Practice Arena

Practice problems aimed to improve your coding skills.

All My Descendants

LAB-PRAC-14_SORT-MISC

All My Descendants [20 marks]

Problem Statement
We have read about binary trees in the tutorial. These are data structures where all nodes, except the root, have exactly one parent, and all nodes may have a left child and a right child. Nodes that have no children are called leaves. In this problem, we will construct a binary tree using the concepts we learnt while studying linked lists.

The first line of the input will give you n, a strictly positive integer. This will be the number of nodes in the tree. The next line will give you n integers, separated by a space. These are the occupants of a binary tree we are going to construct. However, the binary tree does not contain these elements in this order. Store these n integers in an array arr (please refer to the hint below to see how to do so).

After this, in the next line, we will give you the index of the root of the binary tree in the array arr. After this, there will be n lines telling you which are the children of these n nodes. Each line will contain a triplet of numbers a b c (i.e. the three numbers will be separated by a space). This will indicate that the element at index a in the array arr has the element at index b in the array arr, as its left child, and the element at index c in the array arr, as its right child. If a node has no left child, b will be -1. If a node has no right child, c will be -1. The last line of the input will give you the index q of an element in the array arr. Call the node corresponding to q as the query node. q will be between 0 and n-1 (both included).

Create a binary tree with n nodes with the elements in the given arrangement. Please refer to the hints to see an easy way to do so. In the first line of the input, print the value of the number stored at the query node. In the next line, print the value of the number stored in the left child of the query node (if there is no left child print -1). In the next line, print the value of the number stored in the right child of the query node (if there is no right child print -1). In the fourth and final line of the output, print the sum of values stored at q and all its descendants (i.e. the sum of value stored at the query node + sum of values stored in its children, however many they may be + sum of values stored in is grandchildren, however many they may be + values stored at great-grand children, and so on).

P.S. The name of this problem is derived from that of a television serial called "All My Children" that used to air in the US. The serial ran for a ridiculous 41 years, spanning multiple resets and relaunches, and finally ended in 2011

Caution

1. A node may have a left child but no right child. A node may have a right child but no left child. Leaves have neither a left child nor a right child.
2. If the query node is a leaf, then the sum will simply be the number stored at the node itself. Print -1 for values stored in the left and right children since there are no left and right children for leaves.
3. The numbers being stored in the nodes of the tree may be positive, negative or zero. Numbers may repeat in multiple nodes of the tree. Remember, this is just a binary tree, not a binary search tree. So there is no restriction on which number may be stored in which node.
4. n may be any strictly positive integer, even 1 in which the tree will have a single node - the root.
5. There are four lines of output. There should be no extra spaces at the end of any of the lines.

HINTS: Following these steps might make your life easier:

1. Create a structure node to store an integer and a left pointer and a right pointer.
2. Make an array of n nodes to store the n integers as given as well as the pointers to the left and the right children.
3. Store the location of the root node
4. For each triplet (a b c) given as input, make the node at index a in the array point to the node at index b in the array as its left child and the point to the node at index c in the array as its right child
5. Try to write a recursive function to print the sum of all descendants of a node.

Note that we are asking you to use a static data structure like an array, to store a binary tree, which is a dynamic data structure, only to make this problem less complicated. What you have created is not a very efficient dynamic data structure.

EXAMPLE:
INPUT

```
9
1 2 3 4 5 6 7 8 9
4
4 3 1
2 7 8
7 –1 –1
3 6 5
5 –1 –1
6 –1 –1
8 –1 –1
1 2 0
0 –1 –1
3
```

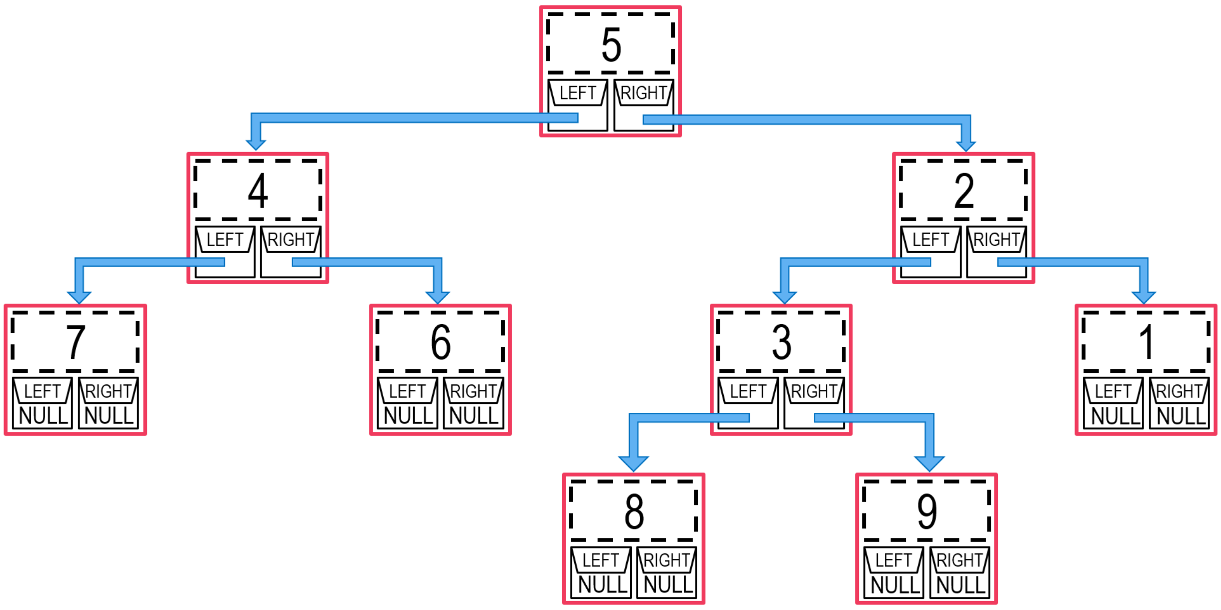
OUTPUT:

```
4
7
6
17
```

Explanation: See the following image to see what this tree looks like. The node at index 3 in the array is 4 and its children are 7 and 6. The sum of all nodes which are descendants of 4 (including 4 itself) is 17.

For example, had q been 1 i.e. the query node been the node that stores the number 2, then the answer would have been

```
2
3
1
23
```



Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are four lines in your output. The first three lines carry 10% weightage and the last line carries 70% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6293)