【华为OD机考 统一考试机试C卷】万能单词拼写 / 掌握单词个数（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

有一个字符串数组 words 和一个字符串 chars。假如可以用 chars 中的字母拼写出 words 中的某个“单词”（字符串），那么我们就认为你掌握了这个单词。

words 的字符仅由 a-z 英文小写字母组成，例如 “abc”

chars 由 a-z 英文小写字母和 “?” 组成。其中英文 “?” 表示万能字符，能够在拼写时当作任意一个英文字母。例如：“?” 可以当作 “a” 等字母。

注意：每次拼写时，chars 中的每个字母和万能字符都只能使用一次。

输出词汇表 words 中你掌握的所有单词的个数。没有掌握任何单词，则输出0。

输入描述

第一行：输入数组 words 的个数，记作N。

第二行 ~ 第N+1行：依次输入数组words的每个字符串元素

第N+2行：输入字符串chars

输出描述

输出一个整数，表示词汇表 words 中你掌握的单词个数

备注

- 1 ≤ words.length ≤ 100
- 1 ≤ words[i].length, chars.length ≤ 100
- 所有字符串中都仅包含小写英文字母、英文问号

用例1

输入

```
4
cat
bt
hat
tree
atach??
```

输出

```
3
```

说明 : 可以拼写字符串"cat"、“bt"和"hat”

用例2

输入

```
3
hello
world
cloud
welldonehohneyr
```

输出

```
2
```

说明：可以拼写字符串"hello"和"world"

用例3

输入

```
3
apple
car
window
welldoneapplec?
```

输出

```
2
```

说明：可以拼写字符串"apple"和“car”

解题思路

1. 我们需要创建一个数组来计算字符集中每个字母的出现次数，同时计算问号的数量。
2. 接下来，我们遍历所有的单词，对于每个单词，我们也需要计算单词中每个字母的出现次数。
3. 然后，我们需要判断是否可以使用字符集来拼写单词。如果单词中的字母数量大于字符集中的字母数量，我们可以使用问号来替代。如果问号的数量不足，那么我们就不能拼写这个单词。如果所有的字母都可以匹配，那么我们就可以拼写这个单词。

C++

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

```

```cpp

// 判断是否可以拼写单词
bool canSpell(vector<int>& wordCount, vector<int>& count, int wildcards) {
    for (int i = 0; i < 26; i++) {
        if (wordCount[i] > count[i]) {
            wildcards -= wordCount[i] - count[i]; // 如果单词中的字母数量大于字符集中的字母数量，则使用问号替代
            if (wildcards < 0) {
                return false; // 如果问号不足，则返回false
            }
        }
    }
    return true; // 如果所有的字母都可以匹配，则返回true
}

int main() {
    int N;
    cin >> N; // 读取单词数量
    cin.ignore(); // 忽略换行符

    vector<string> words(N);
    for (int i = 0; i < N; i++) {
        getline(cin, words[i]); // 读取每个单词
    }

    string chars;
    getline(cin, chars); // 读取字符集

    vector<int> count(26, 0);
    int wildcards = 0;
    for (char c : chars) {
        if (c != '?') {
            count[c - 'a']++; // 如果字符不是问号，则计数
        } else {
            wildcards++; // 计算问号的数量
        }
    }

    int res = 0;
    for (string& word : words) {
        vector<int> wordCount(26, 0);

```

```
        for (char c : word) {
            wordCount[c - 'a']++; // 计数单词中的字母
        }

        if (canSpell(wordCount, count, wildcards)) {
            res++; // 如果可以拼写单词，则结果加1
        }
    }

    cout << res << endl; // 输出结果

    return 0;
}
```

Java

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // 创建一个扫描器用于读取输入
        int N = scanner.nextInt(); // 读取输入的第一个整数，即单词的数量
        scanner.nextLine();  // 消耗掉换行符

        // 创建一个字符串数组用于存储所有的单词
        String[] words = new String[N];
        for (int i = 0; i < N; i++) {
            words[i] = scanner.nextLine(); // 读取每一个单词
        }

        // 读取包含字母和问号的字符串
        String chars = scanner.nextLine();

        // 创建一个数组用于计数字母的出现次数
        int[] count = new int[26];
        for (char c : chars.toCharArray()) {
            if (c != '?') {
                count[c - 'a']++; // 如果字符不是问号，则计数
            }
        }
```

```java
        // 计算问号的数量
        int wildcards = (int)chars.chars().filter(c -> c == '?').count();

        // 初始化结果为0
        int res = 0;

        // 遍历所有的单词
        for (String word : words) {
            // 创建一个数组用于计数单词中字母的出现次数
            int[] wordCount = new int[26];
            for (char c : word.toCharArray()) {
                wordCount[c - 'a']++; // 计数单词中的字母
            }

            // 如果可以拼写单词，则结果加1
            if (canSpell(wordCount, count, wildcards)) {
                res++;
            }
        }

        // 输出结果
        System.out.println(res);
    }

    // 判断是否可以拼写单词
    private static boolean canSpell(int[] wordCount, int[] count, int wildcards) {
        for (int i = 0; i < 26; i++) {
            if (wordCount[i] > count[i]) {
                wildcards -= wordCount[i] - count[i]; // 如果单词中的字母数量大于字符集中的字母数量，则使用问号替代
                if (wildcards < 0) {
                    return false; // 如果问号不足，则返回false
                }
            }
        }
        return true; // 如果所有的字母都可以匹配，则返回true
    }
}
```

javaScript

```javascript
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout
});

let input = [];
readline.on('line', line => {
  input.push(line);
}).on('close', () => {
  const N = parseInt(input[0]); // 读取单词数量
  const words = input.slice(1, N + 1); // 读取单词
  const chars = input[N + 1]; // 读取字符集

  const count = new Array(26).fill(0);
  let wildcards = 0;
  for (let c of chars) {
    if (c !== '?') {
      count[c.charCodeAt() - 'a'.charCodeAt()]++; // 如果字符不是问号，则计数
    } else {
      wildcards++; // 计算问号的数量
    }
  }

  let res = 0;
  for (let word of words) {
    const wordCount = new Array(26).fill(0);
    for (let c of word) {
      wordCount[c.charCodeAt() - 'a'.charCodeAt()]++; // 计数单词中的字母
    }

    if (canSpell(wordCount, count, wildcards)) {
      res++; // 如果可以拼写单词，则结果加1
    }
  }

  console.log(res); // 输出结果
});

function canSpell(wordCount, count, wildcards) {
  for (let i = 0; i < 26; i++) {

```

```
    if (wordCount[i] > count[i]) {
      wildcards -= wordCount[i] - count[i]; // 如果单词中的字母数量大于字符集中的字母数量，则使用问号替代
      if (wildcards < 0) {
        return false; // 如果问号不足，则返回false
      }
    }
  }
  return true; // 如果所有的字母都可以匹配，则返回true
}
```

Python

```python
def can_spell(word_count, count, wildcards):
    for i in range(26):
        if word_count[i] > count[i]:
            wildcards -= word_count[i] - count[i]  # 如果单词中的字母数量大于字符集中的字母数量，则使用问号替代
            if wildcards < 0:
                return False  # 如果问号不足，则返回false
    return True  # 如果所有的字母都可以匹配，则返回true


N = int(input())  # 读取单词数量
words = [input() for _ in range(N)]  # 读取单词
chars = input()  # 读取字符集

count = [0] * 26
wildcards = chars.count('?')  # 计算问号的数量
for c in chars:
    if c != '?':
        count[ord(c) - ord('a')] += 1  # 如果字符不是问号，则计数

res = 0
for word in words:
    word_count = [0] * 26
    for c in word:
        word_count[ord(c) - ord('a')] += 1  # 计数单词中的字母

    if can_spell(word_count, count, wildcards):
        res += 1  # 如果可以拼写单词，则结果加1


```

```
print(res)  # 输出结果
```

C语言

```c
#include <stdio.h>
#include <string.h>

// 判断是否可以拼写单词
int canSpell(int wordCount[26], int count[26], int wildcards) {
    for (int i = 0; i < 26; i++) {
        if (wordCount[i] > count[i]) {
            wildcards -= wordCount[i] - count[i]; // 如果单词中的字母数量大于字符集中的字母数量，则使用问号替代
            if (wildcards < 0) {
                return 0; // 如果问号不足，则返回0
            }
        }
    }
    return 1; // 如果所有的字母都可以匹配，则返回1
}

int main() {
    int N;
    scanf("%d", &N); // 读取单词数量
    getchar(); // 忽略换行符

    char words[N][101];
    for (int i = 0; i < N; i++) {
        fgets(words[i], 101, stdin); // 读取每个单词
        words[i][strcspn(words[i], "\n")] = 0; // 去掉末尾的换行符
    }

    char chars[101];
    fgets(chars, 101, stdin); // 读取字符集
    chars[strcspn(chars, "\n")] = 0; // 去掉末尾的换行符

    int count[26] = {0};
    int wildcards = 0;
    for (int i = 0; i < strlen(chars); i++) {
        if (chars[i] != '?') {
```

```
            count[chars[i] - 'a']++; // 如果字符不是问号，则计数
        } else {
            wildcards++; // 计算问号的数量
        }
    }

    int res = 0;
    for (int i = 0; i < N; i++) {
        int wordCount[26] = {0};
        for (int j = 0; j < strlen(words[i]); j++) {
            wordCount[words[i][j] - 'a']++; // 计数单词中的字母
        }

        if (canSpell(wordCount, count, wildcards)) {
            res++; // 如果可以拼写单词，则结果加1
        }
    }

    printf("%d\n", res); // 输出结果

    return 0;
}
```

完整用例

用例1

3
dog
god
odg
gdodog

用例2

2
hello

java
a?llohe

用例3

2
hello
world
abc

用例4

4
cat
bat
rat
drat
abcdrt

用例5

2
cat
dog
cat

用例6

3
cat
car
rat
c?t?r

用例7

2
hello

world
h?llo

用例8

2
cat
dog
???

用例9

4
cat
cat
dog
dog
catdog

用例10

2
hello
world
he

文章目录