

【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

- $1 \leq \text{words.length} \leq 100$
- $1 \leq \text{words}[i].\text{length}, \text{chars.length} \leq 100$
- 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

```
1 | 4
2 | cat
3 | bt
4 | hat
5 | tree
6 | atach??
```

输出

```
1 | 3
```

说明：可以拼写字符串"cat"、“bt"和"hat”

用例2

输入

```
1 | 3
2 | hello
3 | world
4 | cloud
5 | welldonehohneyr
```

输出

```
1 | 2
```

说明：可以拼写字符串"hello"和"world"

用例3

输入

```
1 | 3
2 | apple
3 | car
4 | window
5 | welldoneapplec?
```

输出

```
1 | 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++

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 |
5 | using namespace std;
6 |
```

```

7
8 // 判断是否可以拼写单词
9 bool canSpell(vector<int>& wordCount, vector<int>& count, int wildcards) {
10     for (int i = 0; i < 26; i++) {
11         if (wordCount[i] > count[i]) {
12             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
13             if (wildcards < 0) {
14                 return false; // 如果问号不足, 则返回false
15             }
16         }
17     }
18     return true; // 如果所有的字母都可以匹配, 则返回true
19 }
20
21 int main() {
22     int N;
23     cin >> N; // 读取单词数量
24     cin.ignore(); // 忽略换行符
25
26     vector<string> words(N);
27     for (int i = 0; i < N; i++) {
28         getline(cin, words[i]); // 读取每个单词
29     }
30
31     string chars;
32     getline(cin, chars); // 读取字符集
33
34     vector<int> count(26, 0);
35     int wildcards = 0;
36     for (char c : chars) {
37         if (c != '?') {
38             count[c - 'a']++; // 如果字符不是问号, 则计数
39         } else {
40             wildcards++; // 计算问号的数量
41         }
42     }
43
44     int res = 0;
45     for (string& word : words) {
46         vector<int> wordCount(26, 0);
47

```

```

48     for (char c : word) {
49         wordCount[c - 'a']++; // 计数单词中的字母
50     }
51
52     if (canSpell(wordCount, count, wildcards)) {
53         res++; // 如果可以拼写单词, 则结果加1
54     }
55 }
56
57 cout << res << endl; // 输出结果
58
59 return 0;
60 }

```

Java

```

1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in); // 创建一个扫描器用于读取输入
6          int N = scanner.nextInt(); // 读取输入的的第一个整数, 即单词的数量
7          scanner.nextLine(); // 消耗掉换行符
8
9          // 创建一个字符串数组用于存储所有的单词
10         String[] words = new String[N];
11         for (int i = 0; i < N; i++) {
12             words[i] = scanner.nextLine(); // 读取每一个单词
13         }
14
15         // 读取包含字母和问号的字符串
16         String chars = scanner.nextLine();
17
18         // 创建一个数组用于计数字母的出现次数
19         int[] count = new int[26];
20         for (char c : chars.toCharArray()) {
21             if (c != '?') {
22                 count[c - 'a']++; // 如果字符不是问号, 则计数
23             }
24         }
25
26     }
27 }

```

```

25
26 // 计算问号的数量
27 int wildcards = (int)chars.chars().filter(c -> c == '?').count();
28
29 // 初始化结果为0
30 int res = 0;
31
32 // 遍历所有的单词
33 for (String word : words) {
34     // 创建一个数组用于计数单词中字母的出现次数
35     int[] wordCount = new int[26];
36     for (char c : word.toCharArray()) {
37         wordCount[c - 'a']++; // 计数单词中的字母
38     }
39
40     // 如果可以拼写单词, 则结果加1
41     if (canSpell(wordCount, count, wildcards)) {
42         res++;
43     }
44 }
45
46 // 输出结果
47 System.out.println(res);
48 }
49
50 // 判断是否可以拼写单词
51 private static boolean canSpell(int[] wordCount, int[] count, int wildcards) {
52     for (int i = 0; i < 26; i++) {
53         if (wordCount[i] > count[i]) {
54             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
55             if (wildcards < 0) {
56                 return false; // 如果问号不足, 则返回false
57             }
58         }
59     }
60     return true; // 如果所有的字母都可以匹配, 则返回true
61 }
62 }

```

```
1  const readline = require('readline').createInterface({
2    input: process.stdin,
3    output: process.stdout
4  });
5
6  let input = [];
7  readline.on('line', line => {
8    input.push(line);
9  }).on('close', () => {
10    const N = parseInt(input[0]); // 读取单词数量
11    const words = input.slice(1, N + 1); // 读取单词
12    const chars = input[N + 1]; // 读取字符集
13
14    const count = new Array(26).fill(0);
15    let wildcards = 0;
16    for (let c of chars) {
17      if (c !== '?') {
18        count[c.charCodeAt() - 'a'.charCodeAt()]++; // 如果字符不是问号, 则计数
19      } else {
20        wildcards++; // 计算问号的数量
21      }
22    }
23
24    let res = 0;
25    for (let word of words) {
26      const wordCount = new Array(26).fill(0);
27      for (let c of word) {
28        wordCount[c.charCodeAt() - 'a'.charCodeAt()]++; // 计数单词中的字母
29      }
30
31      if (canSpell(wordCount, count, wildcards)) {
32        res++; // 如果可以拼写单词, 则结果加1
33      }
34    }
35
36    console.log(res); // 输出结果
37  });
38
39  function canSpell(wordCount, count, wildcards) {
40    for (let i = 0; i < 26; i++) {
41
```

```

41     if (wordCount[i] > count[i]) {
42         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
43         if (wildcards < 0) {
44             return false; // 如果问号不足, 则返回false
45         }
46     }
47 }
48 return true; // 如果所有的字母都可以匹配, 则返回true
49 }

```

Python

```

1 def can_spell(word_count, count, wildcards):
2     for i in range(26):
3         if word_count[i] > count[i]:
4             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
5             if wildcards < 0:
6                 return False # 如果问号不足, 则返回false
7     return True # 如果所有的字母都可以匹配, 则返回true
8
9
10 N = int(input()) # 读取单词数量
11 words = [input() for _ in range(N)] # 读取单词
12 chars = input() # 读取字符集
13
14 count = [0] * 26
15 wildcards = chars.count('?') # 计算问号的数量
16 for c in chars:
17     if c != '?':
18         count[ord(c) - ord('a')] += 1 # 如果字符不是问号, 则计数
19
20 res = 0
21 for word in words:
22     word_count = [0] * 26
23     for c in word:
24         word_count[ord(c) - ord('a')] += 1 # 计数单词中的字母
25
26     if can_spell(word_count, count, wildcards):
27         res += 1 # 如果可以拼写单词, 则结果加1
28
29

```



```
print(res) # 输出结果
```

C语言

```

1  #include <stdio.h>
2  #include <string.h>
3
4  // 判断是否可以拼写单词
5  int canSpell(int wordCount[26], int count[26], int wildcards) {
6      for (int i = 0; i < 26; i++) {
7          if (wordCount[i] > count[i]) {
8              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
9              if (wildcards < 0) {
10                 return 0; // 如果问号不足, 则返回0
11             }
12         }
13     }
14     return 1; // 如果所有的字母都可以匹配, 则返回1
15 }
16
17 int main() {
18     int N;
19     scanf("%d", &N); // 读取单词数量
20     getchar(); // 忽略换行符
21
22     char words[N][101];
23     for (int i = 0; i < N; i++) {
24         fgets(words[i], 101, stdin); // 读取每个单词
25         words[i][strcspn(words[i], "\n")] = 0; // 去掉末尾的换行符
26     }
27
28     char chars[101];
29     fgets(chars, 101, stdin); // 读取字符集
30     chars[strcspn(chars, "\n")] = 0; // 去掉末尾的换行符
31
32     int count[26] = {0};
33     int wildcards = 0;
34     for (int i = 0; i < strlen(chars); i++) {
35         if (chars[i] != '?') {
36

```

```

36         count[chars[i] - 'a']++; // 如果字符不是问号, 则计数
37     } else {
38         wildcards++; // 计算问号的数量
39     }
40 }
41
42 int res = 0;
43 for (int i = 0; i < N; i++) {
44     int wordCount[26] = {0};
45     for (int j = 0; j < strlen(words[i]); j++) {
46         wordCount[words[i][j] - 'a']++; // 计数单词中的字母
47     }
48
49     if (canSpell(wordCount, count, wildcards)) {
50         res++; // 如果可以拼写单词, 则结果加1
51     }
52 }
53
54 printf("%d\n", res); // 输出结果
55
56 return 0;
57 }
58

```

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

- 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷
- 题目描述
- 输入描述
- 输出描述
- 备注
- 用例1
- 用例2
- 用例3
- 解题思路

C++

Java

JavaScript

Python

C语言

完整用例

用例1

用例2

用例3

用例4

用例5

用例6

用例7

用例8

用例9

用例10

机考真题 华为OD



CSDN @算法大师

