Hello, **2024101067**.

# Balancing BT

Submit solution

My submissions
All submissions
Best submissions

**Points:** 100 (partial)
**Time limit:** 1.0s
**Memory limit:** 256M

**Allowed languages**
C

You are provided with a binary tree represented in the form of a parent array. In this array:

1. The element at index 0 is the root of the tree, indicated by a value of –1 (since it has no parent).
2. For every other index i, the value parent[i] represents the index of its parent node.

A binary tree is defined as balanced if, for every node in the tree, the difference between the height of its left subtree and the height of its right subtree is never more than 1.

Each test case contains only one test.

- The first line contains the number of elements in the tree.
- The second line contains the space separated parent array.

For each test output "Balanced" if the Binary Tree is balanced and "Not balanced" otherwise.

## Constraints:

1 <= n <= 1e6

## Input:

Copy

```
4
-1 0 0 1
```

## Output:

Copy

```
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

// Function to create a new Node
struct Node* newNode(int x) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = x;
    node->left = NULL;
    node->right = NULL;
    return node;
}

-----------------------------------------------------

int n;
scanf("%d", &n);  // Read the number of nodes

int parent[n+1];
struct Node* nodes[n+1];  // Array to store all nodes

// Read the parent array
for (int i = 0; i < n; i++) {
    scanf("%d", &parent[i]);
    nodes[i] = newNode(i);  // Create a node for each index
}

struct Node* root = NULL;

// Construct the binary tree from the parent array
for (int i = 0; i < n; i++) {
    if (parent[i] == -1) {
        root = nodes[i];  // Root node has no parent
    } else {
        struct Node* parentNode = nodes[parent[i]];
        if (parentNode->left == NULL) {
            parentNode->left = nodes[i];  // Assign left child
        } else if (parentNode->right == NULL) {
            parentNode->right = nodes[i];  // Assign right child
        }
    }
}
```

## Clarifications

No clarifications have been made at this time.