

【固定滑窗】-考勤信息

题目描述与示例

题目描述

公司用一个字符串来表示员工的出勤信息:

`absent` : 缺勤

`late` : 迟到

`leaveearly` : 早退

`present` : 正常上班

现需根据员工出勤信息，判断本次是否能获得出勤奖，能获得出勤奖的条件如下：

- 缺勤不超过一次
- 没有连续的迟到/早退
- 任意连续 7 次考勤，缺勤/迟到/早退不超过 3 次

输入描述

用户的考勤数据字符串，记录条数 ≥ 1 ;输入字符串长度 < 10000 ;不存在非法输入如：

```
1 2
2 present
3 present absent present present leaveearly present absent
```

输出描述

根据考勤数据字符串，如果能得到考勤奖，输出 `"true"`，否则输出 `"false"`

对于输入示例的结果应为 `true false`

示例

输入

```
1 2
2 present
3 present absent present present leaveearly present absent
```

输出

```
1 true false
```

解题思路

每个人的考勤都是独立的，所以可以构建一个 `check()` 函数来判断每一个人的情况，记录某一个人特定的人是否可以获得全勤奖。

`check()` 函数的构建包含如下步骤

- 判断是否存在 1 天以上的缺勤
- 判断任意连续的 2 天，是否均为迟到/早退
- 判断任意连续的 7 天，是否存在 3 天以上的迟到/早退/缺勤

这三个只要有一个条件成立，直接返回 `False`，否则返回 `True`。

整体而言，题目难度并不高，只是需要对每一个条件进行较为细致的分类讨论和处理。

另外要注意取哈希表的 `key` 的时候，**字符串单词一定不要拼错**。

缺勤数量的判断

缺勤天数的判断非常好做，用哈希表计数器 `Counter()` 即可完成。

```
1 def check(s):
2     lst = s.split()
3
4     cnt = Counter(lst)
5     if cnt["absent"] > 1:
6         return "false"
7
8     pass
```

连续两天的判断

仅需遍历整个 `lst` 数组，考虑连续的两个元素 `lst[i]` 和 `lst[i+1]`，查看是否均为早退/迟到即可。

```
1 def check(s):
2     lst = s.split()
3
4     pass
5
6     n = len(lst)
7     for i in range(n-1):
8         if lst[i] in {"leaveearly", "late"} and lst[i+1] in {"leaveearly",
9             "late"}:
10             return "false"
11
12     pass
```

连续七天的判断

这是这道题里面稍微有点难度的地方，显然我们需要考虑所有的长度为 `7` 的连续区间，统计其中早退/迟到/缺席的天数的和是否超过 `3`。

很容易想到应该使用固定滑窗来实现。构建一个新的哈希表 `cnt_win`，来表示窗口中各个元素出现的次数。

在滑窗过程中，仅需判断 `cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"]` 是否大于 3 即可。如果是，则可以直接返回 `"false"`

```
1 def check(s):
2     lst = s.split()
3
4     pass
5
6     # 第一个窗口
7     cnt_win = Counter(lst[:7])
8     if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
9         return "false"
10
11    # 滑窗过程
12    for right, info_right in enumerate(lst[7:], 7):
13        cnt_win[info_right] += 1
14
15        left = right-7
16        info_left = lst[left]
17        cnt_win[info_left] -= 1
18
19        if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
20            return "false"
```

如果嫌判断 `cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3` 的写法太麻烦，还可以换一个角度来写，考虑正常上班的天数是否小于 4 天，即判断 `cnt_win["present"] < 4`，这样也可以大大缩短代码量。

PS：由于在Python的切片比较灵活，当某个人的考勤总天数不足 7 天时，`cnt_win = Counter(lst[:7])` 并不会产生越界报错。

但对于使用其他语言的同学来说，是需要考虑越界的，可以将遍历的右边界从 7 改为取 7 和 n 之间的较小值，这样就不会出现越界操作。

当 `n <= 7` 时，也自然不会进入后续的固定滑窗过程了。

```
1 HashMap<String, Integer> cnt_win = new HashMap<>();
2 for (int i = 0; i < Math.min(7, n); ++i) {
3     cnt_win.put(lst[i], cnt_win.getDefault(lst[i], 0) + 1);
4 }
```

```
1 unordered_map<string, int> cnt_win;
2 for (int i = 0; i < min(7, n); ++i) {
3     cnt_win[lst[i]]++;
4 }
```

代码

Python

```
1 # 题目：【固定滑窗】2024E-考勤信息
2 # 分值：100
3 # 作者：闭着眼睛学数理化
4 # 算法：固定滑窗
5 # 代码看不懂的地方，请直接在群上提问
6
7
8 from collections import Counter
9
10
11 # 检查某一个人是否可以拿全勤奖的函数check
12 def check(s):
13     # 对s根据空格进行分割，得到数组lst
14     lst = s.split()
15
16     # 构建哈希表计数器，统计缺席次数
17     cnt = Counter(lst)
18
19     # 如果缺席次数超过1次，直接返回"false"
20     if cnt["absent"] > 1:
21         return "false"
22
23     # 计算这个人的天数
24     n = len(lst)
25     # 遍历lst，考虑连续的两
```

```

26     for i in range(n-1):
27         # 如果lst[i]和lst[i+1]均为迟到/早退, 则直接返回"false"
28         if lst[i] in {"leaveearly", "late"} and lst[i+1] in {"leaveearly",
"late"}:
29             return "false"
30
31     # 固定滑窗过程: 考虑任意连续的7天是否存在3天以上的迟到/早退/缺席
32     # 考虑第一个窗口
33     cnt_win = Counter(lst[:7])
34     if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
35         return "false"
36
37     # 滑窗过程
38     for right, info_right in enumerate(lst[7:], 7):
39         # A1
40         cnt_win[info_right] += 1
41         # A2
42         left = right-7
43         info_left = lst[left]
44         cnt_win[info_left] -= 1
45         # A3
46         if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
47             return "false"
48
49     # 如果经过上述判断, 都没有退出函数返回"false", 则返回"true"
50     return "true"
51
52
53 # 输入人数n
54 n = int(input())
55
56 # 构建答案列表
57 ans = list()
58 # 循环n次, 输入每一个人的情况
59 for _ in range(n):
60     s = input()
61     res = check(s)
62     ans.append(res)
63
64 # 输出结果
65 print(*ans)

```

Java

```

1 import java.util.ArrayList;

```

```
2 import java.util.HashMap;
3 import java.util.Scanner;
4
5 public class Main {
6     // 检查某一个人是否可以拿全勤奖的函数check
7     public static String check(String s) {
8         // 对s根据空格进行分割, 得到数组lst
9         String[] lst = s.split(" ");
10
11         // 构建哈希表计数器, 统计缺席次数
12         HashMap<String, Integer> cnt = new HashMap<>();
13         for (String info : lst) {
14             cnt.put(info, cnt.getOrDefault(info, 0) + 1);
15         }
16
17         // 如果缺席次数超过1次, 直接返回"false"
18         if (cnt.containsKey("absent") && cnt.get("absent") > 1) {
19             return "false";
20         }
21
22         // 计算这个人的天数
23         int n = lst.length;
24         // 遍历lst, 考虑连续的天数
25         for (int i = 0; i < n - 1; ++i) {
26             // 如果lst[i]和lst[i+1]均为迟到/早退, 则直接返回"false"
27             if ((lst[i].equals("leaveearly") || lst[i].equals("late")) &&
28                 (lst[i + 1].equals("leaveearly") || lst[i +
29 1].equals("late")))) {
30                 return "false";
31             }
32         }
33
34         // 固定滑动过程: 考虑任意连续的7天是否存在3天以上的迟到/早退/缺席
35         // 考虑第一个窗口
36         HashMap<String, Integer> cnt_win = new HashMap<>();
37         // 需要总天数考虑不足7天的情况, 所以取右区间为Math.min(7, n)
38         for (int i = 0; i < Math.min(7, n); ++i) {
39             cnt_win.put(lst[i], cnt_win.getOrDefault(lst[i], 0) + 1);
40         }
41         if (cnt_win.getOrDefault("leaveearly", 0) +
42             cnt_win.getOrDefault("late", 0) + cnt_win.getOrDefault("absent", 0) > 3) {
43             return "false";
44         }
45
46         // 滑动过程
47         for (int right = 7; right < n; ++right) {
48             // A1
```

```

47         cnt_win.put(lst[right], cnt_win.getDefault(lst[right], 0) + 1);
48         // A2
49         int left = right - 7;
50         cnt_win.put(lst[left], cnt_win.getDefault(lst[left], 0) - 1);
51         // A3
52         if (cnt_win.getDefault("leaveearly", 0) +
cnt_win.getDefault("late", 0) + cnt_win.getDefault("absent", 0) > 3) {
53             return "false";
54         }
55     }
56
57     // 如果经过上述判断, 都没有退出函数返回"false", 则返回"true"
58     return "true";
59 }
60
61 public static void main(String[] args) {
62     Scanner scanner = new Scanner(System.in);
63
64     // 输入人数n
65     int n = scanner.nextInt();
66     scanner.nextLine(); // 消耗换行符
67
68     // 构建答案列表
69     ArrayList<String> ans = new ArrayList<>();
70     // 循环n次, 输入每一个人的情况
71     for (int i = 0; i < n; ++i) {
72         String s = scanner.nextLine();
73         String res = check(s);
74         ans.add(res);
75     }
76
77     // 输出结果
78     for (String result : ans) {
79         System.out.print(result + " ");
80     }
81     System.out.println();
82 }
83 }
84

```

C++

```

1 #include <iostream>
2 #include <unordered_map>
3 #include <vector>

```



```

4 #include <string>
5
6 using namespace std;
7
8 // 检查某一个人是否可以拿全勤奖的函数check
9 string check(string s) {
10     // 对s根据空格进行分割, 得到数组lst
11     vector<string> lst;
12     string word = "";
13     for (char ch : s) {
14         if (ch == ' ') {
15             lst.push_back(word);
16             word = "";
17         } else {
18             word += ch;
19         }
20     }
21     lst.push_back(word);
22
23     // 构建哈希表计数器, 统计缺席次数
24     unordered_map<string, int> cnt;
25     for (string info : lst) {
26         cnt[info]++;
27     }
28
29     // 如果缺席次数超过1次, 直接返回"false"
30     if (cnt.find("absent") != cnt.end() && cnt["absent"] > 1) {
31         return "false";
32     }
33
34     // 计算这个人的天数
35     int n = lst.size();
36     // 遍历lst, 考虑连续的天数
37     for (int i = 0; i < n - 1; ++i) {
38         // 如果lst[i]和lst[i+1]均为迟到/早退, 则直接返回"false"
39         if ((lst[i] == "leaveearly" || lst[i] == "late") &&
40             (lst[i + 1] == "leaveearly" || lst[i + 1] == "late")) {
41             return "false";
42         }
43     }
44
45     // 固定滑窗过程: 考虑任意连续的7天是否存在3天以上的迟到/早退/缺席
46     // 考虑第一个窗口
47     unordered_map<string, int> cnt_win;
48     // 需要总天数考虑不足7天的情况, 所以取右区间为min(7, n)
49     for (int i = 0; i < min(7, n); ++i) {
50         cnt_win[lst[i]]++;

```

```

51     }
52     if (cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3) {
53         return "false";
54     }
55
56     // 滑窗过程
57     for (int right = 7; right < n; ++right) {
58         // A1
59         cnt_win[lst[right]]++;
60         // A2
61         int left = right - 7;
62         cnt_win[lst[left]]--;
63         // A3
64         if (cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3) {
65             return "false";
66         }
67     }
68
69     // 如果经过上述判断，都没有退出函数返回"false"，则返回"true"
70     return "true";
71 }
72
73 int main() {
74     // 输入人数n
75     int n;
76     cin >> n;
77     cin.ignore(); // 消耗换行符
78
79     // 构建答案列表
80     vector<string> ans;
81     // 循环n次，输入每一个人的情况
82     for (int i = 0; i < n; ++i) {
83         string s;
84         getline(cin, s);
85         string res = check(s);
86         ans.push_back(res);
87     }
88
89     // 输出结果
90     for (string result : ans) {
91         cout << result << " ";
92     }
93     cout << endl;
94
95     return 0;
96 }
97

```

时空复杂度

时间复杂度： $O(NM)$ 。 M 为每一个字符串数组的长度

空间复杂度： $O(M)$ 。