Problem 2900: Longest Unequal Adjacent Groups Subsequence I

Problem Information

Difficulty: Easy
Acceptance Rate: 67.04%
Paid Only: No
Tags: Array, String, Dynamic Programming, Greedy

Problem Description

You are given a string array `words` and a **binary** array `groups` both of length `n`.

A subsequence of `words` is **alternating** if for any two _consecutive_ strings in the sequence, their corresponding elements at the _same_ indices in `groups` are **different** (that is, there _cannot_ be consecutive 0 or 1).

Your task is to select the **longest alternating** subsequence from `words`.

Return _the selected subsequence. If there are multiple answers, return**any** of them._

Note: The elements in `words` are distinct.

Example 1:

Input: words = ["e","a","b"], groups = [0,0,1]

Output: ["e","b"]

Explanation: A subsequence that can be selected is `["e","b"]` because `groups[0] != groups[2]`. Another subsequence that can be selected is `["a","b"]` because `groups[1] != groups[2]`. It can be demonstrated that the length of the longest subsequence of indices that satisfies the condition is `2`.

Example 2:

Input: words = ["a","b","c","d"], groups = [1,0,1,1]

Output: ["a","b","c"]

Explanation: A subsequence that can be selected is `["a","b","c"]` because `groups[0] != groups[1]` and `groups[1] != groups[2]`. Another subsequence that can be selected is `["a","b","d"]` because `groups[0] != groups[1]` and `groups[1] != groups[3]`. It can be shown that the length of the longest subsequence of indices that satisfies the condition is `3`.

Constraints:

* `1 <= n == words.length == groups.length <= 100` * `1 <= words[i].length <= 10` * `groups[i]` is either `0` or `1.` * `words` consists of **distinct** strings. * `words[i]` consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<string> getLongestSubsequence(vector<string>& words, vector<int>&
groups) {

}
};
```

Java:

```
class Solution {
public List<String> getLongestSubsequence(String[] words, int[] groups) {

}
}
```

Python3:

```
class Solution:
def getLongestSubsequence(self, words: List[str], groups: List[int]) ->
List[str]:
```