916. Word Subsets

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**.

916. Word Subsets

```
/*
```

Counting+Hashing

Time compelxity: $O(n_1.l_1+n_2.(l_2+26)+n_1.26)=O(n_1.l_1+n_2.l_2+n_1)$

Space compelxity: $O(n_1+26)=O(n_1)$

n_1 : number of words in the list words1

l_1 : size of each word in words1

n_2 : number of words in the list words2

l_2 : size of each word in words2

```
*/
typedef std::vector<std::string> vs;
typedef std::vector<int> vi;
class Solution {
   public:
      vector<string> wordSubsets(vs& words1, vs& words2){
         // For each word in words1, coompute the frequencies of its letters
         std::unordered_map<std::string,vi> word_freq;
         for(auto& word: words1){
            vi freq(26,0);
            for(auto& letter: word) freq[letter-'a']++;
            word_freq[word]=freq;
         }

         // For each word in words2, compute the frequencies of its letters
         vi freq2(26,0);
         for(auto& word: words2){
            vi freq(26,0);
            for(auto& letter: word) freq[letter-'a']++;

            // For each letter in word, determine its max frequency
            for(int i=0;i<26;++i) freq2[i]=std::max(freq2[i],freq[i]);
         }
```

```
        // For each word (from words1) stored in the hash table
        vs ans;
        for(auto& [word,freq1]: word_freq){
            // Check if all its letters exist in each word from words2
            // if the frequency of each letter the word is greater or equal than the
            // maximum frequency of all words from words2, then the word is universal.
            int i=0;
            while( i<26 && freq2[i]<=freq1[i]) i++;
            if(i==26) ans.push_back(word);
        }
        return ans;
    }
};
```