Report for Assignment 5

Deep Kiran Shroti
Roll Number- 12EC35013

March 9, 2016

Binary Tree

Assignment Statement

- Generate a binary tree of 100 nodes containing distinct integer (both +ve and -ve) keys at random.
- Given a binary tree, carry out an inorder traversal of the tree and output the keys separated by a blank and terminated by double LF.
- Given a binary tree, carry out an preorder traversal of the tree and output the keys separated by a blank and terminated by double LF.
- Given a binary tree, carry out an postorder traversal of the tree and output the keys separated by a blank and terminated by double LF.
- Given a binary tree storing integer keys, find the path from the root to a leaf node that has the maximum sum, output the keys separated by a blank and terminated by double LF.
- Given a binary tree storing integer keys, find the path (between any pair of nodes) in it that has the maximum sum, output the keys separated by a blank and terminated by double LF.

To calculate the path of Maximum sum from the root to leaf.

1 Implementation Details

1.1 Tree Traversal

There are three types of depth first traversals:

- Preorder Visit root, visit left subtree in preorder, visit right subtree in preorder

- Postorder Visit left subtree in postorder, right subtree in postorder, then the root
- Inorder Visit left subtree in inorder, then the root, then the right subtree in inorder

1.2 Maximum Sum

- **Step: 1** The maximum sum is calculated in *findMaxSum()* from the top to the bottom for every possible combination recursively. The maximum sum thus obtained is passed to the *main() function*.
- **Step: 2** The maximum sum is calculated in the function *findMaxSum()* is then passed to the function *findPath()*. In this function the present sum value is compared against the sum that will be generated from the left or right child. If the sum matched the left child, then the left node is printed else the right node is printed. Recursively the path for maximum sum from root to leaf is printed.

Maximum sum, any node to any other node: For this, we are using previous function. First the children are visited recursively, where we treat each child node as the root and try to find the maximum root to node sum at each point.

Now we compare the total of maximum sum given by left child, maximum sum given by the right child and the current key value and this total is compared with maximum sum obtained till now. If the calculated value is greater, maximum sum is updated, This way, we recursively visit children and at each node, try to find the possible maximum sum path.

2 Conclusion

In the algorithm the *findMaxSum()* and *findMaxSum2()*function divides the binary tree into two trees of size n/2. So,

$$T(n) = 2T(n/2) + b$$

Similarly, in case of *findPath()* function also divides the binary tree into two trees of size n/2. Hence,

$$T(n) = 2T(n/2) + b$$

Therefore the complexity is O(n) for both the cases.