**Problem Name:** Find Words That Can Be formed by characters

**Topics:** Hash Table, Array, String

**Companies:** Karat, Indeed, Amazon, Facebook

**Level:** Easy

**Language:** C++

**Problem Statement:** You are given an array of strings words and a string chars.

A string is **good** if it can be formed by characters from chars (each character can only be used once).

Return *the sum of lengths of all good strings in words*.

**Input Format:**

First line of the input contains integer n (no of strings).

Second line contain n space separated strings.

Third and last line contain string chars.

Ex:

4

cat bt hat tree

atach

**Output Format:** Print sum of lengths of all good strings in the words. Ex for above input, output would be

6

**Constraints:**

* 1 <= words.length <= 1000
* 1 <= words[i].length, chars.length <= 100
* words[i] and chars consist of lowercase English letters.

**Examples:**

**Input:** words = ["hello","world","leetcode"], chars = "welldonehoneyr"

**Output:** 10

**Explanation:** The strings that can be formed are "hello" and "world" so the answer is 5 + 5 = 10.

**Brute force Solution:**

**Explanation:**

Our goal to is find whether each string in words can be made by letters in chars string. This can be done if a words[i] of words has count of every letter of it less than or equal to chars letters. For storing and comparision you can use a std::unordered\_map<char,int> to keep count of letters and their frequencies or use a vector<int> to store the count of all lowercase English alphabets where count of a's is at 0th index, b's at 1st index... But let us use **sorting** and try another way of doing it. What we are going to do is that we will:

* Sort every string in words
* Sort the chars string
* For every string in words, one of them will be true. So,
  + Check whether str's letter is **same** as chars's letter. If yes, delete the str's letter. Our job is to make the str a null string. If it gets null, the it means it had count of letters <= count of chars letters of respective letter.
  + Check whether str's **>** char's letter (*ASCII comparision*). If yes, simply iterate further on chars string.
  + Check whether str's **<** char's letter. If yes, break because no other letter of chars will be less than str's letter and str will not become empty at ay point (*because we sorted the string*).
  + If str become null, add its original length to final answer and continue the process on rest of the strings.

**Code:**

#include <bits/stdc++.h>

using namespace std;

int countCharacters(vector<string>& words, string chars) {

    for (auto& str : words)

        sort(str.begin(),str.end());

    sort(chars.begin(),chars.end());

    int length = 0;

    for (auto& str : words){

        auto it\_s = str.begin();

        int str\_length = str.size();

        for (auto it\_c = chars.begin(); it\_c != chars.end(); it\_c++){

            if (\*it\_c == \*it\_s){

                it\_s = str.erase(it\_s);

                if (str.empty()){

                    length += str\_length;

                    break;

                }

            }

            else if (\*it\_c > \*it\_s)

                break;

        }

    }

    return length;

}

int main() {

    string chars;

    int n, result;

    cin>>n;

    vector<string> words(n);

    for(int i=0; i<n; i++){

        cin>>words[i];

    }

    cin>>chars;

    result = countCharacters(words, chars);

    cout<<result;

    return 0;

}

**Time Complexity**: O(N2)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:**

Create a map to store the occurrences of each character in "chars". Each character is mapped to their frequency <char, int>. if a character is found, increment its frequency. Declare an integer to store sum of all the strings that are valid. Create a flag to switch to false if the string cannot be formed from "chars". Create a map to store the frequency of characters in current string. Loop through string and if the frequency of any character in the string is greater than the frequency of the same character in the "chars" string, make flag false and end loop (word is invalid). Append string's length to "count" if the flag is still true(valid)

**Code:**

#include <bits/stdc++.h>

using namespace std;

int countCharacters(vector<string>& words, string chars) {

    vector<int> chFreq(26, 0);

    for(auto ch : chars) chFreq[ch - 'a']++;

    int count = 0;

    for(auto str : words){

        bool flag = true;

        vector<int> copyFreq(26, 0);

        for(auto ch : str) {

            if(++copyFreq[ch - 'a'] > chFreq[ch - 'a']){

                flag = false;

                break;

            }

        }

        if(flag) count += str.size();

    }

    return count;

}

int main() {

    string chars;

    int n, result;

    cin>>n;

    vector<string> words(n);

    for(int i=0; i<n; i++){

        cin>>words[i];

    }

    cin>>chars;

    result = countCharacters(words, chars);

    cout<<result;

    return 0;

}

**Time Complexity**: O(m\*n)

**Space Complexity:** O(1)