Informatics II
Exercise 8

Week 9

Binary Search Tree

Task 1. (Properties of Binary Search Tree) Binary search tree is constructed by inserting nodes one by one. Assume there is a sequence of nodes `[12, 35, 17, 29, 45, 28, 9, 50]`,

a. Draw the tree structure after inserting each node starting with 12.

b. Calculate the following properties of the tree and justify your answers.

   (a) Height of the tree.

   (b) The depth of node `45`.

c. Is it true that the time complexity of search in a binary search tree is $O(1)$ in the best case? Identify when this best case is achieved.

d. Is it true that the time complexity of search in a binary search tree is $O(\log n)$ in the worst case where n is the number of nodes? Identify when this worst case is achieved.

Task 2. Binary Search Tree in C The structure of the tree node is defined as follow. The entry to a binary search tree (the root) is also a tree node.

```
struct TreeNode {
  int val;
  struct TreeNode* left;
  struct TreeNode* right;
};
```

Write a C program that contains the following functions:

a. Write the function *struct TreeNode* insert(struct TreeNode* root, int val)* that inserts an integer `val` into the binary search tree. The function returns the root of the tree.

b. Write the function *struct TreeNode* search(struct TreeNode* root, int val)* that finds the tree node with value `val` and returns the node. If not exist, return NULL;

c. Write the function *struct TreeNode* delete(struct TreeNode* root, int val)* that deletes the node with value `val` from the tree.

d. Write *void printTree(tree *root)* which prints all edges of the tree from root in the console in the format `Node A -- Node B`, and each edge is printed in a separate line. The ordering of the printed edges does not matter and may vary based on your implementation.

Task 3. Balance Binary Search Tree The time complexity for searching a key is $O(\log n)$ for a balanced binary search tree. Hence, it is useful if we can transform a binary search tree to a balanced binary search tree. Write a function that converts binary search tree to a balanced binary search tree, with the following sub-tasks:

a. Write a pseudocode for this function.

b. Implement the function in C.

Hint: If you perform the inorder tree walk to a binary search tree, what pattern can you find in the output array? With the help of this pattern, can you construct the balanced binary search tree?

Task 4. Range Query by Trimming Binary Search Tree One important application of the binary search tree is for finding elements. Suppose you need to find all elements in the range [`low`, `high`], write a pseudocode that returns the binary search tree with all its element lies in the range [`low`, `high`]. Write a pseudocode and C implementation for this task.

Note: You should not change the relative structure of the elements that will remain in the tree. For example, assume the original binary search tree is shown in Figure 1 and suppose you need to find all elements in the range $[17, 45]$, then the trimmed binary search tree should be shown in Figure 2.

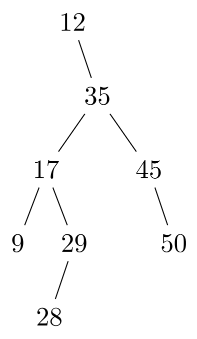

Figure 1: Original Binary Search Tree

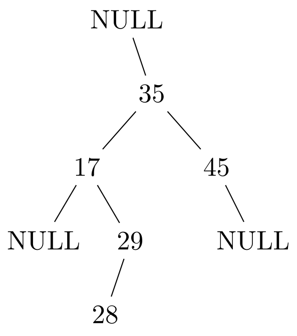

Figure 2: Trimmed Tree, the `NULL` node is used to indicate the nodes that are deleted from the original tree.