目录

The 17th UESTC Fun Programming Contest Round 1 1
A.广告 1
B.十二平均律 3
C.魔法 6
D.三相电 9
E.星际导演 12
F.奶奶茶店的神秘排单 15

The 17th UESTC Fun Programming Contest Round 1

A.广告

题目描述

夏美子喜欢看广告。

一天，杏里在樱之丘高等学校的电梯显示屏上投放了广告，并热情地邀请夏美子测试广告的投放效果。

杏里一共投放了 n 种不同的广告，编号为 $1, 2, ..., n$。显示屏会按照编号顺序依次播放这些广告，第 n 条播完后从第 1 条重新开始。每次播放会恰好持续 m 分钟，广告切换之间没有间隔。

夏美子可以选择在任意一条广告**恰好开始播放**的时刻走进电梯，连续观看广告恰好 k 分钟后立即离开。

以她进入电梯的时刻为 0，她的观看时段为左闭右开区间 $[0, k)$，因此对于夏美子离开时恰好开始播放的广告，我们认为她**没有看过**。只要她在这段时间内看到了某条广告的一部分，无论是否看完都算看过这条广告。

为了更好地测试广告的投放效果，夏美子希望看到的广告**种类数**尽可能多。夏美子想知道，对于这 n 种广告，她最多能看到其中的多少种。

输入格式

本题有**多组测试数据**。

第一行输入一个正整数 T $(1 \le T \le 2 \times 10^5)$，表示测试数据组数。

接下来 T 行分别表示 T 组测试数据，对于每组测试数据：

- 第一行输入三个正整数 n, m, k $(1 \le n, m, k \le 10^9)$，表示广告种类数、一条广告播放的时长与夏美子的总观看时长。

输出格式

共输出 T 行。

对于每组测试数据，输出一个正整数，表示夏美子最多能看到多少种广告。

样例

1.in / 1.out

```
1 5
2 5 2 7
3 5 2 8
4 5 2 11
5 4 2 1
6 6 3 8
```

```
4
4
5
1
3
```

注释

对于第一组测试数据，一种方案是选择在广告 3 开始播放的时刻走进电梯，这样她会依次观看到编号为 $3,4,5,1$ 的广告，并在广告 1 播放了恰好 1 分钟时立即离开。这样夏美子可以看到 4 种不同的广告。

对于第二组测试数据，一种方案是选择在广告 1 开始播放的时刻走进电梯，这样她会依次观看到编号为 $1,2,3,4$ 的广告，并在广告 4 播放完时立即离开，注意由于夏美子离开时广告 5 恰好开始播放，因此我们认为夏美子**没有看过**这条广告。这样夏美子可以看到 4 种不同的广告。

对于第三组测试数据，一种方案是选择在广告 2 开始播放的时刻走进电梯，这样她会依次观看到编号为 $2,3,4,5,1,2$ 的广告，并在广告 2 **再次**播放了恰好 1 分钟时立即离开。这样夏美子可以看到 5 种不同的广告。

题解

status: `6 +2`

首先有一个傻子这题吃了 40min 的罚时。

显然答案就是 $\min\left(\left\lceil\frac{m}{k}\right\rceil,n\right)$，所以打出来就结束了……

cpp
```cpp
#include <iostream>
#include <map>
#include <stdio.h>
#include <string>
#include <vector>
typedef long long i64;
typedef __int128 i128;

#define maxn 200007

int T;
int n, m, k;

inline int max(int a, int b) { return a > b ? a : b; }
inline int min(int a, int b) { return a < b ? a : b; }

int main() {
    scanf("%d", &T);
    while (T--) {
        scanf("%d%d%d", &n, &m, &k);
        printf("%d\n", min(n, (k + m - 1) / m));
    }
    return 0;
}
```

B.十二平均律

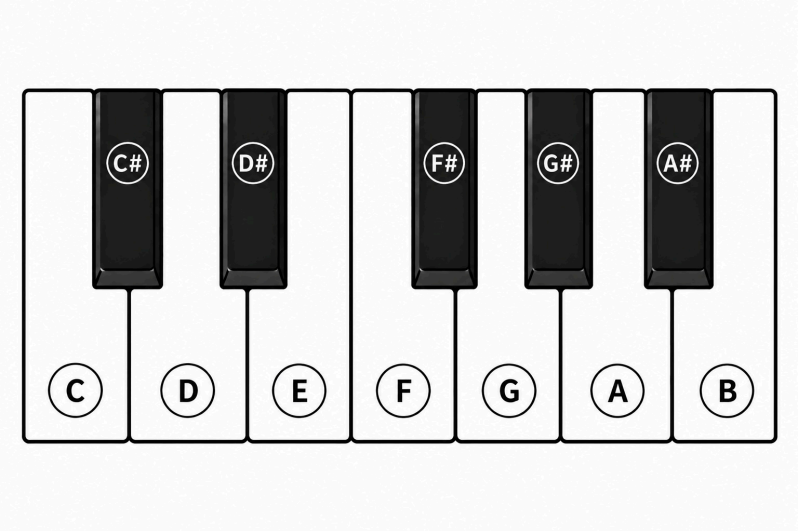
题目描述

音乐具有优美的数学性质!

你听到了一段旋律，并且清晰地分辨出了这段旋律的音高，为了分析这段旋律为何动听，你想知道在某一个调式下，每个音符各自的音级。

为了便于理解与解题，本题所涉及的概念均高度简化，可能与专业的乐理知识存在差异，请以本题的描述为准。给出以下定义：

如下图所示，有一个局部的钢琴键盘，总共有 7 个白键，5 个黑键，从左到右依次为：白-黑-白-黑-白-白-黑-白-黑-白-黑-白。

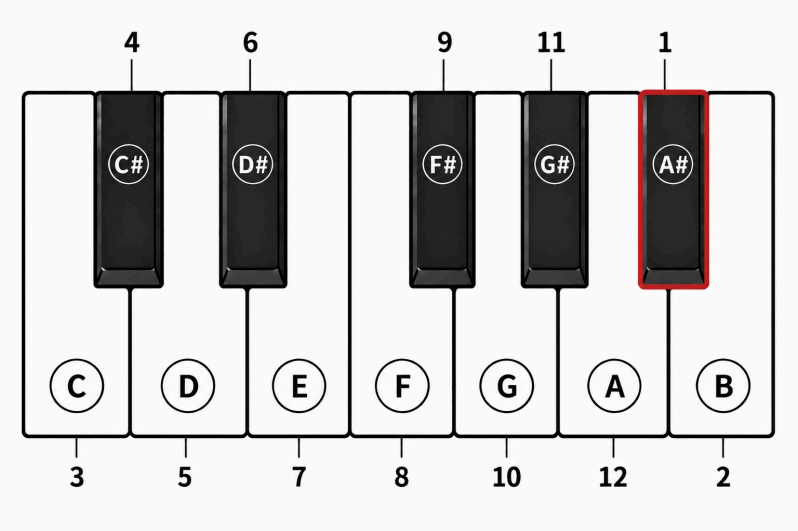


音高类别：由 1 至 2 个字符表示（其中，如果有第 2 个字符，则它一定为 #）。**音高类别**与琴键一一对应，已在图中标注，12 个琴键对应的**音高类别**，从左到右依次为：C, C#, D, D#, E, F, F#, G, G#, A, A#, B。

注意，在本题中，仅有上述的 12 种合法**音高类别**，例如，E#, B#, H 等字符串，在本题都**不是**合法的**音高类别**。

X 自然大调：其中，X 为一个**音高类别**。考虑为所有琴键编号：将 12 个琴键看作一个环（即对于最右侧的琴键，它的右侧相邻琴键为最左侧的琴键），从 X 对应的琴键开始，向右依次重新编号为 1 至 12。我们认为这一组新的编号，代表了 **X 自然大调**。

以 A# **自然大调**为例，参考下图：首先，找到 A# 对应的琴键（即图中红色描边的黑键），将其编号为 1；其右侧琴键为 B，将其编号为 2；此时已经达到了键盘的最右侧，因此从最左侧的琴键继续编号，将 C 对应的琴键编号为 3；继续向右，直到所有琴键完成编号。



音级：由一个字符串表示。在规定了 **X 自然大调**的前提下，每个**音高类别**对应着一个编号，据编号对应着一个**音级**，对应关系如下表所示：

编号	1	2	3	4	5	6	7	8	9	10	11	12
音级	I	#I	II	#II	III	IV	#IV	V	#V	VI	#VI	VII

应特别注意**音高类别**和**音级**的格式，且本题涉及的英文字母均为大写。

输入格式

第一行一个整数 N $(1 \leq N \leq 10^5)$，表示音符总数。

第二行一个**音高类别** X，表示规定的 **X 自然大调**。

第三行 N 个**音高类别**，用空格分隔，你需要计算它们依次对应的**音级**。

输出格式

一行 N 个字符串，中间用空格分隔，表示这 N 个**音高类别**依次对应的**音级**。音级的格式参考上述表格。

请注意，为了便于排版，下面 3 个样例的输入、输出都有额外的换行，在实际测试数据中则**不存在**这些换行。具体地，样例输入的第 4 行实际为第 3 行的后续，样例输出的第 2 行实际为第 1 行的后续。

样例

1.in / 1.out

```
20
A#
G D A# A G A# A G F# A D
G D A D A# A G A D
```

```
VI III I VII VI I VII VI #V VII III
VI III VII III I VII VI VII III
```

2.in / 2.out

```
21
A#
F D F D C A# A# A# D F
```

```
4 G G G F D A# D# D D# G F
```

```
1 V III V III II I I I III V
2 VI VI VI V III I IV III IV VI V
```

3.in / 3.out

```
1 24
2 A
3 C# E F# E C# B B B A B E C#
4 C# E F# E C# D D D C# D F# G#
```

```
1 III V VI V III II II II I II V III
2 III V VI V III IV IV IV III IV VI VII
```

注释

样例解释：

对于样例一：规定在 A# **自然大调**下分析音级，此时，A# 音级为 I，B 音级为 #I，C 音级为 II，C# 音级为 #II，D 音级为 III，其余音符以此类推。因此，前 5 个音符的音级依次为：VI, III, I, VII, VI。

另外，乐谱中的 F# 音符的音级为 #V。

对于样例二：同样在 A# **自然大调**下分析音级，前 5 个音符的音级依次为：V, III, V, III, II。

对于样例三：规定在 A **自然大调**下分析音级，此时，A 音级为 I，A# 音级为 #I，B 音级为 II，C 音级为 #II，C# 音级为 III，其余音符以此类推。

样例一节选自 J.S.Bach 的 BWV578 的主题乐句。样例二、三节选自真实歌曲的乐谱片段。

题解

一道模拟，考察建立映射。

直接使用 `map<string, int>` 和 `map<int, string>` 来映射音高类别和音级。

```cpp
#include <iostream>
#include <map>
#include <string>

#define maxn

std::map<std::string, int> m;
std::map<int, std::string> mp;

std::string b, x, r;

int n;
int main() {

    m["C"] = 0, m["C#"] = 1, m["D"] = 2, m["D#"] = 3, m["E"] = 4, m["F"] = 5,
    m["F#"] = 6, m["G"] = 7, m["G#"] = 8, m["A"] = 9, m["A#"] = 10, m["B"] = 11;

    mp[1] = "I", mp[2] = "#I", mp[3] = "II", mp[4] = "#II", mp[5] = "III",
```

```
    mp[6] = "IV", mp[7] = "#IV", mp[8] = "V", mp[9] = "#V", mp[10] = "VI",
    mp[11] = "#VI", mp[12] = "VII";

    std::cin >> n >> b;
    int base = m[b];
    for (int i = 1; i <= n; i++) {
        std::cin >> r;
        int t = (m[r] + 12 - base) % 12 + 1;
        std::cout << mp[t] << " ";
    }

    return 0;
}
```

C.魔法

题目描述

千代田桃喜欢研究魔法。

桃的魔法阵可以看作一个长度为 n 的正整数序列 $a_1, a_2, ..., a_n$，其中第 i 个位置的魔力值为 a_i。

桃对这个魔法阵做了 q 次测试。在一次测试中，桃会指定三个参数 l, r, p，对于完全包含在 $[l, r]$ 内的每个非空子区间 $[i, j]$，桃计算其中所有位置的魔力值之和，将这些和相乘，再对 2^p 取模[1]，即为此次测试的效果值。形式化地表述，单次测试的效果值 $f(l, r, p)$ 定义为：

$$f(l,r,p) = \left(\prod_{l \le i \le j \le r} \left(\sum_{k=i}^{j} a_k \right) \right) \bmod 2^p$$

遗憾的是桃不小心弄丢了测试结果，只有序列 a 和每次测试的参数保留了下来。桃希望你依据这些信息，按顺序还原出所有测试的效果值。

另外，对于使用 C/C++/Java 语言进行编程的选手，为了避免在运算过程中出现数据溢出的问题，善良的桃建议您使用 **64 位整型变量**来存储数据并进行中间运算。

输入格式

第一行输入两个正整数 n, q $(1 \le n, q \le 2 \times 10^5)$，表示序列 a 的长度和总测试次数。

第二行输入 n 个正整数 $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^9)$，表示序列 a。

接下来 q 行分别表示 q 次测试。其中的第 i 行包含三个正整数 l_i, r_i, p_i $(1 \le l_i \le r_i \le n, 1 \le p_i \le 30)$，表示第 i 次测试的参数。

输出格式

共输出 q 行，第 i 行输出一个非负整数，表示第 i 次测试的效果值。

---

[1] a 对 b 取模，即 $a \bmod b$，表示 a 除以 b 所得的余数。例如：

- $7 \bmod 3 = 1$，因为 $7 = 3 \times 2 + 1$；
- $15 \bmod 4 = 3$，因为 $15 = 4 \times 3 + 3$；
- $21 \bmod 3 = 0$，因为 $21 = 3 \times 7 + 0$。

同时，模运算对加法和乘法满足以下性质：

- $(a + b) \bmod M = ((a \bmod M) + (b \bmod M)) \bmod M$
- $(a \times b) \bmod M = ((a \bmod M) \times (b \bmod M)) \bmod M$

因此您可以在运算过程中通过及时取模来避免数据溢出问题。

样例

1.in / 1.out

```
3 4
1 2 1
1 3 4
1 3 7
2 3 1
1 3 2
```

```
8
72
0
0
```

2.in / 2.out

```
6 5
1145 1419 19810 9982 44 353
1 3 30
2 5 30
3 6 4
3 6 30
1 5 5
```

```
67333712
887369728
0
531574784
0
```

注释

对于第一组样例：

第一次测试中，区间 $[1,3]$ 包含子区间 $[1,1],[2,2],[3,3],[1,2],[2,3],[1,3]$，这六个子区间的魔力值之和分别为 $1,2,1,3,3,4$，因此效果值为 $(1\times 2\times 1\times 3\times 3\times 4)\bmod 2^4=72\bmod 16=8$。

第三次测试中，区间 $[2,3]$ 包含子区间 $[2,2],[3,3],[2,3]$，这三个子区间的魔力值之和分别为 $2,1,3$，因此效果值为 $(2\times 3\times 1)\bmod 2^1=6\bmod 2=0$。

题解

想了一小会高超的区间统计技术……然后发现简直是白瞎。

你先思考我们是在 $\bmod 2^p$，如果说答案是 2^p 的倍数，或者说，含有 p 个因子 2，是不是 $\bmod 2^p$ 的余数就一定是 0 了呢？

而我们这里是在对任何的区间求和再求积，如果这些和里面有 p 个偶数，也能说明 $\bmod$ 后的值为 0.

容易说明，区间足够长的时候，我们的加法必然产生相当多的偶数，让答案为 0.

具体多长呢？群友说 12 以上就可以了，但是实际上没有必要卡的那么严格，我当时猜了 20，然后可以证明这些区间和里至少有 30 个偶数让答案为 0.

大概写一下。假设这 20 个数里面有 k 个偶数，那么我们取 1 个数求和的时候，就有 k 种取法使得和为偶数，取 2 个数的时候，就能够取相邻的两个偶数或相邻的两个奇数使得和为偶数，取三个数的时候，可以取 3 个偶数，或者 2 个奇数 1 个偶数……

能够直观感觉到，不管到底有多少个偶数，最终所有区间求和那么多数里面必然有很多个偶数。（其实是我不会证……）

所以当区间长度小于 20 的时候，暴力计算结果，长度大于 20 的时候，我们直接输出 0.

```cpp
#include <iostream>
#include <map>
#include <stdio.h>
#include <string>
#include <vector>
typedef unsigned long long i64;
typedef __int128 i128;

#define maxn 200007

i64 li[maxn], si[maxn];

int n, q, l, r, p;
int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++)
        scanf("%lld", li + i), si[i] = li[i] + si[i - 1];

    // r-l -> max -> sum mod 2^p = 0!!!!!

    for (int i = 1; i <= q; i++) {
        scanf("%d%d%d", &l, &r, &p);
        if (r - l >= 20)
            puts("0");
        else {
            i128 m = 1, mod = (i128)1 << p;
            for (int i = l; i <= r; i++)
                for (int j = i; j <= r; j++) {
                    m = m * (si[j] - si[i - 1]) % mod;
                }

            printf("%d\n", (int)m);
        }
    }

    return 0;
}
```

D.三相电

题目描述

标题仅供参考，题目具体内容请以题面为准。

这是一道通信题。 你的程序将会被运行两次，两次运行之间相互独立，第一次运行结束后，程序的内存状态不会保留。你需要设计一种编码与解码方案，利用第一次运行产生的信息，在第二次运行中还原原始数据。

具体来说，RainF 想要向 FlameR 传递一个仅由 0, 1, 2 三种字符组成的字符串 s。然而，RainF 的通信设备有一个特殊限制：

- 传输后的字符串中，任意两个相邻字符都不能相同。

这个限制给通信带来了一些麻烦。为了完成这次传输，RainF 决定请你设计一套编码方案，使得信息能够顺利传递。

你的程序需要完成以下两个阶段：

- **第一阶段：编码** ($\mathsf{op} = 1$) 给定原始字符串 s，你需要将其编码为另一个字符串 t，使得 t 仅包含 0, 1, 2，且任意两个相邻字符都不相同。同时，为了避免传输过多信息，要求编码后的字符串长度满足 $1 \le |t| \le 2|s|$。
- **第二阶段：解码** ($\mathsf{op} = 2$) 在第二次运行中，你将获得第一阶段产生的编码字符串 t。你需要根据 t 还原出原始字符串 s，并保证还原结果与第一阶段给定的原始字符串完全一致。

注意：两次运行之间不会保留任何程序内存中的信息。你必须通过第一阶段产生的编码结果完成信息传递，不能依赖第一次运行时的变量或其他内存状态。

RainF 希望你的方案不仅能够满足通信设备的限制，还能保证信息不会丢失。接下来，就请你帮助 RainF 完成这次特殊的通信任务吧！

输入格式

第一行给定两个整数：T ($1 \le T \le 10^5$)，op ($\mathsf{op} \in \{1, 2\}$)，分别代表测试组数以及接下来需要执行的操作。

当 $\mathsf{op} = 1$ 时，每组测试有 2 行输入。第一行给定整数 n ($1 \le n \le 3 \times 10^5$)，表示计划传递的信息的长度；第二行给定仅包含 0, 1, 2 的字符串 s，表示计划传递的信息字符串。保证所有样例的 n 之和（即 $\sum n$）满足 $\sum n \le 3 \times 10^5$。

当 $\mathsf{op} = 2$ 时，每组测试有 2 行输入。第一行给定整数 m ($1 \le m \le 6 \times 10^5$)，表示 RainF 传递的信息的长度；第二行给定仅包含 0, 1, 2 且任意相邻两位不相等的字符串 t，表示 RainF 传递的信息字符串。保证所有样例的 m 之和（即 $\sum m$）满足 $\sum m \le 6 \times 10^5$。

输出格式

当 $\mathsf{op} = 1$ 时，每组测试输出两行内容。第一行输出 RainF 需要传递的信息的长度 m ($1 \le m \le 2n$)；第二行输出 RainF 需要传递的信息字符串 t。

当 $\mathsf{op} = 2$ 时，每组测试输出两行内容。第一行输出计划传递的信息长度 n；第二行输出计划传递的信息字符串 s。

样例

1.in / 1.out

```
1 3 1
2 1
3 0
4 3
```

```
5 012
6 6
7 000011
```

```
1 2
2 01
3 4
4 0102
5 8
6 01201210
```

2.in / 2.out

```
1 3 2
2 2
3 01
4 4
5 0102
6 8
7 01201210
```

```
1 1
2 0
3 3
4 012
5 6
6 000011
```

注释

样例仅用于为题目要求和输入输出格式提供参考。

题解

直接跳过 BC，拿到了本题的一血，爽到了。（但是又因为没有注意输入输出格式吃了一发罚时）

思考这样一个问题：我的字符串可以是原来的两倍长度，就意味着我其实可以把原来的字符串夹在不同的字符中间，这样的话我只用取 $1, 3, 5, 7, \ldots$ 这些字符就能得到原串。

例如: $10210 \to 1[2]0[1]2[0]1[2]0$，这样插入完了之后就是两两不同的了。

实现也很简单，因为相邻两个字符至多只有两种，而我有三种字符，所以必然可以构造出夹在两个字符中间的。不同的字符。

cpp

```cpp
#include <iostream>
#include <map>
#include <stdio.h>
#include <string>
#include <vector>
typedef long long i64;
typedef __int128 i128;

#define maxn

```

```cpp
int n;
int T, op, siz;
std::string s, r;

int mk[3];
void cl_mk() { mk[0] = mk[1] = mk[2] = 0; }
int fd_mk() {
    for (int i = 0; i < 3; i++)
        if (mk[i] == 0)
            return i;
}

int main() {
    std::cin >> n >> op;
    if (op == 1) {
        for (int i = 1; i <= n; i++) {
            std::cin >> siz >> s;
            r.clear();

            // if (siz == 1) {
            //     std::cout << s[0] << ((s[0] - '0') ^ 1);
            //     std::cout << std::endl;
            // } else
            {
                // std::cout << s[0];
                r += s[0];
                for (int i = 1; i < siz; i++) {
                    cl_mk();
                    mk[s[i] - '0'] = 1;
                    mk[s[i - 1] - '0'] = 1;
                    r += char(fd_mk() + '0');
                    r += s[i];
                    // std::cout << fd_mk() << s[i];
                }
                std::cout << r.size() << std::endl << r << std::endl;
            }
        }
    } else {
        for (int i = 1; i <= n; i++) {
            std::cin >> siz >> s;
            for (int i = 0; i < siz; i += 2)
                std::cout << s[i];
            std::cout << std::endl;
        }
    }
    return 0;
}
```

E.星际导演

题目描述

星际聊天室正在直播。共有 n 位外星人，第 i 位外星人在整数时刻 $[l_i, r_i]$ 内在线，两个端点均包含在内。

导演想从录像中剪出片段 $[x, y]$，其中 x, y 为整数，且 $1 \le x \le y \le 2 \times 10^5$。如果能找到两位不同的外星人，他们从时刻 x 到时刻 y 始终在线，就称这段片段合格。用于判定的两位外星人不能在片段中途更换。

形式化地，片段 $[x, y]$ 合格，当且仅当存在两个不同的下标 i, j，使得 $l_i \le x \le y \le r_i$ 且 $l_j \le x \le y \le r_j$。

导演只保留极大片段。一个合格片段 $[x, y]$ 是极大的，当且仅当不存在比它更长、且完整包含它的合格片段。换句话说，无论向左、向右，还是同时向两侧延长，都无法得到更长的合格片段。

请输出所有不同的极大片段。若两个片段的左右端点分别相同，则视为同一个片段。

输入格式

第一行包含一个整数 n（$1 \le n \le 2 \times 10^5$）。

接下来 n 行，每行包含两个整数 l_i, r_i（$1 \le l_i \le r_i \le 2 \times 10^5$），表示第 i 位外星人的在线时间。

输出格式

第一行一个整数 m，表示极大片段的数量。

接下来 m 行，每行两个整数 x, y，表示一个极大片段 $[x, y]$。按 x 从小到大输出。

若 $m = 0$，只输出一行 0。

样例

1.in / 1.out

```
4
1 3
1 2
2 4
3 4
```

```
3
1 2
2 3
3 4
```

2.in / 2.out

```
3
1 2
3 3
4 4 6
```

```
0
```

3.in / 3.out

```
8
```

```
2 1 8
3 2 6
4 3 4
5 5 5
6 10 10
7 10 10
8 12 13
9 13 13
```

```
1 3
2 2 6
3 10 10
4 13 13
```

注释

样例 1：三个答案各有固定的两人全程在线。$[1,3]$ 虽然每一刻都有两人在线，却没有固定的两人覆盖全段；$[2,2]$ 能延长，因此也不是答案。

样例 2：在线区间互不相交，答案为 0。

样例 3：第 1、2 位共同覆盖 $[2,6]$；第 5、6 位得到 $[10,10]$；第 7、8 位只在时刻 13 同时在线。三个片段均无法延长。

题解

这道题做出来也相当爽！感觉自己确实比几年前有比较大的进步。

其实有点像一个区间覆盖问题，然后往下想，因为我们的 $l_i, r_i \leq n$，我们是可以枚举值域的。

好，然后就思考，”往右不能延伸“这个条件转化成：对于每一个时刻 t 去枚举**包含它的**区间的右侧次远点 r，这个时刻的答案就是 $[t,r]$

为什么是枚举次远点？你想想，最远点相当于我们挑选的第一名嘉宾，次远点相当于我们挑选的第二名嘉宾，**被包含**这个条件就限定了这两名嘉宾必须在 i 时刻在场，而次远点就是两名嘉宾之中先离场的那个。

然后，”往左不能延伸“这个条件转化成：对于所有上面的区间，剔除完全被包含的区间。

实现上，我这里将右端点合并，然后取左端点的最小值。

做完感觉相当通透。

```cpp
#include <algorithm>
#include <iostream>
#include <map>
#include <set>
#include <stdio.h>
#include <string>
#include <vector>
typedef long long i64;
typedef __int128 i128;

#define maxn 200007

std::multiset<int> s;
```

```cpp
std::vector<int> ins[maxn], outs[maxn], seqs[maxn];

int n, l, r;

int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d%d", &l, &r);
        ins[l].push_back(r);
        outs[r].push_back(r);
        // outs[r].push_back(l);
    }

    for (int i = 0; i < maxn; i++) {
        for (auto x : ins[i])
            s.insert(x);

        if (s.size() >= 2) {
            int mx = *(--s.end());
            int mx2 = *(--(--s.end()));

            // printf("%d, %d\n", i, mx2);

            // (i, mx2)
            seqs[mx2].push_back(i);
        }

        for (auto x : outs[i])
            s.erase(x);
    }

    std::vector<std::pair<int, int>> ret;
    for (int i = 0; i < maxn; i++)
        if (not seqs[i].empty()) {
            ret.push_back(
                {*std::min_element(seqs[i].begin(), seqs[i].end()), i});
        }

    printf("%d\n", ret.size());
    for (auto [u, v] : ret) {
        printf("%d %d\n", u, v);
    }
    return 0;
}
```

F.奶奶茶店的神秘排单

题目描述

奶茶店在时刻 0 收到了 n 个订单，编号为 1 到 n。每个订单需要制作一杯奶茶。店里只有一台煮茶机和一台封口机，时刻 0 时两台机器都空闲。

每杯奶茶都必须依次经过以下工序：

1. **煮茶**：第 i 杯奶茶需要煮茶 a_i 秒。
2. **冷却**：第 i 杯奶茶煮好后，需要冷却 c_i 秒，之后才能开始封口。
3. **封口**：第 i 杯奶茶需要封口 b_i 秒。

你需要确定一个处理订单的顺序。这个顺序是一个 1 到 n 的排列，每个订单恰好出现一次。煮茶机和封口机都必须严格按照这个顺序处理订单。

具体规则如下：

- 煮茶机和封口机同一时刻都只能处理一杯奶茶。
- 煮茶和封口一旦开始，就必须连续进行到该工序结束，不能中途暂停。
- 冷却不占用机器，多杯奶茶可以同时冷却。
- 第 i 杯奶茶在时刻 t 煮好后，最早可以在时刻 $t + c_i$ 开始封口。如果此时封口机正忙，或者还没有轮到它封口，那么它需要继续等待。
- 封口机空闲时，只能等待顺序中的下一杯奶茶满足封口条件，不能跳过它去封口后面的奶茶。
- 两台机器可以同时处理不同的奶茶。

请选择一个订单顺序，使得所有奶茶都完成封口的时间尽可能早。求这个最早完成时刻。

输入格式

第一行输入一个整数 n，表示订单数量。

接下来 n 行，第 i 行输入三个整数 a_i, b_i, c_i，分别表示第 i 杯奶茶的煮茶时间、封口时间和最短冷却时间，单位均为秒。

对于所有测试数据，$1 \le n \le 2 \times 10^5$，$1 \le a_i, b_i \le 10^9$，$0 \le c_i \le 10^9$。

输出格式

输出一行一个整数，表示所有奶茶都完成封口的最早时刻，单位为秒。

样例

1.in / 1.out

```
1 4
2 4 2 5
3 2 5 0
4 3 3 1
5 1 2 5
```

```
1 17
```

注释

一种最优的处理顺序是第 2、4、1、3 杯。按照这个顺序，四杯奶茶分别在时刻 2、3、7、10 煮好，并分别在时刻 2、8、12、11 满足冷却要求。

第 2 杯无需冷却，在时刻 2 开始封口，到时刻 7 完成。此时煮茶机已经煮好了第 4 杯和第 1 杯。第 4 杯冷却到时刻 8，随后封口，到时刻 10 完成。

时刻 11，第 3 杯已经冷却完成，但排在它前面的第 1 杯还需要冷却到时刻 12。因此封口机先等待，再从时刻 12 到 14 为第 1 杯封口，最后从时刻 14 到 17 为第 3 杯封口。

所有奶茶最早在时刻 17 完成封口。

题解

是个什么鬼结论……等会写
