e(/explore/) Problems(/problemset/all/) Contest(/contest/) Discuss(/discuss/) Interview Store 0 (/problems/painting-the-walls/) (/subs ref=lp premiu

100077. Longest Unequal Adjacent Groups Subsequence II

My Submissions (/contest/biweekly-contest-115/problems/longest-unequal-adjacent-groups-subsequence-ii/submissions/)

Back to Contest (/contest/biweekly-contest-115/)

User Accepted:	37
User Tried:	107
Total Accepted:	37
Total Submissions:	135
Difficulty:	Medium

You are given an integer `n`, a **0-indexed** string array `words`, and a **0-indexed** array `groups`, both arrays having length `n`.

The **hamming distance** between two strings of equal length is the number of positions at which the corresponding characters are **different**.

You need to select the **longest subsequence** from an array of indices `[0, 1, ..., n - 1]`, such that for the subsequence denoted as $[i_0, i_1, ..., i_{k - 1}]$ having length `k`, the following holds:

- For **adjacent** indices in the subsequence, their corresponding groups are **unequal**, i.e., $groups[i_j] \,!= groups[i_{j + 1}]$, for each `j` where `0 < j + 1 < k`.
- $words[i_j]$ and $words[i_{j + 1}]$ are **equal** in length, and the **hamming distance** between them is `1`, where `0 < j + 1 < k`, for all indices in the subsequence.

Return *a string array containing the words corresponding to the indices **(in order)** in the selected subsequence*. If there are multiple answers, return *any of them*.

A **subsequence** of an array is a new array that is formed from the original array by deleting some (possibly none) of the elements without disturbing the relative positions of the remaining elements.

Note: strings in `words` may be **unequal** in length.

Example 1:

```
Input: n = 3, words = ["bab","dab","cab"], groups = [1,2,2]
Output: ["bab","cab"]
Explanation: A subsequence that can be selected is [0,2].
- groups[0] != groups[2]
- words[0].length == words[2].length, and the hamming distance between them is 1.
So, a valid answer is [words[0],words[2]] = ["bab","cab"].
Another subsequence that can be selected is [0,1].
- groups[0] != groups[1]
- words[0].length == words[1].length, and the hamming distance between them is 1.
So, another valid answer is [words[0],words[1]] = ["bab","dab"].
It can be shown that the length of the longest subsequence of indices that satisfies the conditions is 2.
```

Example 2:

```
Input: n = 4, words = ["a","b","c","d"], groups = [1,2,3,4]
Output: ["a","b","c","d"]
Explanation: We can select the subsequence [0,1,2,3].
It satisfies both conditions.
Hence, the answer is [words[0],words[1],words[2],words[3]] = ["a","b","c","d"].
It has the longest length among all subsequences of indices that satisfy the conditions.
Hence, it is the only answer.
```

Constraints:

- `1 <= n == words.length == groups.length <= 1000`
- `1 <= words[i].length <= 10`
- `1 <= groups[i] <= n`
- `words` consists of **distinct** strings.
- `words[i]` consists of lowercase English letters.

JavaScript

```javascript
const getWordsInLongestSubsequence = (n, a, b) => {
    let dp = Array(n).fill(0), from = Array(n).fill(-1), longest = 0, res = [];
    for (let i = 0; i < n; i++) {
        dp[i] = 1;
        for (let j = 0; j < i; j++) {
            if (b[i] != b[j] && dp[j] + 1 > dp[i] && ok(a[i], a[j])) {
                dp[i] = dp[j] + 1;
                from[i] = j;
            }
        }
        longest = Math.max(longest, dp[i]);
    }
    for (let i = 0; i < n; i++) {
        if (dp[i] == longest) {
            let cur = i;
            while (cur >= 0) {
                res.push(a[cur]);
                cur = from[cur];
            }
            break;
        }
    }
    return res.reverse();
};

const ok = (s, t) => s.length == t.length && ham(s, t) == 1

const ham = (s, t) => {
    let n = s.length, res = 0;
    for (let i = 0; i < n; i++) {
        if (s[i] != t[i]) res++;
    }
    return res;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/1075148861/) More Details (/submissions/detail/1075148861/)

Share your acceptance!

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

United States (/region)