# Windy Round 1.5 & Radiant Round 3.5

## 备用题面

| **中文名称** | 压缩打表 | 计算器 | 旅人 1977 | 谜 | 移球游戏 | RMSQ | ACP-II |
|---|---|---|---|---|---|---|---|
| **英文名称** | `table` | `calc` | `space` | `mystery` | `ball` | `rmsq` | `acp2` |
| **题目类型** | 传统 | 传统 | 传统 | 传统 | 传统 | 传统 | 传统 |
| **答案比较** | 全文 | 全文 | 全文 | 全文 | SPJ | 全文 | SPJ |
| **计分方式** | 加和 | 捆绑测试 | 捆绑测试 | 捆绑测试 | 捆绑测试 | 捆绑测试 | 捆绑测试 |
| **时间限制** | 1 sec | 1 sec | 1.2 sec | 1.5 sec | 3 sec | 3.8 sec | 2.5 sec |
| **空间限制** | 128 MB | 250 MB | 500 MB | 512 MB | 512 MB | 1.2 GB | 512 MB |
| **题目负责** | yzy1 | 小波 | 八云蓝 | rin & yzy1 | 八云蓝 | yzy1 | yzy1 |

编译选项

| **对于 C++ 语言** | `g++ -DONLINE_JUDGE -Wall -fno-asm -O2` |
|---|---|
| **对于 C 语言** | `bash -c "echo H6_6Q AK IOI! >> /dev/stderr ; /bin/false"` |
| **对于 Pascal 语言** | `bash -c "echo H6_6Q AK NOI! >> /dev/stderr ; /bin/false"` |

**注意事项（请仔细阅读）**

1. C/C++ 中函数 `main()` 的返回值类型必须是 `int`，程序正常结束时的返回值必须是 0。
2. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
3. 由于这是在线测试，选手可以使用搜索引擎或通讯软件等工具获取与题目相关的帮助信息，但是我们**并不推荐**这么做。
4. 由于这是在线测试，选手可以使用 ssh 等命令获取并非属于自己的代码片段，但是我们**并不推荐**这么做。
5. 评测使用在线 OJ。选手没有必要同时也禁止使用文件输入输出。
6. 输入文件不保证是否存在行末空格和文末回车，且不保证换行符统一。请选手使用较为完善的读入方式。
7. 本 pdf 仅供备用题面使用，对于题面差异请以 OJ 题面为准。
8. 本备用题面最终解释权归出题组全体成员所有。

# 压缩打表（table）

**传统，1 sec，128 MB，答案比较：全文，计分方式：加和，题目负责：yzy1**

## 【题目描述】

小 A 想要打表过题，但是代码长度过长。他想到了把数字用十六进制表示有时比用十进制表示更短，决定把答案表用十六进制进行改写。

我们定义下文中的「十六进制」均用数字 $0 \sim 9$ 和大写英文字母 `A` ～ `F` 表示，且每个十六进制数字都包含前缀 `0x`。

给出若干个用大括号包裹，用逗号分隔的十进制整数，代表小 A 的答案表。你需要对答案表内的每个数字进行改写：

- 若该整数使用十六进制表示所占字符数小于或等于十进制表示，则将该数改写为十六进制。
- 否则，该数保留十进制不变。

输出改写后的答案表。

## 【输入格式】

输入一行一个字符串，字符串中仅包含大括号、逗号和数字，不包含空格和其他字符等。具体格式可以参考 C++ 语言的数组定义格式。代表小 A 的答案表，保证答案表格式正确。

## 【输出格式】

输出一行一个字符串，表示改写后的答案表。

## 【样例 1 输入】

```
{1,314159,3141592653589793}
```

## 【样例 1 输出】

```
{1,314159,0xB29430A256D21}
```

## 【样例 1 解释】

- 1 用十六进制表示为 `0x1`，共 3 个字符，长度大于十进制的 1 个字符。
- 314159 用十六进制表示为 `0x4CB2F`，共 7 个字符，长度大于十进制的 6 个字符。
- 3141592653589793 用十六进制表示为 `0xB29430A256D21`，共 15 个字符，长度小于十进制的 16 个字符。

## 【样例 2 输入】

```
{}
```

## 【样例 2 输出】

```
{}
```

### 【样例 2 解释】

输入为空数组，所以输出也应为空数组。

### 【数据范围及约定】

本题共五个测试点，每个测试点 20 分，总分数为各测试点分数之和。

记答案表中共有 $n$ 个整数，这些整数分别为 $a_1, a_2, \cdots, a_n$。则对于 $100\%$ 的数据，$0 \leq n \leq 10^3$，$0 \leq a_i < 2^{64}$。

# 计算器（calc）

**传统，1 sec，250 MB，答案比较：全文，计分方式：捆绑测试，题目负责：小波**

## 【题目描述】

魔理沙的计算器可以进行 $b$ 进制的运算，屏幕上可以显示 $k$ 位数字（不包含小数点）。进行计算后，若某个数字超出了屏幕，就会被舍去（例如 $b = 10$ 时 $1 \div 7 = 0.142857\cdots$，若屏幕大小为 $4$，那么最终显示为 $0.142$）。

魔理沙用计算器计算了 $1 \div n = n'$（$n'$ 为显示在屏幕上的结果，下同），再计算 $1 \div n' = n''$。魔理沙希望知道，有多少个正整数 $n$ 使得 $n = n''$。你只需要输出这个答案对 $998,244,353$ 取模后的结果即可。

## 【输入格式】

- 第一行有一个正整数 $T$，表示数据组数。
- 接下来 $T$ 行，每行有两个正整数 $b, k$，分别表示计算器的进制、屏幕上能显示的数字个数。

## 【输出格式】

输出共 $T$ 行。每行输出一个整数。第 $i$ 行的整数表示第 $i$ 组数据中合法的 $n$ 的总数对 $998,244,353$ 取模后的结果。

## 【样例 1 输入】

```
3
4 2
5 3
12 99
```

## 【样例 1 输出】

```
3
3
19503
```

## 【样例 1 解释】

- 对于第一组询问，符合条件的数为 $1, 2, 4$。
- 对于第二组询问，符合条件的数为 $1, 5, 25$。

## 【数据范围及约定】

| subtask | 分值 | $b \leq$ | $k \leq$ | 特殊性质 | subtask 依赖 |
|---|---|---|---|---|---|
| 1 | 20 | 10 | 7 | - | - |
| 2 | 20 | $10^5$ | 2 | $k = 2$ | - |
| 3 | 10 | $10^5$ | 3 | $k = 3$ | - |
| 4 | 50 | $10^5$ | 500 | - | $1, 2, 3$ |

对于 100% 的数据，满足 $1 \leq T \leq 10$，$2 \leq b \leq 10^5$，$1 \leq k \leq 500$。

# 旅人 1977（space）

**传统，1.2 sec，500 MB，答案比较：全文，计分方式：捆绑测试，题目负责：八云蓝**

## 【题目描述】

给出一张 $n$ 个点、$m$ 条边的有向图。点无点权，边有边权，图可能有重边自环。但保证至少有一条路径可以从 $S$ 走到 $T$，其中 $S$、$T$ 为题目给定量。第 $i$ 条有向边起点为 $f_i$，终点为 $t_i$，它的权值用一个有序三元组 $(l_i, r_i, w_i)$ 表示。

定义一条从 $S$ 到 $T$ 的有向路径的「美观度」计算方式如下：

1. 初始化一颗线段树，维护一个初始值均为 0 的长度为 $k$ 的数组 $a$，具体方式可以参考下面的伪代码。
2. 按经过的顺序遍历路径中的每一条边，设它的编号为 $i$，则用线段树将 $a$ 数组的 $[l_i, r_i]$ 区间加 $w_i$。
3. 将线段树上所有节点的懒标记（即伪代码中的 tag）加和，得到的结果即为这条路径的美观度。

你需要构造一条从 $S$ 到 $T$ 的路径，满足这条路径的「美观度」最小，输出这个最小值。

以下是线段树的伪代码：（为了方便选手阅读，题目附件中给出了线段树的 C++ 源代码）

**Algorithm:** SegTree

```
Input. 长度为 k 的 a 数组，初始全为 0
Output. a 数组进行若干次区间加操作后得到的结果
Method.
Add(L, R, x)
    Add0(L, R, x, root, 1, k)
Add0(L, R, x, u, l, r)
    if L ≤ l and r ≤ R
        tag(u) ← tag(u) + x
        return
    mid ← ⌊(l+r)/2⌋
    tag(lson(u)) ← tag(lson(u)) + tag(u)
    tag(rson(u)) ← tag(rson(u)) + tag(u)
    tag(u) ← 0
    if L ≤ mid
        Add0(L, R, x, lson(u), l, mid)
    if mid < R
        Add0(L, R, x, rson(u), mid + 1, r)
```

## 【输入格式】

- 第一行五个整数 $n, m, k, S, T$。
- 接下来 $m$ 行，每行五个整数 $f_i, t_i, l_i, r_i, w_i$。

## 【输出格式】

对于每组数据，输出一行一个整数，表示最小值。

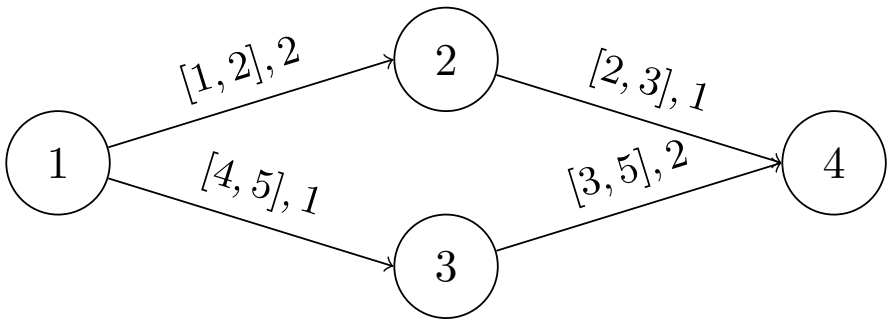
**【样例 1 输入】**

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

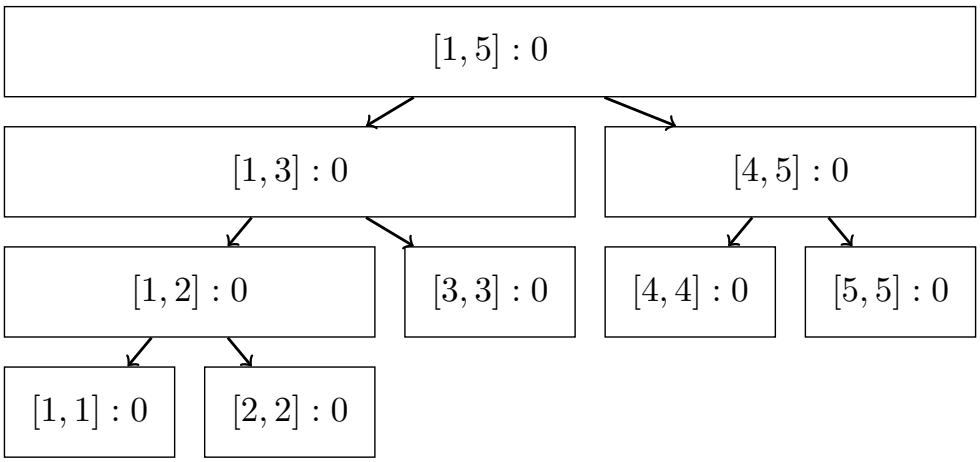
**【样例 1 输出】**

```
5
```

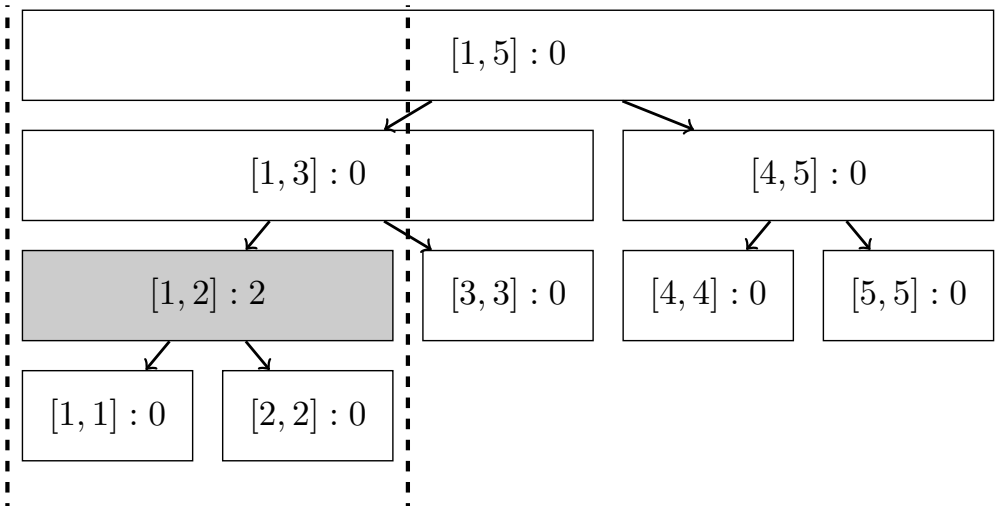
**【样例 1 解释】**



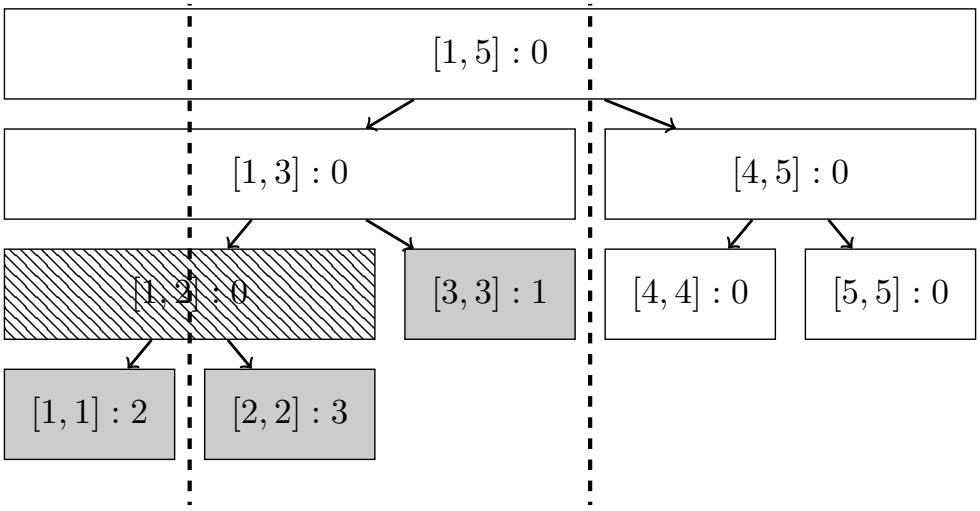

容易发现，样例 1 中有且仅有两条可能的路径：$1 \to 2 \to 4$ 与 $1 \to 3 \to 4$。下面分别计算这两条路径最终 tag 的权值和。




考虑画出这棵 $k = 5$ 的线段树。

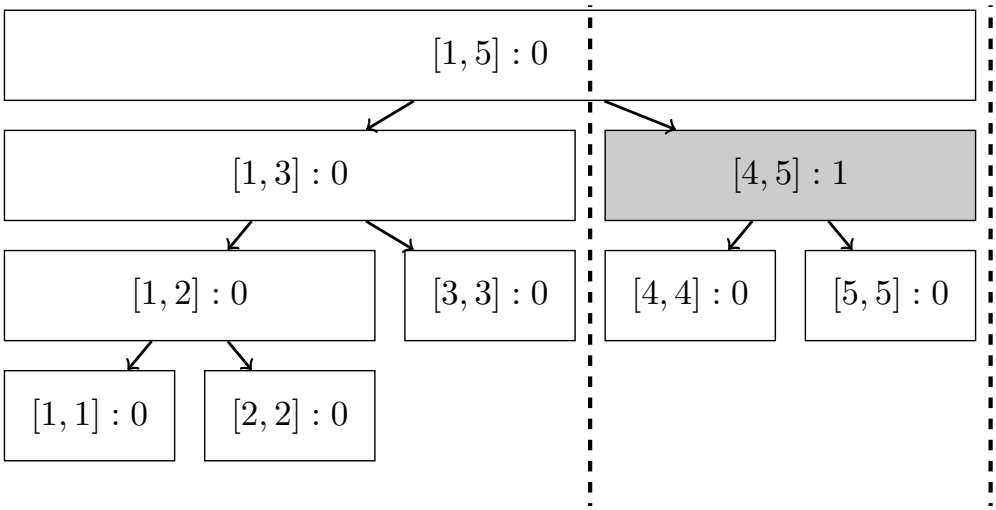



走了边 $1 \to 2$ 后，$[1, 2]$ 节点被打上了权值为 $2$ 的 tag。

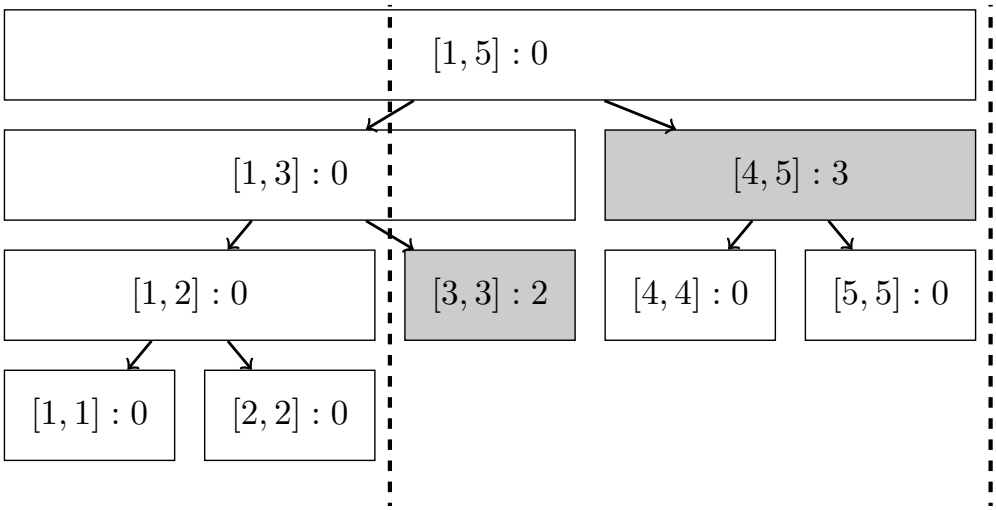



走了 $2 \to 4$ 后，$[2, 2]$ 节点和 $[3, 3]$ 节点被打上了值为 $1$ 的 tag；但是 $[1, 2]$ 节点的标记进行了下推（因为使 $[2, 3]$ 区间 $+1$ 的时候会访问到 $[1, 2]$ 节点，而 $[1, 2] \not\subseteq [2, 3]$，故而发生标记下推），因此 $[1, 1]$ 节点和 $[2, 2]$ 节点的 tag 分别加上了 $2$，最终成了如图所示的模样。

因此走到 $4$ 之后所有结点的 tag 之和为 $2 + 3 + 1 = 6$。




对于另外一条路径，首先对 $[4,5]$ 加上 $1$。




接着对 $[3,5]$ 加上 $2$。未发生带有 tag 的节点的标记下推，因此最终的权值为 $2+3=5$。

由于 $6>5$，因而最终的答案为 $5$。

**【样例 2 输入】**

```
10 19 5 6 1
2 1 1 3 592
6 8 3 5 488
10 9 4 4 548
10 4 1 4 442
6 5 1 3 422
9 7 1 4 529
5 8 1 1 559
5 9 1 5 560
```

```
5 8 2 3 434
5 9 3 3 592
4 7 2 2 594
7 9 5 5 595
4 1 4 4 501
3 9 1 2 410
10 6 2 4 509
6 10 4 5 455
2 4 2 5 444
4 3 4 5 541
8 7 1 1 463
```

**【样例 2 输出】**

```
2295
```

**【数据范围及约定】**

| subtask | 分值 | $n \leq$ | $m \leq$ | $k \leq$ | 特殊性质 | subtask 依赖 |
|---|---|---|---|---|---|---|
| 1 | 10 | 10 | 30 | 5 | - | 1 |
| 2 | 5 | 30 | 30 | 12 | **AB** | - |
| 3 | 20 | 30 | 500 | 12 | **B** | 2 |
| 4 | 15 | 200 | $3 \times 10^3$ | 25 | **B** | 3 |
| 5 | 30 | 200 | $3 \times 10^3$ | 25 | - | 4 |

- **特殊性质 A**：保证有且仅有一条从 $S$ 到 $T$ 的路径。
- **特殊性质 B**：保证图中不存在环。

对于 100% 的数据，有 $1 \leq n \leq 200$，$1 \leq m \leq 3 \times 10^3$，$1 \leq l_i \leq r_i \leq k \leq 25$，$1 \leq w_i \leq 10^3$。

**【提示】**

在附件中有两个版本的线段树。Lite 版本仅包含了在本题中你会用到的下推标记的操作，而标准版则较为完整地支持区间加、区间求和。选手可根据自己的喜好使用。

# 谜（mystery）

**传统，1.5 sec，512 MB，答案比较：全文，计分方式：捆绑测试，题目负责：rin & yzy1**

## 【题目描述】

给出一个长度为 $n$ 的单调不降整数数列 $\{a_i\}$ 和一个整数 $k$。

我们定义两个长度均为 $p$ 的序列 $\{x_i\}, \{y_i\}$ 的「差异度」$F(x, y, p) = \sum_{i=1}^{p} |x_i - y_i|$。

现在对于每个整数 $l \in [1, n]$，你都需要构造一个长度为 $l$ 的序列 $\{b_{l,i}\}$。满足对于任意 $1 \leq i < l$，$b_{l,i+1} \geq b_{l,i} + k$；且 $F(a_{[1\cdots l]}, b_l, l)$ 最小。其中 $a_{[1\cdots l]}$ 表示 $\{a_i\}$ 的长度为 $l$ 的前缀，即 $\{a_1, a_2, \cdots, a_l\}$。注意，$b_{l,i}$ 没必要是整数。

## 【输入格式】

第一行输入两个整数 $n, k$。
第二行输入 $n$ 个整数，代表 $\{a\}$。
第三行输入一个整数 $T$，代表答案输出方式。具体解释请参考「输出格式」。

## 【输出格式】

- 若 $T = 0$，则输出 $n$ 个整数，每个整数单独占一行。第 $l$ 行的整数代表 $F(a_{[1\cdots l]}, b_l, l)$。
- 若 $T = 1$，则你仅需输出一行一个整数，表示 $F(a, b_n, n)$。

## 【样例 1 输入】

```
5 2
2 3 4 5 6
0
```

## 【样例 1 输出】

```
0
2
4
8
12
```

## 【样例 1 解释】

如下是一种可能的构造方案：

$$b_1 = \{2\}$$
$$b_2 = \{2, 4\}$$
$$b_3 = \{1, 3, 5\}$$
$$b_4 = \{1, 3, 5, 7\}$$
$$b_5 = \{0, 2, 4, 6, 8\}$$

**【样例 2 输入】**

```
6 2
1 1 4 5 6 8
0
```

**【样例 2 输出】**

```
0
4
4
6
8
10
```

**【样例 2 解释】**

如下是一种可能的构造方案：

$$b_1 = \{1\}$$
$$b_2 = \{0, 2\}$$
$$b_3 = \{0, 2, 4\}$$
$$b_4 = \{0, 2, 4, 6\}$$
$$b_5 = \{\text{-}1, 1, 3, 5, 7\}$$
$$b_6 = \{\text{-}1, 1, 3, 5, 7, 9\}$$

**【样例 3 输入】**

```
20 4
4 6 7 9 19 21 30 32 33 35 49 50 58 67 75 77 78 89 91 91
0
```

**【样例 3 输出】**

```
0
4
10
20
20
24
24
28
34
44
44
50
50
50
50
54
60
60
64
72
```

**【数据范围及约定】**

| subtask | 分值 | $n \leq$ | $T =$ | $k, a_i \leq$ | subtask 依赖 |
|---|---|---|---|---|---|
| 1 | 30 | 100 | 0 | 100 | - |
| 2 | 30 | $10^5$ | 0 | $10^8$ | 1 |
| 3 | 40 | $10^6$ | 1 | $10^8$ | - |

对于 100% 的数据，$1 \leq n \leq 10^6$，$1 \leq k, a_i \leq 10^8$，$T \in \{0, 1\}$。

# 移球游戏（ball）

**传统，3 sec，512 MB，答案比较：SPJ，计分方式：捆绑测试，题目负责：八云蓝**

## 【题目描述】

小 C 正在玩一个移球游戏，他面前有一个 $n$ 个点，$m$ 条边的连通无向图。图可能有自环和重边。点从 $1 \sim n$ 标号，边从 $1 \sim m$ 编号。且小 C 有黑白两种颜色的球各一个。

初始时，小 C 可以决定一个点 $S$，并把两个球放到点 $S$ 上。而小 C 的任务是让两个球**各自**经过图的每一个节点至少 1 次，并且最后都回到点 $S$。

小 C 可以通过若干次操作完成这个目标，操作分为三种：

- `Ran x`：将黑球移动至与它当前节点有直接连边的节点 $x$。
- `Chen x`：将白球移动至与它当前节点有直接连边的节点 $x$。
- `Swap`：交换黑白两个球的位置。

小 C 被难住了，但他相信难不倒你，请你给出一个操作方案完成小 C 的目标。合法的方案可能有多种，你只需要给出任意一种，题目保证一定存在一个合法方案。

## 【输入格式】

- 第一行两个整数 $n, m$，表示该图有 $n$ 个节点，$m$ 条边。
- 接下来 $m$ 行，每行两个整数 $u, v$，表示有一条连接 $u, v$ 的无向边。

## 【输出格式】

- 第一行输出两个整数 $S$ 和 $k$。其中 $S$ 的含义见题目描述，$k$ 表示你的方案的操作次数。
- 接下来 $k$ 行，每行一个字符串和零到一个整数，表示一次操作。

## 【样例 1 输入】

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

## 【样例 1 输出】

```
1 5
Ran 2
Chen 3
Swap
Ran 1
Chen 1
```

## 【样例 1 解释】

| 操作次数 | 黑球位置 | 白球位置 |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 1 |
| 2 | 2 | 3 |
| 3 | 3 | 2 |
| 4 | 1 | 2 |
| 5 | 1 | 1 |

## 【判分方式】

**本题使用 Special Judge。**

对于每组数据，若你输出的方案不合法（含不合法的移动操作，或者至少一球没有经过每个结点至少 1 次，或者至少一球最后没有在 $S$ 点），你的分数为零分。否则你的分数将这样计算：

- 当 $k \leq 4 \cdot n$ 时，你将获得该测试点 20% 的分数；
- 当 $k \leq 3 \cdot n$ 时，你将获得该测试点 40% 的分数；
- 当 $k \leq \lfloor \frac{11}{4} \cdot n \rfloor$ 时，你将获得该测试点 70% 的分数；
- 当 $k \leq \lfloor \frac{8}{3} \cdot n \rfloor$ 时，你将获得该测试点所有的分数。

## 【数据范围及约定】

**本题采用捆绑测试，且仅有一个 subtask，总成绩取各测试点最低分。**

对于 100% 的数据，$3 \leq n, m \leq 5 \times 10^5$。

## 【提示】

为了方便选手测试，在附件中我们下发了 `checker.cpp` 文件，选手可以编译该程序，并使用它校验自己的输出文件。但请注意它与最终评测时所使用的校验器并不完全一致。你也不需要关心其代码的具体内容。

编译命令为：`g++ checker.cpp -o checker -std=c++14`。

checker 的使用方式为：`./checker <inputfile> <outputfile>`，参数依次表示输入文件与你的输出文件。

若你输出的数字大小范围不合法，则校验器会给出相应提示。若你的输出数字大小范围正确，但方案错误，则校验器会给出简要的错误信息：

1. `A x`，表示进行到第 $x$ 个操作时不合法。
2. `B x`，表示操作执行完毕后 $x$ 球没有经过每个节点至少一次，其中 $x = 0$ 表示黑球，$x = 1$ 表示白球。
3. `C x`，表示操作执行完毕后 $x$ 球没有回到 $S$ 点。其中 $x = 0$ 表示黑球，$x = 1$ 表示白球。
4. `Illeagl Output`，表示你输出了错误的操作。请检查你的输出格式是否错误。

若你的方案正确，校验器会给出 `OK`。

# RMSQ（rmsq）

**传统，3.8 sec，1.2 GB，答案比较：全文，计分方式：捆绑测试，题目负责：yzy1**

## 【题目描述】

给出一个长度为 $m$ 的**排列** $b$ 和一个长度为 $n$ 的**排列** $a$。

如果一个序列 $S$，满足其按位置从左到右依次匹配 $b$ 的一个区间从左到右的位置，那么我们说 $S$ 是一个「优美序列」。

给出 $q$ 次询问。每次询问给出两个整数 $l$ 和 $r$。你需要找到一个 $a$ 的 $[l,r]$ 子区间中的一个最长的满足「优美序列」条件的子序列长度。

## 【输入格式】

**本题采用强制在线。**

- 第一行输入三个整数 $m, n, q$。
- 第二行输入 $m$ 个整数 $b_1, b_2, \cdots, b_m$。
- 第三行输入 $n$ 个整数 $a_1, a_2, \cdots, a_n$。
- 接下来 $q$ 行每行输入两个整数 $l', r'$，你需要将 $l'$ 和 $r'$ 按位异或 *lastans* 来得到真正的 $l, r$。其定义为上一次询问操作得到的答案，若之前没有询问操作，则为 $0$。

## 【输出格式】

输出共 $q$ 行。第 $i$ 行输出一个整数，表示第 $i$ 次询问的答案。

## 【样例 1 输入】

```
4 6 6
1 2 3 4
1 2 3 2 3 4
1 3
2 7
1 7
0 7
0 4
2 5
```

## 【样例 1 输出】

```
3
3
2
2
3
```

```
4
```

### 【样例 1 解释】

*lastans* 解密后的询问为：

```
1 3
1 4
2 4
2 5
2 6
1 6
```

### 【样例 2】

见选手目录下的 ***rmsq/rmsq2.in*** 与 ***rmsq/rmsq2.ans***。

### 【数据范围及约定】

**本题采用捆绑测试，且仅有一个 subtask，总成绩取各测试点最低分。**

对于 100% 的数据，$1 \leq l \leq r \leq n \leq 3 \times 10^5$，$1 \leq a_i \leq m \leq 3 \times 10^5$，$1 \leq q \leq 1 \times 10^6$。

# ACP-II（acp2）

**传统，2.5 sec，512 MB，答案比较：SPJ，计分方式：捆绑测试，题目负责：yzy1**

## 【题目描述】

**这是一道传统题。**

你需要控制机器人玩躲避子弹的游戏。游戏的画面是一个在第一象限，坐标范围从 $(0,0)$ 至 $(n,m)$ 的矩形。机器人初始生成在坐标 $(0,0)$。你可以通过以下几种指令来控制这个机器人：

| 指令 | 描述 |
|---|---|
| **0** | 机器人静止不动 |
| **1** | 机器人向左移动 $1$ 单位长度，即从 $(x,y)$ 移动到 $(x\text{ - }1,y)$ |
| **2** | 机器人向下移动 $1$ 单位长度，即从 $(x,y)$ 移动到 $(x,y\text{ - }1)$ |
| **3** | 机器人向上移动 $1$ 单位长度，即从 $(x,y)$ 移动到 $(x,y+1)$ |
| **4** | 机器人向右移动 $1$ 单位长度，即从 $(x,y)$ 移动到 $(x+1,y)$ |

你构造的指令用一个数字串 $C'$ 表示。每种指令都会耗费一定的费用。具体来讲，你构造的**原始指令** $C'$ 中每包含一个 $i$ 号命令，就会耗费 $P_i$ 的费用。

你的指令会被重复 $k$ 遍，作为机器人的移动指令 $C$。例如当 $C'=1123$，$k=3$ 时，机器人收到的指令 $C=112311231123$。

游戏中会生成 $b$ 个子弹，每个子弹用一个有序六元组 $(l_i,r_i,x_i,y_i,p_i,q_i)$ 表示。它的意思是这颗子弹在 $l_i$ 秒生成，第 $r_i$ 秒销毁。生成时的坐标为 $(x_i,y_i)$。每秒沿着 $x$ 轴正方向移动 $p_i$ 单位长度，沿着 $y$ 轴正方向移动 $q_i$ 单位长度。

游戏共进行 $d$ 秒，若 $d$ 秒内机器人碰撞到子弹或者移动到画面以外，机器人就会爆炸，游戏失败。如果在第 $d$ 秒结束时机器人没有爆炸，则游戏胜利。

游戏中的每一秒分为五个阶段，每个阶段结束后才会执行下一个阶段：

1. 机器人移动阶段。机器人会在第 $i$ 秒执行 $C_i$ 指令。若 $C$ 的长度 $<i$，也就是说所有指令都执行完了，则静止不动。
2. 子弹移动阶段，所有已经生成且还未销毁的子弹会进行一次移动。具体来讲，设当前是第 $c$ 秒，对于满足 $l_i<c\le r_i$ 的每个子弹 $(l_i,r_i,x_i,y_i,p_i,q_i)$，它的坐标会变为 $(x_i+(c\text{ - }l_i)p_i,y_i+(c\text{ - }l_i)q_i)$。
3. 子弹生成阶段。设当前是第 $c$ 秒，对于所有 $l_i=c$ 的子弹，将它们生成进画面，放置各自的初始位置 $(x_i,y_i)$ 上。
4. 判定阶段。若此时机器人的位置超出了画面，或者碰撞到了任意一颗已经被生成且尚未销毁的子弹，则机器人就会受击爆炸。具体来说，对于每一颗在画面上的子弹，设这颗子弹在这一秒初的位置为 $P=(x',y')$，当前的位置为 $Q=(x'',y'')$。若子弹是在这一秒内刚生成的则 $P=Q$。连接线段 $PQ$，若当前机器人的坐标在这条线段上（包括在线段的端点上），则视为机器人碰撞到了子弹。

5. 子弹销毁阶段。设当前是第 $c$ 秒，对于所有 $r_i = c$ 的子弹，将它们销毁。

**【输入格式】**

- 第一行输入六个整数 $n, m, b, d, k, maxc$。
- 第二行输入五个整数 $P_0, P_1, P_2, P_3, P_4$。
- 第三行至第 $b + 2$ 行，每行输入六个整数。其中第 $i + 2$ 行输入的整数分别代表 $l_i, r_i, x_i, y_i, p_i, q_i$。

**【输出格式】**

- 若 $maxc \geq 0$，则你需要输出一行一个字符串，表示你构造的指令，你需要保证指令费用 $\leq maxc$。
- 否则，你需要输出一行一个整数，表示在所有可以成功完成游戏的指令中费用最少的指令的费用是多少。

**【样例 1 输入】**

```
1 1 3 100 1 5
1 1 1 2 3
1 100 0 0 1 0
1 100 1 0 0 0
2 3 0 1 0 0
```

**【样例 1 输出】**

```
34
```

**【样例 1 解释】**

示意图里用 `0` 表示子弹，`1` 表示机器人，`.` 表示空位。

第 1 秒钟：机器人移动到了 $(0, 1)$；在 $(0, 0)$ 和 $(1, 0)$ 分别生成了一颗子弹。

```
1.
00
```

第 2 秒钟：机器人移动到了 $(1, 1)$；在 $(0, 0)$ 处的子弹移动到了 $(0, 1)$；在 $(0, 1)$ 生成了一颗子弹。所以现在在 $(0, 1)$ 处有两颗子弹（在下图中因为子弹重合只标出了一颗）。

```
01
.0
```

第 3 秒钟：机器人位置不变；其中一颗位于 $(0, 1)$ 的子弹飞出了画面；另外一颗位于 $(0, 1)$ 的子弹被销毁。

```
.1
.0
```

第 4 秒至第 100 秒：机器人没有移动；位于画面外的子弹移动后仍然在画面之外，在画面内的那颗子弹没有移动，画面情况同第三秒的情况。

综上所述，这个指令可以成功完成游戏，费用为 $1 \times 3 + 1 \times 2 = 5$。

## 【数据范围及约定】

**本题采用捆绑测试。**

| **subtask** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **分值** | 10 | 10 | 10 | 10 | 15 | 15 | 15 | 15 |

对于 100% 的数据，保证 $n, m, b, d, P_i \geq 0$，$k \geq 1$，$1 \leq l \leq r$，$maxc \geq -1$。

你可能会想，这算什么数据范围？怎么没有上界？而且表格里怎么没有 subtask 的特殊限制？实际上，出题人也不知道数据的上界和特殊限制。

由于出题人的误操作，数据生成器源代码意外丢失，只剩下可执行程序。下发文件中提供了在 Windows / Linux / Mac 下编译的可执行程序。不同操作系统下编译的文件名如下：

- 对于 Windows 系统（文件名 genX-w32）：`g++ genX.cpp -o genX -O3 -std=c++14`，在 Windows10，Dev-C++ 5.50，TDM-GCC 4.9.2 32bit 下编译。
- 对于 Linux/Mac 系统（32 位，文件名 genX-l32）：`g++ genX.cpp -o genX -O3 -std=c++14 -m32`，在 NOI Linux 2.0，GCC 9.3.0 下编译。
- 对于 Linux/Mac 系统（64 位，文件名 genX-l64）：`g++ genX.cpp -o genX -O3 -std=c++14 -m64`，在 NOI Linux 2.0，GCC 9.3.0 下编译。

数据生成器需要在运行后输入一个 $[1, 2^{30})$ 范围内的整数作为随机数种子，它会生成一份输入至 `0.in` 中。出题人无法保证生成的数据范围，但是可以保证：

- 同一个生成器所生成出的数据拥有同一个特殊限制。
- 所有生成器在 NOI Linux 2.0，处理器 i5-4200m 下运行，所用时间不超过 3s，内存占用不超过 2G。
- 生成出来的数据只与你输入的种子有关，与当前时间、操作系统等其它可变因素无关。
- 部分数据生成器会有小概率生成一个无解数据，但保证 OJ 上的所有数据均有解。
- 你输入的种子只会当作随机数的种子使用，生成器不会出现特判种子生成干扰数据等行为。

数据中的每个 subtask 分别对应一个数据生成器所生成的数据。编号为 $X$ 的 subtask 对应的生成器是 `genX`。

**注意：由于输入格式中没有要求输入「subtask 编号」，你需要自己判断当前数据所对应的 subtask。且本题的 subtask 为乱序排布，你需要自己判断每个 subtask 的难易程度。**

同时，在下发文件分发有 `checker.cpp`。使用 `g++ checker.cpp -o checker -std=c++14` 将 `checker.cpp` 编译为可执行文件后运行 `./checker <inputfile> <outputfile>`，`checker` 就会给出游戏的输赢情况。其中 `<inputfile>` 为数据生成器生成的 `.in` 文件；`<outputfile>` 中存放你的输出。**本 checker 仅供理解游戏规则使用，和实际使用的 checker 有所不同，且不保证 checker 的时间复杂度正确。**