<https://leetcode.com/problems/group-anagrams/>

**Group Anagrams**

**Given an array of strings strs, group the anagrams together. You can return the answer in any order.**

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.**

Example 1:

Input: strs = ["eat","tea","tan","ate","nat","bat"]

Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Example 2:

Input: strs = [""]

Output: [[""]]

Example 3:

Input: strs = ["a"]

Output: [["a"]]

Constraints:

1 <= strs.length <= 104

0 <= strs[i].length <= 100

strs[i] consists of lowercase English letters.

**Method 1: (Categorize by Sorted String)**

Two strings are anagrams if and only if their sorted strings are equal.

Use a hashmap with key as sorted string to store indices of strings in the same group.

Traverse the hashmap add anagram strings in the result 2d vector.

Time Complexity: O(nklogk)  *[ ]*

Space Complexity: O(nk) *[result requires same space as input strs]*

vector<vector<string>> groupAnagrams(vector<string>& strs) {

        vector<vector<string>> result;

        map<string,vector<string>> mp;

        for(int i=0; i<strs.size();i++){ O(nklogk)

            string ordered\_string; //let k= max length of string

            ordered\_string=strs[i];

            sort(ordered\_string.begin(), ordered\_string.end());//O(klogk)

            mp[ordered\_string].push\_back(strs[i]);

        }

        for(auto m: mp)

            result.push\_back(m.second);

        return result;

    }

**Method 2: (Categorize by Count)**

Two strings are anagrams if and only if their character counts (respective number of occurrences of each character) are the same.

Use a hashmap with key as character frequency to store strings in the same group.

Traverse the hashmap add anagram strings in the result 2d vector.

Time Complexity: O(nk) *[n= size of input string vector, k= max length of each string]*

Space Complexity: O(nk) *[result requires same space as input strs]*

vector<vector<string>> groupAnagrams(vector<string>& strs) {

        vector<vector<string>> result;

        map<string,vector<string>> mp;

        for(int i=0; i<strs.size();i++){ //O(nk)

            string str = strs[i]; //let k= max length of string

            string keyStr = "";

            vector<int> count(26,0);

            for(int j=0; j<str.size(); j++) //O(k)

                count[str[j]-'a']++;

            for(int k=0;k<26;k++){

                keyStr.append(to\_string(count[k]));

                keyStr.append("#");

            }

            mp[keyStr].push\_back(str);

        }

        for(auto m: mp){

            result.push\_back(m.second);

        }

        return result;

   }

    }

for(auto m: mp){

result.push\_back(m.second);

}

return result;

}