【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：

```
2
present
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

```
2
present
present absent present present leaveearly present absent
```

输出

```
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。

```
def check(s):
    lst = s.split()

    cnt = Counter(lst)
    if cnt["absent"] > 1:
        return "false"

    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。

```
def check(s):
    lst = s.split()

    pass

    n = len(lst)
    for i in range(n-1):
        if lst[i] in {"leaveearly", "late"} and lst[i+1] in {"leaveearly", "late"}:
            return "false"

    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"`

```python
def check(s):
    lst = s.split()

    pass

    # 第一个窗口
    cnt_win = Counter(lst[:7])
    if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
        return "false"

    # 滑窗过程
    for right, info_right in enumerate(lst[7:], 7):
        cnt_win[info_right] += 1

        left = right-7
        info_left = lst[left]
        cnt_win[info_left] -= 1

        if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
            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。

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

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

代码

Python

```python
# 题目：【固定滑窗】2024E-考勤信息
# 分值：100
# 作者：闭着眼睛学数理化
# 算法：固定滑窗
# 代码看不懂的地方，请直接在群上提问


from collections import Counter


# 检查某一个人是否可以拿全勤奖的函数check
def check(s):
    # 对s根据空格进行分割，得到数组lst
    lst = s.split()

    # 构建哈希表计数器，统计缺席次数
    cnt = Counter(lst)

    # 如果缺席次数超过1次，直接返回"false"
    if cnt["absent"] > 1:
        return "false"

    # 计算这个人的天数
    n = len(lst)
    # 遍历lst，考虑连续的两天
```

```
    for i in range(n-1):
        # 如果lst[i]和lst[i+1]均为迟到/早退，则直接返回"false"
        if lst[i] in {"leaveearly", "late"} and lst[i+1] in {"leaveearly",
"late"}:
            return "false"

    # 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
    # 考虑第一个窗口
    cnt_win = Counter(lst[:7])
    if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
        return "false"

    # 滑窗过程
    for right, info_right in enumerate(lst[7:], 7):
        # A1
        cnt_win[info_right] += 1
        # A2
        left = right-7
        info_left = lst[left]
        cnt_win[info_left] -= 1
        # A3
        if cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3:
            return "false"

    # 如果经过上述判断，都没有退出函数返回"false"，则返回"true"
    return "true"


# 输入人数n
n = int(input())

# 构建答案列表
ans = list()
# 循环n次，输入每一个人的情况
for _ in range(n):
    s = input()
    res = check(s)
    ans.append(res)

# 输出结果
print(*ans)
```

Java

```
import java.util.ArrayList;
```

```java
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    // 检查某一个人是否可以拿全勤奖的函数check
    public static String check(String s) {
        // 对s根据空格进行分割，得到数组lst
        String[] lst = s.split(" ");

        // 构建哈希表计数器，统计缺席次数
        HashMap<String, Integer> cnt = new HashMap<>();
        for (String info : lst) {
            cnt.put(info, cnt.getOrDefault(info, 0) + 1);
        }

        // 如果缺席次数超过1次，直接返回"false"
        if (cnt.containsKey("absent") && cnt.get("absent") > 1) {
            return "false";
        }

        // 计算这个人的天数
        int n = lst.length;
        // 遍历lst，考虑连续的两天
        for (int i = 0; i < n - 1; ++i) {
            // 如果lst[i]和lst[i+1]均为迟到/早退，则直接返回"false"
            if ((lst[i].equals("leaveearly") || lst[i].equals("late")) &&
                    (lst[i + 1].equals("leaveearly") || lst[i +
1].equals("late"))) {
                return "false";
            }
        }

        // 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
        // 考虑第一个窗口
        HashMap<String, Integer> cnt_win = new HashMap<>();
        // 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)
        for (int i = 0; i < Math.min(7, n); ++i) {
            cnt_win.put(lst[i], cnt_win.getOrDefault(lst[i], 0) + 1);
        }
        if (cnt_win.getOrDefault("leaveearly", 0) +
cnt_win.getOrDefault("late", 0) + cnt_win.getOrDefault("absent", 0) > 3) {
            return "false";
        }

        // 滑窗过程
        for (int right = 7; right < n; ++right) {
            // A1
```

```
            cnt_win.put(lst[right], cnt_win.getOrDefault(lst[right], 0) + 1);
            // A2
            int left = right - 7;
            cnt_win.put(lst[left], cnt_win.getOrDefault(lst[left], 0) - 1);
            // A3
            if (cnt_win.getOrDefault("leaveearly", 0) +
cnt_win.getOrDefault("late", 0) + cnt_win.getOrDefault("absent", 0) > 3) {
                return "false";
            }
        }

        // 如果经过上述判断，都没有退出函数返回"false"，则返回"true"
        return "true";
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 输入人数n
        int n = scanner.nextInt();
        scanner.nextLine(); // 消耗换行符

        // 构建答案列表
        ArrayList<String> ans = new ArrayList<>();
        // 循环n次，输入每一个人的情况
        for (int i = 0; i < n; ++i) {
            String s = scanner.nextLine();
            String res = check(s);
            ans.add(res);
        }

        // 输出结果
        for (String result : ans) {
            System.out.print(result + " ");
        }
        System.out.println();
    }
}

```

C++

```
#include <iostream>
#include <unordered_map>
#include <vector>
```

```cpp
#include <string>

using namespace std;

// 检查某一个人是否可以拿全勤奖的函数check
string check(string s) {
    // 对s根据空格进行分割，得到数组lst
    vector<string> lst;
    string word = "";
    for (char ch : s) {
        if (ch == ' ') {
            lst.push_back(word);
            word = "";
        } else {
            word += ch;
        }
    }
    lst.push_back(word);

    // 构建哈希表计数器，统计缺席次数
    unordered_map<string, int> cnt;
    for (string info : lst) {
        cnt[info]++;
    }

    // 如果缺席次数超过1次，直接返回"false"
    if (cnt.find("absent") != cnt.end() && cnt["absent"] > 1) {
        return "false";
    }

    // 计算这个人的天数
    int n = lst.size();
    // 遍历lst，考虑连续的两天
    for (int i = 0; i < n - 1; ++i) {
        // 如果lst[i]和lst[i+1]均为迟到/早退，则直接返回"false"
        if ((lst[i] == "leaveearly" || lst[i] == "late") &&
            (lst[i + 1] == "leaveearly" || lst[i + 1] == "late")) {
            return "false";
        }
    }

    // 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
    // 考虑第一个窗口
    unordered_map<string, int> cnt_win;
    // 需要总天数考虑不足7天的情况，所以取右区间为min(7, n)
    for (int i = 0; i < min(7, n); ++i) {
        cnt_win[lst[i]]++;
```

```cpp
    }
    if (cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3) {
        return "false";
    }

    // 滑窗过程
    for (int right = 7; right < n; ++right) {
        // A1
        cnt_win[lst[right]]++;
        // A2
        int left = right - 7;
        cnt_win[lst[left]]--;
        // A3
        if (cnt_win["leaveearly"] + cnt_win["late"] + cnt_win["absent"] > 3) {
            return "false";
        }
    }

    // 如果经过上述判断，都没有退出函数返回"false"，则返回"true"
    return "true";
}

int main() {
    // 输入人数n
    int n;
    cin >> n;
    cin.ignore(); // 消耗换行符

    // 构建答案列表
    vector<string> ans;
    // 循环n次，输入每一个人的情况
    for (int i = 0; i < n; ++i) {
        string s;
        getline(cin, s);
        string res = check(s);
        ans.push_back(res);
    }

    // 输出结果
    for (string result : ans) {
        cout << result << " ";
    }
    cout << endl;

    return 0;
}

```

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)`。