Shortest Completing Word

Find the minimum length word from a given dictionary `words`, which has all the letters from the string `licensePlate`. Such a word is said to *complete* the given string `licensePlate`

Here, for letters we ignore case. For example, `"P"` on the `licensePlate` still matches `"p"` on the word.

It is guaranteed an answer exists. If there are multiple answers, return the one that occurs first in the array.

The license plate might have the same letter occurring multiple times. For example, given a `licensePlate` of `"PP"`, the word `"pair"` does not complete the `licensePlate`, but the word `"supper"` does.

Example 1:

```
Input: licensePlate = "1s3 PSt", words = ["step", "steps", "stripe", "stepple"]
Output: "steps"
Explanation: The smallest length word that contains the letters "S", "P", "S", and "T
".
Note that the answer is not "step", because the letter "s" must occur in the word twi
ce.
Also note that we ignored case for the purposes of comparing whether a letter exists
in the word.
```

Example 2:

```
Input: licensePlate = "1s3 456", words = ["looks", "pest", "stew", "show"]
Output: "pest"
Explanation: There are 3 smallest length words that contains the letters "s".
We return the one that occurred first.
```

Note:

1. `licensePlate` will be a string with length in range `[1, 7]`.
2. `licensePlate` will contain digits, spaces, or letters (uppercase or lowercase).
3. `words` will have a length in the range `[10, 1000]`.
4. Every `words[i]` will consist of lowercase letters, and have length in range `[1, 15]`.

Solution 1

```
class Solution {
    public String shortestCompletingWord(String licensePlate, String[] words) {
        String target = licensePlate.toLowerCase();
        int [] charMap = new int[26];
        // Construct the character map
        for(int i = 0 ; i < target.length(); i++){
            if(Character.isLetter(target.charAt(i))) charMap[target.charAt(i) - 'a'
]++;
        }
        int minLength = Integer.MAX_VALUE;
        String result = null;
        for (int i = 0; i < words.length; i++){
            String word = words[i].toLowerCase();
            if(matches(word, charMap) && word.length() < minLength) {
                minLength = word.length();
                result  = words[i];
            }
        }
        return result;
    }
    private boolean matches(String word, int[] charMap){
        int [] targetMap = new int[26];
        for(int i = 0; i < word.length(); i++){
            if(Character.isLetter(word.charAt(i))) targetMap[word.charAt(i) - 'a']++
;
        }

        for(int i = 0; i < 26; i++){
            if(charMap[i]!=0 && targetMap[i]<charMap[i]) return false;
        }
        return true;
    }
}
```

written by roshan108 original link here

Solution 2

```
def shortestCompletingWord(self, licensePlate, words):
        regex = re.compile('[^a-zA-Z]')
        licensePlate = regex.sub('', licensePlate)
        counter = Counter(licensePlate.lower())
        res = ''
        for word in words:
            if self.contains(counter, word):
                if res == '' or len(word) < len(res):
                    res = word
        return res


    def contains(self, counter1, w2):
        c2 = Counter(w2)
        c2.subtract(counter1)
        return all(map(lambda x: x >= 0,c2.values()))
```

written by johnyrufus16 original link here

Solution 3

```
class Solution {
    public String shortestCompletingWord(String licensePlate, String[] words) {
        int[] array = new int[26];
        String str = licensePlate.toLowerCase();
        for(int i=0;i<str.length();i++){
            if(str.charAt(i)>='a' && str.charAt(i)<='z'){
                array[str.charAt(i)-'a']++;
            }
        }


        int minLen = Integer.MAX_VALUE;
        int min_index = 0;

        int p=-1;
        for(String word:words){
            p++;
            int[] temp = new int[26];
            for(char c:word.toCharArray()){
                temp[c-'a']++;
            }

            boolean check = true;
            for(int i=0;i<26;i++){
                if(temp[i]<array[i]){
                    check = false;
                    break;
                }
            }

            if(check==true && minLen>word.length()){
                minLen = word.length();
                min_index = p;
            }
        }

        return words[min_index];
    }
}
```

written by tiandiao123 original link here