Hello, **2024101067**.

Mr. J is Back

Submit solution

My submissions
All submissions
Best submissions

Points: 100 (partial)
Time limit: 1.5s
Memory limit: 256M

Allowed languages
C

No story in the Himalayas is complete without mentioning Mr. J—our ever-busy TA. Known for his unwavering dedication to the gym, he somehow manages to disappear whenever it's time to create tasks for the DSA course. Always seen roaming with his new friend, he is nowhere to be found when work calls. So, while you debug your BST or tackle a tricky query, spare a thought for Mr. J and his legendary unavailability! Your task is to work with this BST by performing two operations: - **Insert:** Add a new node with a given integer value into the BST. - **Query:** For a given node with value `x`, calculate the sum of all its leaf descendants. If the node does not exist or has no leaf descendants, return `0`. if the query is called on leaf node, print 0.

Input Format

1. The first line contains two space-separated integers, `n` and `q`, where:
   - `n` is the initial number of nodes in the BST.
   - `q` is the number of operations.
2. The next `n` lines each contain a distinct integer representing the initial nodes inserted into the BST (inserted in the given order).
3. Each of the next `q` lines represents an operation and is in one of the following formats:
   - **Insertion:** `1 x` where `x` (1 ≤ `x` ≤ 10^9) is the value to be inserted.
   - **Query:** `2 x` where `x` is the value of a node in the BST for which you must output the sum of its leaf children.(leafs that are in the subtree of that value)

Note: The input is designed so that the BST remains balanced (i.e., no skewed trees).

Output Format

proudly powered by **DMOJ** | English (en)

Hello, **2024101067**.

Constraints

The problem consists of two batches:

1. **Batch 1:**
   - `n` $\leq 10^3$
   - `q` $\leq 10^3$
2. **Batch 2:**
   - `n` $\leq 10^6$
   - `q` $\leq 10^6$

All values in the BST are guaranteed to be unique.

Template code for BST (Can be modified)

Copy

Hello, **2024101067**.

```
struct node {
    element_type val;
    struct node* left;
    struct node* right;
};

typedef struct node* Tree;

Tree init(element_type val) {
    Tree ret = malloc(sizeof(struct node));
    ret->val = val;
    ret->left = NULL;
    ret->right = NULL;
    return ret;
}

Tree insert(Tree root, element_type val) {
    if (root == NULL)
        return init(val);
    if (val < root->val)
        root->left = insert(root->left, val);
    else
        root->right = insert(root->right, val);

    return root;
}

Tree findnode(Tree root, element_type val) {
    if (root == NULL || root->val == val)
        return root;
    if (val < root->val)
        return findnode(root->left, val);
    return findnode(root->right, val);
}
```

Sample Test Case

Input:

Copy

```
50 30 70 20 40
2 30
1 60
2 70
2 50
1 80
2 70
```

Output:

```
60
60
120
140
```

Copy

Explanation

1. **BST Initialization:**
    - **Insert 50:** This becomes the root.
    - **Insert 30:** Since 30 < 50, it is placed as the left child of 50.
    - **Insert 70:** Since 70 > 50, it is placed as the right child of 50.
    - **Insert 20:** Compared to 50 and then 30, 20 is placed as the left child of 30.
    - **Insert 40:** Compared to 50 and then 30, 40 is placed as the right child of 30.

    The BST structure now looks like:

    ```
      50
     /  \
    30   70
    /  \
    20   40
    ```

    Copy

    The initial leaf nodes are: **20, 40, 70**.

2. **Operation 1: Query (`2 30`)**

- For node **30**, the leaf descendants are **20** and **40**.
- **Sum:** 20 + 40 = **60**.

1. **Operation 2: Insertion (`1 60`)**

- Insert **60:** Since 60 > 50, go to the right of 50. Then, since 60 < 70, it becomes the left child of 70.

Updated BST:

Hello, **2024101067**.

```
      50
     /  \
    30   70
   / \   /
  20 40 60
```

Now, the leaf nodes are: **20, 40, 60**.

1. **Operation 3: Query (`2 70`)**

- For node **70**, the only leaf descendant is **60**.
- **Sum: 60**.

1. **Operation 4: Query (`2 50`)**

- For node **50**, the leaf descendants include **20**, **40**, and **60**.
- **Sum:** 20 + 40 + 60 = **120**.

1. **Operation 5: Insertion (`1 80`)**

- Insert **80:** Since 80 > 50, go right; then 80 > 70, so it becomes the right child of 70.

Updated BST:

```
        50
      /    \
    30      70
   / \     /  \
  20  40   60  80
```

Copy

The current leaf nodes are: **20, 40, 60, 80**.

1. **Operation 6: Query (`2 70`)**

- For node **70**, the leaf descendants are now **60** and **80**.
- **Sum:** 60 + 80 = **140**.

Can you harness the power of the BST to unlock the secrets of Binaryland and ensure the kingdom's everlasting prosperity?

Clarifications

Request clarification

No clarifications have been made at this time.

Hello, **2024101067**.