链滴

PAT 甲级刷题实录——1017

作者：aopstudio
原文链接：https://ld246.com/article/1580449685444
来源网站：链滴

原题链接

https://pintia.cn/problem-sets/994805342720868352/problems/994805491530579968

思路

这题也出现了时间，我们根据上一题的经验同样把时分秒转换为自零点零分零秒开始经过的秒数。这也出现了窗口排队，之前也有题目可以参考，而且这次每个窗口最多排一个，更加简单。另外，之前过的 sort 排序这里也有用处。本来我的想法是先填满所有窗口，将没排进窗口的统一放在一个队列存储，但这个方法比较复杂，而且容易出错。参考了网上的一个方法，将 17 点之后到的在输入时就除掉，而之前到的全部放在一个 vector 中，之后遍历 vector，同时遍历查找每个窗口完成当前接待最早时间，之后进行更新。

代码

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
struct record
{
    int time, process;    //time代表到达时间，为从00:00:00开始经过的秒数
};
bool compare(record a, record b)
{
    return a.time < b.time;
}
int main()
{
    const int EIGHT = 8 * 3600;    //八点代表的秒数
    const int SEVENTEEN = 17 * 3600;    //十七点代表的秒数
    int N, K;
    double totalTime = 0.0;    //所有顾客的等待时间
    cin >> N >> K;
    vector<record> records;    //存储所有记录
    for (int i = 0; i < N; i++)
    {
        record newRecord;
        int hour, minute, second;
        scanf("%d:%d:%d %d", &hour, &minute, &second, &newRecord.process);
        newRecord.time = hour * 3600 + minute * 60 + second;
        if (newRecord.time > SEVENTEEN)    //如果晚于17：00则不计入
            continue;
        newRecord.process *= 60;    //分钟转换为秒
        records.push_back(newRecord);    //添加新记录
    }
    sort(records.begin(), records.end(), compare);
    vector<int> linesTime(K, EIGHT);    //每个窗口完成当前接待的时间,初始化都为8点
    for (int i = 0; i < records.size(); i++)
    {
        int min = linesTime[0], index = 0;
        for (int i = 1; i < K; i++)    //找出最早完成前一个处理的窗口
```

```
        {
            if (linesTime[i] < min)
            {
                min = linesTime[i];
                index = i;
            }
        }
        record r = records[i];
        if (linesTime[index] <= r.time)    //来的时间晚于窗口完成前一个处理的时间
        {
            linesTime[index] = r.time + r.process;    //到达时间加处理时间为窗口完成当前处理的时间
等待时间为0，无需更新
        }
        else //来的时间早于窗口完成前一个处理的时间
        {
            totalTime += linesTime[index] - r.time;    //等待时间加上窗口完成当前处理的时间减到达
间
            linesTime[index] += r.process;    //窗口完成当前处理的时间为完成前一个处理的时间加处
时间
        }
    }
    if (records.size() == 0)    //没有符合的条件，直接输出0.0，因为0不能作除数
        cout << 0.0;
    else
    {
        double average = totalTime / 60.0 / records.size();    //计算平均等待时间（单位为分钟）
        printf("%.1lf", average);
    }
    return 0;
}
```