ILLINOIS

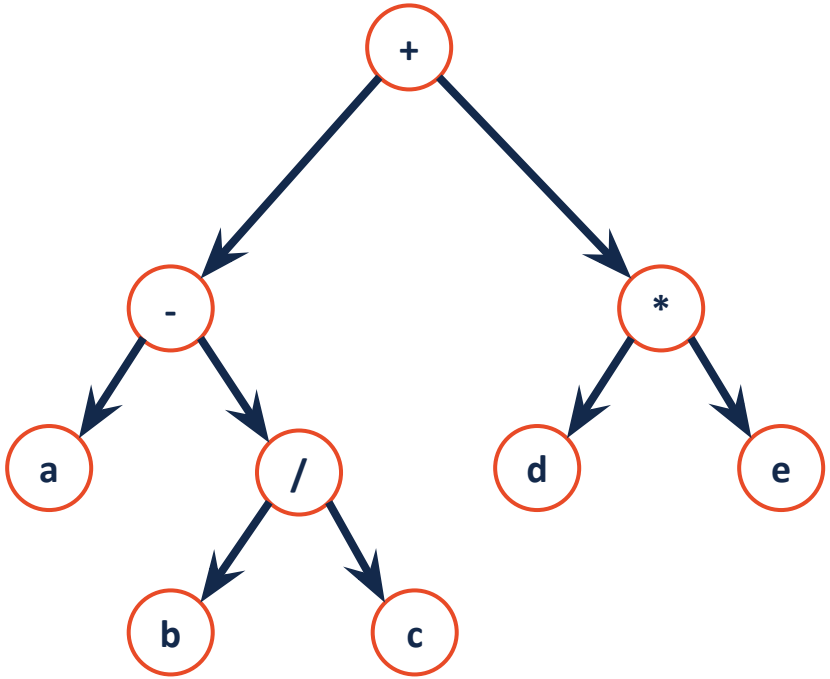
Tree Reconstruction, Traversals, and Search

Join Code: **225**

Learning Objectives

1. Reconstruct a tree from traversals
2. Implement a level order traversal
3. Analyze the runtime of traversals
4. Analyze the space and time complexities of searches

Accessing all nodes



Combining information

Preorder Output: + - a / b c * d e

Postorder Output: a b c / - d e * +

Combining information

Preorder Output: + - a / b c * d e

Postorder Output: a b c / - d e * +

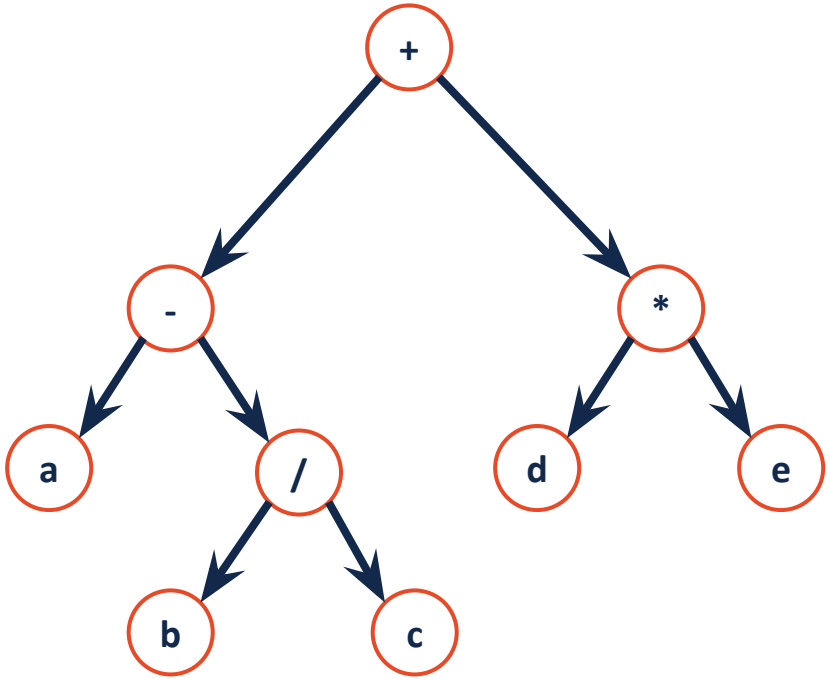
Accessing all nodes

In-order

Pre-order

Post-order

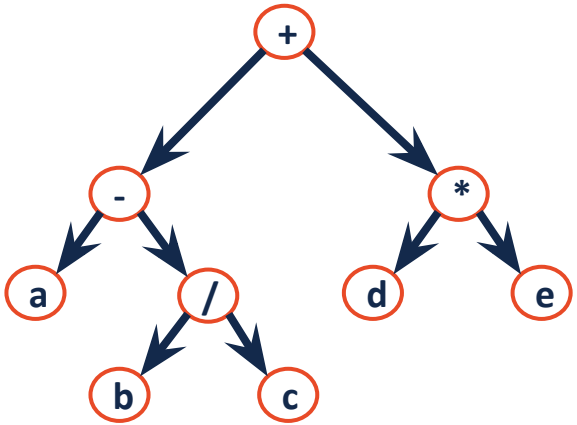
Level order



What data structure can I use to help implement this?

Level Order Traversal

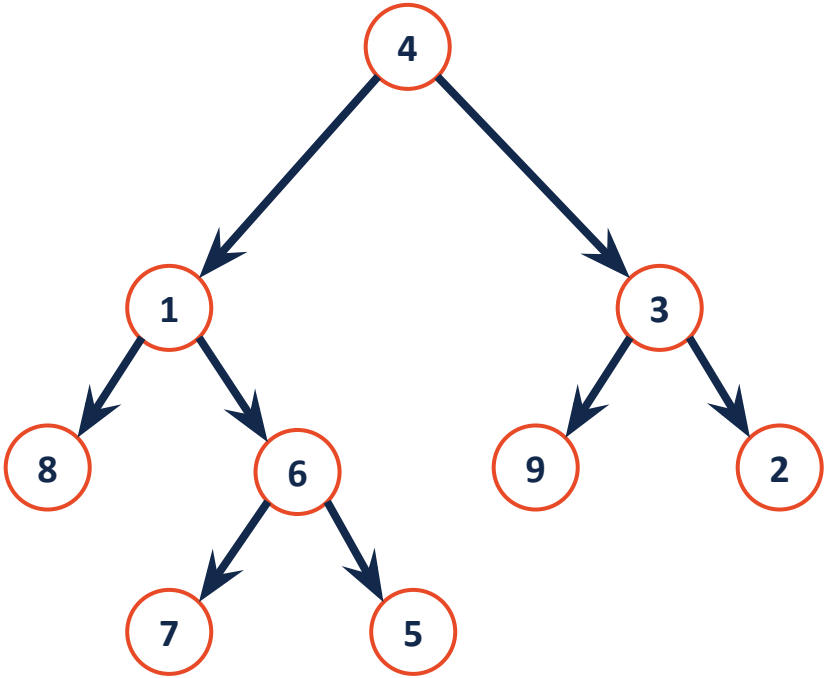
```
1  template<class T>
2  void BinaryTree<T>::levelOrder(TreeNode *
3  root) {
4      std::queue<TreeNode *> q;
5      q.push(root);
6      while(!q.empty()){
7          TreeNode* n = q.front();
8          q.pop();
9          if(n){
10             std::cout << n->data;
11             q.push(n->left);
12             q.push(n->right);
13         }
14     }
15 }
```



What is the runtime?

Tree Search

Tree Search - find a node of value v in the tree or return that it does not exist



Tree Search

Search Type	Traversal Used	Time Complexity	Space Complexity

Large Trees

Breadth-First -

Depth-First -