



**Marmara University Engineering Faculty**

**Department of Computer Engineering**

**CSE2025**

**DATA STRUCTURES**

**PROJECT #2**

**Yasin Alper BİNGÜL**

**170517033**

**Development Enviorment**

**Programming Language :** C Programming Language

**Input/Output Files :** input.txt

**Source Code File :** YasinAlperBingul.c

**CSE2025 Data Structures PROJECT #1 - Report**

1. **Problem Definition and Purpose of the Program**

In this Project, the program construct a BST with the keys that are in the given input file. The BST has the following properties:

RULE 1: Its nodes are distributed over 3 ⌊ log4 n ⌋ depth levels;

RULE 2: It provides the best possible total access time

1. **Problem Solution Description / Software Design**
   1. **Functions, Structs**
      * **struct** BTNodeType{

**struct** BTNodeType \*left;

**struct** BTNodeType \*right;

**int** key;

**int** height;

}

* + - **struct** BTNodeType\* newNode(**int** key)

This function creates new Binary Search Tree node. Basically creates a new root.

* + - **struct** BTNodeType \*rightRotate(**struct** BTNodeType \*p)

This function rotate right the Binary Search Tree to balance tree (for better access time). This rotation process applying similar to AVL trees. Program firstly collects the related nodes and swap them as to be rotated right. At the end the heigh values are update.

* + - **struct** BTNodeType \*leftRotate(**struct** BTNodeType \*p)

This function rotate left the Binary Search Tree to balance tree (for better access time). This rotation process applying similar to AVL trees. Program firstly collects the related nodes and swap them as to be rotated left. At the end the heigh values are update.

* + - **int** compareHeights(**struct** BTNodeType \*p)

This function compares left and right heights of entered node as parameter. Then subtracts the left node from the right node height.

* + - **int** findBigger(**int** num1, **int** num2)

This function compares two values and returns the bigger one

* + - **int** findHeight(**struct** BTNodeType \*p)

This function returns the height value of entered node as parameter

* + - **struct** BTNodeType\* insert(**struct** BTNodeType\* p, **int** key)

This function inserts a new node according to rules

* + - **int** inputValidation(**int**\* numbers)

Getting input informations from the file. Then determines the values are valid or not.

* + - **int** numberOfInput(**int**\* numbers)

Calculates the number of input numbers.

* + - **int** depthLevel(**int** numOfInputs)

This function calculates the depth level of Binary Search Tree.

* + - **int** getDepthLevel(**struct** BTNodeType \*p, **int** key)

This function is to find depthLevel of entered key value as parameter.

* + - **int** getElement(**struct** BTNodeType \*p, **int** key)

This function returns the element number of entered key value as parameter.

* + - **struct** BTNodeType\* findLeaf(**struct** BTNodeType\* p)

This function finds a leaf and return it.

* + - **struct** BTNodeType\* findAnotherLeaf(**struct** BTNodeType\* p)

This function finds another leaf and return it.

* + - **struct** BTNodeType\* removeLeaf(**struct** BTNodeType\* p, **int** key)

This function removes a leaf.

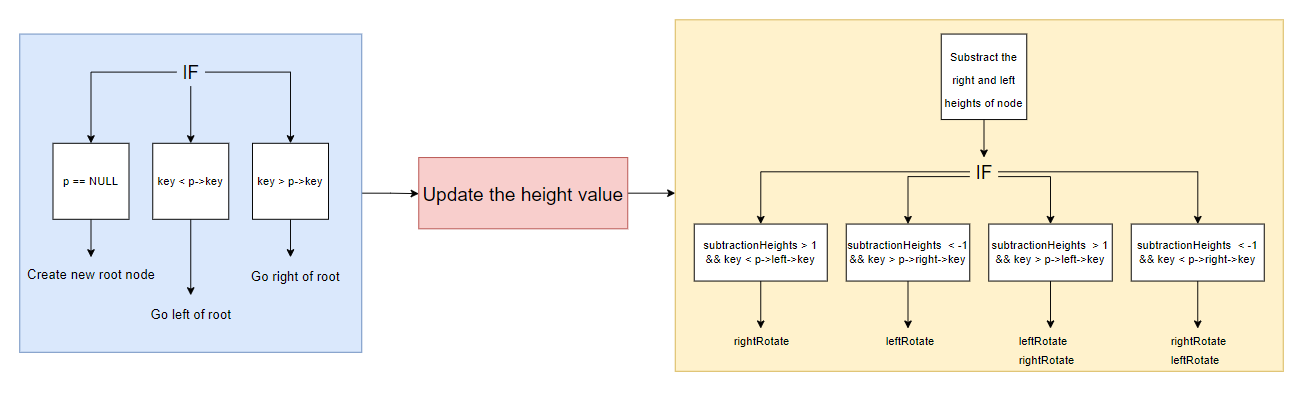
* + - **struct** BTNodeType\* insertDepthMost(**struct** BTNodeType\* p,**int** key)

This function inserts removed node leaf to the deepest depth.

* 1. **Algorithm Design (Step by step Description)**

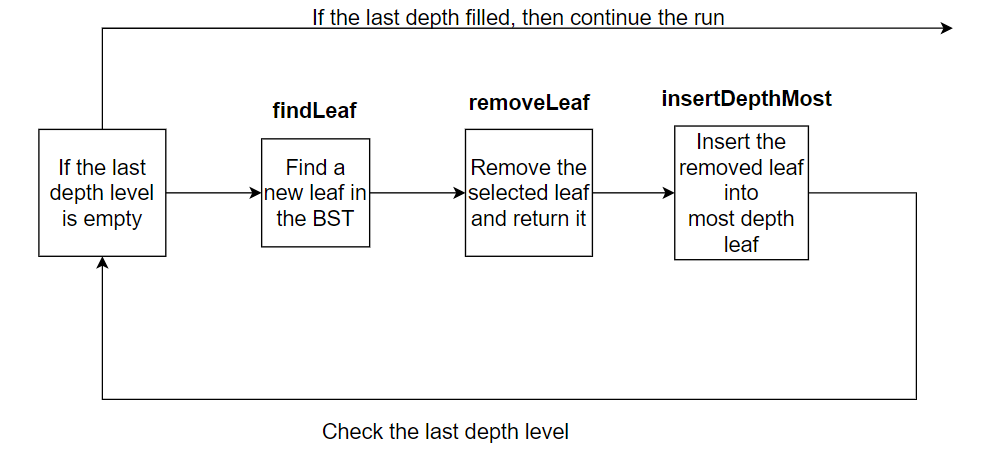
1. Firstly, **inputValidation** function runs to determine the values are valid or not. Then the values are store in array with related file operation codes.

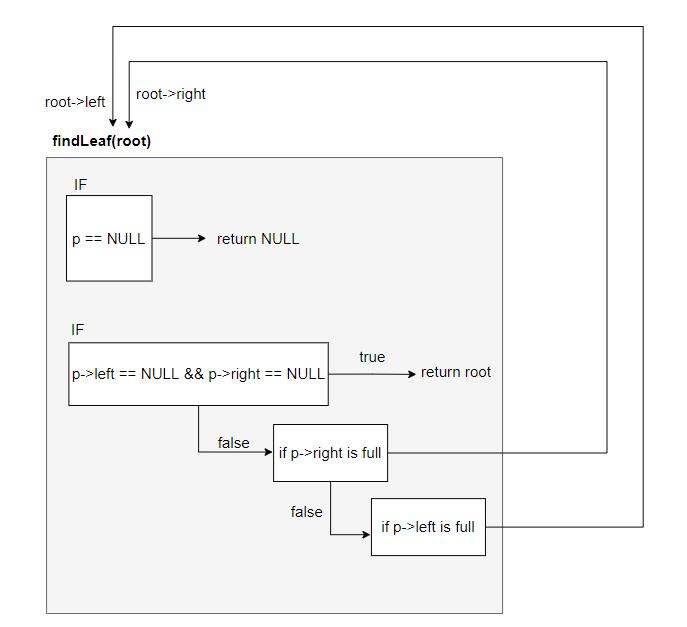
ii. If values are valid, then program creates a Binary Search Tree via **insert** function

 with using input numbers. Insert function runs like shown diagram down below.

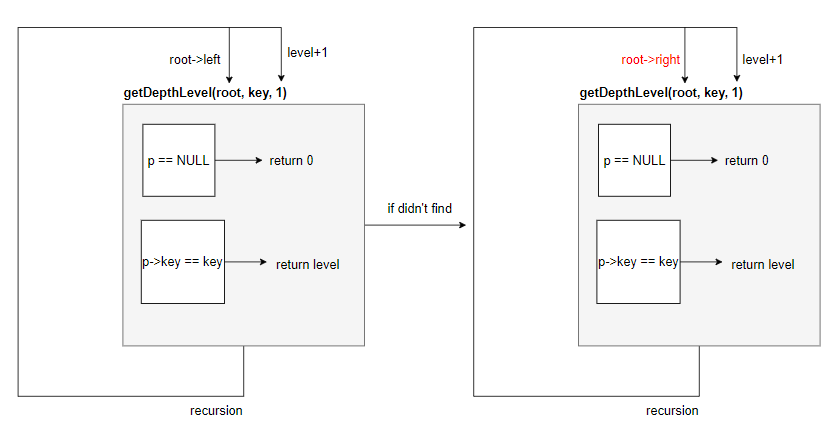
iii. Program gets depth level information with **getDepthLevel** function. And prints to console.

iv. If it doesn't fit with the rules, program fix the BST tree according to first requirement. (If the last depth level is empty, then fill that depth.) This process happens like diagram below.





v. In this step, the BST is completed. So user enters key value to search in BST. The depth value prints with **getDepthLevel** function like shown below diagram. But number of element of key value can’t printed.



1. **Results – Example Cases**

**Case 1:** 11 12 1 2 6 7 14 15 16 17 18 19 3 4 5 8 9 10 13 (Works fine)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Case 2:** 10 5 3 25 70 97 115 2 65 69 32 152 1 21 49 8 12 78 (Works fine)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Case 3:** 11 12 1 61 62 2 6 7 14 15 16 17 3 4 5 8 9 52 53 65 66 54 55 56 57 58 18 19 59 60 63 10 13 22 23 24 25 26 49 27 28 50 29 30 31 32 33 34 35 48 36 37 38 39 40 41 42 43 44 45 46 47 51 64

(Violates the second rule)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Case 4:** 11 12 1 61 62 2 6 7 14 15 16 17 3 4 5 8 9 52 53 65 66 -5 54 41 65 98 22 35 64

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Case 5:** 11 12 1 61 62 2 6 7 14 15

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Case 6:** 15 11 12 1 61 62 2 6 7 14 1 2 3 15 4 8 7 9

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

As a result, program works fine number of inputs with between 16 – 64. But if input number greater than 64, then program violates the second rule like case 3. In addition to this, unfortunately, number of element of key value can’t printed.

**Utilized resources**

* [**https://www.geeksforgeeks.org/get-level-of-a-node-in-a-binary-tree/**](https://www.geeksforgeeks.org/get-level-of-a-node-in-a-binary-tree/)
* [**https://www.geeksforgeeks.org/relationship-number-nodes-height-binary-tree/**](https://www.geeksforgeeks.org/relationship-number-nodes-height-binary-tree/)
* [**https://www.geeksforgeeks.org/avl-tree-set-1-insertion/**](https://www.geeksforgeeks.org/avl-tree-set-1-insertion/)