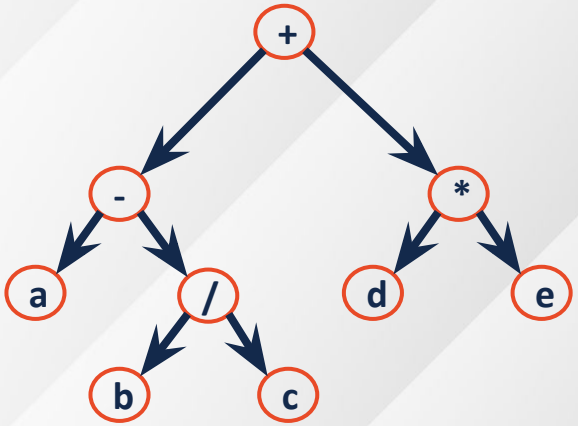
Graph Traversals

Learning Objectives

1. Implement BFS for Graph Traversals

Level Order Traversal

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



Graph Traversal - BFS

```
1  BFS(G):
2    Input: Graph, G
3    Output: A labeling of the edges on
4        G as discovery and cross edges
5
6    foreach (Vertex v : G.vertices()):
7      setLabel(v, UNEXPLORED)
8    foreach (Edge e : G.edges()):
9      setLabel(e, UNEXPLORED)
10   foreach (Vertex v : G.vertices()):
11     if getLabel(v) == UNEXPLORED:
12       BFS(G, v)
```

Discovery Edge
- An edge taken to find a new node

Cross Edge
- An edge taken to find an already visited node

Graph Traversal - BFS from a vertex

```
14  BFS(G, v):
15    Queue q
16    q.enqueue(v)
17
18    while !q.empty():
19      v = q.dequeue()
20      If (getLabel(v) != VISITED){
21          setLabel(v, VISITED)
22          foreach (Vertex w : G.adjacent(v)):
23            if getLabel(w) == UNEXPLORED:
24               setLabel(v, w, DISCOVERY)
25               setLabel(w, VISITED)
26               q.enqueue(w)
27            elseif getLabel(v, w) == UNEXPLORED:
28               setLabel(v, w, CROSS)
29      }
```

Node Labels
- Visited
- Unexplored

Edge Labels
- Discovery
- Cross
- Unexplored