

# 第 31 届全国信息学奥林匹克竞赛

## CCF NOI 2014

### 第二试

**竞赛时间：2014 年 7 月 29 日 8:00–13:00**

题目名称	动物园	随机数生成器	购票
目录	zoo	random	ticket
可执行文件名	zoo	random	ticket
输入文件名	zoo.in	random.in	ticket.in
输出文件名	zoo.out	random.out	ticket.out
每个测试点时限	1 秒	5 秒	3 秒
内存限制	512MB	256MB	512MB
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有附加文件	是	是	是

提交源程序须加后缀

对于 Pascal 语言	zoo.pas	random.pas	ticket.pas
对于 C 语言	zoo.c	random.c	ticket.c
对于 C++ 语言	zoo.cpp	random.cpp	ticket.cpp

**注意：最终测试时，所有编译命令均不打开任何优化开关。**

## 动物园

### 【问题描述】

近日，园长发现动物园中好吃懒做的动物越来越多了。例如企鹅，只会卖萌向游客要吃的。为了整治动物园的不良风气，让动物们凭自己的真才实学向游客要吃的，园长决定开设算法班，让动物们学习算法。

某天，园长给动物们讲解 KMP 算法。

园长：“对于一个字符串  $S$ ，它的长度为  $L$ 。我们可以在  $O(L)$  的时间内，求出一个名为  $next$  的数组。有谁预习了  $next$  数组的含义吗？”

熊猫：“对于字符串  $S$  的前  $i$  个字符构成的子串，既是它的后缀又是它的前缀的字符串中（它本身除外），最长的长度记作  $next[i]$ 。”

园长：“非常好！那你能举个例子吗？”

熊猫：“例  $S$  为 abcababc，则  $next[5]=2$ 。因为  $S$  的前 5 个字符为 abcab，ab 既是它的后缀又是它的前缀，并且找不到一个更长的字符串满足这个性质。同理，还可得出  $next[1]=next[2]=next[3]=0$ ， $next[4]=next[6]=1$ ， $next[7]=2$ ， $next[8]=3$ 。”

园长表扬了认真预习的熊猫同学。随后，他详细讲解了如何在  $O(L)$  的时间内求出  $next$  数组。

下课前，园长提出了一个问题：“KMP 算法只能求出  $next$  数组。我现在希望求出一个更强大  $num$  数组——对于字符串  $S$  的前  $i$  个字符构成的子串，既是它的后缀同时又是它的前缀，并且该后缀与该前缀不重叠，将这种字符串的数量记作  $num[i]$ 。例如  $S$  为 aaaaa，则  $num[4]=2$ 。这是因为  $S$  的前 4 个字符为 aaaa，其中 a 和 aa 都满足性质‘既是后缀又是前缀’，同时保证这个后缀与这个前缀不重叠。而 aaa 虽然满足性质‘既是后缀又是前缀’，但遗憾的是这个后缀与这个前缀重叠了，所以不能计算在内。同理， $num[1]=0, num[2]=num[3]=1, num[5]=2$ 。”

最后，园长给出了奖励条件，第一个做对的同学奖励巧克力一盒。听了这句话，睡了一节课的企鹅立刻就醒过来了！但企鹅并不会做这道题，于是向参观动物园的你寻求帮助。你能否帮助企鹅写一个程序求出  $num$  数组呢？

特别地，为了避免大量的输出，你不需要输出  $num[i]$  分别是多少，你只需要输出  $\prod_{i=1}^L (num[i] + 1)$  对 1,000,000,007 取模的结果即可。

其中  $\prod_{i=1}^L (num[i] + 1) = (num[1] + 1) \times (num[2] + 1) \times \cdots \times (num[L] + 1)$ 。

### 【输入格式】

从文件 **zoo.in** 中读入数据。

输入文件的第 1 行仅包含一个正整数  $n$ ，表示测试数据的组数。

随后  $n$  行，每行描述一组测试数据。每组测试数据仅含有一个字符串  $S$ ， $S$  的定义详见题目描述。数据保证  $S$  中仅含小写字母。

输入文件中不会包含多余的空行，行末不会存在多余的空格。

【输出格式】

输出到文件 `zoo.out` 中。

输出文件应包含  $n$  行，每行描述一组测试数据的答案，答案的顺序应与输入数据的顺序保持一致。对于每组测试数据，仅需要输出一个整数，表示这组测试数据的答案对  $1,000,000,007$  取模的结果。

输出文件中不应包含多余的空行。

【样例输入 1】

```
3
aaaaa
ab
abcababc
```

【样例输出 1】

```
36
1
32
```

【样例输入输出 2】

见选手目录下的 `zoo/zoo.in` 与 `zoo/zoo.ans`。

【数据规模与约定】

所有测试点的范围和特点如下表所示

测试点编号	约定
1	$n \leq 5, L \leq 50$
2	$n \leq 5, L \leq 200$
3	$n \leq 5, L \leq 200$
4	$n \leq 5, L \leq 10,000$
5	$n \leq 5, L \leq 10,000$
6	$n \leq 5, L \leq 100,000$
7	$n \leq 5, L \leq 200,000$
8	$n \leq 5, L \leq 500,000$
9	$n \leq 5, L \leq 1,000,000$
10	$n \leq 5, L \leq 1,000,000$

## 随机数生成器

### 【问题描述】

小 H 最近在研究随机算法。随机算法往往需要通过调用随机数生成函数（例如 Pascal 中的 `random` 和 C/C++ 中的 `rand`）来获得随机性。事实上，随机数生成函数也并不是真正的“随机”，其一般都是利用某个算法计算得来的。

比如，下面这个二次多项式递推算法就是一个常用算法：

算法选定非负整数  $x_0, a, b, c, d$  作为随机种子，并采用如下递推公式进行计算。

$$\text{对于任意 } i \geq 1, \quad x_i = (a \cdot x_{i-1}^2 + b \cdot x_{i-1} + c) \bmod d$$

这样可以得到一个任意长度的非负整数数列 $\{x_i\}_{i \geq 1}$ ，一般来说，我们认为这个数列是随机的。

利用随机序列 $\{x_i\}_{i \geq 1}$ ，我们还可以采用如下算法来产生一个 1 到  $K$  的随机排列 $\{T_i\}_{i=1}^K$ ：

- 1、初始设  $T$  为 1 到  $K$  的递增序列；
- 2、对  $T$  进行  $K$  次交换，第  $i$  次交换，交换  $T_i$  和  $T_{(x_i \bmod i)+1}$  的值。

此外，小 H 在这  $K$  次交换的基础上，又额外进行了  $Q$  次交换操作，对于第  $i$  次额外交换，小 H 会选定两个下标  $u_i$  和  $v_i$ ，并交换  $T_{u_i}$  和  $T_{v_i}$  的值。

为了检验这个随机排列生成算法的实用性，小 H 设计了如下问题：

小 H 有一个  $N$  行  $M$  列的棋盘，她首先按照上述过程，通过  $N \times M + Q$  次交换操作，生成了一个  $1 \sim N \times M$  的随机排列 $\{T_i\}_{i=1}^{N \times M}$ ，然后将这  $N \times M$  个数逐行逐列依次填入这个棋盘：也就是第  $i$  行第  $j$  列的格子上所填入的数应为  $T_{(i-1) \cdot M + j}$ 。

接着小 H 希望从棋盘的左上角，也就是第一行第一列的格子出发，每次向右走或者向下走，在不走出棋盘的前提下，走到棋盘的右下角，也就是第  $N$  行第  $M$  列的格子。

小 H 把所经过格子上的数字都记录了下来，并从小到大排序，这样，对于任何一条合法的移动路径，小 H 都可以得到一个长度为  $N + M - 1$  的升序序列，我们称之为路径序列。

小 H 想知道，她可能得到的字典序最小的路径序列应该是怎样的呢？

### 【输入格式】

从文件 `random.in` 中读入数据。

输入文件的第 1 行包含 5 个整数，依次为  $x_0, a, b, c, d$ ，描述小 H 采用的随机数生成算法所需的随机种子。

第 2 行包含三个整数  $N, M, Q$ ，表示小 H 希望生成一个 1 到  $N \times M$  的排列来填入她  $N$  行  $M$  列的棋盘，并且小 H 在初始的  $N \times M$  次交换操作后，又进行了  $Q$  次额外的交换操作。

接下来  $Q$  行，第  $i$  行包含两个整数  $u_i, v_i$ ，表示第  $i$  次额外交换操作将交换  $T_{u_i}$

和  $T_{v_i}$  的值。

**【输出格式】**

输出到文件 *random.out* 中。

输出一行，包含  $N + M - 1$  个由空格隔开的正整数，表示可以得到的字典序最小的路径序列。

**【样例输入 1】**

```
1 3 5 1 71
3 4 3
1 7
9 9
4 9
```

**【样例输出 1】**

```
1 2 6 8 9 12
```

**【样例输入 2】**

```
654321 209 111 23 70000001
10 10 0
```

**【样例输出 2】**

```
1 3 7 10 14 15 16 21 23 30 44 52 55 70 72 88 94 95 97
```

**【样例输入 3】**

```
123456 137 701 101 10000007
20 20 0
```

**【样例输出 3】**

```
1 10 12 14 16 26 32 38 44 46 61 81 84 101 126 128 135
140 152 156 201 206 237 242 243 253 259 269 278 279 291 298
338 345 347 352 354 383 395
```

**【样例说明】**

对于样例 1，根据输入的随机种子，小 H 所得到的前 12 个随机数  $x_i$  为：

9 5 30 11 64 42 36 22 1 9 5 30

根据这 12 个随机数，小 H 在进行初始的 12 次交换操作后得到的排列为：

6 9 1 4 5 11 12 2 7 10 3 8

在进行额外的 3 次交换操作之后，小 H 得到的最终的随机排列为：

12 9 1 7 5 11 6 2 4 10 3 8

这个随机排列可以得到如右侧的棋盘：

最优路径依次经过的数字为：  
12→9→1→6→2→8。

12	9	1	7
5	11	6	2
4	10	3	8

对于样例 3，由于卷面宽度不够，在样例输出中出现了换行。请注意，这里的换行仅作展示用途，事实上，样例输出有且仅有一行，所有的数字都应该出现在同一行中。

#### 【样例输入输出 4】

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

#### 【数据规模与约定】

所有测试数据的范围和特点如下表所示

测试点编号	$N, M$ 的规模	$Q$ 的规模	约定
1	$2 \leq N, M \leq 8$	$Q = 0$	$0 \leq a \leq 300$
2	$2 \leq N, M \leq 200$		
3			
4	$2 \leq N, M \leq 2000$	$0 \leq Q \leq 50000$	$0 \leq b, c \leq 10^8$
5			
6			
7	$2 \leq N, M \leq 5000$		$0 \leq x_0 < d \leq 10^8$
8			
9			
10			

#### 【特别提示】

本题的空间限制是 256 MB，请务必保证提交的代码运行时所使用的总内存空间不超过此限制。

一个 32 位整数（例如 C/C++ 中的 int 和 Pascal 中的 Longint）为 4 字节，因而在程序中声明一个长度为  $1024 \times 1024$  的 32 位整型变量的数组，将会占用 4 MB 的内存空间。

## 购票

### 【问题描述】

今年夏天, NOI 在 SZ 市迎来了她 30 周岁的生日。来自全国  $n$  个城市的 OIer 们都会从各地出发, 到 SZ 市参加这次盛会。

全国的城市构成了一棵以 SZ 市为根的有根树, 每个城市与它的父亲用道路连接。为了方便起见, 我们将全国的  $n$  个城市用 1 到  $n$  的整数编号。其中 SZ 市的编号为 1。对于除 SZ 市之外的任意一个城市  $v$ , 我们给出了它在这棵树上的父亲城市  $f_v$  以及到父亲城市道路的长度  $s_v$ 。

从城市  $v$  前往 SZ 市的方法为: 选择城市  $v$  的一个祖先  $a$ , 支付购票的费用, 乘坐交通工具到达  $a$ 。再选择城市  $a$  的一个祖先  $b$ , 支付费用并到达  $b$ 。以此类推, 直至到达 SZ 市。

对于任意一个城市  $v$ , 我们会给出一个交通工具的距离限制  $l_v$ 。对于城市  $v$  的祖先  $a$ , 只有当它们之间所有道路的总长度不超过  $l_v$  时, 从城市  $v$  才可以通过一次购票到达城市  $a$ , 否则不能通过一次购票到达。对于每个城市  $v$ , 我们还会给出两个非负整数  $p_v, q_v$  作为票价参数。若城市  $v$  到城市  $a$  所有道路的总长度为  $d$ , 那么从城市  $v$  到城市  $a$  购买的票价为  $dp_v + q_v$ 。

每个城市的 OIer 都希望自己到达 SZ 市时, 用于购票的总资金最少。你的任务就是, 告诉每个城市的 OIer 他们所花的最少资金是多少。

### 【输入格式】

从文件 *ticket.in* 中读入数据。

输入文件的第 1 行包含 2 个非负整数  $n, t$ , 分别表示城市的个数和数据类型 (其意义将在后面提到)。

输入文件的第 2 到  $n$  行, 每行描述一个除 SZ 之外的城市。其中第  $v$  行包含 5 个非负整数  $f_v, s_v, p_v, q_v, l_v$ , 分别表示城市  $v$  的父亲城市, 它到父亲城市道路的长度, 票价的两个参数和距离限制。

请注意: 输入不包含编号为 1 的 SZ 市, 第 2 行到第  $n$  行分别描述的是城市 2 到城市  $n$ 。

### 【输出格式】

输出到文件 *ticket.out* 中。

输出包含  $n - 1$  行, 每行包含一个整数。其中第  $v$  行表示从城市  $v + 1$  出发, 到达 SZ 市最少的购票费用。

同样请注意: 输出不包含编号为 1 的 SZ 市。

### 【样例输入 1】

```
7 3
1 2 20 0 3
```

```

1 5 10 100 5
2 4 10 10 10
2 9 1 100 10
3 5 20 100 10
4 4 20 0 10

```

## 【样例输出 1】

```

40
150
70
149
300
150

```

## 【样例说明 1】

样例如右图所示。

从每个城市出发到达 SZ 的路线如下(其中箭头表示一次直达):

城市 2: 只能选择  $2 \rightarrow 1$ , 花费为  $2 \times 20 + 0 = 40$ 。

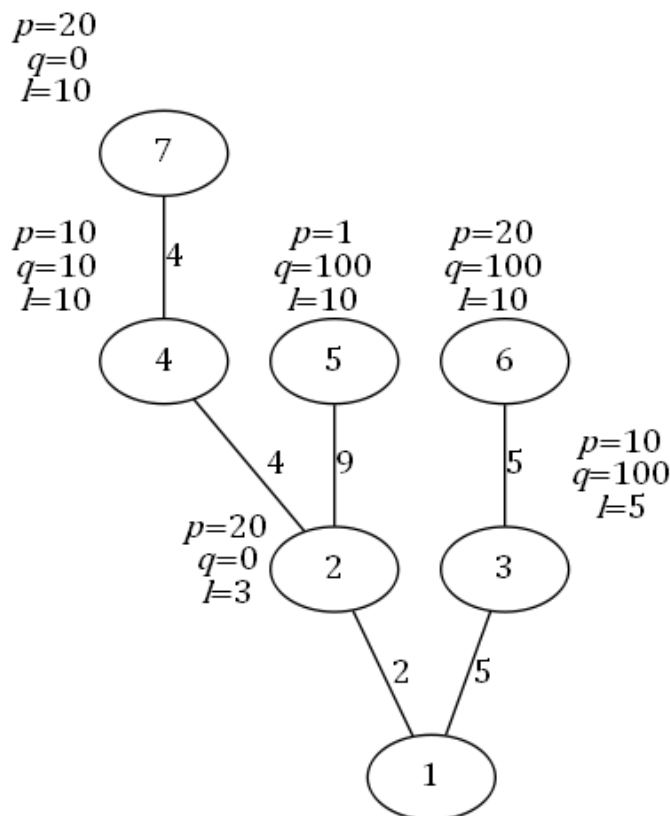
城市 3: 只能选择  $3 \rightarrow 1$ , 花费为  $5 \times 10 + 100 = 150$ 。

城市 4: 由于  $4 + 2 = 6 \leq l_4 = 10$ , 故可以选择  $4 \rightarrow 1$ 。若选择  $4 \rightarrow 1$ , 花费为  $(4 + 2) \times 10 + 10 = 70$ ; 若选择  $4 \rightarrow 2 \rightarrow 1$ , 则花费为  $(4 \times 10 + 10) + (2 \times 20 + 0) = 90$ ; 因此选择  $4 \rightarrow 1$ 。

城市 5: 只能选择  $5 \rightarrow 2 \rightarrow 1$ , 花费为  $(9 \times 1 + 100) + (2 \times 20 + 0) = 149$ ; 无法选择  $5 \rightarrow 1$ , 因为  $l_5 = 10$ , 而城市 5 到城市 1 总路程为  $9 + 2 = 11 > l_5$ , 城市 5 不能直达城市 1。

城市 6: 若选择  $6 \rightarrow 1$ , 花费为  $(5 + 5) \times 20 + 100 = 300$ ; 若选择  $6 \rightarrow 3 \rightarrow 1$ , 花费为  $(5 \times 20 + 100) + (5 \times 10 + 100) = 350$ ; 因此选择  $6 \rightarrow 1$ 。

城市 7: 选择  $7 \rightarrow 4 \rightarrow 1$ , 花费为  $(4 \times 20 + 0) + ((4 + 2) \times 10 + 10) = 150$ ; 其他方案均比该方案差。





## 【样例输入输出 2】

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

该组样例按照城市的编号几乎平均地被分为了 4 个部分，每个部分有不同的特点。你可以使用它们进行针对性的测试。

## 【数据规模与约定】

对于所有测试数据，保证  $0 \leq p_v \leq 10^6$ ， $0 \leq q_v \leq 10^{12}$ ， $1 \leq f_v < v$ ；保证  $0 < s_v \leq l_v \leq 2 \times 10^{11}$ ，且任意城市到 SZ 市的总路程长度不超过  $2 \times 10^{11}$ 。

输入的  $t$  表示数据类型， $0 \leq t < 4$ ，其中：

当  $t = 0$  或  $2$  时，对输入的所有城市  $v$ ，都有  $f_v = v - 1$ ，即所有城市构成一个以 SZ 市为终点的链；

当  $t = 0$  或  $1$  时，对输入的所有城市  $v$ ，都有  $l_v = 2 \times 10^{11}$ ，即没有移动的距离限制，每个城市都能到达它的所有祖先；

当  $t = 3$  时，数据没有特殊性质。

每组测试数据的  $n$  和  $t$  如下所示

测试点编号	$n$	$t$
1	$n = 2 \times 10$	$t = 2$
2	$n = 2 \times 10^3$	$t = 0$
3		$t = 3$
4	$n = 2 \times 10^5$	$t = 0$
5		$t = 2$
6		$t = 1$
7		
8		
9		$t = 3$
10		

## 【特别提示】

最终评测时，调用栈占用的空间大小不会有单独的限制。如果你的程序涉及到调用栈溢出的问题，请阅读 *ticket/stack.pdf*。请注意，调用栈占用的空间会计入总空间占用中，和程序其他部分占用的内存共同受到内存限制。

数据的输入输出需要用到 64 位整型。如果你在计算中需要用到两个 64 位整型相乘，请务必注意结果是否会溢出。