(/) Explore(/explore/) Problems(/problemset/all/) Interview New (/contest/) Contest Discuss(/discuss/) Store ☆ Premium (/subscribe?ref=nb_npl) 0tree-zigzag-level-order-traversal/)

6363. Find the String with LCP

My Submissions (/contest/weekly-contest-333/problems/find-the-string-with-lcp/submissions/) Back to Contest (/contest/weekly-contest-333/)

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

We define the `lcp` matrix of any **0-indexed** string `word` of `n` lowercase English letters as an `n x n` grid such that:

- `lcp[i][j]` is equal to the length of the **longest common prefix** between the substrings `word[i,n-1]` and `word[j,n-1]`.

Given an `n x n` matrix `lcp`, return the alphabetically smallest string `word` that corresponds to `lcp`. If there is no such string, return an empty string.

A string `a` is lexicographically smaller than a string `b` (of the same length) if in the first position where `a` and `b` differ, string `a` has a letter that appears earlier in the alphabet than the corresponding letter in `b`. For example, `"aabd"` is lexicographically smaller than `"aaca"` because the first position they differ is at the third letter, and `'b'` comes before `'c'`.

Example 1:

```
Input: lcp = [[4,0,2,0],[0,3,0,1],[2,0,2,0],[0,1,0,1]]
Output: "abab"
Explanation: lcp corresponds to any 4 letter string with two alternating letters. The lexicographically smallest of them is "a
```

Example 2:

```
Input: lcp = [[4,3,2,1],[3,3,2,1],[2,2,2,1],[1,1,1,1]]
Output: "aaaa"
Explanation: lcp corresponds to any 4 letter string with a single distinct letter. The lexicographically smallest of them is "
```

Example 3:

```
Input: lcp = [[4,3,2,1],[3,3,2,1],[2,2,2,1],[1,1,1,3]]
Output: ""
Explanation: lcp[3][3] cannot be equal to 3 since word[3,...,3] consists of only a single letter; Thus, no answer exists.
```

Constraints:

- `1 <= n == lcp.length == lcp[i].length <= 1000`
- `0 <= lcp[i][j] <= n`

JavaScript

```
const initialize2DArray = (n, m) => [...Array(n)].map(() => Array(m).fill(0));
const char = (ascii) => String.fromCharCode(ascii);

const findTheString = (g) => {
    let n = g.length, res = Array(n).fill(-1), max = 1, dp = initialize2DArray(n, n);
    res[0] = 0;
    for (let i = 1; i < n; i++) {
        let match = false;
        for (let j = 0; j < i; j++) {
            if (g[i][j] > 0) {
                res[i] = res[j];
                match = true;
                break;
            }
        }
        if (!match) {
            res[i] = max;
            if (res[i] >= 26) return '';
            max++;
        }
    }
    dp[n - 1][n - 1] = 1;
    for (let i = n - 2; ~i; i--) {
        for (let j = i; j < n; j++) {
            if (res[i] != res[j]) {
                dp[i][j] = 0;
            } else {
                dp[i][j] = j + 1 < n ? dp[i + 1][j + 1] + 1 : 1;
            }
        }
    }
    for (let i = 0; i < n; i++) {
        for (let j = i; j < n; j++) {
            if (dp[i][j] != g[i][j] || dp[i][j] != g[j][i]) return '';
        }
    }
    return res.map(x => char(x + 97)).join("");
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/902738741/) More Details (/submissions/detail/902738741/)

Share your acceptance!

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

United States (/region)