6191. Number of Good Paths

My Submissions (/contest/weekly-contest-312/problems/number-of-good-paths/submissions/) Back to Contest (/contest/weekly-contest-312/)

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

There is a tree (i.e. a connected, undirected graph with no cycles) consisting of `n` nodes numbered from `0` to `n - 1` and exactly `n - 1` edges.

You are given a **0-indexed** integer array `vals` of length `n` where `vals[i]` denotes the value of the i^{th} node. You are also given a 2D integer array `edges` where `edges[i] = [a_i, b_i]` denotes that there exists an **undirected** edge connecting nodes a_i and b_i.

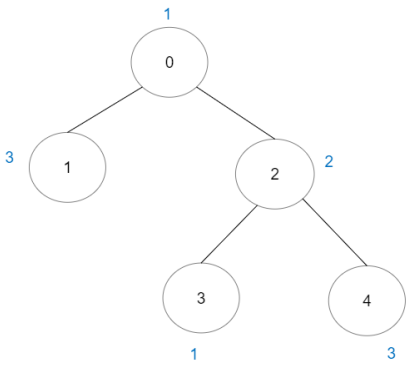
A **good path** is a simple path that satisfies the following conditions:

1. The starting node and the ending node have the **same** value.
2. All nodes between the starting node and the ending node have values **less than or equal to** the starting node (i.e. the starting node's value should be the maximum value along the path).

Return *the number of distinct good paths*.

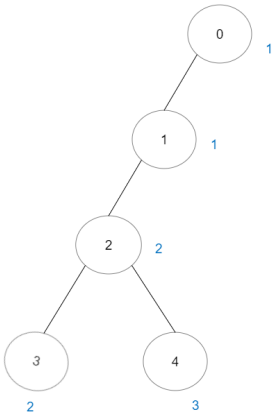
Note that a path and its reverse are counted as the **same** path. For example, `0 -> 1` is considered to be the same as `1 -> 0`. A single node is also considered as a valid path.

Example 1:




```
Input: vals = [1,3,2,1,3], edges = [[0,1],[0,2],[2,3],[2,4]]
Output: 6
Explanation: There are 5 good paths consisting of a single node.
There is 1 additional good path: 1 -> 0 -> 2 -> 4.
(The reverse path 4 -> 2 -> 0 -> 1 is treated as the same as 1 -> 0 -> 2 -> 4.)
Note that 0 -> 2 -> 3 is not a good path because vals[2] > vals[0].
```

Example 2:



```
Input: vals = [1,1,2,2,3], edges = [[0,1],[1,2],[2,3],[2,4]]
Output: 7
Explanation: There are 5 good paths consisting of a single node.
There are 2 additional good paths: 0 -> 1 and 2 -> 3.
```

Example 3:


```
Input: vals = [1], edges = []
Output: 1
Explanation: The tree consists of only one node, so there is one good path.
```

Constraints:

- n == vals.length
- 1 <= n <= 3 * 10^4
- 0 <= vals[i] <= 10^5
- edges.length == n - 1
- edges[i].length == 2
- 0 <= a_i, b_i < n
- a_i != b_i
- edges represents a valid tree.

JavaScript

```
const packUG = (g, edges) => { for (const [u, v] of edges) { g[u].push(v); g[v].push(u); } };
const initializeGraph = (n) => { let g = []; for (let i = 0; i < n; i++) { g.push([]); } return g; };

function DJSet(n) {
    // parent[i] < 0, -parent[i] is the group size which root is i. example: (i -> parent[i] -> parent[parent[i]] -> parent[parent[parent[i]]] ...)
    // parent[i] >= 0, i is not the root and parent[i] is i's parent. example: (... parent[parent[parent[i]]] -> parent[parent[i]] -> parent[i] -> i)
    let parent = Array(n).fill(-1);
    return { find, union, count, equiv, par }
    function find(x) {
        return parent[x] < 0 ? x : parent[x] = find(parent[x]);
    }
    function union(x, y) {
        x = find(x);
        y = find(y);
        if (x == y) return false;
        if (parent[x] < parent[y]) [x, y] = [y, x];
        parent[x] += parent[y];
        parent[y] = x;
        return true;
    }
    function count() { // total groups
        return parent.filter(v => v < 0).length;
    }
    function equiv(x, y) { // isConnected
        return find(x) == find(y);
    }
    function par() {
        return parent;
    }
}

const numberOfGoodPaths = (a, edges) => {
    let n = a.length, g = initializeGraph(n), f = Array(n).fill(0), ds = new DJSet(n), res = 0;
    packUG(g, edges);
    let d = a.map((x, i) => [x, i]);
    d.sort((x, y) => {
        if (x[0] != y[0]) return x[0] - y[0];
        return x[1] - y[1];
    })
    for (let r = 0; r < n;) { // l: start node  r: end node
```

```
        let l = r;
        while (r < n && d[l][0] == d[r][0]) r++; // condition 1
        for (let i = l; i < r; i++) {
            let cur = d[i][1];
            for (const child of g[cur]) {
                if (a[child] <= d[l][0]) ds.union(child, cur); // condition 2
            }
        }
        for (let i = l; i < r; i++) { // loop the path
            let cur = d[i][1];
            res += ++f[ds.find(cur)];
        }
        for (let i = l; i < r; i++) {
            let cur = d[i][1];
            f[ds.find(cur)]--;
        }
    }
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/808047826/) | More Details > (/submissions/detail/808047826/)

Share your acceptance!