# CS 2050
# Computer Science II

Thyago Mota

METROPOLITAN STATE UNIVERSITY OF DENVER
LIVES TRANSFORMED

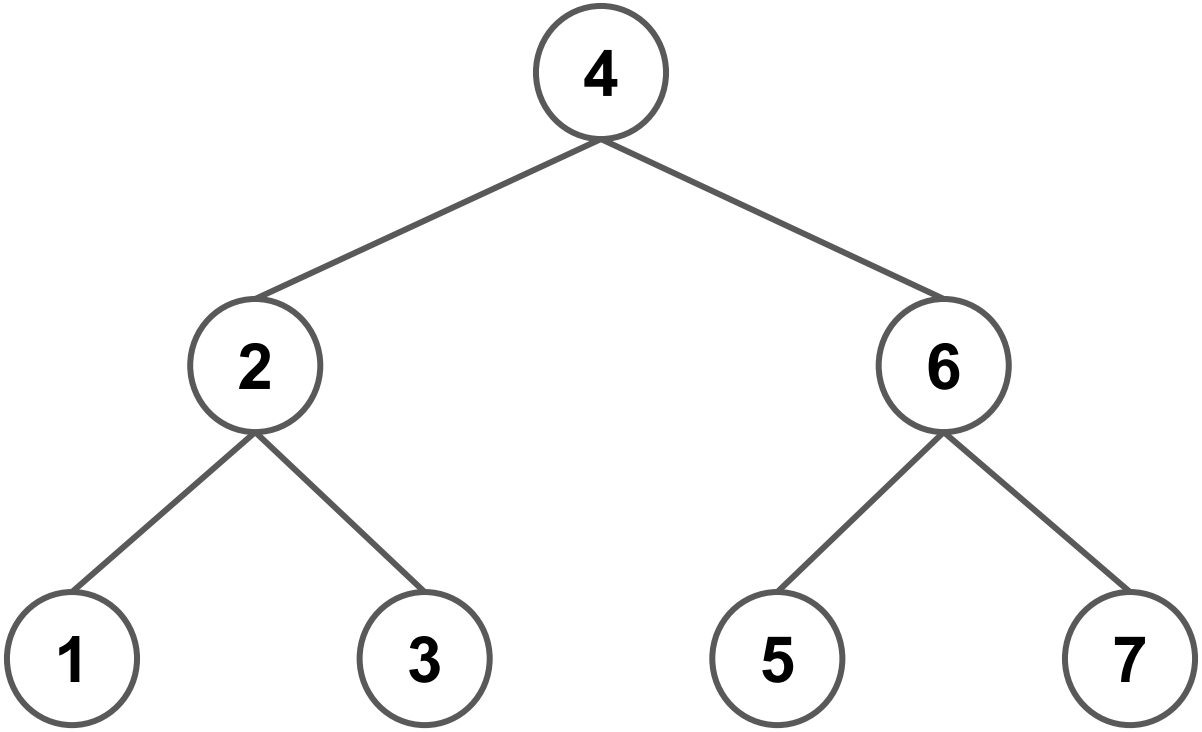
# Agenda

- Binary Trees:
  - Tree Traversal

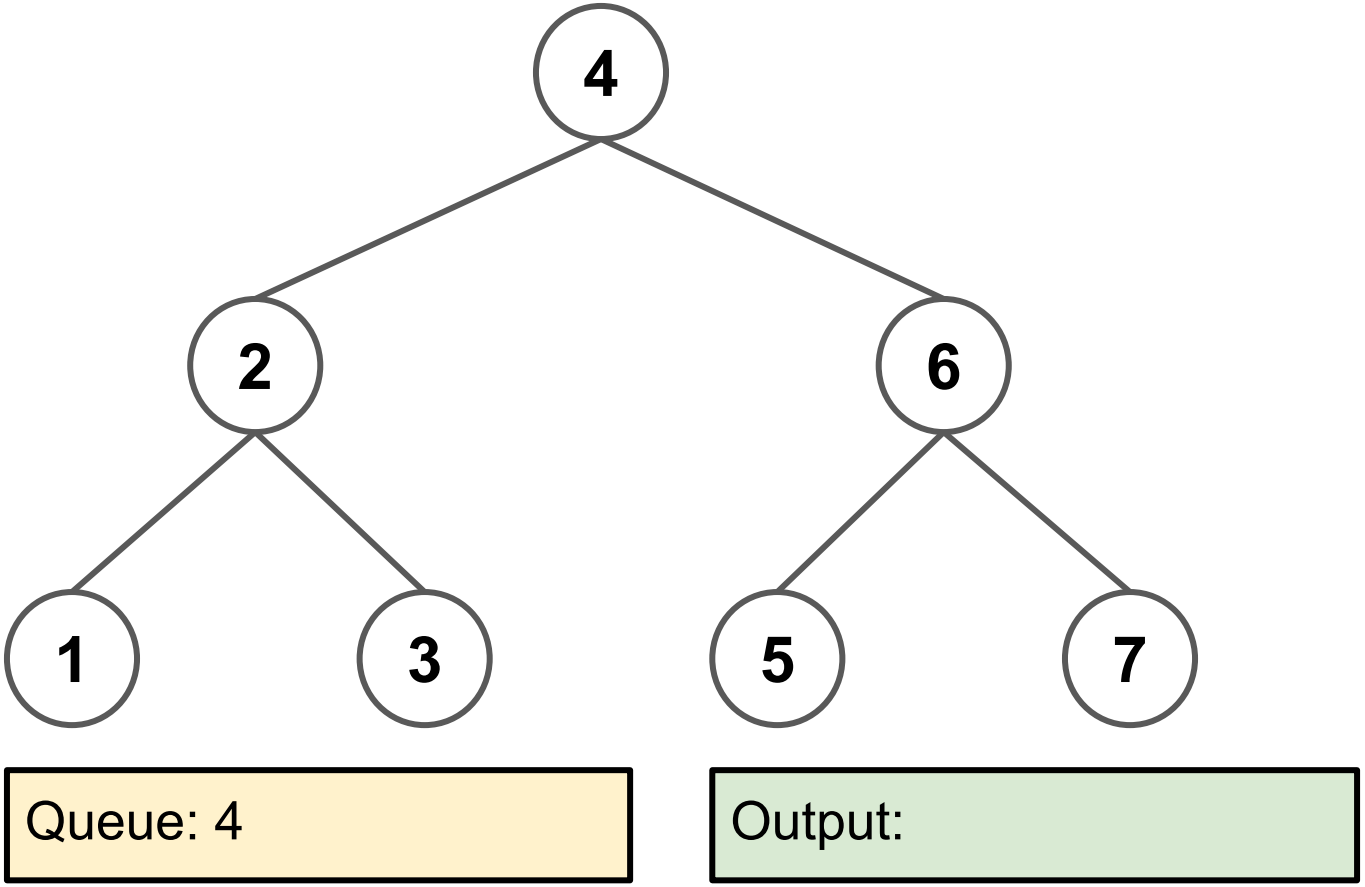
# Tree Traversal

- Because trees are non-linear data structures, there are more than one way to traverse a tree
- They fall into two categories:
  - Breadth First
  - Depth First

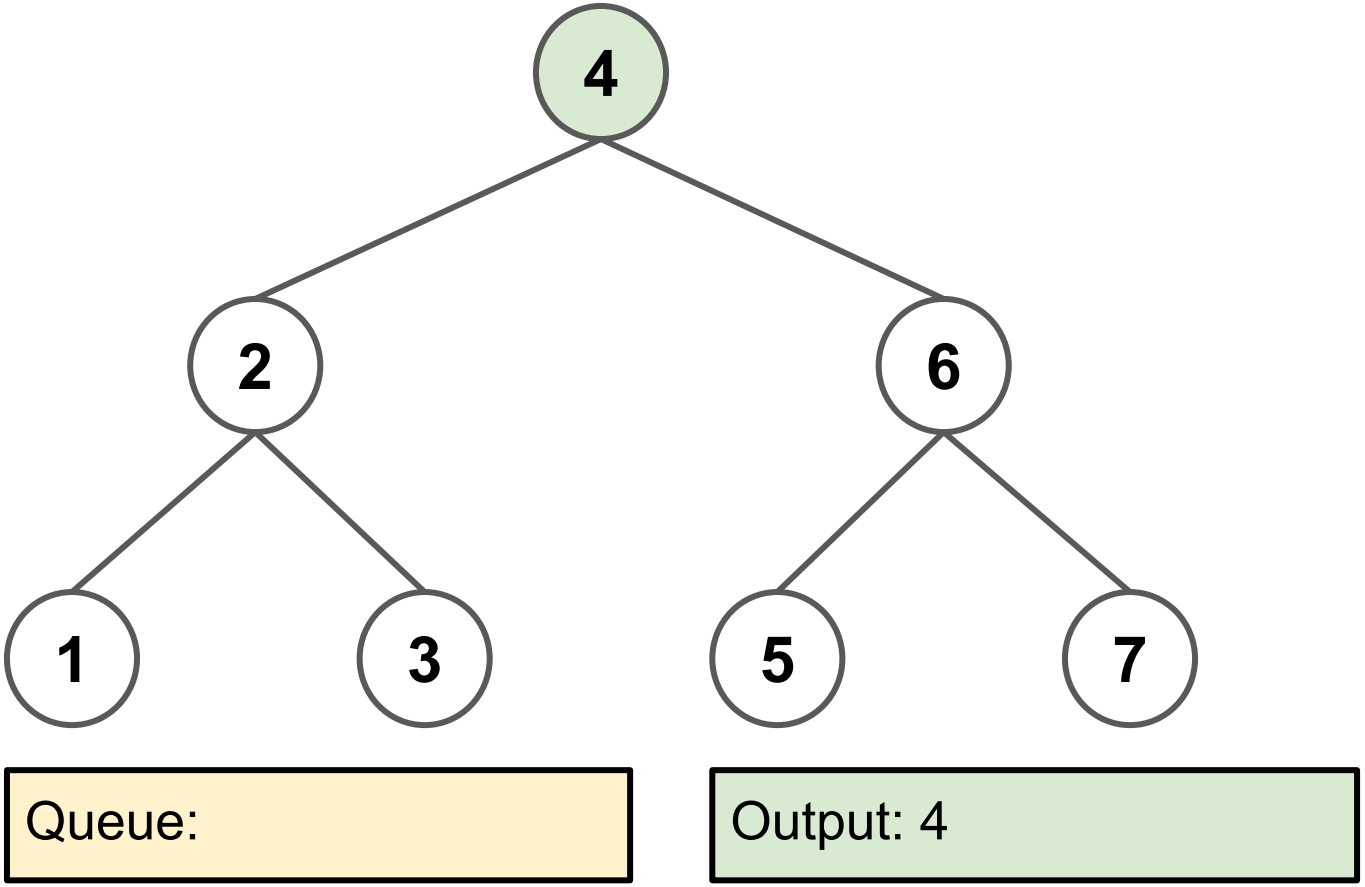
# Tree Traversal: Breadth First



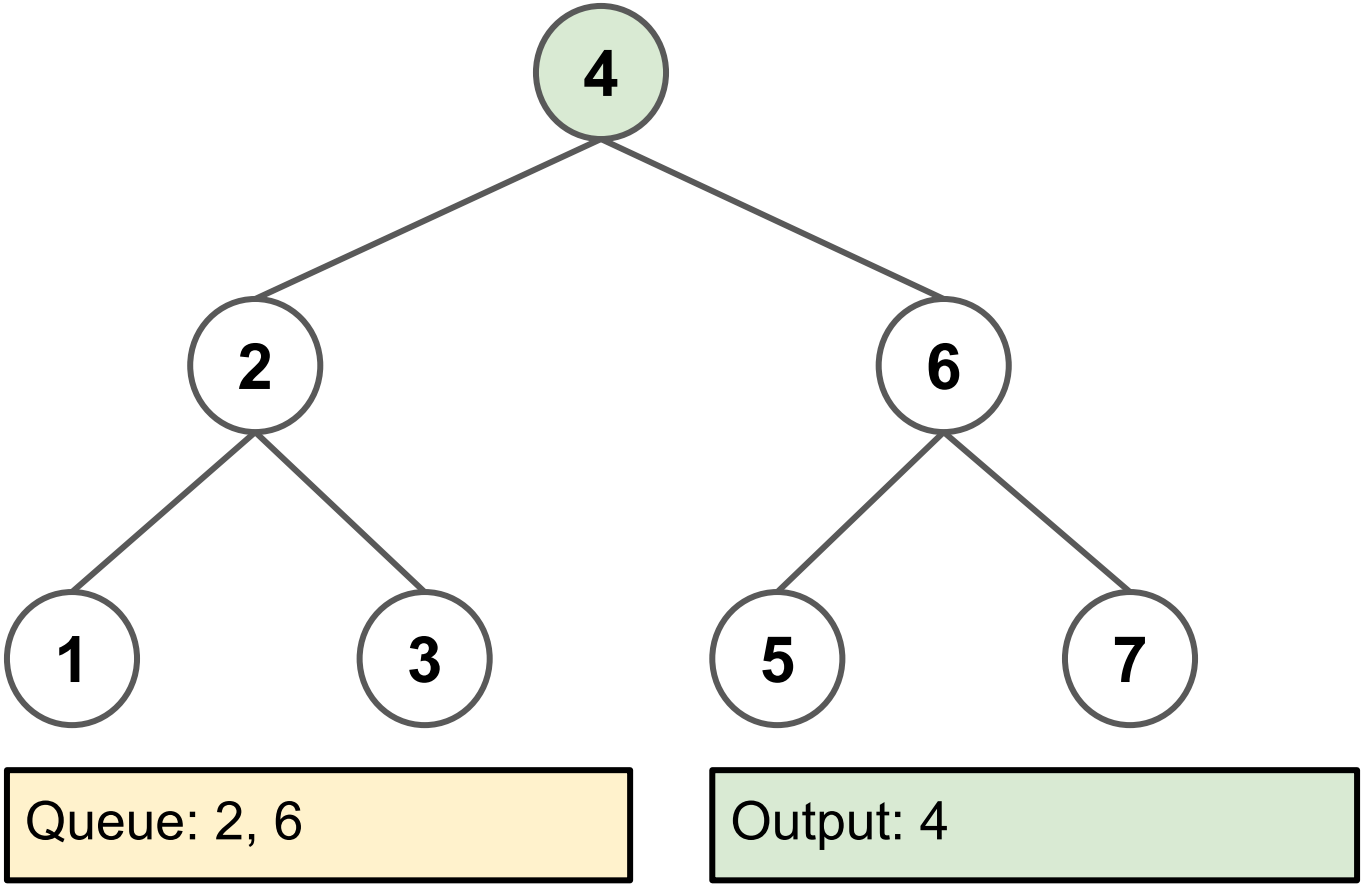
# Tree Traversal: Breadth First



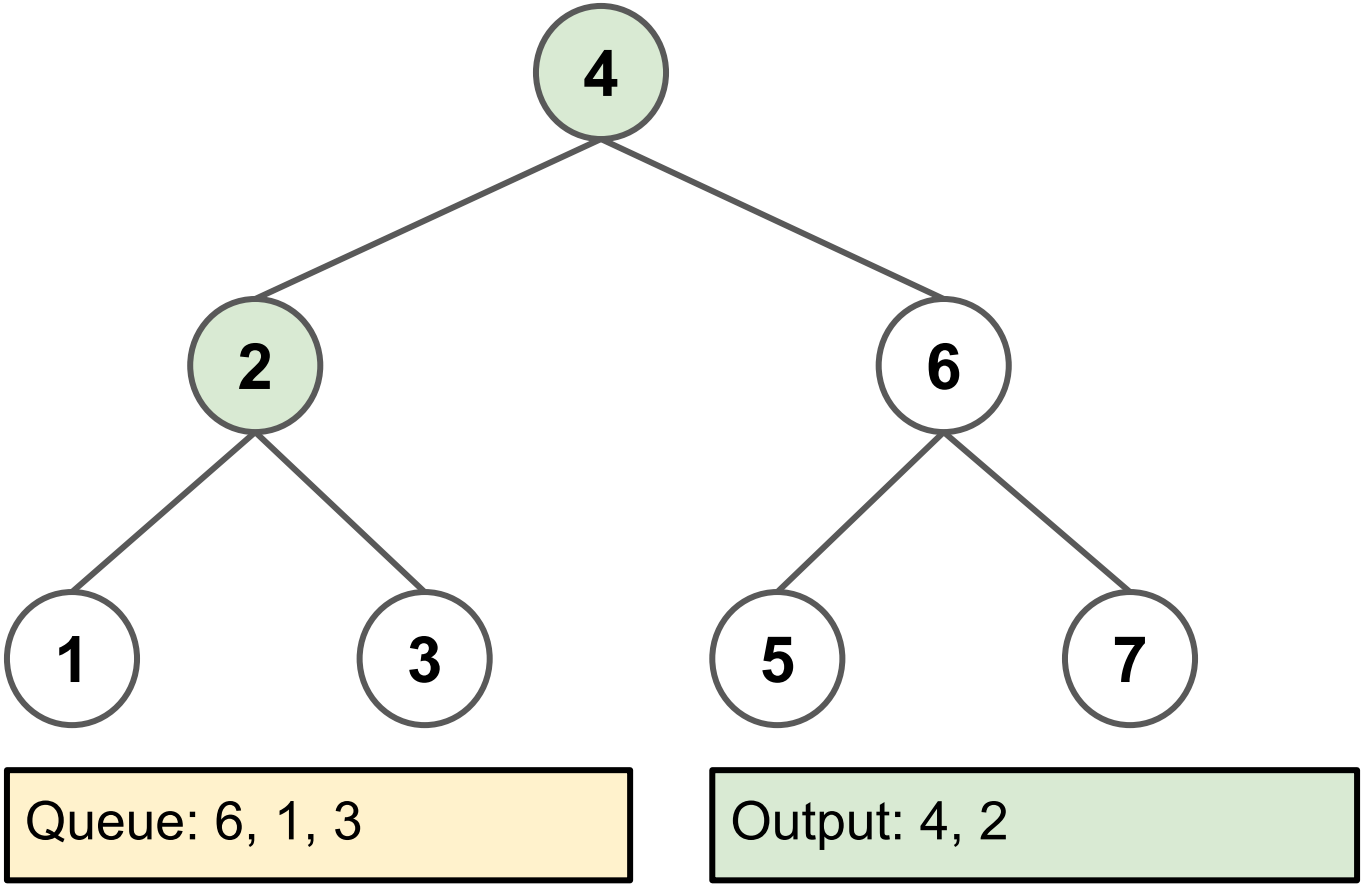
# Tree Traversal: Breadth First



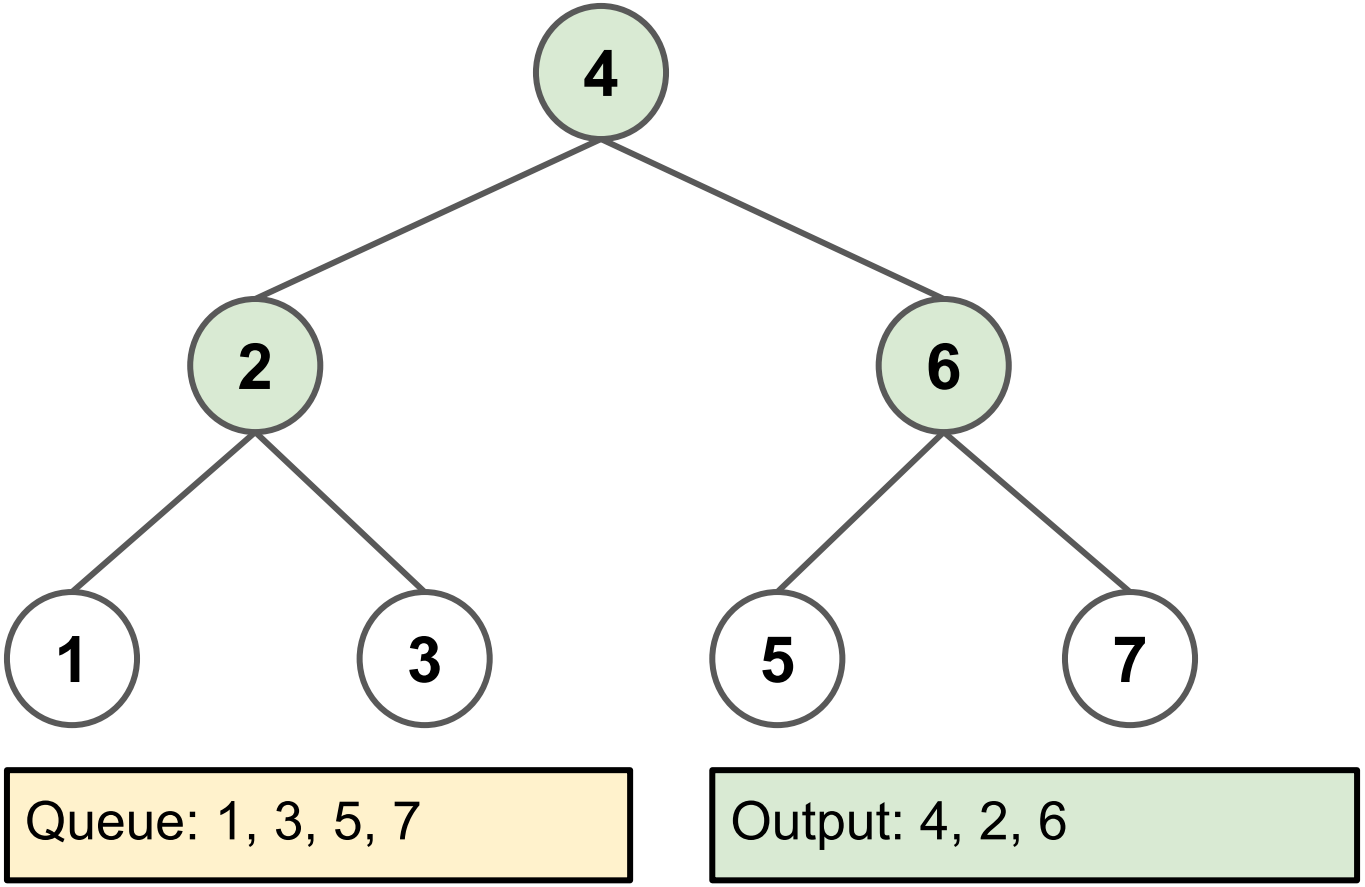
# Tree Traversal: Breadth First



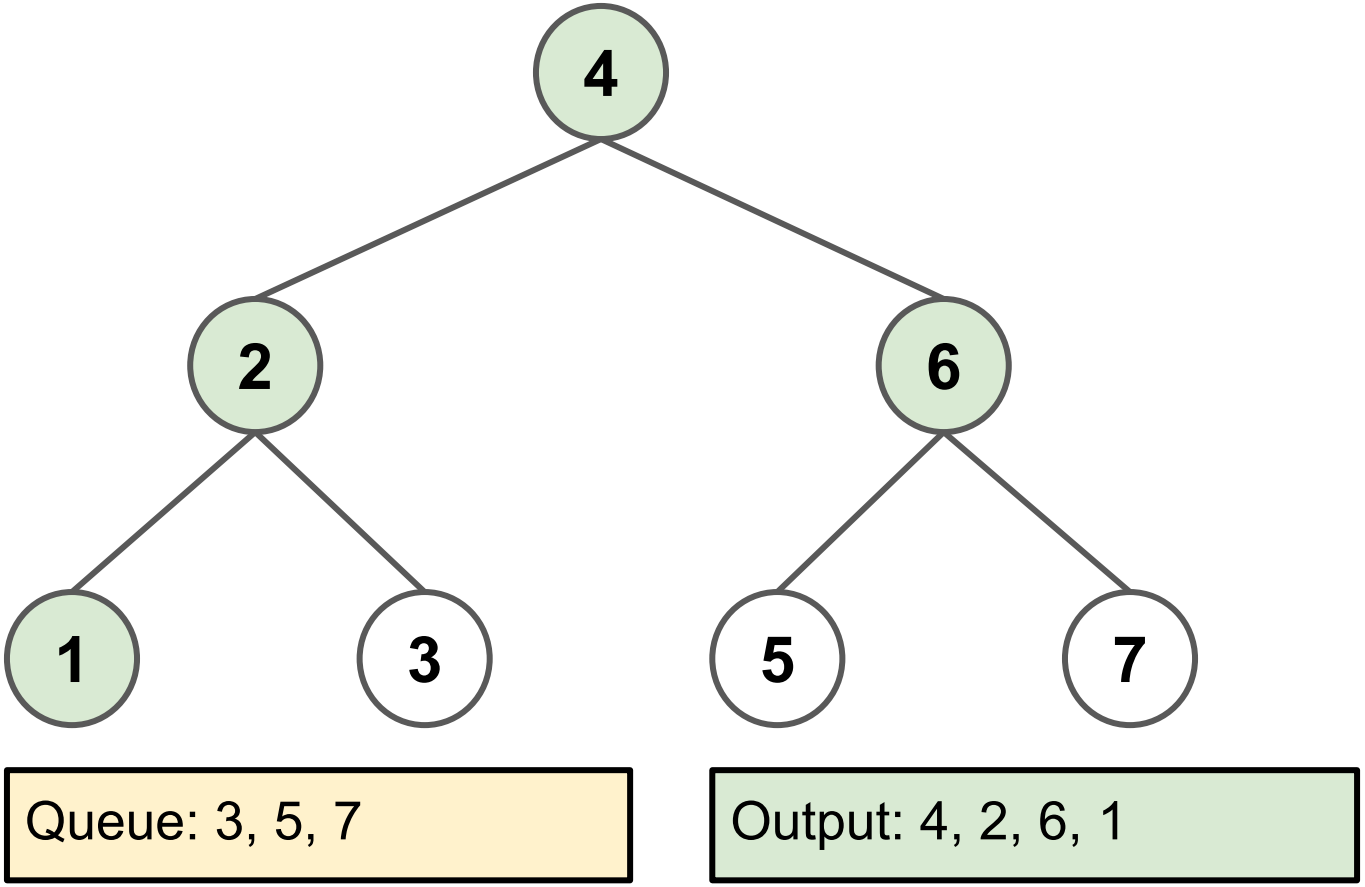
# Tree Traversal: Breadth First



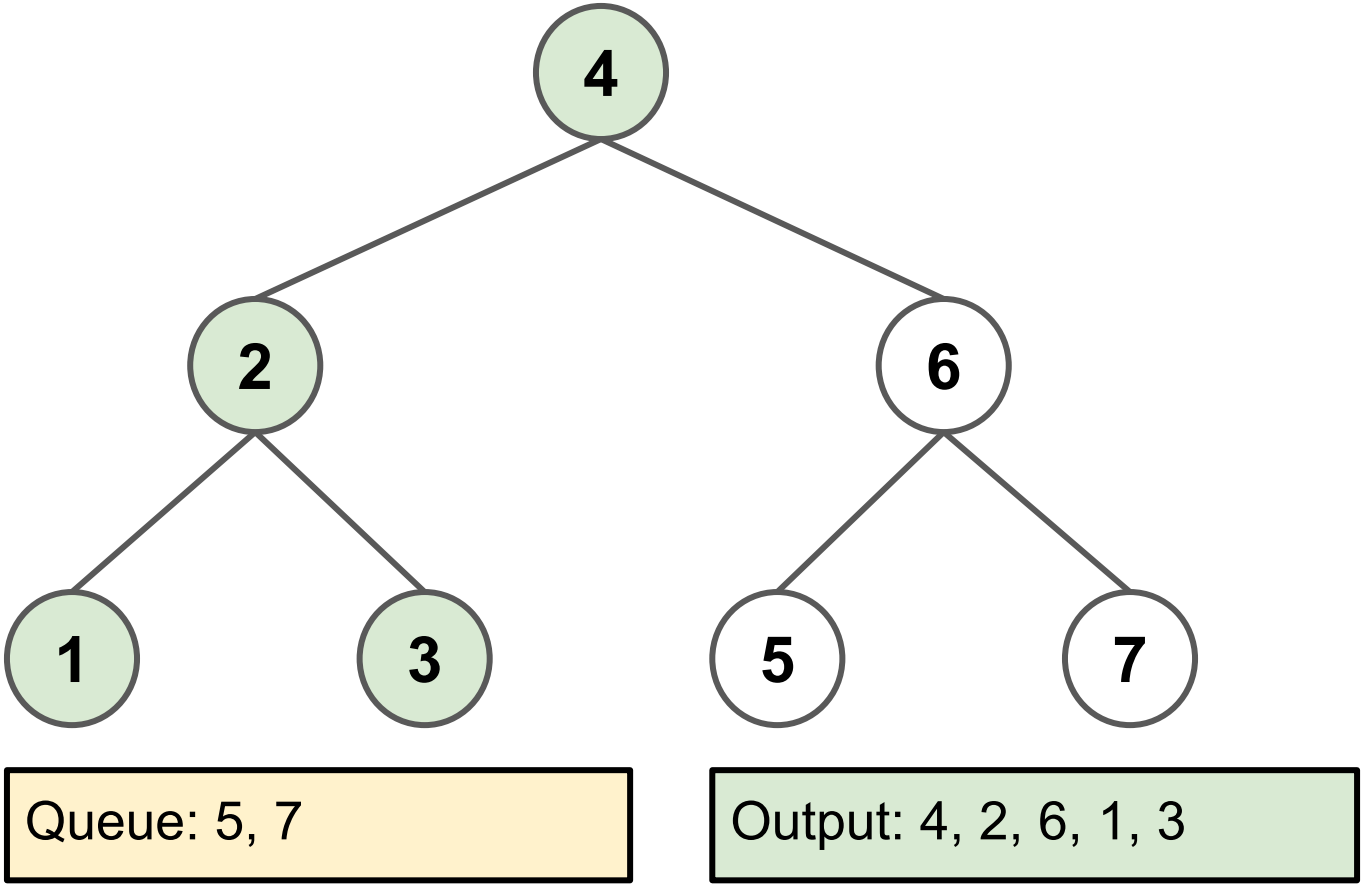
# Tree Traversal: Breadth First



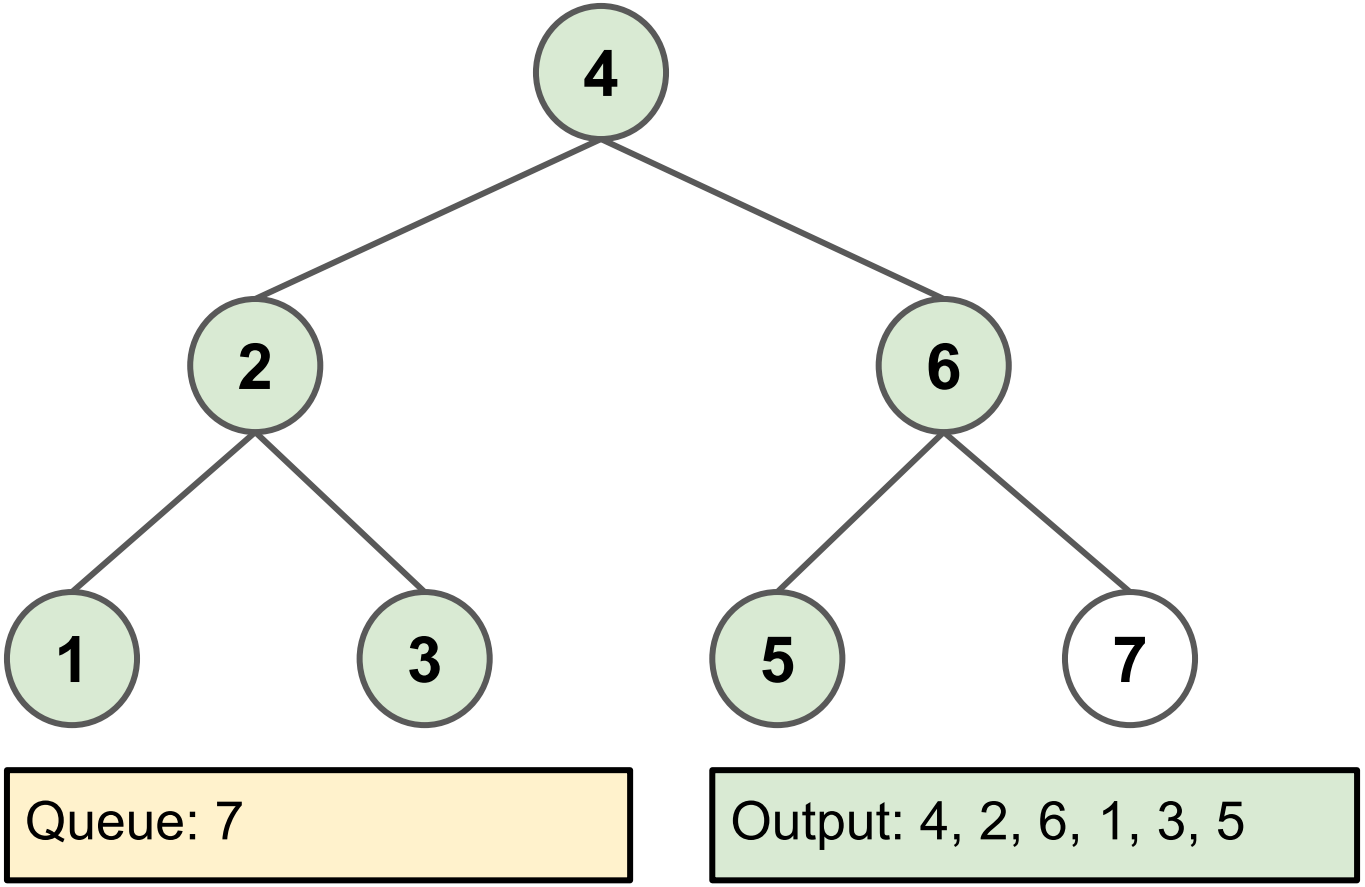
# Tree Traversal: Breadth First



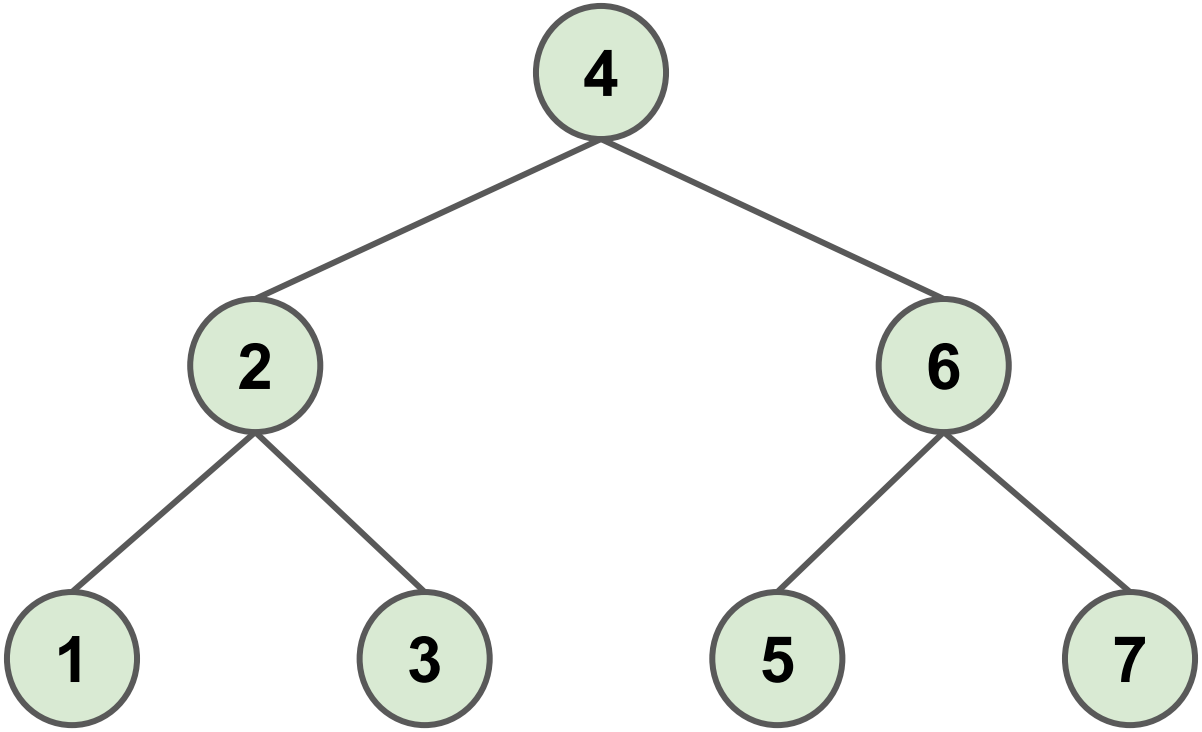
# Tree Traversal: Breadth First



# Tree Traversal: Breadth First



# Tree Traversal: Breadth First



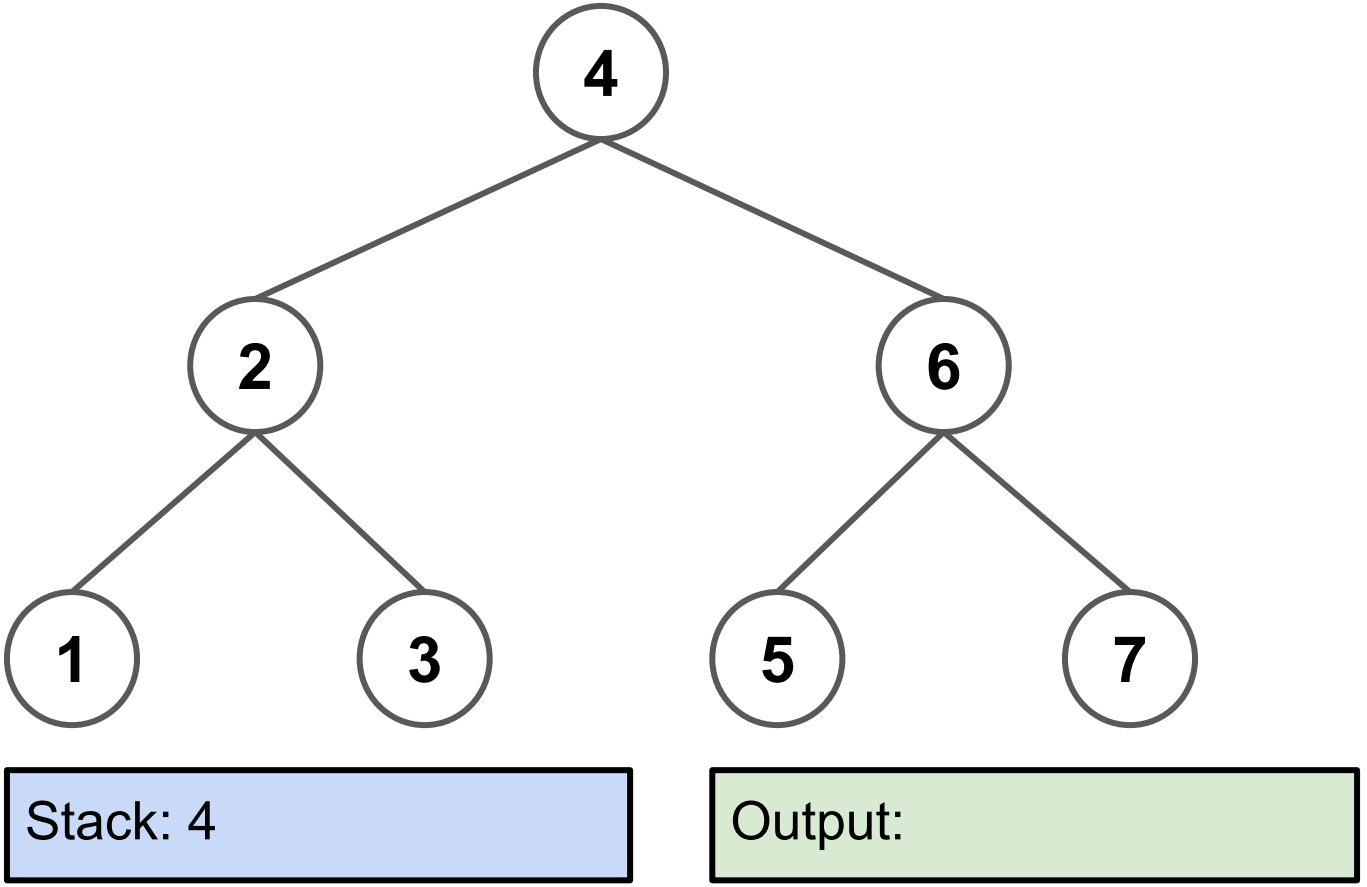

Queue:

Output: 4, 2, 6, 1, 3, 5, 7

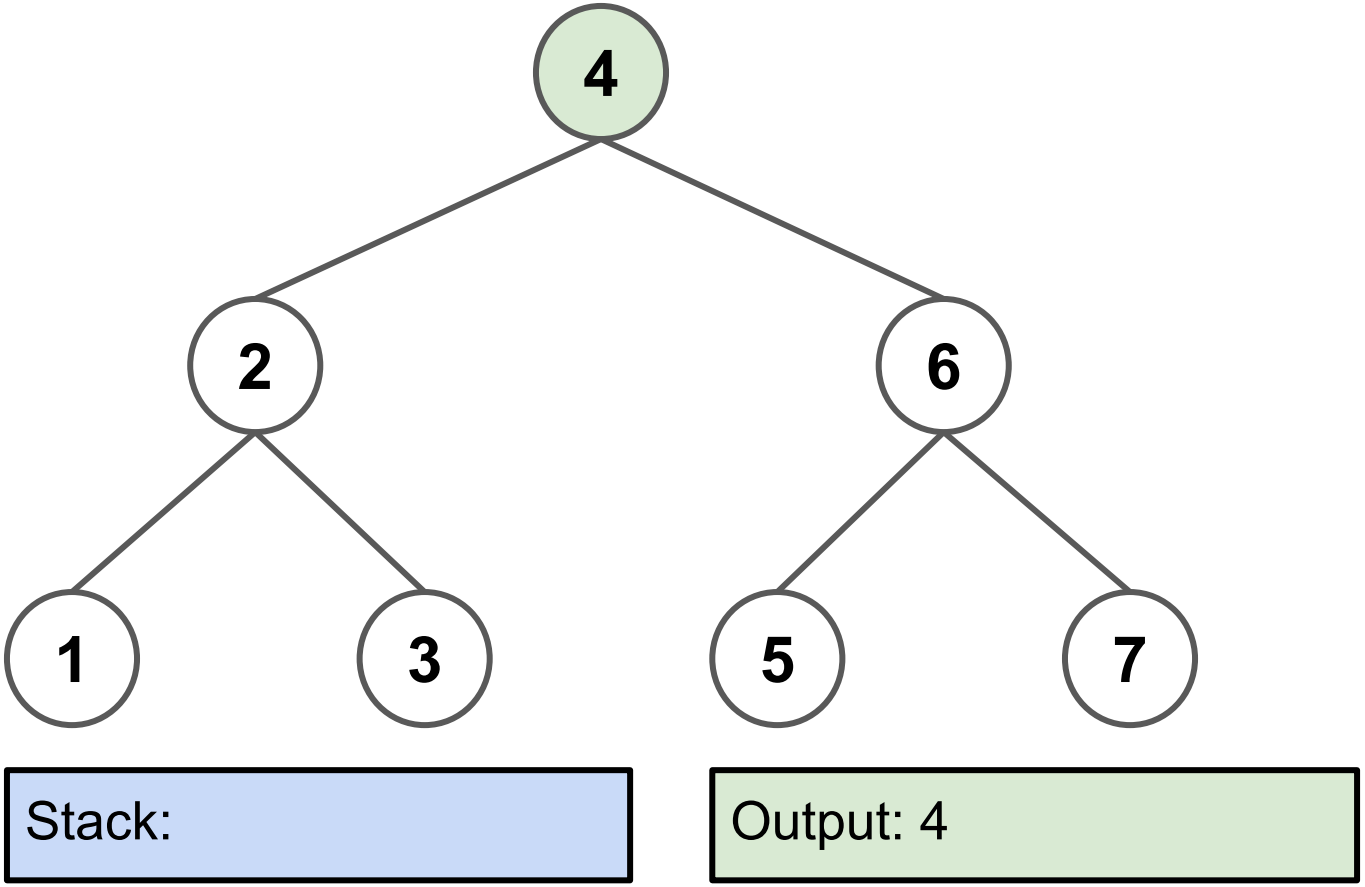
# Tree Traversal: Breadth First

```
38      @Override
39      public String toString() {
40          Queue<BinNode<T>> queue = new Queue<>();
41          BinNode<T> current = root;
42          queue.push(current);
43          String str = "";
44          while (!queue.isEmpty()) {
45              current = queue.pop();
46              str += current.getData() + " ";
47              if (current.getLeft() != null)
48                  queue.push(current.getLeft());
49              if (current.getRight() != null)
50                  queue.push(current.getRight());
51          }
52          return str;
53      }
```

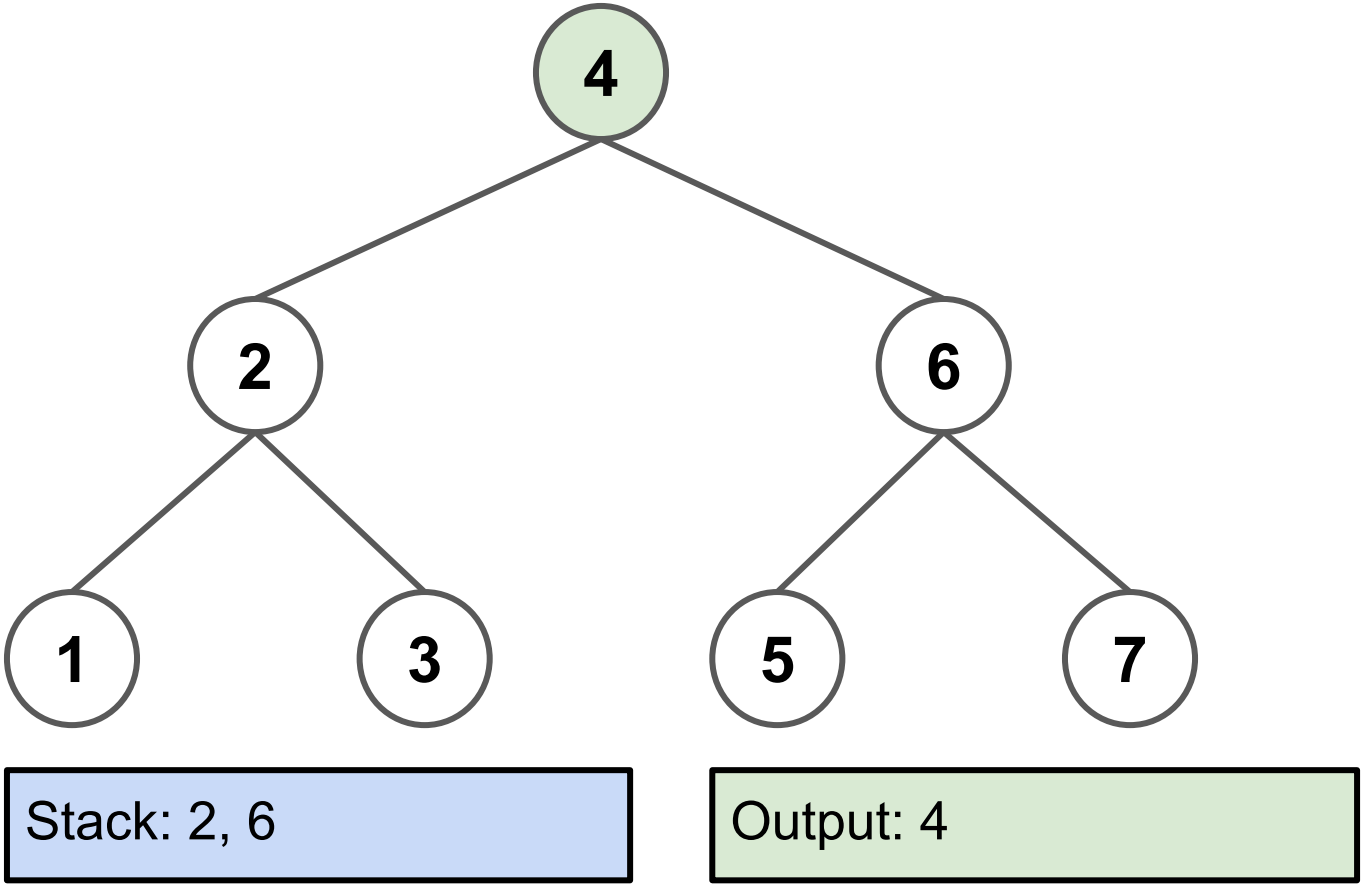
# Tree Traversal: Depth First



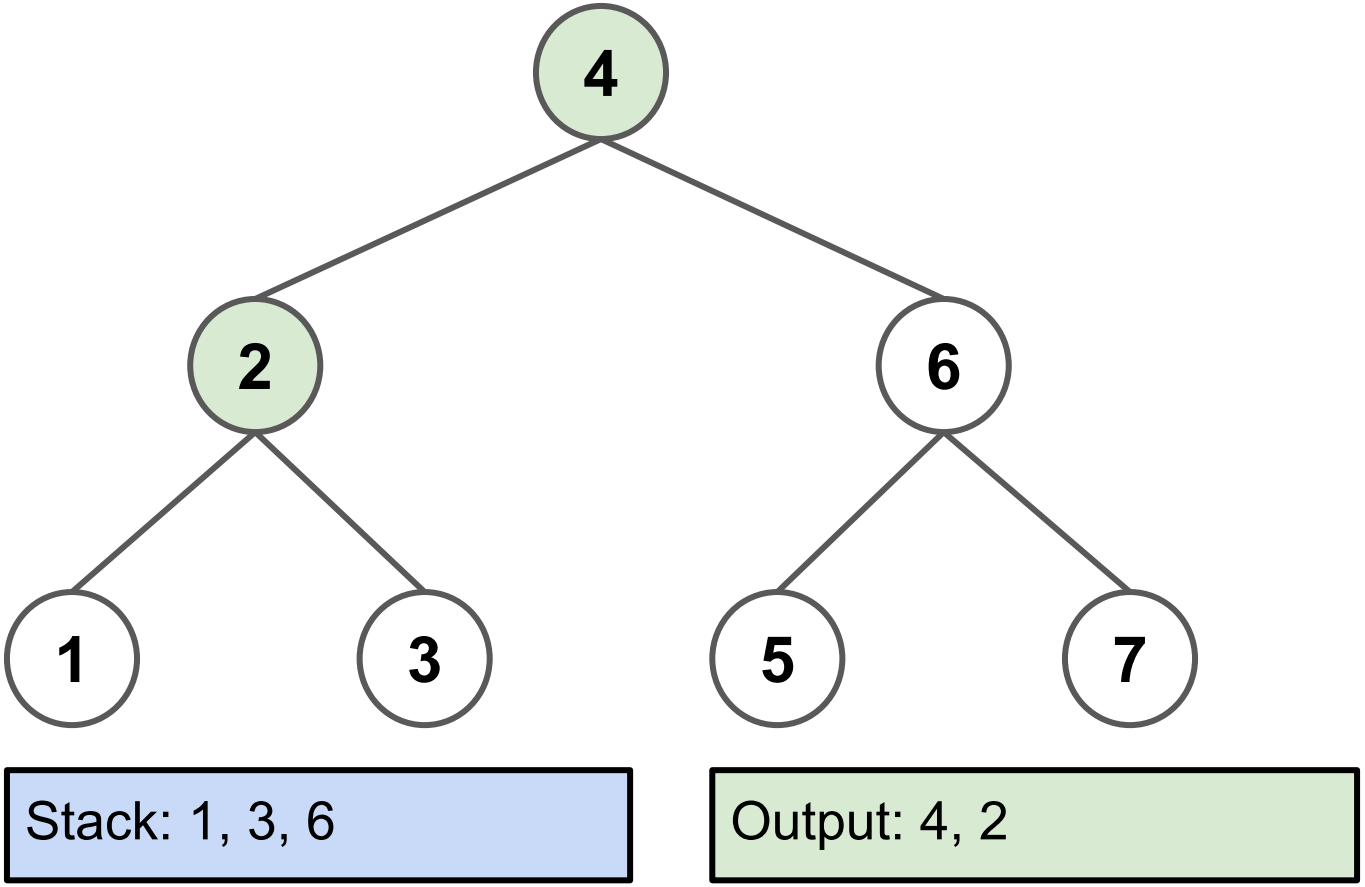
# Tree Traversal: Depth First



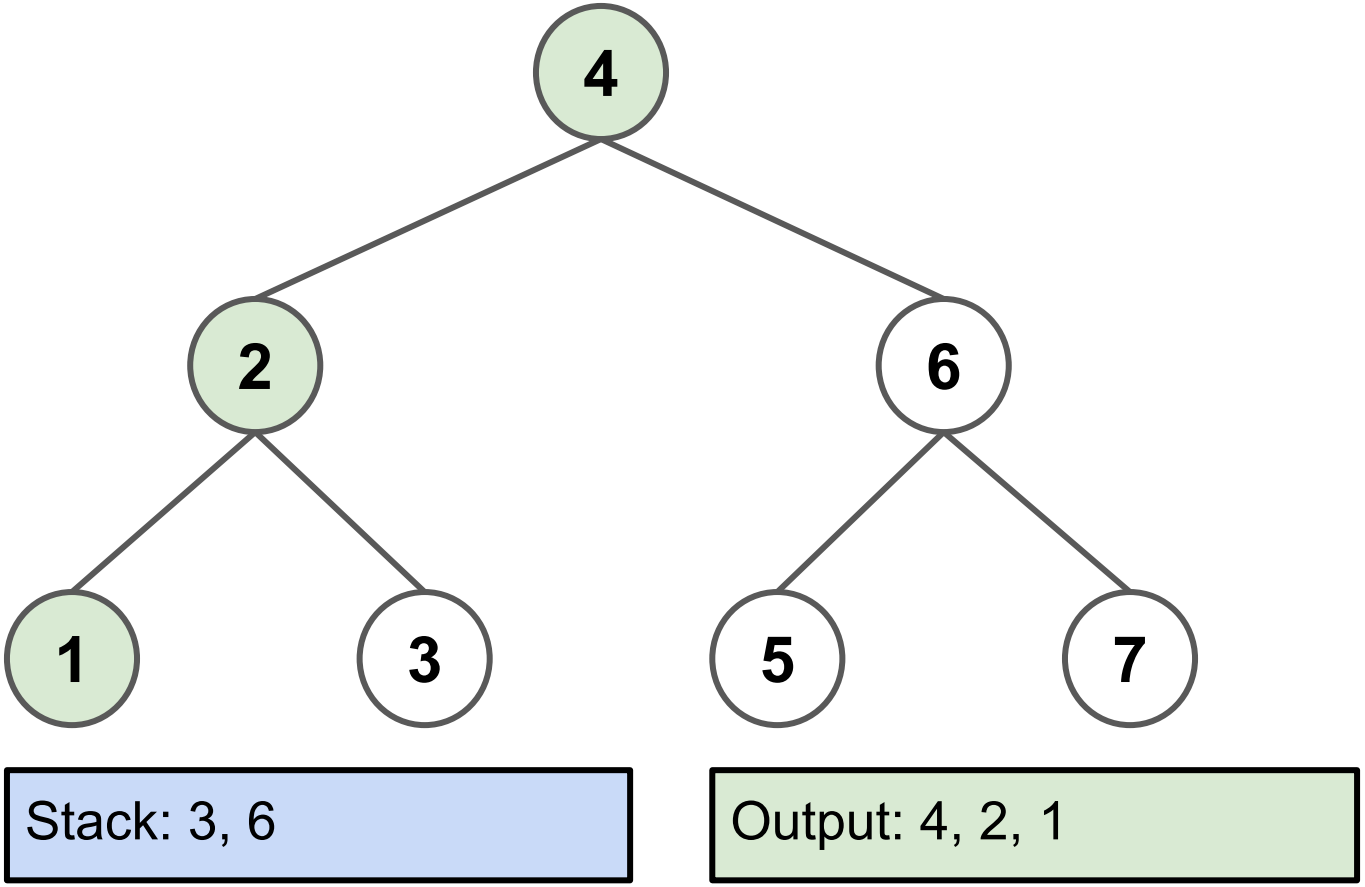
# Tree Traversal: Depth First



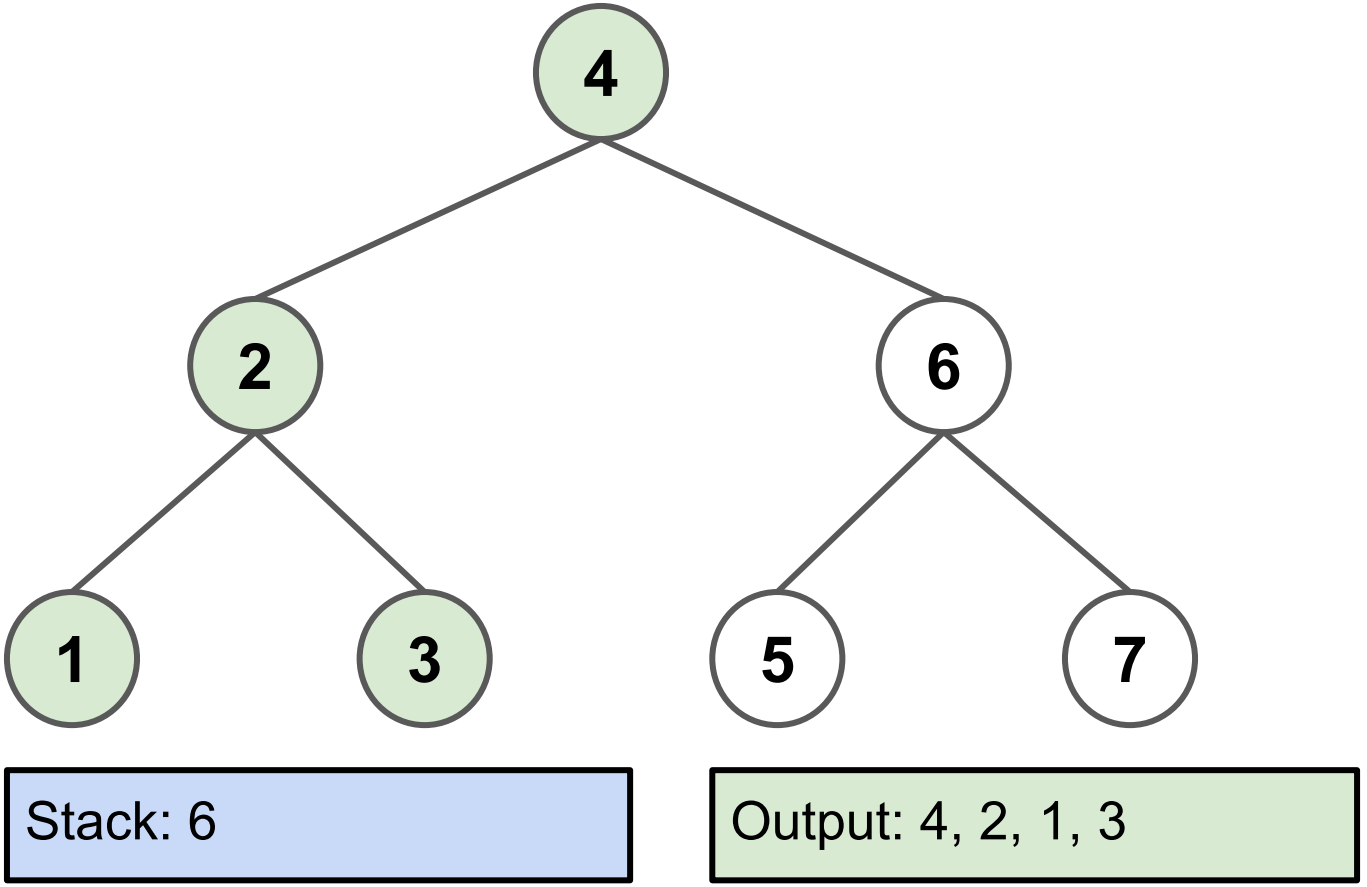
# Tree Traversal: Depth First



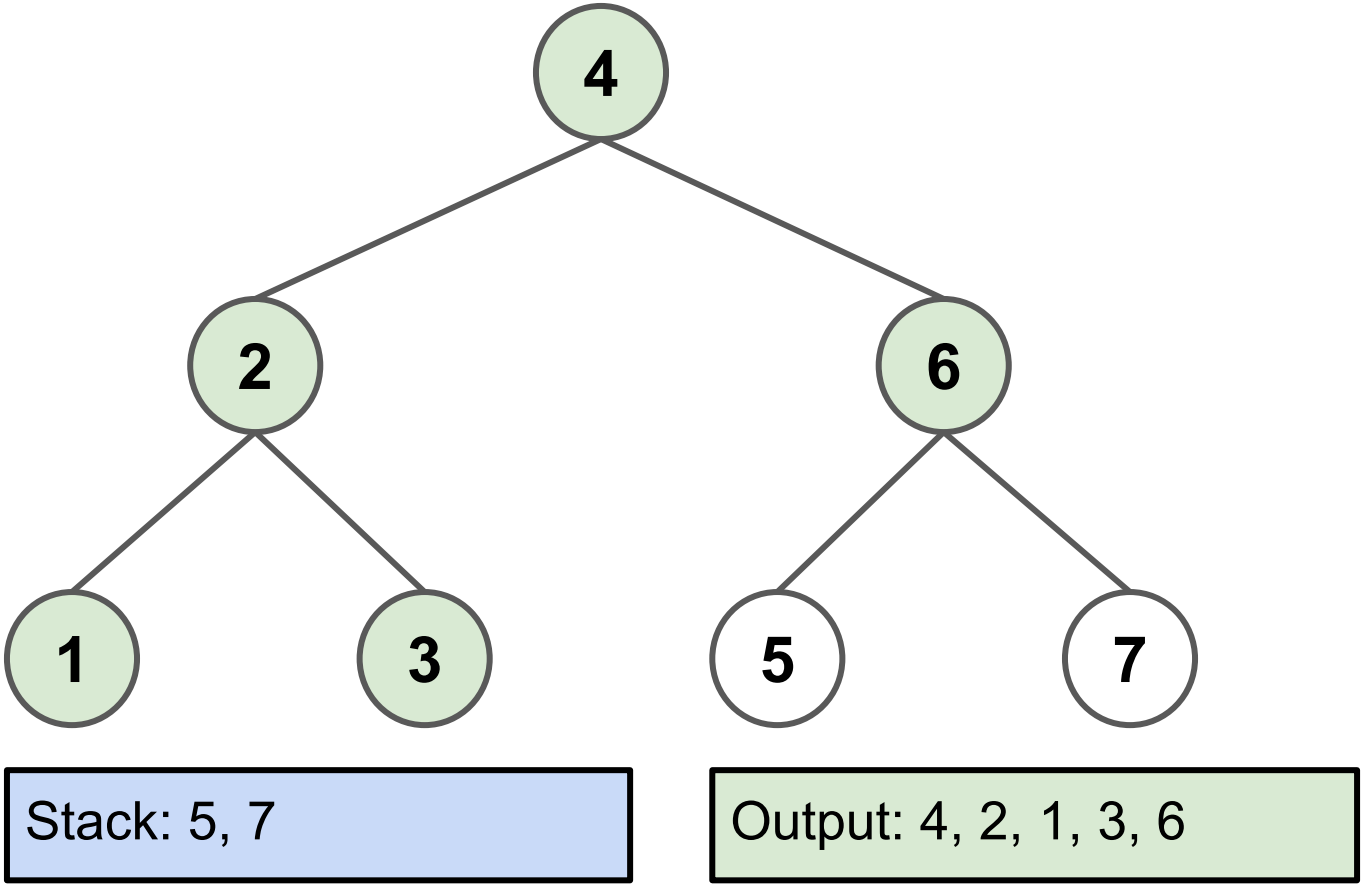
# Tree Traversal: Depth First



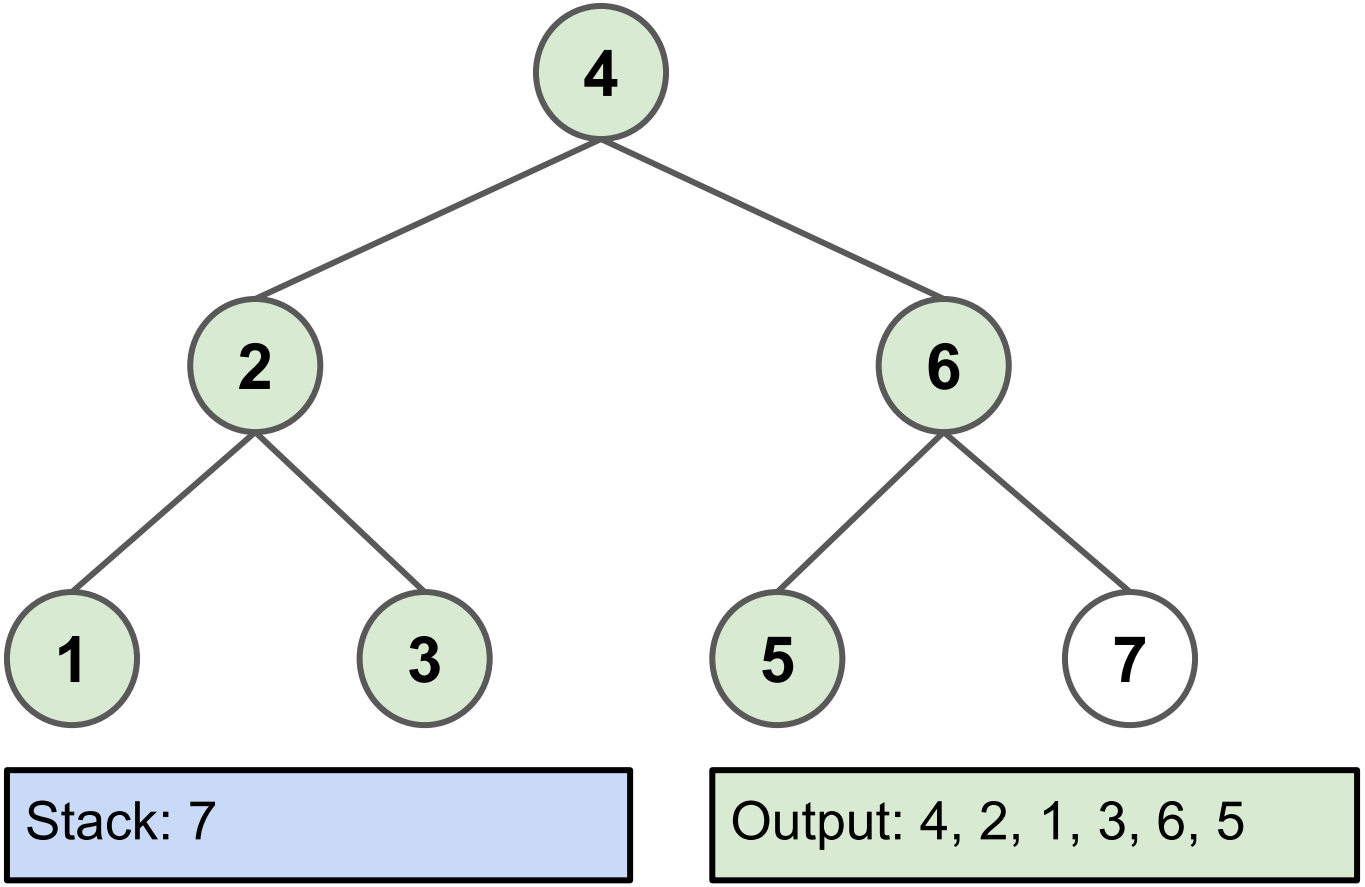
# Tree Traversal: Depth First



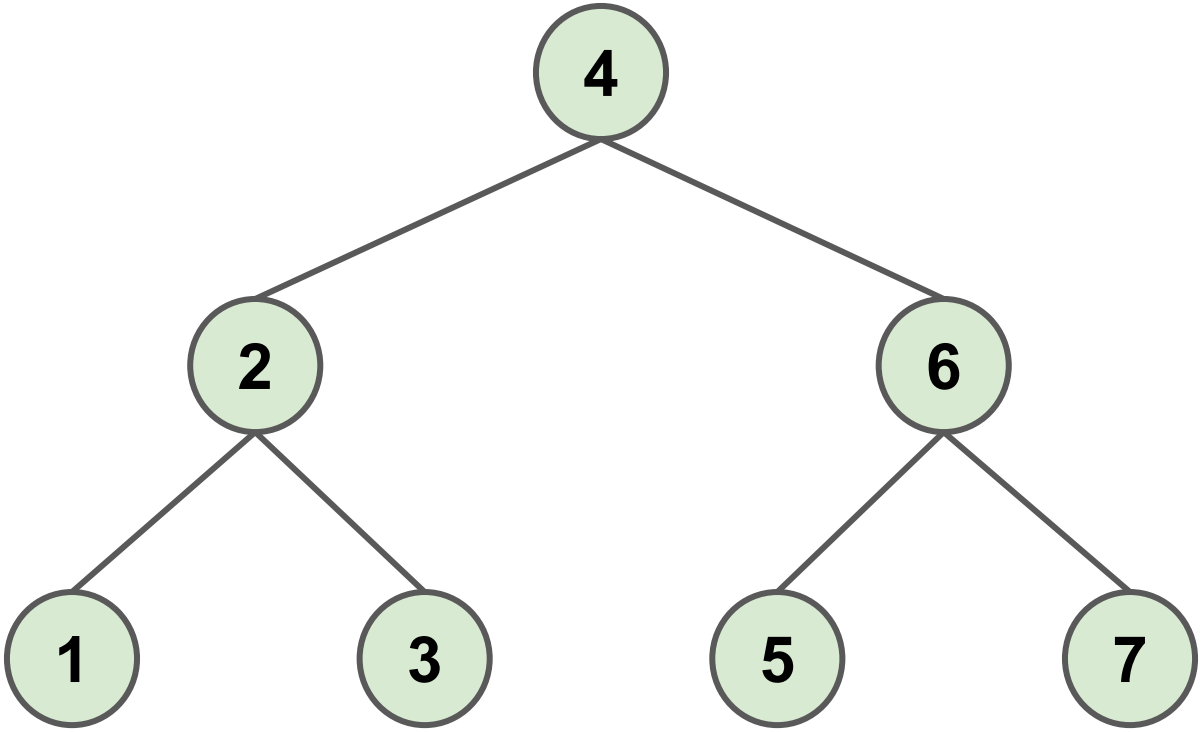
# Tree Traversal: Depth First



# Tree Traversal: Depth First



# Tree Traversal: Depth First




Stack:

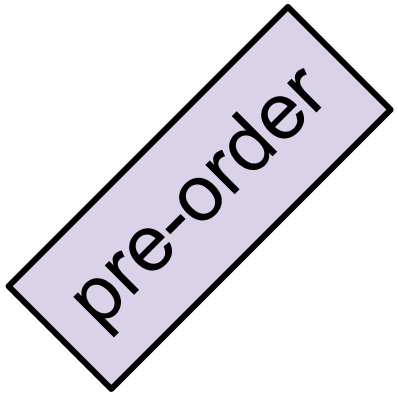
Output: 4, 2, 1, 3, 6, 5, 7

# Tree Traversal: Depth First

```
38      @Override
39      public String toString() {
40          Stack<BinNode<T>> stack = new Stack<>();
41          BinNode<T> current = root;
42          stack.push(current);
43          String str = "";
44          while (!stack.isEmpty()) {
45              current = stack.pop();
46              str += current.getData() + " ";
47              if (current.getRight() != null) // right node pushed 1st
48                  stack.push(current.getRight());
49              if (current.getLeft() != null) // left node pushed 2nd
50                  stack.push(current.getLeft());
51
52          }
53          return str;
54      }
```

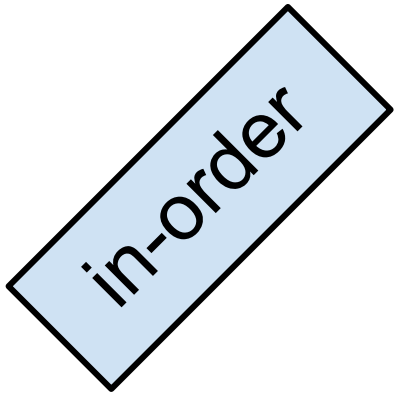
# Tree Traversal: Depth First

```
38      public String preOrder(final BinNode<T> current) {
39          if (current != null)
40              return  current.getData() + " "      +
41                      preOrder(current.getLeft()) +
42                      preOrder(current.getRight());
43          return "";
44      }
45
46      @Override
47      public String toString() {
48          return preOrder(root);
49      }
```



# Tree Traversal: Depth First

```
46      public String inOrder(final BinNode<T> current) {
47          if (current != null)
48              return  inOrder(current.getLeft())  +
49                      current.getData() + " "     +
50                      inOrder(current.getRight());
51          return "";
52      }
53
54      @Override
55      public String toString() {
56          return inOrder(root);
57      }
```

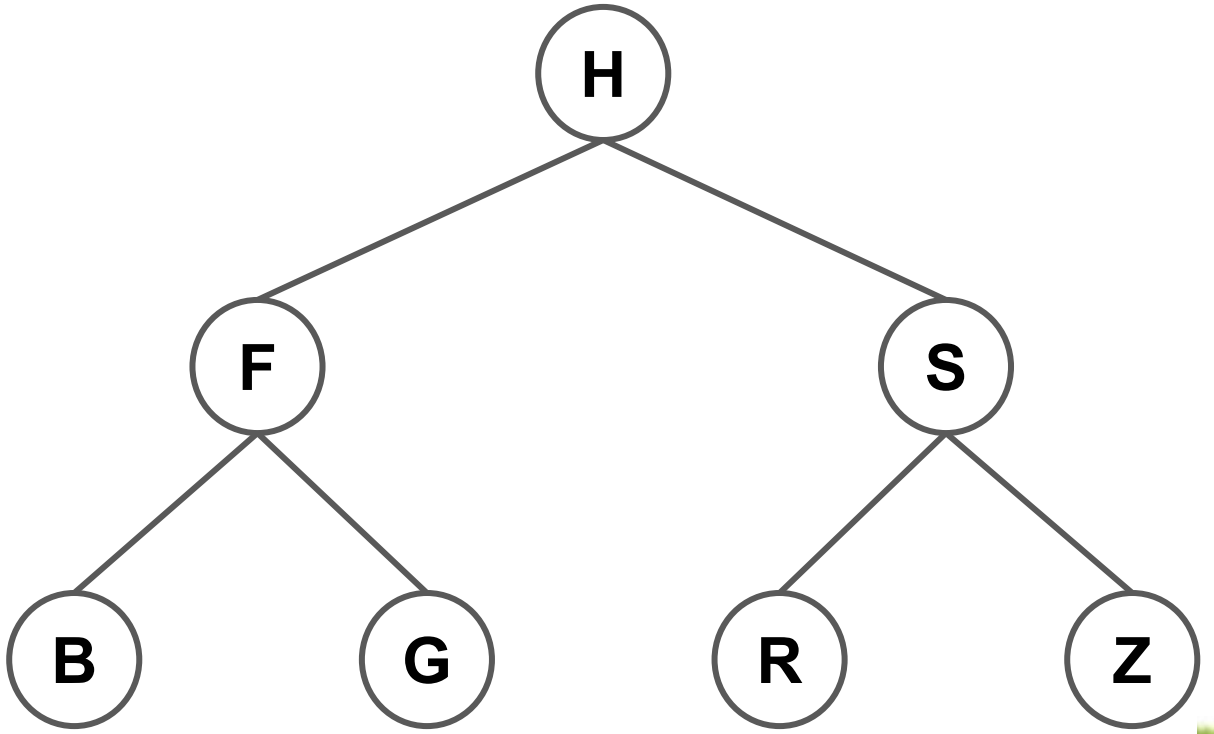


# Tree Traversal: Depth First

```
54      public String postOrder(final BinNode<T> current) {
55          if (current != null)
56              return  postOrder(current.getLeft())  +
57                      postOrder(current.getRight()) +
58                      current.getData() + " ";
59          return "";
60      }
61
62      @Override
63      public String toString() {
64          return postOrder(root);
65      }
```

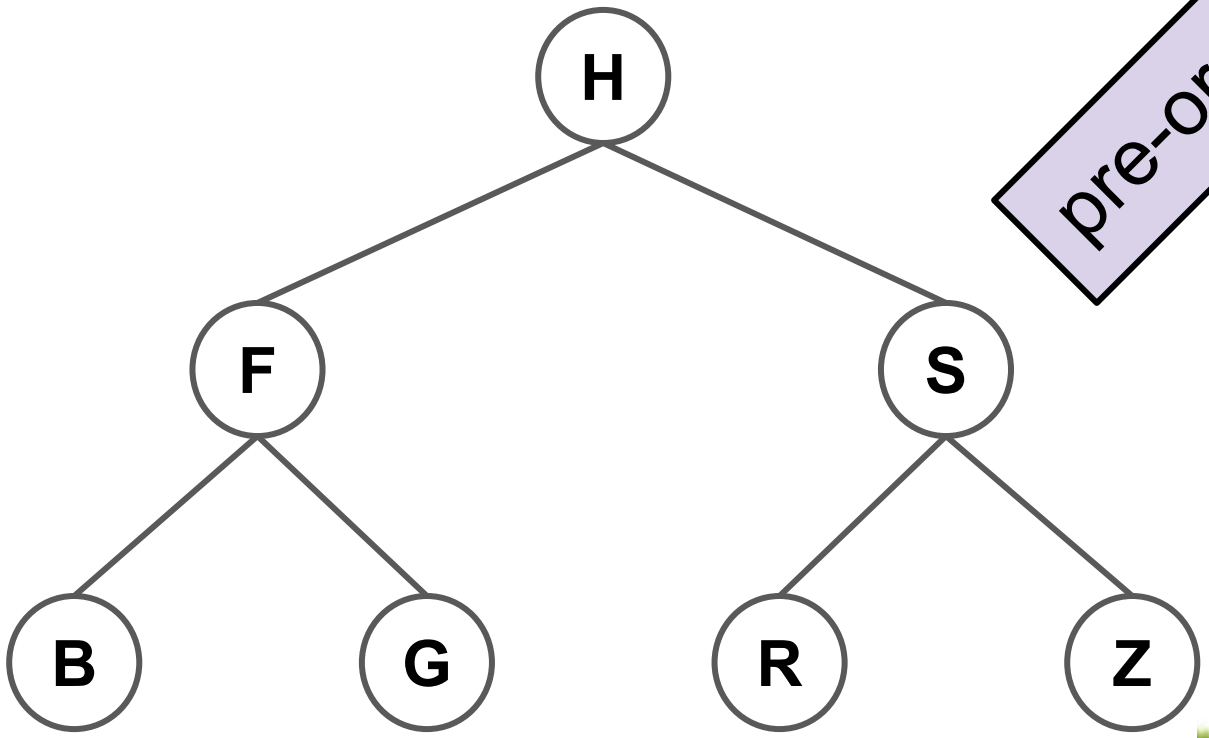
# Tree Traversal: Depth First

- Practice:




# Tree Traversal: Depth First

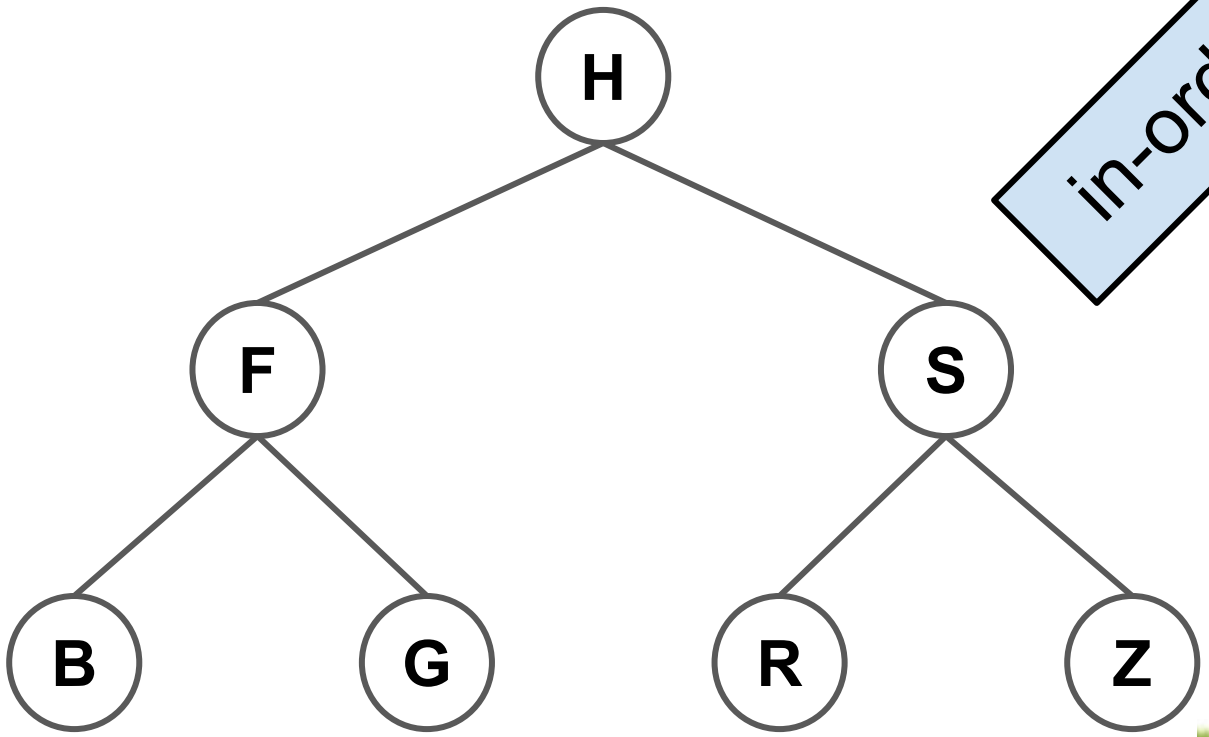
- Practice:




pre-order

# Tree Traversal: Depth First

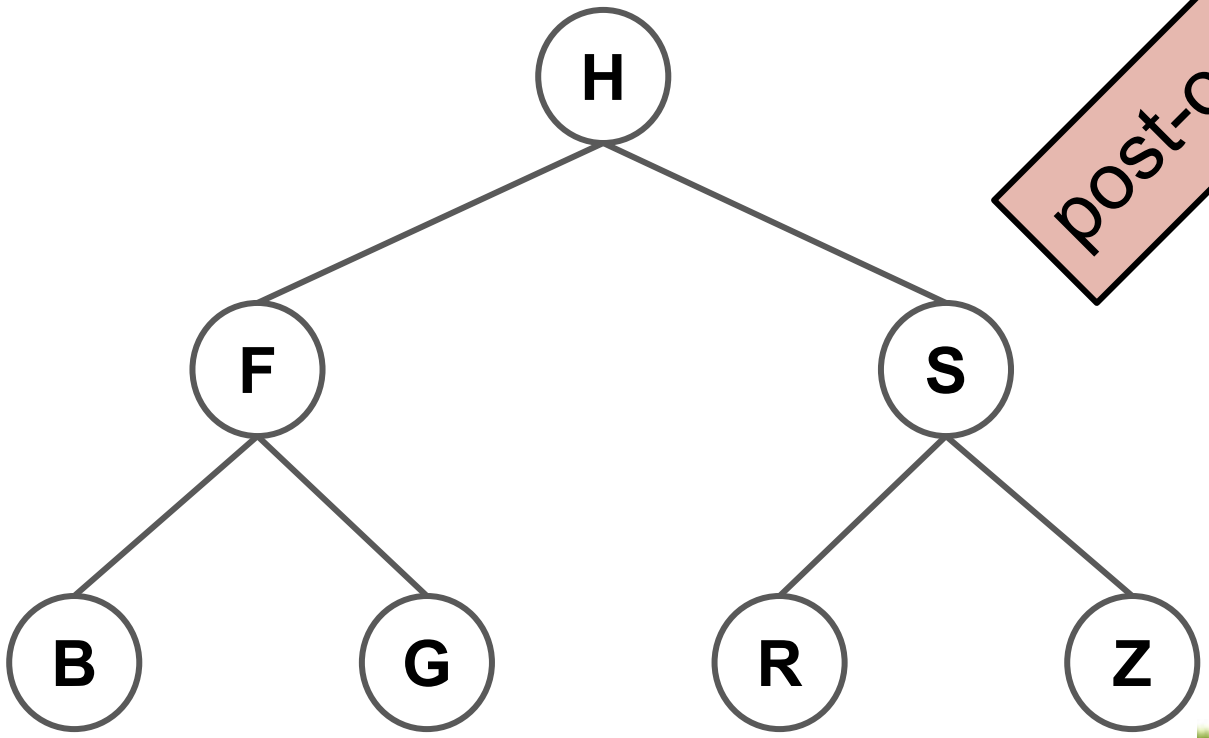
- Practice:




in-order

# Tree Traversal: Depth First

- Practice:




post-order