全国青少年信息学奥林匹克竞赛

CCF-NOIP-2018

提高组（复赛）模拟试题

中文题目名称	楼梯问题	数独	疏散演习
英文题目名称	stair	sudoku	practice
输入文件名	stair.in	sudoku.in	practice.in
输出文件名	stair.out	sudoku.out	practice.out
每个测试点时限	1 秒	1 秒	1 秒
内存限制	512MB	512MB	512MB
测试点数目	20	10	20
每个测试点分值	5	10	5
结果比较方式	全文比较（过滤行末空格及文末回车）		
题目类型	传统	传统	传统

提交源程序须加后缀

对于 Pascal 语言	stair.pas	sudoku.pas	practice.pas
对于 C　　语言	stair.c	sudoku.c	practice.c
对于 C++　语言	stair.cpp	sudoku.cpp	practice.cpp

注意：最终测试时，所有编译命令均不打开任何优化开关。

楼梯问题

【问题描述】

一层楼共有 n 级台阶，一次可以上至少 1 级但不超过 m 级台阶，求有多少种不同的上楼方案数。由于结果可能很大，你只需要输出结果对 10,007 取模的值即可。

【输入格式】

输入数据只有一行，包含两个正整数 n 和 m。

【输出格式】

一个整数，表示所求结果对 10,007 取模的值。

【样例输入 1】

```
4 3
```

【样例输出 1】

```
7
```

【样例说明 1】

共有 7 种不同的上楼梯方案，分别为：

1. 第 1 步上 1 级，第 2 步上 1 级，第 3 步上 1 级，第 4 步上 1 级。
2. 第 1 步上 1 级，第 2 步上 1 级，第 3 步上 2 级。
3. 第 1 步上 1 级，第 2 步上 2 级，第 3 步上 1 级。
4. 第 1 步上 2 级，第 2 步上 1 级，第 3 步上 1 级。
5. 第 1 步上 2 级，第 2 步上 2 级。
6. 第 1 步上 1 级，第 2 步上 3 级。
7. 第 1 步上 3 级，第 2 步上 1 级。

【样例输入 2】

```
1024 5
```

【样例输出 2】

```
8590
```

【数据规模与约定】

所有测试点的数据规模与约定如下：

测试点编号	n 的规模	m 的规模
1	$1 \le n \le 10$	$m = 1$
2		
3		
4		
5		$m = 2$
6		
7		
8		
9		$m = 3$
10		
11	$1 \le n \le 10{,}000$	$2 \le m \le 10$
12		
13	$1 \le n \le 100{,}000$	
14		
15	$n = 233{,}333{,}333$	$m = 5$
16	$n = 666{,}666{,}666$	$2 \le m \le 5$
17	$1 \le n \le 10^9$	$m = 2$
18	$1 \le n \le 10^{12}$	
19	$1 \le n \le 10^{15}$	$2 \le m \le 10$
20	$1 \le n \le 10^{18}$	

数独

【问题描述】

数独是一个有趣的游戏。你需要在一个 9×9 的矩阵中的每个格子中填入 1～9 的数字，使得没有两个相同的数字填入同一行、同一列或同一个九宫格中。

整个矩阵被划分为 9 个九宫格，若两个格子同时在最左三列、最右三列或中间三列，且同时在最左三行、最右三行或中间三行，则这两个格子在同一九宫格中。

如果两个相同的数同行、同列或同九宫格，则构成一对冲突。如下列状态中，两个 1 在同一行中，两个 2 在同一列中，两个 3 在同一九宫格中，分别是三对冲突；但两个 4 不是一对冲突。

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

现在有一个数独的初始状态，出题人想对其进行一些修改和询问操作。需要注意：在操作时，初始状态中的数也可以被删除或者合并时被替换。

1. 向目前状态中的指定位置填入一个数：但有可能这个位置已经有一个数了，此时你需要输出一行 `Error!`，然后不进行这次修改操作；在指定的这个位置没有数的情况下，这个数已经与之前存在的在同一行、列或九宫格中的数构成冲突，此时，你需要按照行、列、九宫格的顺序，找到第一种冲突的情况，输出一行 `Error:row!`，`Error:column!` 或 `Error:square!`，然后不进行这次修改操作；否则，你需要输出 `OK!`，并在指定位置填入该数。

2. 删除目前状态中的一个位置上的数：若这个位置没有数字，此时你需要输出一行 `Error!`，然后不进行任何操作；否则你需要输出一行 `OK!`，并将该位置的数删除。

3. 查询目前状态中的某个位置能填入多少种数字；若被查询的位置已经有数字了，你需要输出一行 `Error!`；否则，输出一行一个整数 n 表示能填

入的数字个数，随后 *n* 行每行一个整数，按照从小到大的顺序输出能填入的数字。

4. 将之前的第 *i* 次操作后的数独状态和第 *j* 次操作后的数独状态进行合并，作为当前状态。需要注意：对于所有的 5 种操作，包括但不限于出现 `Error!`或是没有进行任何修改，均被算作一次操作。合并时以行为第一关键字，列为第二关键字的顺序依次考虑每个格子，若第 *i* 次操作后的数独状态中该位置有数且不会与之前冲突则优先填入；否则，在不会与之前冲突的情况下，填入第 *j* 次操作后的数独状态中该位置的数。若均没有数字或均与本次合并中已填入的数字冲突，则不填入任何数。输出一行，包含空格隔开的两个整数，表示最终的结果中有多少数字来自第 *i* 次操作后的数独状态中，多少来自第 *j* 次操作后的数独状态中。

5. 查询整个数独的状态，你需要使用方阵格式将整个数独目前的状态输出。方阵格式是一个 19×19 的二维字符数组，具体格式如下，其中用 0 表示该位置还未填入数字。

```
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|5|7|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|7|1|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|9|8|7|6|5|4|3|2|1|
+-+-+-+-+-+-+-+-+-+
```

【输入格式】

输入的前 19 行为一个二维字符数组，为数独的初始状态的方阵格式。

随后一行一个整数 T 表示操作的次数。

随后 T 行，每行为下列形式：

`Insert x y k`，表示在(x, y)位置插入数 k。

`Delete x y`，表示删除(x, y)位置的数。

`Query x y`，表示查询(x, y)位置能填入且不会出现冲突的数。

`Merge i j`，表示合并第 i 次操作后的状态和第 j 次操作后的状态。

`Print`，表示查询整个数独的状态。

其中 x 表示行数，从上到下分别为 1 到 9，y 表示列数，从左到右分别为 1 到 9。

【输出格式】

对于每个操作，你需要按照题目描述进行对应的输出。

【样例输入输出 1】

见题目目录下的 **1.*in*** 与 **1.*ans***。

【样例输入输出 2】

见题目目录下的 **2.*in*** 与 **2.*ans***。

该样例的数据规模与第 6 / 7 个测试点相同。

【数据规模与约定】

所有测试点的数据规模与约定如下：

<table>
<tr><th>测试点</th><th>约定 1</th><th>约定 2</th><th>约定 3</th></tr>
<tr><td>1</td><td rowspan="3">否</td><td>否</td><td rowspan="5">否</td></tr>
<tr><td>2</td><td rowspan="2">是</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="7">是</td><td rowspan="2">否</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td rowspan="2">是</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td>否</td><td rowspan="3">是</td></tr>
<tr><td>9</td><td rowspan="2">是</td></tr>
<tr><td>10</td></tr>
</table>

约定 1：存在插入和删除操作。

约定 2：存在查询单个格子的操作。

约定 3：存在合并操作。

对于所有的数据，$1 \le T \le 100$，$1 \le x, y, k \le 9$，对于第 a 个操作，若是 Merge 操作，则$1 \le i, j < a$。保证第一个操作不是 Merge 操作。

对于所有的数据，均可能存在查询整个数独的操作，且保证初始状态不存在冲突。

疏散演习

【问题描述】

小 A 所在的高中每年都要进行疏散演习，以应对未来可能的突发情况。在以前的演习中，每个班级的学生都从自己的班级出发，最终全部聚集到操场上。对此，小 A 认为所有人都到达操场会使得场地变得拥挤，他想请你来解决下面的问题。

我们设学校中共有 n 个区域，不同的区域之间由 $n-1$ 条无向边连接，构成一棵无根树，其中第 i 个区域有 a_i 个人。在一次演习中，第 x 个区域和第 y 个区域将被作为**聚集点**。定义第 i 个区域到第 j 个区域的距离为 $D(i,j)$，这个值表示从第 i 个区域到第 j 个区域所经过的最少边数。定义 $C(i,j)$ 表示将区域 i 的人全部转移到区域 j 所需要的**转移代价**，$C(i,j)=a_i\times D(i,j)$。定义**总转移代价** $T=\sum_{i=1}^{n}\min\{C(i,x),C(i,y)\}$。小 A 想请你来决定 x 和 y 的值，使得 T 的值最小。

【输入格式】

第一行包括一个正整数 n 表示区域数。

接下来 $n-1$ 行每行两个正整数 u 和 v，表示区域 u 和区域 v 之间有一条无向边连接。

接下来一行包含 n 个正整数，第 i 个正整数为 a_i，表示第 i 个区域的人数。

【输出格式】

只有一个整数，表示 T 的最小值。

【样例输入 1】

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

【样例输出 1】

```
14
```

【样例说明 1】

当 $x=2$，$y=3$ 时，$T=14$。

【样例输入 2】

```
12
1 2
1 3
1 4
3 5
4 6
6 7
4 8
8 9
6 10
4 11
10 12
116 585 895 707 897 331 94 634 935 875 360 77
```

【样例输出 2】

```
7545
```

【数据规模与约定】

所有测试点的数据规模如下：

测试点编号	n 的规模
1	$n = 2$
2	$n = 3$
3	$n = 50$
4	$n = 100$
5	$n = 300$
6	$n = 500$
7	$n = 800$
8	$n = 1{,}000$
9	$n = 3{,}000$
10	$n = 5{,}000$
11	$n = 8{,}000$
12	$n = 10{,}000$
13	$n = 15{,}000$
14	$n = 20{,}000$
15	$n = 25{,}000$
16	$n = 30{,}000$
17	$n = 35{,}000$
18	$n = 40{,}000$
19	$n = 45{,}000$
20	$n = 50{,}000$

对于全部测试数据满足 $1 \le a_i \le 1{,}000$。

【关于数据中无根树的生成方式】

首先随机生成一个 1 到 n 的排列 x_i。对于第 i 条边，令这条边的某一个端点为 $i + 1$，另一个端点为取值在 1 到 i 之间的某个随机数，我们设此时这条边的两个端点分别为 p 和 q，接下来执行 $p \leftarrow x_p$，$q \leftarrow x_q$。最后将这 $n - 1$ 条边打乱输出至输入文件中。