# [ Presentation ]

2020095178 Yoon Seon Choi (최윤선)

# Contents

Part 1

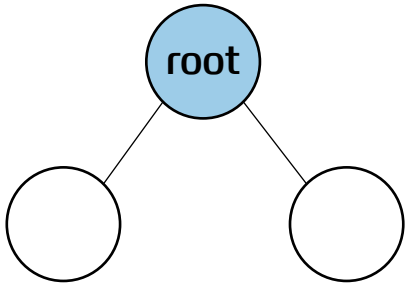
# Problem Statement

## Problem: Find maximum path sum

Given a binary tree, find maximum path sum. For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along with the parent-child connections.

### <Sequences from the root node of the subtree>

1. The sequence from the only current root node



Part 1

# Problem Statement

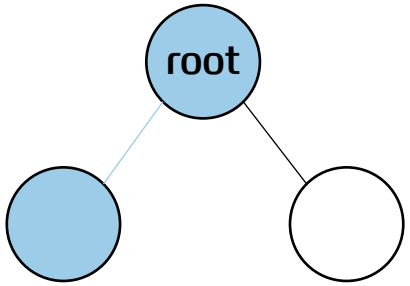
## Problem: Find maximum path sum

Given a binary tree, find maximum path sum. For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along with the parent-child connections.

### <Sequences from the root node of the subtree>

2. The sequence from the current root node and the left node of the root node

   (Starting node is the left node)



Part 1

# Problem Statement

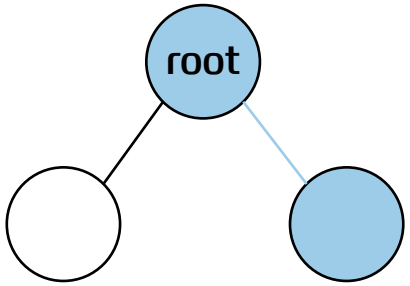
## Problem: Find maximum path sum

Given a binary tree, find maximum path sum. For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along with the parent-child connections.

### <Sequences from the root node of the subtree>

3. The sequence from the current root node and the right node of the root node

   (Starting node is the right node)



Part 1

# Problem Statement

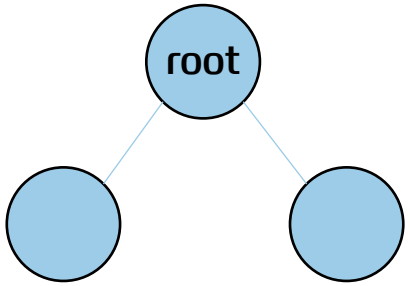
## Problem: Find maximum path sum

Given a binary tree, find maximum path sum. For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along with the parent-child connections.

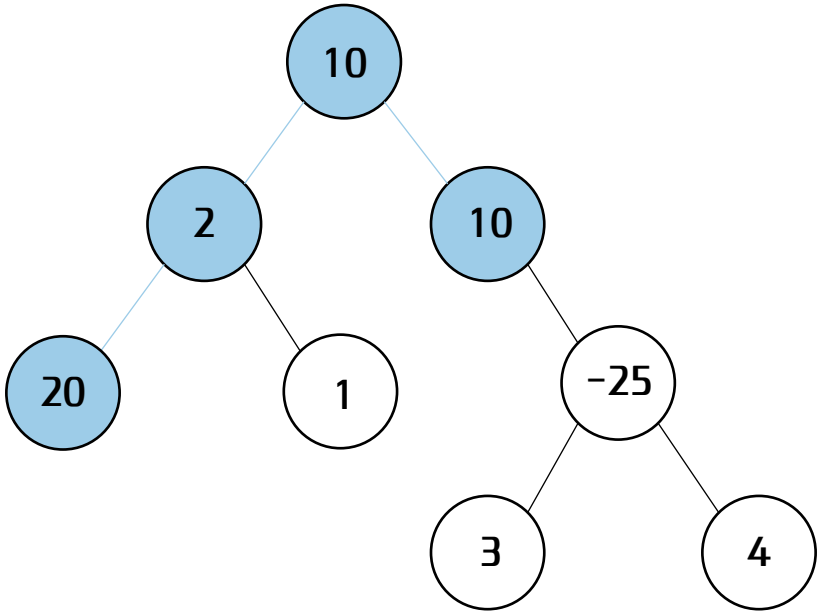
### <Sequences from the root node of the subtree>

4. The sequence from the current root node, the left and the right node of the root node

   (From the left node to the right node)



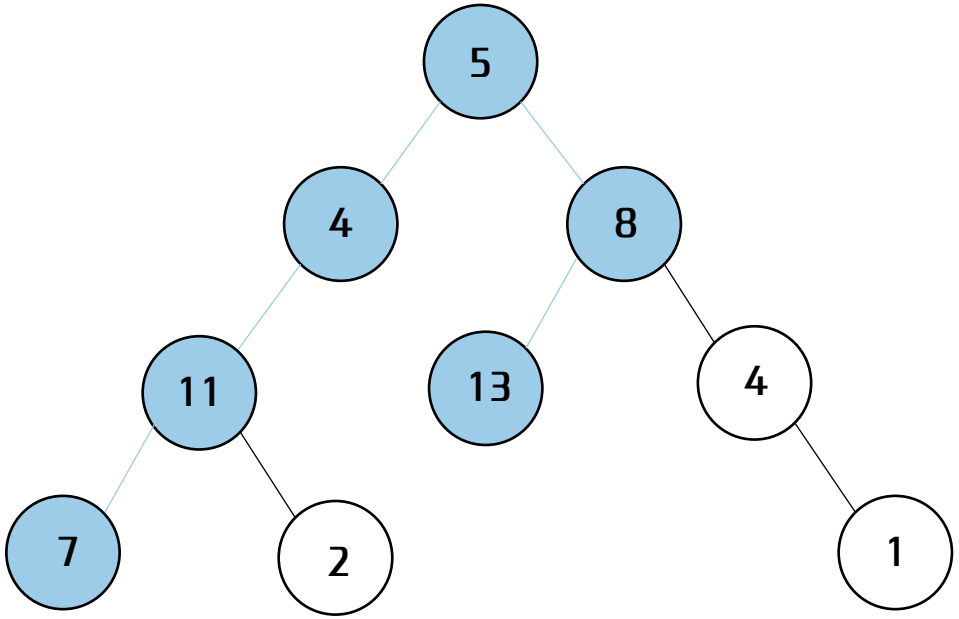
# Problem Example




Input: {0:10, 1:2, 2:10, 3:20, 4:1, 6:-25, 13:3, 14:4}

Output: 42

# Problem Example




Input: {0:5, 1:4, 2:8, 3:11, 5:13, 6:4, 7:7, 8:2, 14:1}

Output: 48

Part 2

# Solve the problem

## Skeleton Code (Python)

```
def maxPathSum(root, res=-sys.maxsize):
    if root is None:
        return 0, res
    # TODO


if __name__ == '__main__':
    nodes = {0:10, 1:2, 2:10, 3:20, 4:1, 6:-25, 13:3, 14:4}
    T = Tree()
    T.build(nodes)
    res = maxPathSum(T.root)[1]
    print(res)
```

- **Hint 1**

Use recursive function

- **Hint 2**

Traverse the tree in a bottom-up manner

Part 3

# Solution

## Solution 1. Simple solution

Traverse the tree and, for every node, calculate the maximum sum path starting from it and passing through its left and right child.

- Time complexity: $O(n^2)$
- Space complexity: $O(n)$

Where n is the number of the nodes

## Solution 2. Recursive approach

We can reduce the time complexity to linear by traversing the tree in a bottom-up manner. Each node returns the maximum path sum starting at that node to its parent.

# Solution

## Solution Code (Python)

```
def maxPathSum(root, res=-sys.maxsize):
    if root is None:
        return 0, res

    l, res = maxPathSum(root.left, res)
    r, res = maxPathSum(root.right, res)

    res = max(res, root.data)
    res = max(res, root.data+l)
    res = max(res, root.data+r)
    res = max(res, root.data+l+r)

    return max(root.data, root.data+max(l,r)), res
```

→ Initialize the parameter

→ Base case

→ Find maximum path sum starting from the left or right child

→ Four cases of sequence and updated result to maximum sum

→ return the maximum path sum starting from the given node and result

- Time complexity: $O(n)$, where n is the number of the nodes
- Space complexity: $O(h)$, where h is the height of the tree

# Thank you