5908. Count Nodes With the Highest Score

My Submissions (/contest/weekly-contest-264/problems/count-nodes-with-the-highest-score/submissions/)

Back to Contest (/contest/weekly-contest-264/)

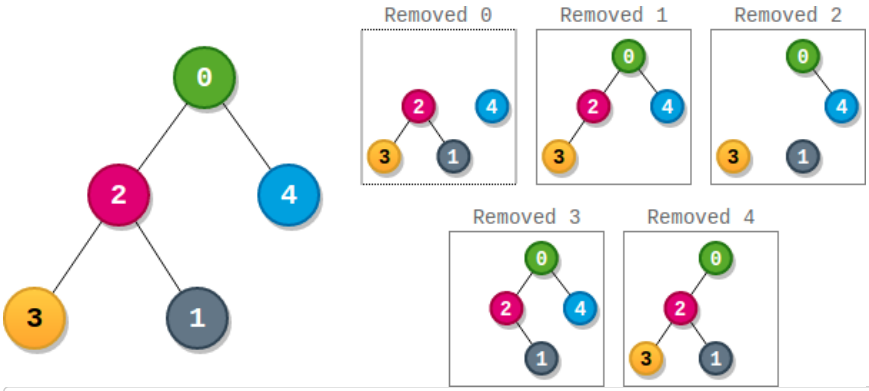
There is a **binary** tree rooted at `0` consisting of `n` nodes. The nodes are labeled from `0` to `n - 1`. You are given a **0-indexed** integer array `parents` representing the tree, where `parents[i]` is the parent of node `i`. Since node `0` is the root, `parents[0] == -1`.

Each node has a **score**. To find the score of a node, consider if the node and the edges connected to it were **removed**. The tree would become one or more **non-empty** subtrees. The **size** of a subtree is the number of the nodes in it. The **score** of the node is the **product of the sizes** of all those subtrees.

Return *the **number** of nodes that have the **highest score***.

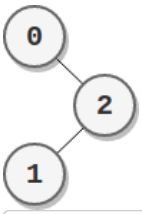
User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:




```
Input: parents = [-1,2,0,2,0]
Output: 3
Explanation:
- The score of node 0 is: 3 * 1 = 3
- The score of node 1 is: 4 = 4
- The score of node 2 is: 1 * 1 * 2 = 2
- The score of node 3 is: 4 = 4
- The score of node 4 is: 4 = 4
The highest score is 4, and three nodes (node 1, node 3, and node 4) have the highest score.
```

Example 2:



```
Input: parents = [-1,2,0]
Output: 2
Explanation:
- The score of node 0 is: 2 = 2
- The score of node 1 is: 2 = 2
- The score of node 2 is: 1 * 1 = 1
The highest score is 2, and two nodes (node 0 and node 1) have the highest score.
```

Constraints:

- `n == parents.length`

- $2 <= n <= 10^5$
- `parents[0] == -1`
- `0 <= parents[i] <= n - 1` for `i != 0`
- `parents` represents a valid binary tree.

JavaScript

```javascript
const initializeGraph = (n) => { let G = []; for (let i = 0; i < n; i++) { G.push([]); } return G; };

let g, n, res, cnt;
const countHighestScoreNodes = (parents) => {
    res = -1, cnt = 0, n = parents.length, g = initializeGraph(n);
    for (let i = 0; i < n; i++) {
        if (parents[i] == -1) continue;
        g[parents[i]].push(i);
    }
    dfs(0);
    return cnt;
};

const dfs = (x) => {
    let subtree = 0, p = 1;
    for (const child of g[x]) {
        let tmp = dfs(child);
        subtree += tmp;
        p *= tmp;
    }
    if (subtree < n - 1) p *= n - 1 - subtree;
    if (p > res) {
        res = p;
        cnt = 1;
    } else if (p == res) {
        cnt++;
    }
    return subtree + 1;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/576232014/) | More Details > (/submissions/detail/576232014/)

Share your acceptance!