.      bfn ← bfn + 1
6.      for each edge (v, w) ∈ E
7.          if w is marked unvisited then
8.              Push(w, Q)
9.              mark w visited
10.         end if
11.     end for
12. end while
```

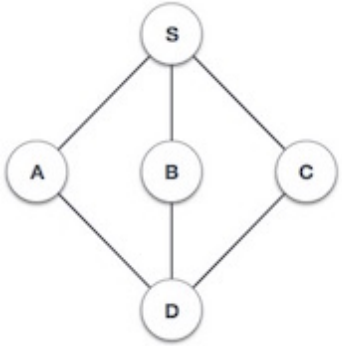
Breadth First Search

Algorithm 9.4 BFS
Input: A directed or undirected graph $G = (V, E)$.
Output: Numbering of the vertices in breadth-first search order.

```
1.  bfn ← 0
2.  for each vertex v ∈ V
3.      mark v unvisited
4.  end for
5.  for each vertex v ∈ V
6.      if v is marked unvisited then bfs(v)
7.  end for
```

Procedure *bfs*(v)

```
1.  Q ← {v}
2.  mark v visited
3.  while Q ≠ {}
4.      v ← Pop(Q)
5.      bfn ← bfn + 1
6.      for each edge (v, w) ∈ E
7.          if w is marked unvisited then
8.              Push(w, Q)
9.              mark w visited
10.         end if
11.     end for
12. end while
```



$Unvisited = \{S, A, B, C, D\}$ $bfs(S)$ $Q = \{S\}$

$visited = \{S\}$ v=S, bfn=1 $Q = \{A, B, C\}$

$visited = \{S, A, B, C\}$ v=A, bfn=2 $Q = \{B, C, D\}$

$visited = \{S, A, B, C, D\}$ v=B, bfn=3 $Q = \{C, D\}$

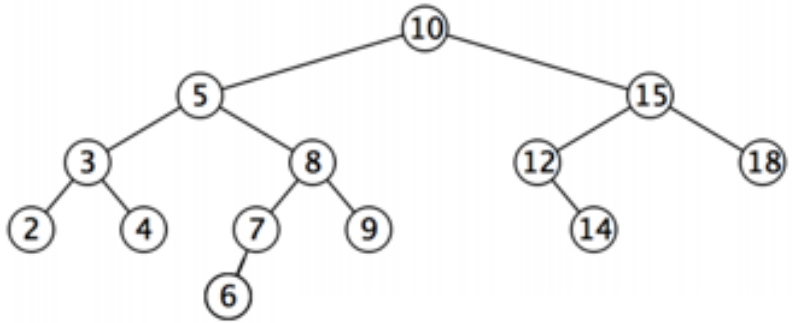
$visited = \{S, A, B, C, D\}$ v=C, bfn=4 $Q = \{D\}$

$visited = \{S, A, B, C, D\}$ v=D, bfn=5 $Q = \{\}$

Graph traversals: BFS

Breadth First Search: a traversal on graphs where you traverse “Level by Level”.

It can be thought of as a sound wave spreading from a starting point, going outwards in all directions possible



BFS traversal: 10, 5, 15, 3, 8, 12, 18, 2, 4, 7, 9, 14, 6

Depth First Search

Approach

Step 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a stack.

Step 2 − If no adjacent vertex is found, remove the first vertex from the stack.

Step 3 − Repeat Step 1 and Step 2 until the queue is emp

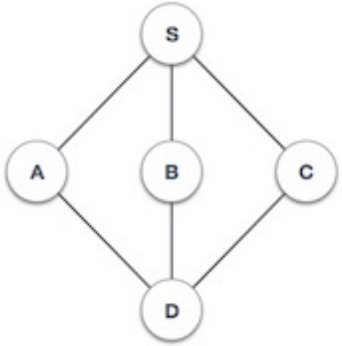
```
init() {
    For each u ∈ G
        u.visited = false
     For each u ∈ G
       DFS(G, u)
}

DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
```

Depth First Search

```
init() {
    For each u ∈ G
        u.visited = false
     For each u ∈ G
       DFS(G, u)
}

DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
```



$Unvisited = \{S, A, B, C, D\}$ DFS (S)

$visited = \{S\}$ $Adj.S = \{A, B, C\}$ DFS (A)

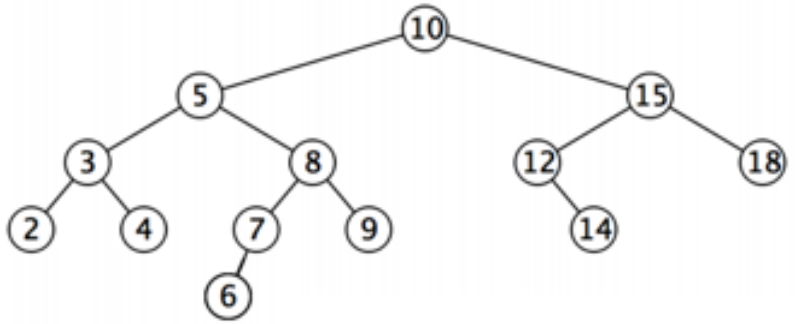
$visited = \{S, A,\}$ $Adj.A = \{D\}$ DFS(D)

$Order\ of\ visit = S, A, D, B, C$

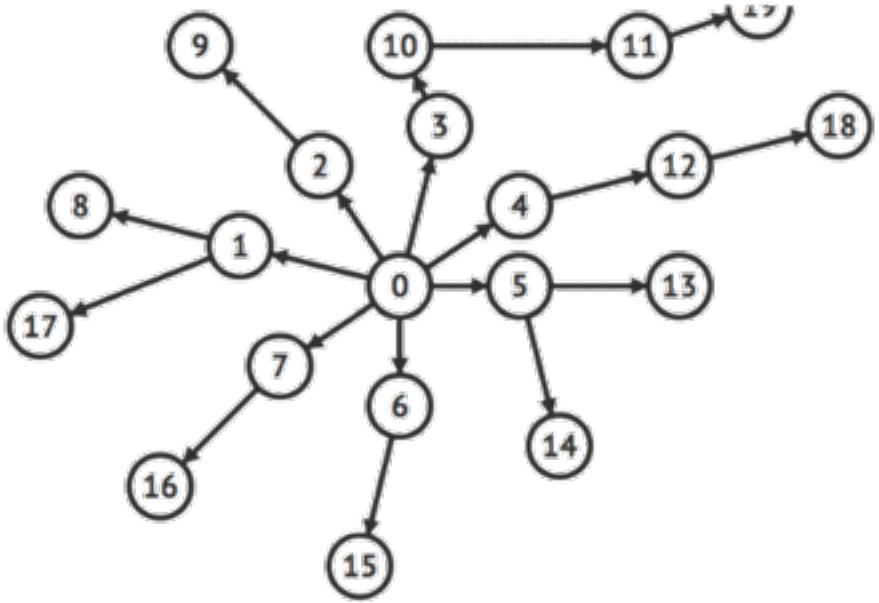
Graph traversals: DFS

Depth First Search: a traversal on graphs where you traverse “deep nodes” before all the shallow ones

you go as far as you can down one path till you hit a dead end. Once you hit a dead end, you backtrack/undo until you find some options/edges that you haven’t actually tried yet.



DFS traversal: 10, 5, 3, 2, 4, 8, 7,6, 9, 15, 12, 14, 18



BFS ordering: ordering within each layer doesn't matter / any ordering is valid

DFS ordering: which path you choose next at any point doesn't matter / any is valid as long as you haven't explored it before

BFS ordering: 0, [1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 12, 13, 14, 15, 16, 17], [11, 18], [19]

DFS ordering: 0, 2, 9, 3, 10, 11, 19, 4, 12, 18, 5, 13, 14, 6, 15, 7, 16, 1,17, 8

Question

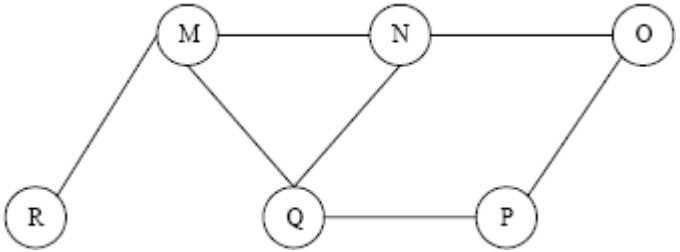
The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is

A. M, N, O, P, Q, R

B. N, Q, M, P, O, R

C. Q, M, N, P, R, O

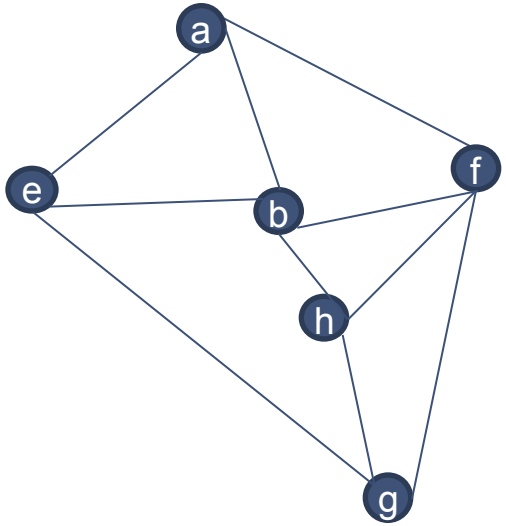
D. Q, M, N, P, O ,R



Question

Which of the following sequence is depth first traversals of the given graph?

1. a b e g h f
2. a b f e h g
3. a b f h g e
4. a f g h b e

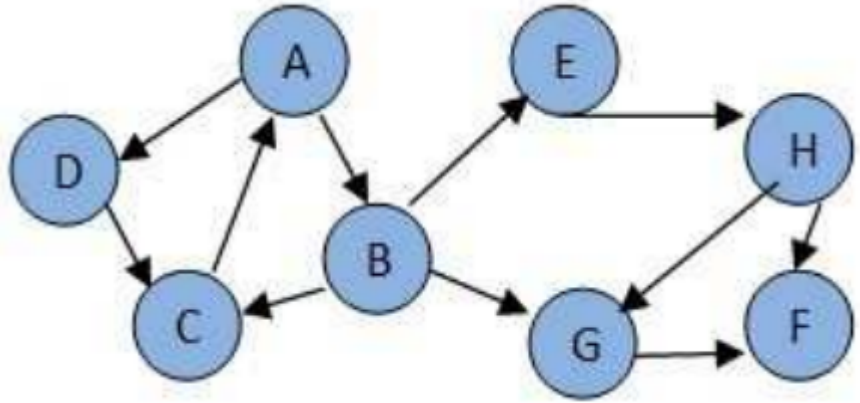


A. 1, 2 and 3
B. 1 and 4
C. 2, 3 and 4
D. 1 , 3 and 4

Question

Consider the following graph. If there is ever a decision between multiple neighbor nodes in the BFS or DFS algorithms, assume we always choose the letter closest to the beginning of the alphabet first.

In what order will the nodes be visited using a Breadth First Search and Depth First Search?



BFS: ABDCEGHF

DFS: ABCEHFGD