**Find the Closest Element in BST :-**

Medium Accuracy: 47.51% Submissions: 51K+ Points: 4

Given a [**BST**](http://quiz.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/) and an integer. Find the least absolute difference between any node value of the BST and the given integer.

**Example 1:**

**Input:**

10

  / \

  2 11

  / \

  1 5

  / \

  3 6

  \

  4

K = 13

**Output:** 2

**Explanation:** K=13. The node that has

value nearest to K is 11. so the answer

is 2

**Example 2:**

**Input:**

8

  / \

  1 9

  \ \

  4 10

  /

  3

K = 9

**Output:** 0

**Explanation:** K=9. The node that has

value nearest to K is 9. so the answer

is 0.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function**minDiff()**that takes the root of the BST and an integer K as its input and returns the minimum absolute difference between any node value of the BST and the integer K.

**Expected Time Complexity:**O(Height of the BST).  
**Expected Auxiliary Space:**O(Height of the BST).

**Constraints:**  
1 <= Number of nodes <= 100000

**Code :-**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

#define MAX\_HEIGHT 100000

// Tree Node

struct Node {

int data;

Node \*left;

Node \*right;

Node(int val) {

data = val;

left = right = NULL;

}

};

// } Driver Code Ends

class Solution

{

public:

void func(Node \*root, int k, int &ans){

if(root==NULL)

return;

//left traversal

func(root->left, k, ans);

//calculations

int dif = abs(root->data - k);

ans = min(ans, dif);

//right traversal

func(root->right, k, ans);

}

//Function to find the least absolute difference between any node

//value of the BST and the given integer.

int minDiff(Node \*root, int k){

int ans=INT\_MAX;

func(root, k, ans);

return ans;

}

};

//{ Driver Code Starts.

// Function to Build Tree

Node\* buildTree(string str)

{

// Corner Case

if(str.length() == 0 || str[0] == 'N')

return NULL;

// Creating vector of strings from input

// string after spliting by space

vector<string> ip;

istringstream iss(str);

for(string str; iss >> str; )

ip.push\_back(str);

// Create the root of the tree

Node\* root = new Node(stoi(ip[0]));

// Push the root to the queue

queue<Node\*> queue;

queue.push(root);

// Starting from the second element

int i = 1;

while(!queue.empty() && i < ip.size()) {

// Get and remove the front of the queue

Node\* currNode = queue.front();

queue.pop();

// Get the current node's value from the string

string currVal = ip[i];

// If the left child is not null

if(currVal != "N") {

// Create the left child for the current node

currNode->left = new Node(stoi(currVal));

// Push it to the queue

queue.push(currNode->left);

}

// For the right child

i++;

if(i >= ip.size())

break;

currVal = ip[i];

// If the right child is not null

if(currVal != "N") {

// Create the right child for the current node

currNode->right = new Node(stoi(currVal));

// Push it to the queue

queue.push(currNode->right);

}

i++;

}

return root;

}

int main() {

int t;

string tc;

getline(cin, tc);

t=stoi(tc);

while(t--)

{

string s;

getline(cin, s);

Node\* root1 = buildTree(s);

getline(cin, s);

int k = stoi(s);

// getline(cin, s);

Solution ob;

cout << ob.minDiff(root1, k);

cout << endl;

//cout<<"~"<<endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(no. of nodes)**

**S.C :- O(height of tree)**