lee215 ★ 47711 Last Edit: December 2, 2019 6:41 PM 2.2K VIEWS

50

Solution 1: DFS

Build up the mapping of parent and its children.
Recursively find the sum and the count of subtree.
Time `O(N)`, Space `O(N)`

Python:

```
    def deleteTreeNodes(self, n, parent, value):
        sons = {i: set() for i in xrange(n)}
        for i, p in enumerate(parent):
            if i: sons[p].add(i)

        def dfs(x):
            total, count = value[x], 1
            for y in sons[x]:
                t, c = dfs(y)
                total += t
                count += c
            return total, count if total else 0
        return dfs(0)[1]
```

Solution 2: One Pass

Wrote this solution in 2019-11-30,
don't have the premium so I'll lose the access to my this post.

Hidden condition:
`parent[i] < i` for all `i > 0`
a parent always has have smaller index of its children.

Intuition

Don't ask me why, my friend told me.
He said it was an intuition.
We hardly see a tree problem represented by array.
Another observation is tha, no test case contains a tree with `sum = 0`.

The problem writer was lasy and made some random cases.
I personally think taht he didn't actually do the work.

Complexity

Time `O(N)`
Space `O(N)`

C++