**Presentation**

**Topic: Maximum Path Sum in Binary Tree**

Hello, everyone. My name is Yoon Seon Choi. The contents are as follows and I’ll introduce a problem using a binary tree.

1. **Introduce a problem with an example**

The problem that I prepared is to find maximum path sum given a binary tree. 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. Let me explain four cases of sequences. First case the sequence from the only current root node. Second, the sequence from the current root node and the left node of the root node. In this case, starting node is the left node. Third, the sequence from the current root node and the right node of the root node. In this case, starting node is the right node. And finally, the sequence from the current root node, the left and the right node of the root node. In this case, the order of sequence is the left node, root node, and right node. The path can start and end at any node of the binary tree. Some examples are as follows. In this binary tree, we can make some sequences, like this, like this or like this… And you can get the sum of nodes for each sequence. The nodes’ sum of this sequence is -18, and the nodes’ sum of this sequence 23. After finding each sum in this way, you can find the largest value. Then, in this case the maximum path sum is 42 in this sequence. Do you understand this problem? Okay, good. When you calculate the following example in the same way, the output is 48 in this sequence.

Then, let’s solve the problem. I’ll give you some hints after 5 minutes.

1. **Give hints**

The skeleton code is this. And I referred to the code for implementing the binary tree in the previous practice code.

First hint is you can use recursive function when you calculate the sum of the left or right subtree.

Second hint is you had better traverse the tree in a bottom-up manner.

1. **Present solutions, Q&A, and discussion**

Actually, there is two ways to solve this problem. The first solution is simpler than the second solution that I prepared. A simple solution would traverse the tree and, for every node, calculate the maximum sum path starting from it and passing through its left and right child. You could keep track of the maximum among all the maximum sum paths found and return it after all nodes are processed. In this solution, the time complexity is O(n2) and the space complexity is O(n) where n is the total number of nodes in the binary tree.

But 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. In order to ensure that the maximum path sum starts at that node, at most, one child of the node should be involved.

Look at this code. Initialize the parameter, result to minus sys.maxsize. And this is the base case. These codes are for finding maximum path sum starting from the left or right child. We can traverse in the bottom-up manner by using the recursive function. These four lines are the four cases of sequence and result is updated to maximum sum. Finally return the maximum path sum starting from the given node and result. In the main function, we get only result.

In this solution, the time complexity is O(n), where n is the number of nodes, and the space complexity is O(h), where h is the height of the tree.

This is the end of my presentation. Thank you for listening.