ECE 312 (SP26):

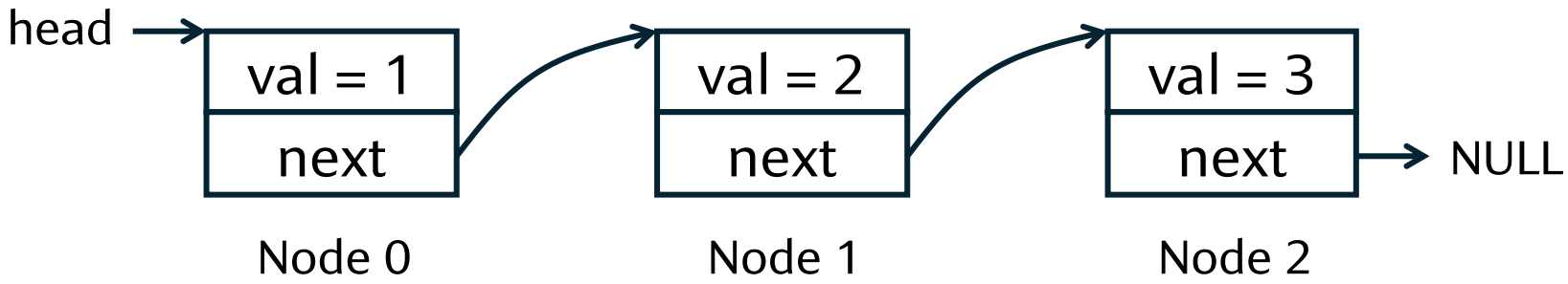
Tree

Neil Zhao

neil.zhao@utexas.edu

Linked List

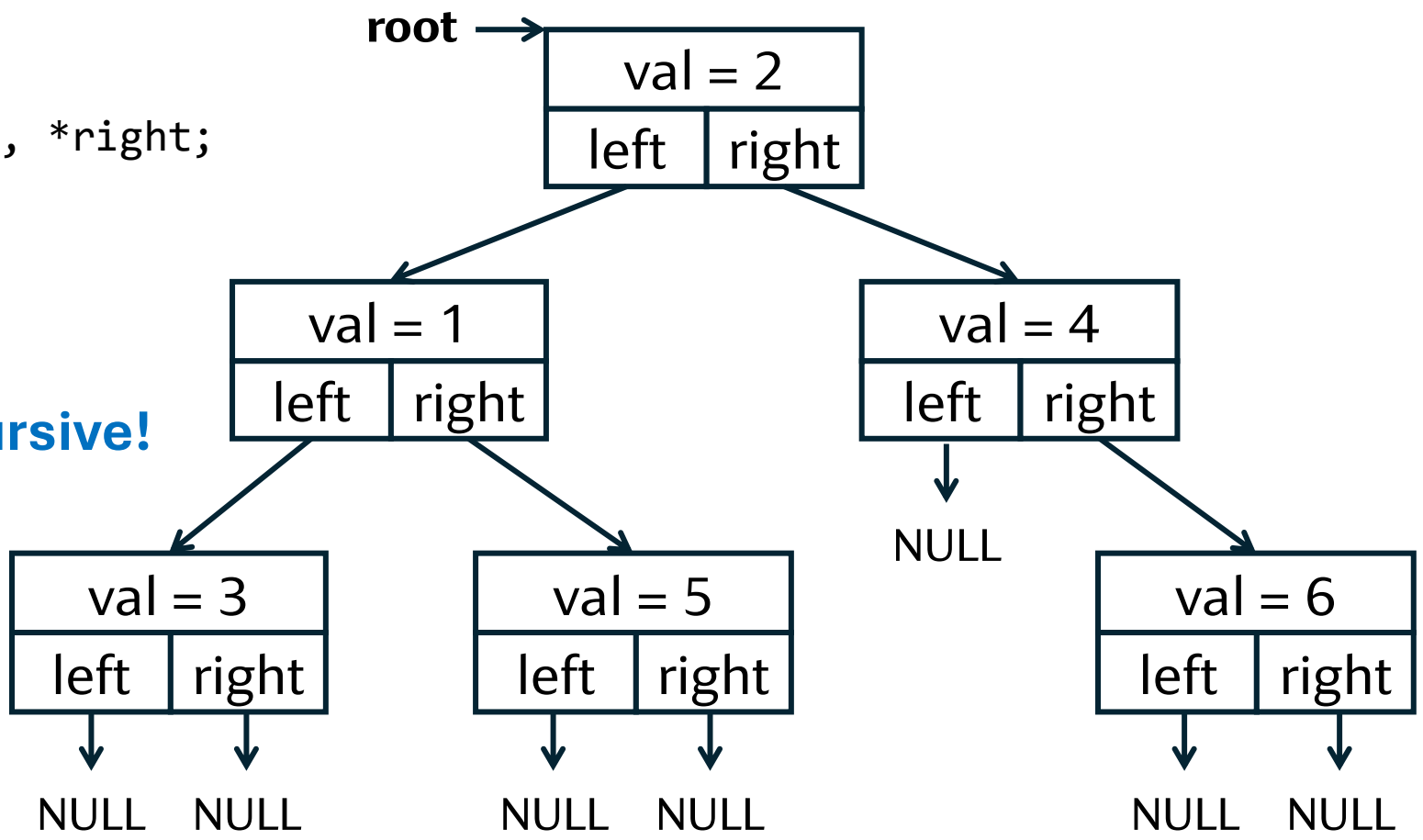
```
typedef struct ListNode {
    int val;
    struct ListNode *next;
} ListNode;
```



Tree

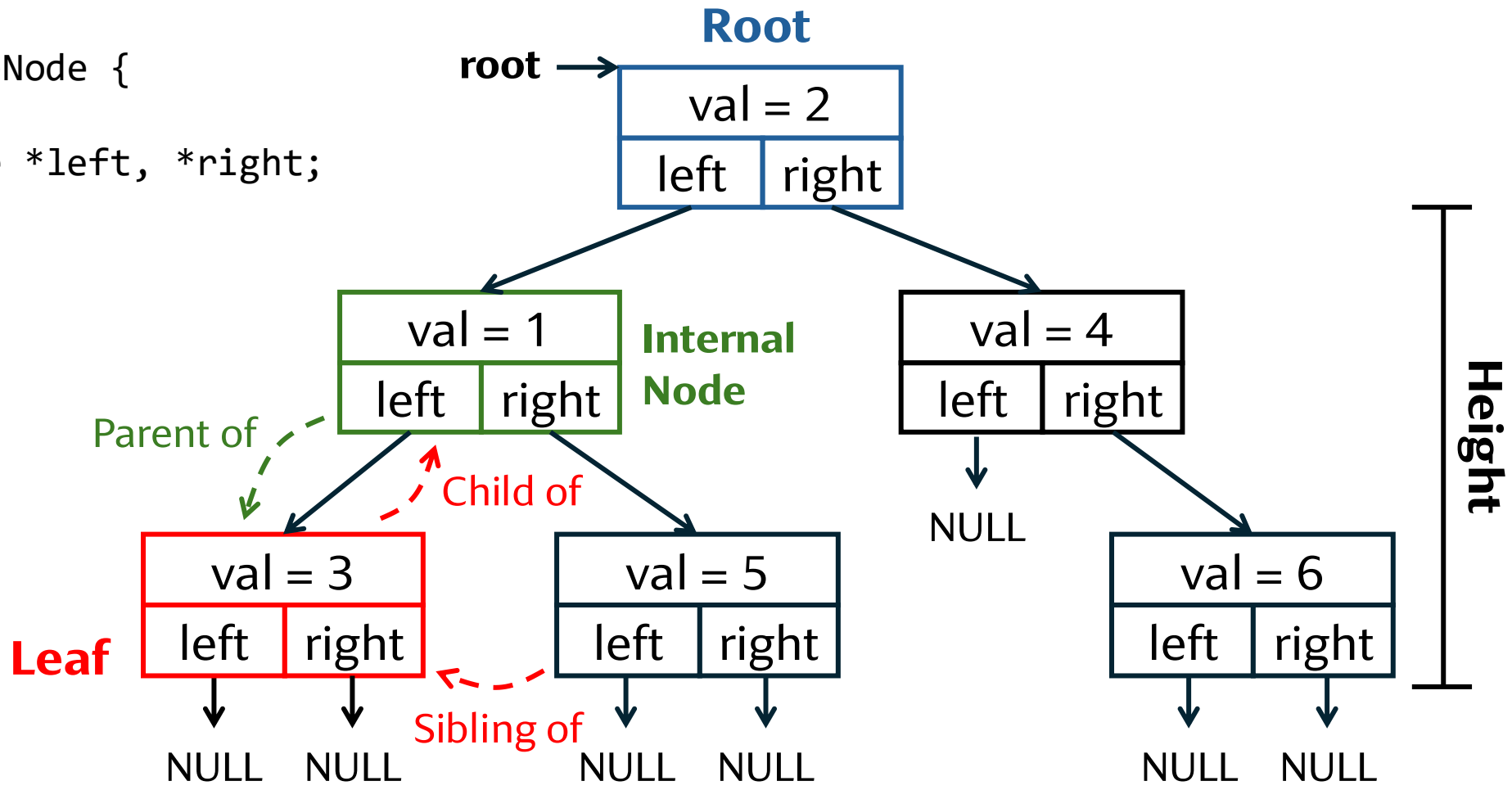
```
typedef struct TreeNode {
    int val;
    struct TreeNode *left, *right;
} TreeNode;
```

Trees are inherently recursive!

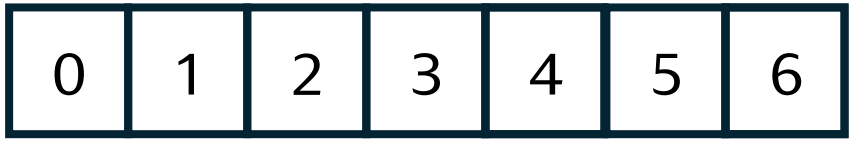


Tree

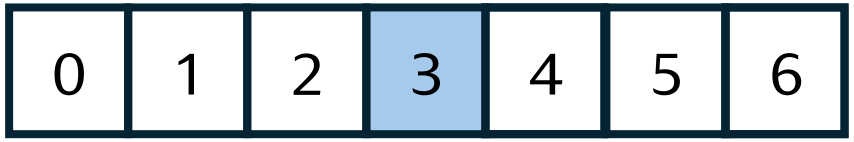
```
typedef struct TreeNode {
    int val;
    struct TreeNode *left, *right;
} TreeNode;
```



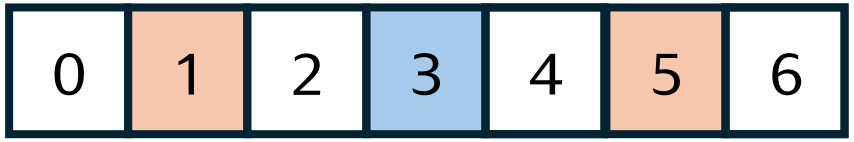
Revisiting Binary Search



Revisiting Binary Search

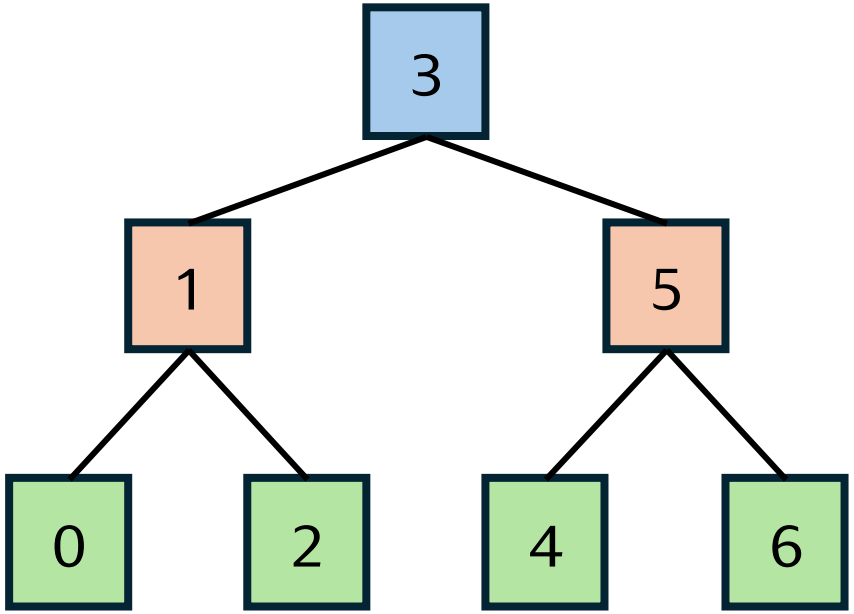


Revisiting Binary Search



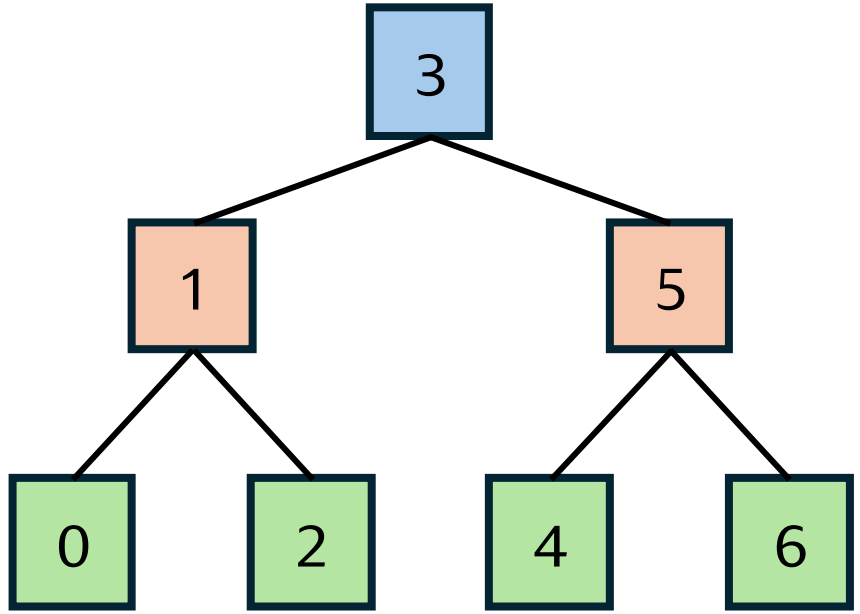
Revisiting Binary Search

Revisiting Binary Search



Searching in the Binary Search Tree (BST)

- Start from the root, target value?
  - Yes => Found!
  - Root is smaller
    - Search the right half
  - Root is larger
    - Search the left half



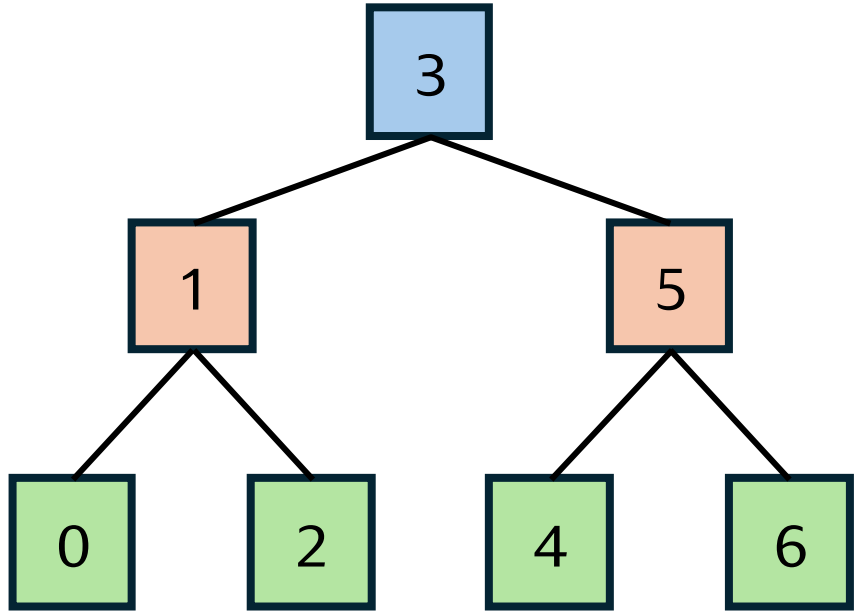
Traversing a Tree

```
void traverse(TreeNode *node) {
  if (!node) return;
  printf("%d ", node->val);
  traverse(node->left);

  traverse(node->right);

}
```

Preorder traversal: 3 1 0 2 5 4 6



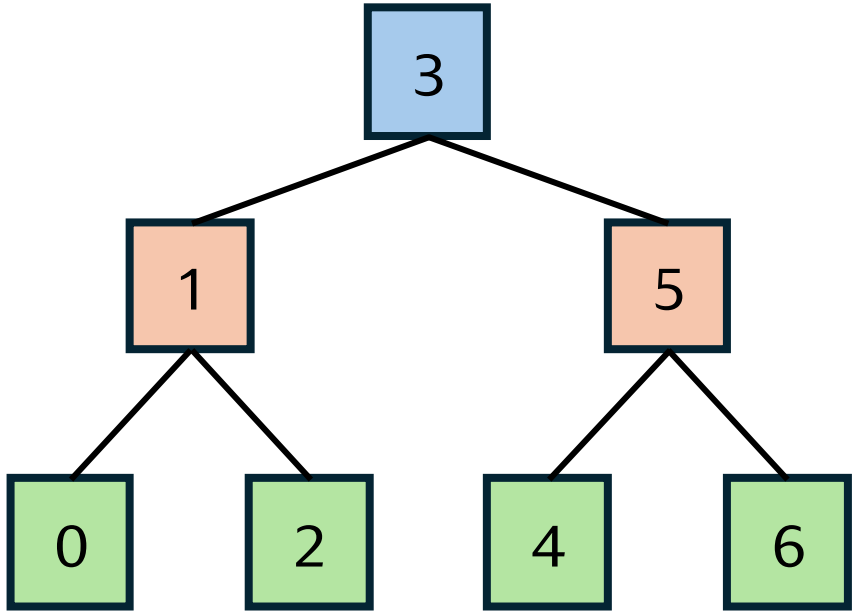
Traversing a Tree

```
void traverse(TreeNode *node) {
  if (!node) return;

  traverse(node->left);
  printf("%d ", node->val);
  traverse(node->right);

}
```

Inorder traversal: 0 1 2 3 4 5 6



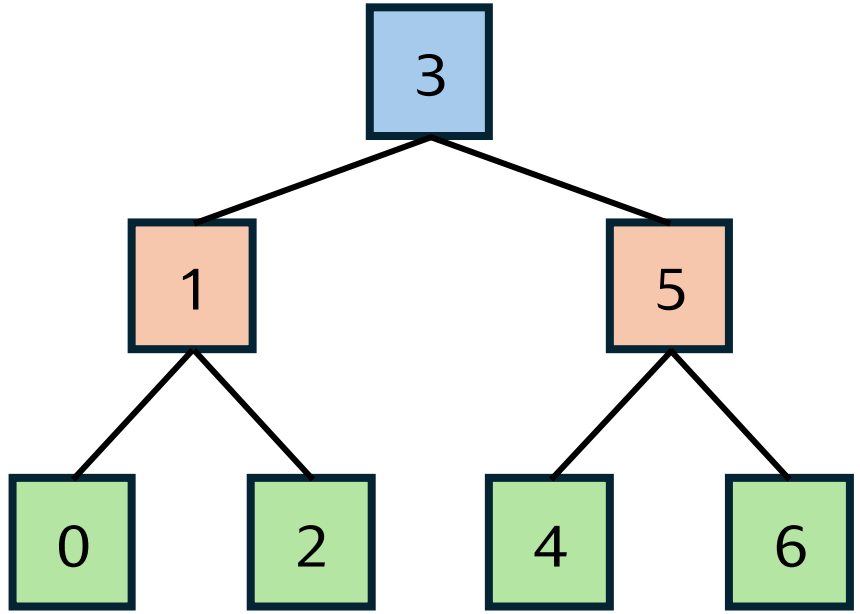
Traversing a Tree

```
void traverse(TreeNode *node) {
  if (!node) return;

  traverse(node->left);

  traverse(node->right);
  printf("%d ", node->val);
}
```

Postorder traversal: 0 2 1 4 6 5 3

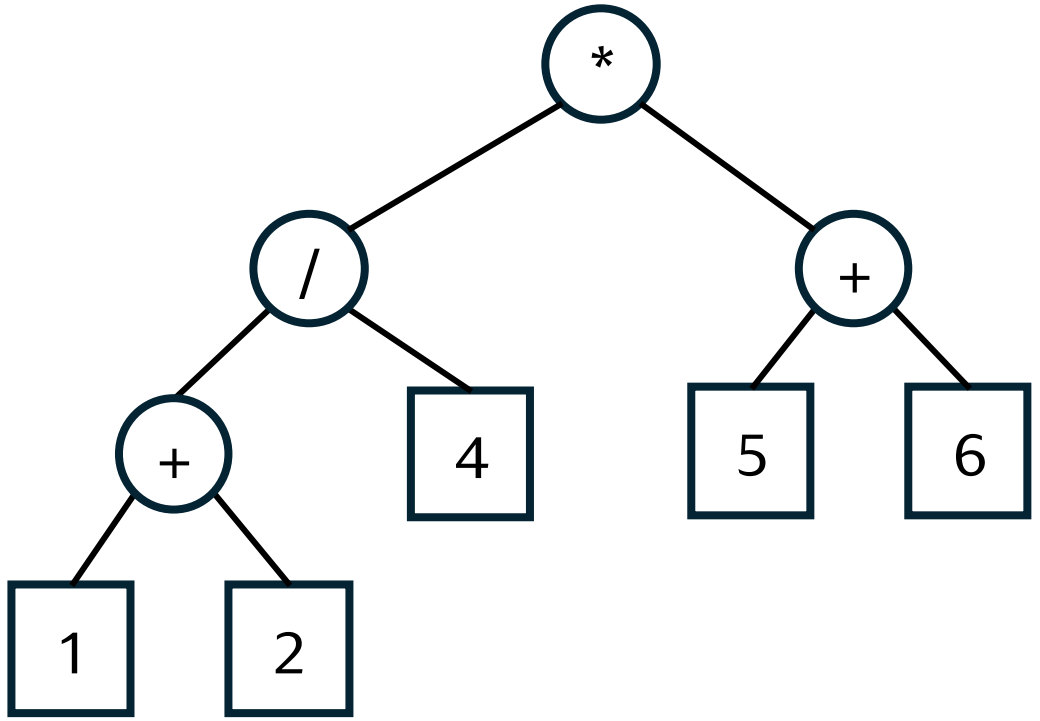


Expression Tree

Infix: `((1 + 2) / 4) * (5 + 6)`
Postfix: `1 2 + 4 / 5 6 + *`

How to evaluate the tree?

Postorder traversal

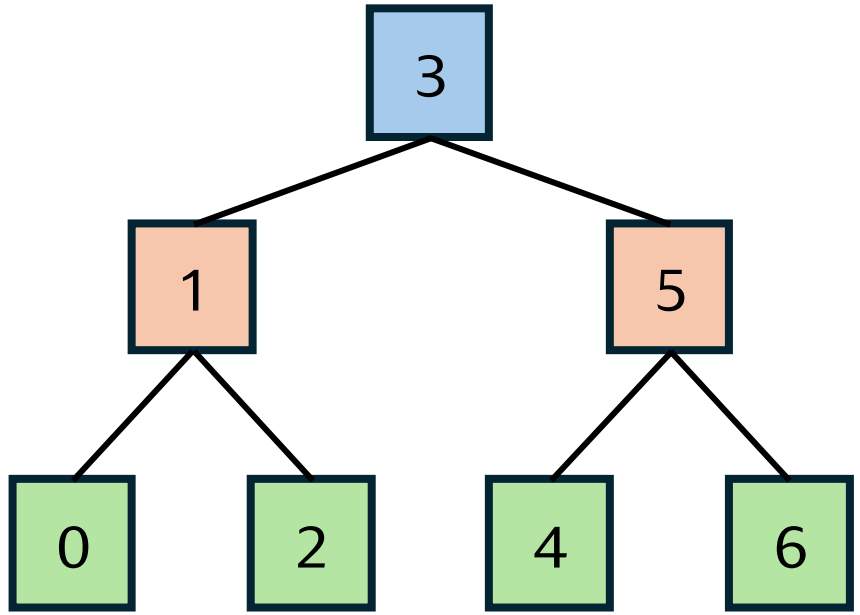


Breadth-First Traversal

```
void bf_traverse(TreeNode *node) {
  Queue q = {node};
  while (q.size) {
    TreeNode *n = queue_pop(&q);
    printf("%d ", n->val);

    if (n->left)
      queue_push(&q, n->left);

    if (n->right)
      queue_push(&q, n->right);
  }
}
```



Breadth-first traversal: 3 1 5 0 2 4 6