VE281

Data Structures and Algorithms

Binary Tree Traversal

Learning Objectives:

- Know the effect and procedure of pre-order, post-order, and in-order depth-first traversal
- Know the effect and procedure of level-order traversal

Binary Tree Traversal

- Many binary tree operations are done by performing a **traversal** of the binary tree.

- In a traversal, each node of the binary tree is visited **exactly once**.

- During the visit of a node, all actions (making a clone, displaying, evaluating the operator, etc.) with respect to this node are taken.

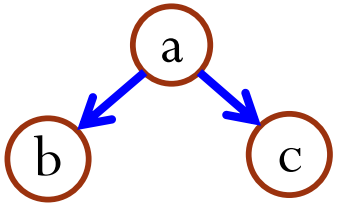
Binary Tree Traversal Methods

- Depth-first traversal
  - Pre-order
  - Post-order
  - In-order
- Level-order traversal

Pre-Order Depth-First Traversal

Procedure

- Visit the node
- Visit its left subtree
- Visit its right subtree

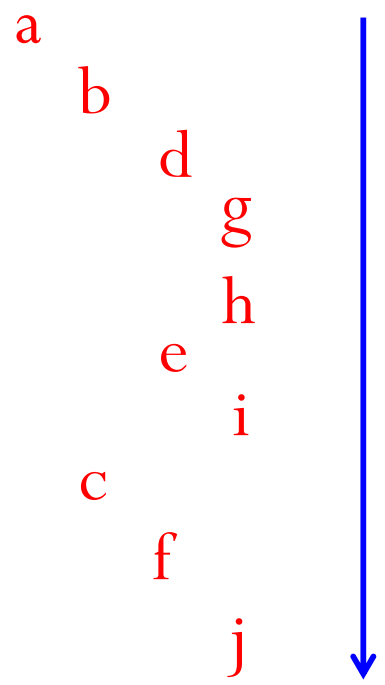


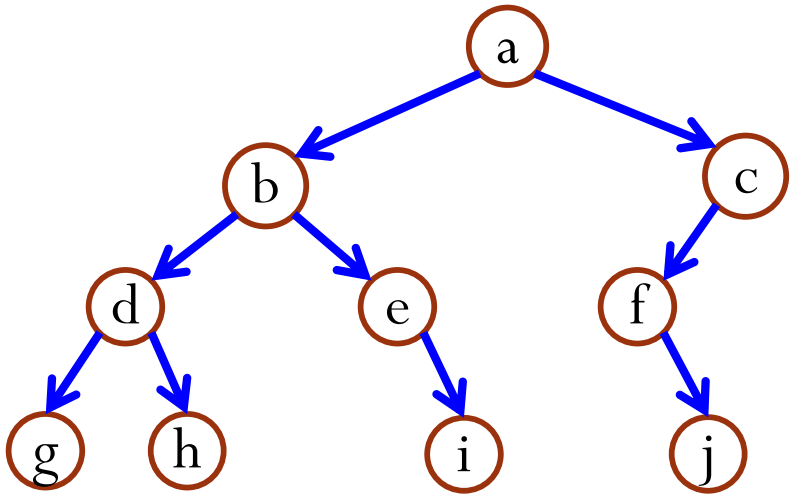
a→b→c

```
void preOrder(node *n) {
  if(!n) return;
  visit(n);
  preOrder(n->left);
  preOrder(n->right);
}
```

Pre-Order Depth-First Traversal

Example



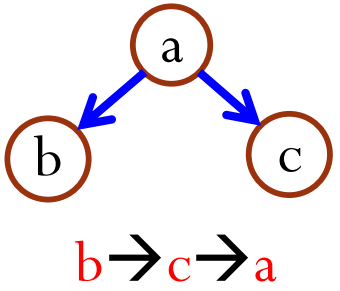


a→b→d→g→h→e→i→c→f→j

Post-Order Depth-First Traversal

Procedure

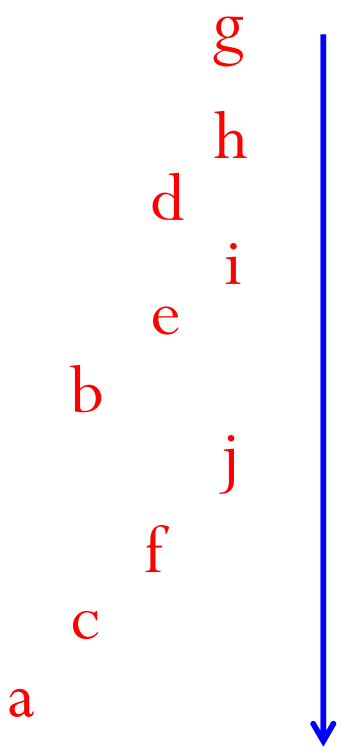
- Visit the left subtree
- Visit the right subtree
- Visit the node

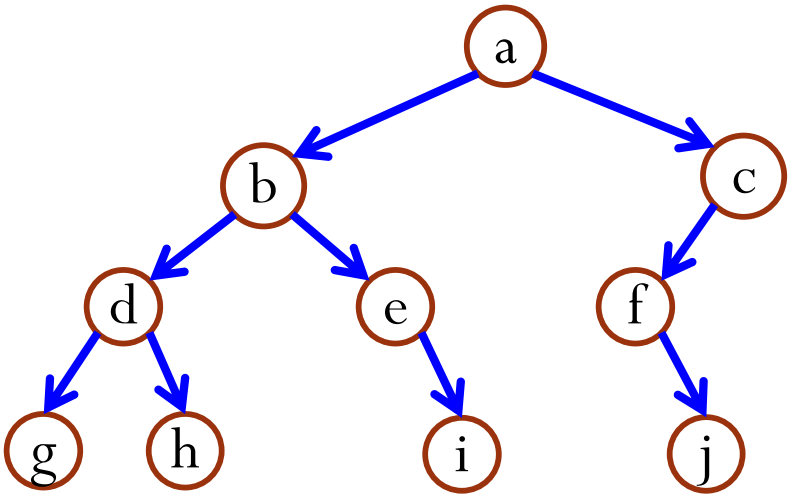



```
void postOrder(node *n) {
  if(!n) return;
  postOrder(n->left);
  postOrder(n->right);
  visit(n);
}
```

Post-Order Depth-First Traversal

Example



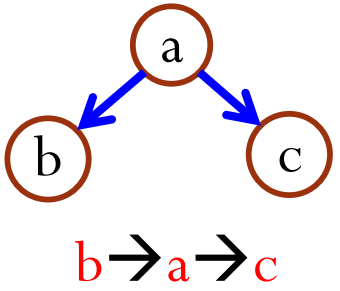


g→h→d→i→e→b→j→f→c→a

In-Order Depth-First Traversal

Procedure

- Visit the left subtree
- Visit the node
- Visit the right subtree




```
void inOrder(node *n) {
  if(!n) return;
  inOrder(n->left);
  visit(n);
  inOrder(n->right);
}
```

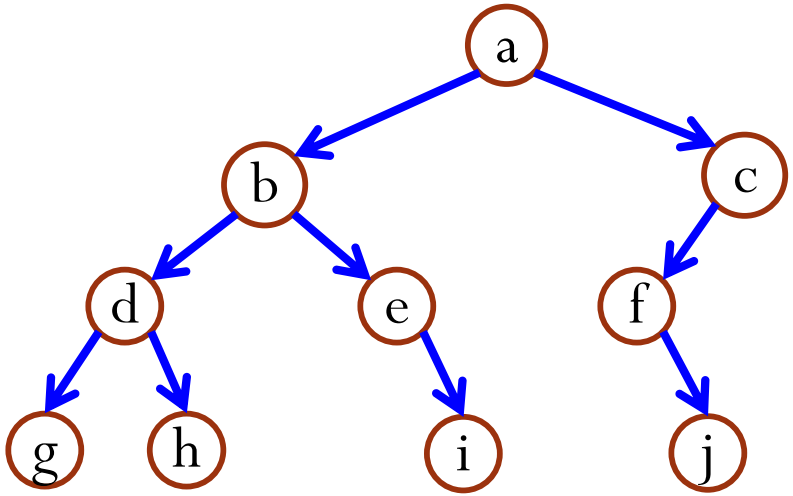
What Is the Result of In-Order Depth-First Traversal?

A. g, d, h, b, e, i, a, c, f, j

B. g, d, h, b, e, i, a, f, j, c

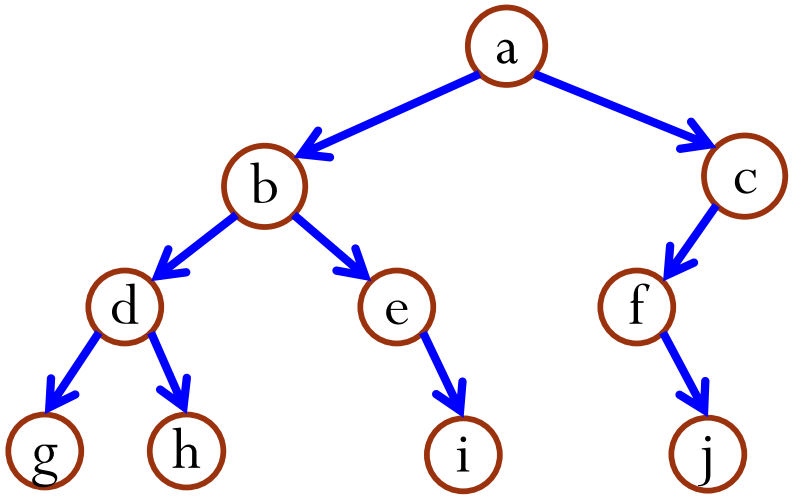
C. g, d, h, b, i, e, a, j, f, c

D. g, d, h, b, i, e, a, f, j, c



Level-Order Traversal

- We want to traverse the tree level by level **from top to bottom**.
- Within each level, traverse **from left to right**.



How can we implement this traversal?

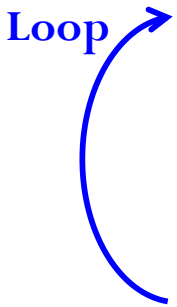
a→b→c→d→e→f→g→h→i→j

Level-Order Traversal

Procedure

- Use a queue!

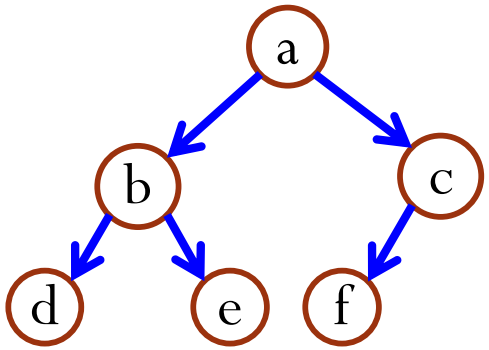
1. Enqueue the root node into an empty queue.
2. While the queue is not empty, dequeue a node from the front of the queue.
    1. Visit the node.
    2. Enqueue its left child (if exists) and right child (if exists) into the queue.



Level-Order Traversal

Code and Example

```
void levelOrder(node *root) {
  queue q; // Empty queue
  q.enqueue(root);
  while(!q.isEmpty()) {
    node *n = q.dequeue();
    visit(n);
    if(n->left) q.enqueue(n->left);
    if(n->right) q.enqueue(n->right);
  }
}
```



Queue: a b c d e f

Output: a b c d e f

Binary Tree Traversal

Application

- The expression $a/b + (c - d)e$ has been encoded as a tree T.
  - The leaves are **operands**.
  - The internal nodes are **operators**.
- How would you traverse the tree T to print out the expression (ignoring parentheses)?
  - In-order depth-first traversal.
- What is the expression printed out by post-order depth-first traversal?
  - $ab/cd - e * +$
  - **Reverse Polish Notation**