#Two Sum IV - Input in BST

Problem statement Send feedback

You have been given a Binary Search Tree and a target value. You need to find out whether there exists a pair of node values in the BST, such that their sum is equal to the target value.
A binary search tree (BST), also called an ordered or sorted binary tree, is a rooted binary tree whose internal nodes each store a value greater than all the values keys in the node's left subtree and less than those in its right subtree.
Follow Up:

Can you solve this in O(N) time, and O(H) space complexity?

Detailed explanation (Input/output format, Notes, Images)

Constraints:

1 <= T <= 100
1 <= N <= 3000
-10^9 <= node data <= 10^9, (where node data != -1)
-10^9 <= target value <= 10^9

Where N denotes is the number of nodes in the given tree.

Time Limit: 1 second

Sample Input 1:

1
10 6 12 2 8 11 15 -1 -1 -1 -1 -1 -1 -1 -1
14

Sample Output 1:

True

→ Approach:

↳ Use morris traversal to traverse through the array.
↳ Use a set to store the complement of the nodes data.
↳ If a nodes data is found in set return true.
↳ Else false.

Time complexity : O(N)
Space complexity : O(N)

```
#include <bits/stdc++.h>

BinaryTreeNode<int> *getSuccessor(BinaryTreeNode<int> *curr) {
  BinaryTreeNode<int> *node = curr;
  if (curr->left) {
    curr = curr->left;
    while (curr->right != NULL && curr->right != node)
      curr = curr->right;
  }
  return curr;
}

bool twoSumInBST(BinaryTreeNode<int> *root, int target) {
  unordered_set<int> mpp;

  BinaryTreeNode<int> *curr = root;

  while (curr != NULL) {
    if (mpp.find(curr->data) != mpp.end())
      return true;
    else {
      int required = target - curr->data;
      mpp.insert(required);
    }
    if (curr->left) {
      BinaryTreeNode<int> *successor = getSuccessor(curr);
      if (successor->right) {
        successor->right = NULL;
        curr = curr->right;
      } else {
        successor->right = curr;
        curr = curr->left;
      }
    } else {
      curr = curr->right;
    }
  }
  return false;
}
```