<https://leetcode.com/problems/word-subsets/description/>

You are given two string arrays words1 and words2.

A string b is a subset of string a if every letter in b occurs in a including multiplicity.

For example, "wrr" is a subset of "warrior" but is not a subset of "world".

A string a from words1 is universal if for every string b in words2, b is a subset of a.

Return an array of all the universal strings in words1. You may return the answer in any order.

**Example 1:**

**Input:** words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["e","o"]

**Output:** ["facebook","google","leetcode"]

**Example 2:**

**Input:** words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["l","e"]

**Output:** ["apple","google","leetcode"]

**Constraints:**

1 <= words1.length, words2.length <= 10^4

1 <= words1[i].length, words2[i].length <= 10

words1[i] and words2[i] consist only of lowercase English letters.

All the strings of words1 are **unique**.

**Attempt 1: 2024-11-30**

**Solution 1: Hash Table (10 min)**

class Solution {

    public List<String> wordSubsets(String[] words1, String[] words2) {

        int[] freq = new int[26];

        for(String word : words2) {

            int[] tmp = new int[26];

            for(char c : word.toCharArray()) {

                tmp[c - 'a']++;

            }

            for(int i = 0; i < 26; i++) {

                freq[i] = Math.max(freq[i], tmp[i]);

            }

        }

        List<String> result = new ArrayList<>();

        for(String word : words1) {

            int[] tmp = new int[26];

            for(char c : word.toCharArray()) {

                tmp[c - 'a']++;

            }

            int i;

            for(i = 0; i < 26; i++) {

                if(tmp[i] < freq[i]) {

                    break;

                }

            }

            if(i == 26) {

                result.add(word);

            }

        }

        return result;

    }

}

Time Complexity: O(n)

Space Complexity: O(n)

**Refer to**

<https://leetcode.com/problems/word-subsets/solutions/175854/java-c-python-straight-forward/>

Explanation

For each word b in B,

we use function counter to count occurrence of each letter.

We take the maximum occurrences of counts, and use it as a filter of A.

public List<String> wordSubsets(String[] A, String[] B) {

int[] count = new int[26], tmp;

int i;

for (String b : B) {

tmp = counter(b);

for (i = 0; i < 26; ++i)

count[i] = Math.max(count[i], tmp[i]);

}

List<String> res = new ArrayList<>();

for (String a : A) {

tmp = counter(a);

for (i = 0; i < 26; ++i)

if (tmp[i] < count[i])

break;

if (i == 26) res.add(a);

}

return res;

}

int[] counter(String word) {

int[] count = new int[26];

for (char c : word.toCharArray()) count[c - 'a']++;

return count;

}