贪心习题

TEACHING COURSEWARE POWPOINT

授课时间：20XX.01.01

目录

01

题目描述

TEACHING
COURSEWARE

TEACH

通用模版

题目描述

PVZ 这款游戏中，有一种坚果保龄球。zombie 从地图右侧不断出现，向左走，玩家需要从左侧滚动坚果来碾死他们。

我们可以认为地图是一个行数为 6，列数为 60 的棋盘。zombie 出现的那一秒站在这一行的第 60 列，之后每秒向左移动一步。玩家可以随时在屏幕最某一行第一列摆放坚果，这一行的 zombie 瞬间全被滚过去的坚果碾死。如果 zombie 走到第 1 列没有被消灭，如果再向左走，则你的大脑就会被 zombie 吃掉。

现在有 n 只 zombie！告诉你每只 zombie 出现的时间以及在出现的行数（可能会同时出现同一位置的僵尸），请问至少需要多少坚果才能消灭所有的 zombie。

输入格式

第一行一个正整数 n，表示 zombie 数量。

之后 n 行中，每行两个正整数 P 和 t，分别表示 zombie 所在行和 zombie 出现的时间。

输出格式

一个正整数，最少需要的坚果数。

样例输入：

3

1 1

1 61

2 10

样例输出：

3

02

知识回归

TEACHING
COURSEWARE

TEACH

知识回归

贪心，不考虑未来，不后悔之前，
总结成四个字的性质：

03

解题思路

TEACHING
COURSEWARE

TEACH

解题思路

从 “僵尸移动” 的角度直接理解，最合适的放坚果时机是：在当前需要处理的僵尸即将走到最左端（第 1 列）的那一刻放坚果，这样能最大限度覆盖之后出现的僵尸。

解题思路

也就是说，只有在第t到t+59秒之间放坚果，才能打到这个僵尸。早于t秒放（僵尸还没出现）、晚于t+59秒放（僵尸已逃走）都没用。

解题思路

问题就变成了：**选最少的时间点x，让每个僵尸的有效区间[t，t+59]都包含至少一个x**

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include<bits/stdc++.h>
using namespace std;
// 定义结构体存储每个僵尸的有效时间区间
struct node{
    int l;  // 区间左端点：僵尸出现的时间t（最早可放置坚果的时间）
    int r;  // 区间右端点：t+59（最晚可放置坚果的时间，此时僵尸刚到第1列）
};
// 定义二维数组存储6行的僵尸区间（行号1-6，每行最多2000个僵尸）
node a[7][2005];
// 排序规则：按区间右端点（r）从小到大排序
// 这是贪心策略的关键，保证我们优先处理结束早的区间
bool cmp(node a,node b) {
    return a.r < b.r;
}
```

标准程序


```
int main() {
    cin >> n;  // 读入僵尸总数
    // 循环读入每个僵尸的信息，按行存储区间
    for(int i = 1 ; i <= n ; i++) {
        int t1, t2;  // t1：僵尸所在行（1-6）；t2：僵尸出现的时间
        cin >> t1 >> t2;
        xiabiao[t1]++;  // 第t1行的僵尸数量加1
        int XiaBiao = xiabiao[t1];  // 当前僵尸在该行的序号（从1开始）
        // 计算并存储当前僵尸的有效区间
        a[t1][XiaBiao].l = t2;       // 左端点=出现时间t2
        a[t1][XiaBiao].r = t2 + 59; // 右端点=出现时间+59（走到第1列的时间）
    }
    // 对每行的僵尸区间按右端点从小到大排序
    // 排序后，区间按结束时间从早到晚排列，便于贪心选点
    for(int i = 1 ; i <= 6 ; i++) {
        sort(a[i] + 1, a[i] + 1 + xiabiao[i], cmp);
    }
```

标准程序


```
int ans = 0;  // 总坚果数量（最终答案）

// 按行处理，每行单独计算所需坚果数
for(int i = 1 ; i <= 6 ; i++) {
    if(xiabiao[i] == 0) continue;  // 该行没有僵尸，直接跳过

    // 初始化：第一个坚果放在第1个区间的右端点（贪心选点的起点）
    int last = a[i][1].r;  // 记录上一个坚果的放置时间（覆盖到的位置）
    int cnt = 1;           // 该行的坚果数量，至少需要1个

    // 从第2个区间开始遍历该行所有僵尸
    for(int j = 2 ; j <= xiabiao[i] ; j++) {
        // 如果当前区间的左端点 > 上一个坚果的时间，说明上一个坚果覆盖不到
        if(a[i][j].l > last) {
            cnt++;                  // 新增一个坚果
            last = a[i][j].r;       // 放在当前区间的右端点（最大化覆盖范围）
        }
    }

    ans += cnt;  // 累加该行的坚果数到总数
}

cout << ans << endl;  // 输出最少需要的坚果总数
return 0;
```