贪心习题

TEACHING COURSEWARE POWPOINT

授课时间：20XX.01.01

目录

01

题目描述

TEACHING
COURSEWARE

TEACH

题目描述

题目背景

一条街的一边有几座房子，因为环保原因居民想要在路边种些树。

题目描述

路边的地区被分割成块，并被编号成 $1, 2, \ldots, n$。每个部分为一个单位尺寸大小并最多可种一棵树。

每个居民都想在门前种些树，并指定了三个号码 b，e，t。这三个数表示该居民想在地区 b 和 e 之间（包括 b 和 e）种至少 t 棵树。

居民们想种树的各自区域可以交叉。你的任务是求出能满足所有要求的最少的树的数量。

输入格式

输入的第一行是一个整数，代表区域的个数 n。

输入的第二行是一个整数，代表房子个数 h。

第 3 到第 $(h+2)$ 行，每行三个整数，第 $(i+2)$ 行的整数依次为 b_i, e_i, t_i，代表第 i 个居民想在 b_i 和 e_i 之间种至少 t_i 棵树。

题目描述

输出格式

输出一行一个整数，代表最少的树木个数。

输入输出样例

输入 #1 复制

```
9
4
1 4 2
4 6 2
8 9 2
3 5 2
```

输出 #1 复制

```
5
```

说明/提示

数据规模与约定

对于 100% 的数据，保证：

- $1 \le n \le 3 \times 10^4$，$1 \le h \le 5 \times 10^3$。
- $1 \le b_i \le e_i \le n$，$1 \le t_i \le e_i - b_i + 1$。

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

贪心，要种树种得少，就要使一棵树给多个区间使用，这样，尽量在重叠区间种树即可，而重叠位置一定是区间尾部。处理问题时，先按所有区间的结束位置从小到大排序，若结束位置相同，则按开始位置从大到小排序。之后依次处理每个区间，先在第一个区间尾部种满足要求的树，对下一个区间，看差多少棵就在该区间尾部种多少。

步骤：

①先按照b[]从小到大排序

②对每个区间依次处理

a. 从前到后扫描这个区间，统计点的个数；

b. 若没有超过要求的点数，则从该区间后向前扫描，添加覆盖点。

③输出ans

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include <bits/stdc++.h>
using namespace std;
int n, h, ans = 0;  // n: 区域总数; h: 居民数量; ans: 最少树的数量
struct stu {
    int b, e, t;  // 每个居民的要求: [b,e]区间至少种t棵树
} a[5005];
int planted[30005] = {0};  // 记录每个位置的树的数量（0=未种，1=已种）
// 按区间结束位置e从小到大排序（优先处理结束早的区间，树尽量种在右侧）
bool cmp(stu x, stu y) {
    return x.e < y.e;
}
```

标准程序


```
int main() {
    cin >> n >> h;
    for (int i = 1; i <= h; i++) {
        cin >> a[i].b >> a[i].e >> a[i].t;
    }
    sort(a + 1, a + h + 1, cmp);
    for (int i = 1; i <= h; i++) {
        int b = a[i].b;
        int e = a[i].e;
        int t = a[i].t;
        // 统计当前区间内已种的树（直接累加planted数组的值）
        int cnt = 0;
        for (int x = b; x <= e; x++) {
            cnt += planted[x];  // 因为planted[x]只能是0或1，累加即总数
        }
```

标准程序


```
        // 计算还需要种的树
        int need = t - cnt;
        if (need <= 0) continue;
        // 从右侧向左补种
        for (int x = e; x >= b && need > 0; x--) {
            if (planted[x] == 0) {  // 该位置未种树
                planted[x] = 1;     // 种一棵树
                ans++;
                need--;
            }
        }
    }
    cout << ans;
    return 0;
}
```