6942. Count Paths That Can Form a Palindrome in a Tree

My Submissions (/contest/weekly-contest-355/problems/count-paths-that-can-form-a-palindrome-in-a-tree/submissions/)

Back to Contest (/contest/weekly-contest-355/)

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

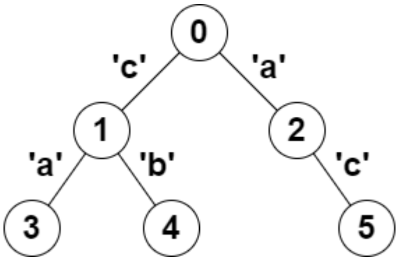
You are given a **tree** (i.e. a connected, undirected graph that has no cycles) **rooted** at node `0` consisting of `n` nodes numbered from `0` to `n - 1`. The tree is represented by a **0-indexed** array `parent` of size `n`, where `parent[i]` is the parent of node `i`. Since node `0` is the root, `parent[0] == -1`.

You are also given a string `s` of length `n`, where `s[i]` is the character assigned to the edge between `i` and `parent[i]`. `s[0]` can be ignored.

Return *the number of pairs of nodes* `(u, v)` *such that* `u < v` *and the characters assigned to edges on the path from* `u` *to* `v` *can be* ***rearranged*** *to form a* ***palindrome***.

A string is a **palindrome** when it reads the same backwards as forwards.

Example 1:



```
Input: parent = [-1,0,0,1,1,2], s = "acaabc"
Output: 8
Explanation: The valid pairs are:
- All the pairs (0,1), (0,2), (1,3), (1,4) and (2,5) result in one character which is always a palindrome.
- The pair (2,3) result in the string "aca" which is a palindrome.
- The pair (1,5) result in the string "cac" which is a palindrome.
- The pair (3,5) result in the string "acac" which can be rearranged into the palindrome "acca".
```

Example 2:

```
Input: parent = [-1,0,0,0,0], s = "aaaaa"
Output: 10
Explanation: Any pair of nodes (u,v) where u < v is valid.
```

Constraints:

- `n == parent.length == s.length`
- `1 <= n <=` 10^5
- `0 <= parent[i] <= n - 1` for all `i >= 1`
- `parent[0] == -1`
- `parent` represents a valid tree.
- `s` consists of only lowercase English letters.

JavaScript

```
const ord = (c) => c.charCodeAt();
const addOneOrManyMap = (m, x, cnt = 1) => m.hasOwnProperty(x) ? m[x] += cnt : m[x] = 1;

let s, p, memo;
const countPalindromePaths = (parent, ss) => {
    s = ss, p = parent, memo = new Map();
    let n = p.length, f = {}, res = 0, vals = [];
    for (let i = 0; i < n; i++) vals.push(dfs(i));
    for (const v of vals) {
        res += f[v] || 0;
        for (let i = 0; i < 26; i++) res += f[v ^ (1 << i)] || 0;
        addOneOrManyMap(f, v);
    }
    return res;
};

const dfs = (idx) => {
    if (idx == 0) return 0;
    if (memo.has(idx)) return memo.get(idx);
    let par = p[idx];
    let res = dfs(par) ^ (1 << ord(s[idx]) - 97);
    memo.set(idx, res);
    return res;
};
```

□ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/1001561072/) More Details > (/submissions/detail/1001561072/)

Share your acceptance!

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)