SN 省队训练 4th

题目名称	动态树	蛇	排序
输入文件名	tree.in	snake.in	sorting.in
输出文件名	tree.out	snake.out	sorting.out
每个测试点时限	2s	1s	1s
内存限制	512 MB	512 MB	512 MB
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	无	无	无
题目类型	传统型	传统型	传统型

提交源程序须加后缀

对于 Pascal 语言	.pas	.pas	.pas
对于 C　　语言	.c	.c	.c
对于 C++　语言	.cpp	.cpp	.cpp

注意：最终测试时，所有编译命令均不打开任何优化开关。

动态树（tree）

【问题描述】

动态树就是那些会动的树。

现在有一棵 n 个节点的树，它的根被标号为 1，对于树上每个节点，你会得到一张表以表示它的儿子。

现在，你需要完成以下三种操作：

1. 输出节点 u 到节点 v 的距离。

2. 给出 x,h 将节点 x 与其父亲之间的边断开并连到其第 h 个祖先上作为 其最后一个儿子节点。我们将 x 到根节点的路径列出：a1,a2,...,ak(h<k)，那么 a1=x 且 ak=1，断开 x 与 a2 之间的边将其与 x(h+1)相连。

3. 输出这棵树 dfs 序列中最后一个与根节点距离为 k 的节点编号。

为了体现在线思想，每次操作输入的数都要异或上一次的答案（**操作编号不异或**）。初始时设答案为 0。

【输入格式】(tree.in)

第一行两个整数 n,m，表示节点数为 n，共 m 次操作。

接下来 n 行，每行第一个整数 si 表示 i 号节点的儿子数量，之后 si 个正整数表示节点 i 的儿子。

之后 m 行每行一种操作。

【输出格式】(tree.out)

对于每次询问输出一个答案，每个答案占一行。

【样例输入】

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

【样例输出】

```
1
2
```

【数据说明】

对于 20%的数据，n,m≤1000。

另有 10%的数据，仅存在操作 1。

另有 10%的数据，操作 2 中 h≤10。

对于 100%的数据，n,m≤10^5。

蛇（snake）

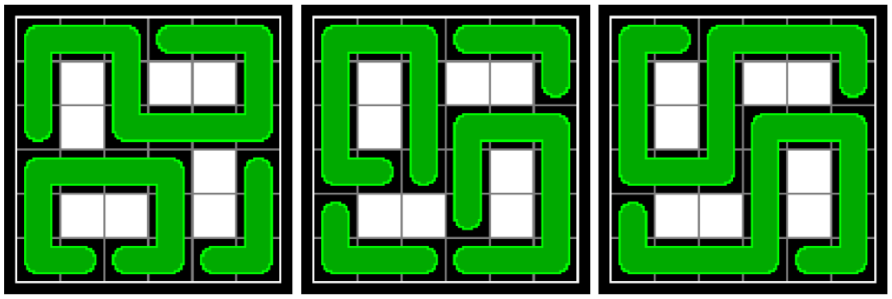
【问题描述】

一些蛇覆盖了一个网格。每个格子要么是一个障碍物，要么是蛇的一部分。每条蛇占据了一条折线 (拐角处只能水平和竖直连接)，且至少占据两个格子。蛇与蛇之间不重叠，蛇也不会与自己重叠。每条蛇还必须满足以下两个条件中的一个：

- 两个端点所在的格子在网格的边界。这样的蛇至少长度为 2。
- 蛇构成一个环，即两个端点相邻(垂直或水平)。注意这个条件意味着一条构成环的蛇至少需要占据 4 个格子。

现在我们给定一个网格，请你求出在满足前面所述的条件下覆盖所有空地，并使得端点在网格边界 (即不构成环) 的蛇尽量少。

例如以下是几种合法的覆盖方案(白色格子代表障碍物，黑色格子代表蛇所在的格子)。



我们知道还有其他方案存在，但对于上图中的网格，不存在仅用一条或更少不构成环的蛇覆盖整个网格的方案。

【输入格式】(snake.in)

输入文件第一行包括两个整数 n 和 m，分别表示网格的行数和列数。

接下来 n 行，每行 m 个字符描述了整个网格，其中井号(#)表示障碍物，点号(.)表示空地。

【输出格式】(snake.out)

输出一行，为合法覆盖方案中，端点在网格边界蛇的最少数量。

如果不存在合法的方案，输出-1。

【样例输入】

```
7 8
#.....##
...#....
........
..#.#.#.
......#.
```

...#..#.
#.....

【样例输出】

1

【数据说明】

30%的数据满足 max{n,m}≤6。
另外 30%的数据满足 min{n,m}≤12。
100%的数据满足 1≤n,m≤50。

排序（sorting）

【问题描述】

给定一个序列，我们定义位置编号从 1 开始。你每次可以将位置 i 的元素移动到位置 j，该操作不改变其他元素的相对位置。即如果 i>j，则位于[j,i)的元素的位置都比原来+1；如果 i<j，则位于(i,j]的元素的位置都比原来-1。一次移动的代价定义为 i+j。请编程确定一个移动序列，使得元素按照从高到低排序，使得总代价最小。可以认为元素是两两不同的。

【输入格式】(sorting.in)

输入第一行为一个正整数 N，表示序列长度。
之后 N 行每行一个非负整数，表示序列中的每个元素 Ai。

【输出格式】(sorting.out)

输出第一行为一个整数表示移动的最小总代价。
输出第二行为一个整数表示满足总代价最小时的最小移动次数。

【样例输入】

```
5
20
30
5
15
10
```

【样例输出】

```
11
2
```

【数据说明】

30%的数据满足 N≤10，操作步数≤10。
100%的数据满足 N≤1000，Ai≤10^6