University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatics II
Exercise 8

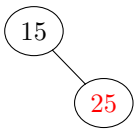
April 8, 2019

Binary Search Trees

Task 1. Binary search tree is created by inserting nodes 15, 25, 18, 31, 16, 2, 12, 3, 1, 49, 20. Show structure of the tree after insertion of each node.

Insert 15

Insert 25



Task 2. Considering binary search tree created in `Task 1`, show the structure of the tree after deleting each of the following nodes 3, 31, 18, 25, 15.
(When the node that is going to be deleted has two children, find the child with the largest value in the left subtree.)

Task 3.

```
1 struct TreeNode {
2   int val;
3   struct TreeNode* left;
4   struct TreeNode* right;
5 };
```

Given the above definition of a binary search tree with positive integer values, create a C program that contains the following functions:

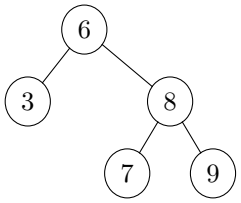
a) the function *void insert(struct TreeNode** root, int val)*, which gets a root node and a value `val` as an input and inserts a node with the value `val` into the proper position of the binary search tree.

b) the function *struct TreeNode* search(struct TreeNode* root, int val)*, which finds the tree node with value `val` and returns the node.

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

d) the function *void print(struct TreeNode* root)*, which prints the elements of a tree in order.

For example, if the values 6, 3, 8, 7 and 9 are inserted into an empty tree, your program should produce the binary tree shown below.



And it should print its elements in order: $3, 6, 7, 8, 9$. Test your program by performing the following operations:

- Create a root node `root` and insert the values 4, 2, 3, 8, 6, 7, 9, 12, 1.
- Print tree to the console.
- Delete the values 4, 7, 2 from the tree.
- Print tree to the console.

Task 4. Extend the program of the previous task writing in pseudocode the following functions:

a) *struct TreeNode* maximum(struct TreeNode* node)*, which returns the node with the largest value in the subtree with root node `n`.

b) *struct TreeNode* minimum(struct TreeNode* node)*, which returns the node with the smallest value in the subtree with root node `n`.

c) *struct TreeNode* successor(struct TreeNode* root, struct TreeNode* node)*, which takes a root node and any other node of the tree as an argument and returns the node with the next largest value. If the node given as the argument has the largest value in the tree, your function should return `NULL`.

d) *struct node* ith_smallest(struct TreeNode* root, int i)* that returns the node with the i^{th}-smallest element of the tree and is implemented using the functions *successor* and *minimum*. If the i^{th}-smallest element does not exist, your function must return `NULL`.

e) *int distanceToRoot(struct TreeNode* root, int x)* that returns the distance of the node with value `x` from the root node `root`.

f) *int distance(struct TreeNode* root, int a, int b)* that returns the distance between the two nodes with values `a` and `b` in a tree with root node `root`. Use the function *distanceToRoot* and assume that `a` $<$ `b`.