Right Map
Time limit per test: 1 seconds
Memory limit per test: 256 megabytes
Input: standard input
Output: standard output

My friend Akash is a son of a very prestigious family. He had heard from his father that their ancestors were zamindars of the region. Because of that, they have many antique things in their house.

One day Akash finds a box from the house store room and a map from the box. So, naturally, a curiosity arises in him. He can see that the map has a key number and an array on the other side of each one.

You should create a BST from this array and add the key number under the last element of the array following BST rules. This map is correct if the BST is not destroyed even after setting the key number. Otherwise, the map is incorrect.

Input
Input starts with the length of array N and the key number K. For example, the following line contains the value of array A[i].

Constraints

1 <= N < = 100

1 <= K <= 10^5

1 <= A[i] <= 10^5

Output
Print "Yes" if the map is correct; otherwise, print "No".

Examples
Input:
6 8
3 5 4 6 2 7

Output:
Yes

Editorial:

প্রবলেমের প্রথমে একটি array এর সাইজ ও একটি key নাম্বার দেওয়া থাকবে। এরপরে array এর সংখ্যা গুলো থাকবে। এই সকল সংখ্যা দিয়ে একটি বাইনারি সার্চ ট্রি বানাতে হবে।

এভাবে আমরা বাইনারি সার্চ ট্রি বানাতে পারি,

```
void insrt(int value){
    node *current_node = new node();
    current_node -> data = value;
    current_node -> left = NULL;
    current_node -> right = NULL;

    if(root == NULL){
        root = current_node;
    }
    else{
        node *temp = root;

        while(true){
            if(current_node -> data <= temp -> data){
                if(temp -> left == NULL){
                    temp -> left = current_node;
                    break;
                }
                else temp = temp -> left;
            }
            else{
                if(temp -> right == NULL){
                    temp -> right = current_node;
                    break;
                }
                else temp = temp -> right;
            }
        }
    }
}
```

প্রাপ্ত array থেকে আমরা বাইনারি সার্চ ট্রি (BST) বানানোর পরে, ট্রি এখন ঠিক এমন দেখা যাচ্ছে,

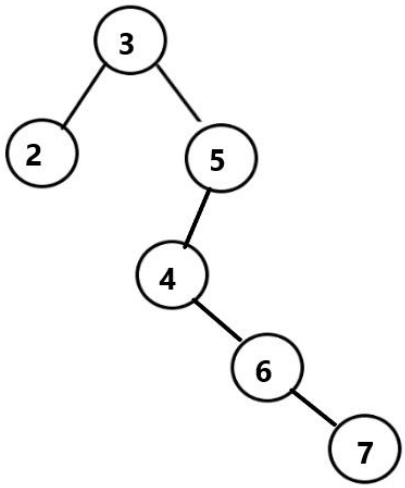

Fig : Binary Search Tree(BST)

এখন key নাম্বারটি array এর শেষ উপাদান অর্থাৎ ৭ এর নিচে BST এর নিয়ম অনুসারে বসাবো,

```
void insInBST (node* root, int key, int val){
    node *current_node = new node();
    current_node -> data = key;
    current_node -> left = NULL;
    current_node -> right = NULL;

    node* cur = root;

    while(true){
        if(cur->data <= val){
            if(cur->right != NULL) cur = cur->right;
            else{
                cur->right = current_node;
                break;
            }
        }
        else{
            if(cur->left != NULL) cur = cur->left;
            else{
                cur->left = current_node;
                break;
            }
        }
    }

}
```

এখন ট্রি ঠিক এমন দেখা যাচ্ছে,

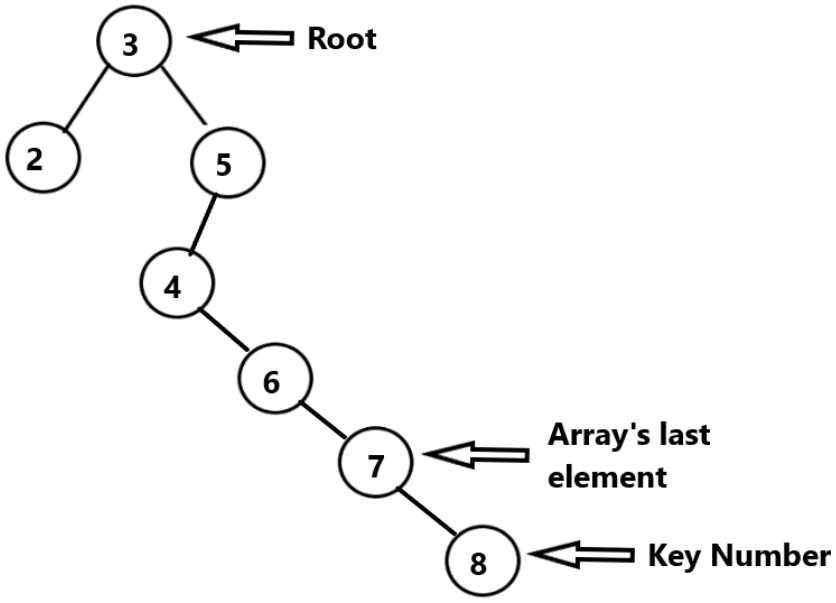

Fig : Binary Search Tree with new Key number.

এরপর, আমরা দেখবো ট্রি এখনো BST আছে কিনা,

এইটা পরীক্ষা করার জন্য আমরা key নাম্বার টি খুঁজে দেখবো । আমরা যদি এই ট্রি থেকে key নাম্বার টি খুঁজে পাই তাহলে বলতে পারবো এটি এখনো Binary search Tree (BST) আছে আর key নাম্বার টি খুঁজে না পেলে এই ট্রি কে আমরা BST বলতে পারবো না ।

```
node *srch (node* root, int key){
    if(root == NULL){
        return NULL;
    }

    if(root->data == key){
        return root;
    }

    if(root->data > key){
        return srch(root->left, key);
    }
    return srch(root->right, key);
}
```

ফলাফল হিসেবে আমাদের “Yes” অথবা “No” দেখাতে হবে। এই ট্রি এর সাথে key নাম্বার যোগ করার পরেও যদি এটি BST থাকে তাহলে আমরা “Yes” প্রিন্ট করবো না হয় “No” প্রিন্ট করবো।

Time Complexity:

Create BST : O(nlogn)

Key number set : O(1)

Search Key element : O(logn)

Full Code : https://paste.ubuntu.com/p/x33vT6yRR5/