2273. Find Resultant Array After Removing Anagrams

Description

You are given a **0-indexed** string array `words`, where `words[i]` consists of lowercase English letters.

In one operation, select any index `i` such that `0 < i < words.length` and `words[i - 1]` and `words[i]` are **anagrams**, and **delete** `words[i]` from `words`. Keep performing this operation as long as you can select an index that satisfies the conditions.

Return `words` *after performing all operations*. It can be shown that selecting the indices for each operation in **any** arbitrary order will lead to the same result.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase using all the original letters exactly once. For example, `"dacb"` is an anagram of `"abdc"`.

Example 1:

```
Input: words = ["abba","baba","bbaa","cd","cd"]
Output: ["abba","cd"]
Explanation:
One of the ways we can obtain the resultant array is by using the following operations:
- Since words[2] = "bbaa" and words[1] = "baba" are anagrams, we choose index 2 and delete words[2].
  Now words = ["abba","baba","cd","cd"].
- Since words[1] = "baba" and words[0] = "abba" are anagrams, we choose index 1 and delete words[1].
  Now words = ["abba","cd","cd"].
- Since words[2] = "cd" and words[1] = "cd" are anagrams, we choose index 2 and delete words[2].
  Now words = ["abba","cd"].
We can no longer perform any operations, so ["abba","cd"] is the final answer.
```

Example 2:

```
Input: words = ["a","b","c","d","e"]
Output: ["a","b","c","d","e"]
Explanation:
No two adjacent strings in words are anagrams of each other, so no operations are performed.
```

Constraints:

- `1 <= words.length <= 100`
- `1 <= words[i].length <= 10`
- `words[i]` consists of lowercase English letters.