Binary Search Tree
BFS

by Alex Joslin

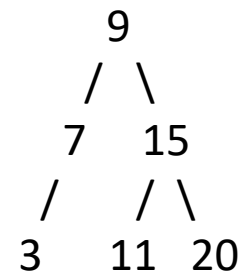
BFS

Graph

```
def bfs(graph, start):
    queue = [start]
    visited = []
    visited.append(start)
    while queue:
        top = queue.pop(0)
        for adj in graph[top]:
            visited.append(adj)
            queue.append(adj)
    print(visited)


d = {9: [7, 15],
     7: [3],
     15: [11, 20],
     3: [],
     11: [],
     20: []}

bfs(d, 9)
```

Binary Search Tree

```
def bfs(root):
    queue = [root]
    visited = []
    visited.append(root.data)
    while queue:
        top = queue.pop(0)
        if top.left:
            queue.append(top.left)
            visited.append(top.left.data)
        if top.right:
            queue.append(top.right)
            visited.append(top.right.data)
    print(visited)


tree = Node(9)
tree.left = Node(7)
tree.right = Node(15)
tree.left.left = Node(3)
tree.right.left = Node(11)
tree.right.right = Node(20)
bfs(tree)
```



[9, 7, 15, 3, 11, 20]

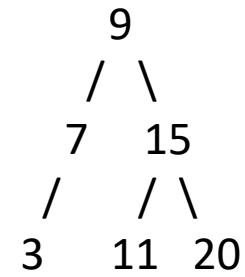
BFS Level Order Traversal

Graph

```
def bfs_lot(graph, start):
    visited = []
    cur_level = [start]
    while cur_level:
        visited.append(cur_level)
        next_level = []
        for node in cur_level:
            for adj in graph[node]:
                next_level.append(adj)
        cur_level = next_level
    print(visited)


d = {9: [7, 15],
     7: [3],
     15: [11, 20],
     3: [],
     11: [],
     20: []}

bfs_lot(d, 9)
```

Binary Search Tree

```
def bfs_lot(root):
    visited = []
    cur_level = [root]
    while cur_level:
        visited.append([n.data for n in cur_level])
        next_level = []
        for node in cur_level:
            if node.left:
                next_level.append(node.left)
            if node.right:
                next_level.append(node.right)
        cur_level = next_level
    print(visited)


tree = Node(9)
tree.left = Node(7)
tree.right = Node(15)
tree.left.left = Node(3)
tree.right.left = Node(11)
tree.right.right = Node(20)
bfs_lot(tree)
```



[[9], [7, 15], [3, 11, 20]]