样题

样题

时间：2016 年 7 月 24 日 08:00 ~ 12:00

题目名称	面试	扫雷	多项式求和
题目类型	传统型	传统型	传统型
目录	interview	mine	polynomial
可执行文件名	interview	mine	polynomial
输入文件名	interview.in	mine.in	polynomial.in
输出文件名	interview.out	mine.out	polynomial.out
每个测试点时限	1 秒	1 秒	1 秒
内存限制	512 MB	512 MB	512 MB
测试点数目	20	20	10
每个测试点分值	5	5	10

提交源程序文件名

对于 C++ 语言	interview.cpp	mine.cpp	polynomial.cpp
对于 C 语言	interview.c	mine.c	polynomial.c
对于 Pascal 语言	interview.pas	mine.pas	polynomial.pas

编译选项

对于 C++ 语言	-lm	-O2 -lm	-O2 -lm
对于 C 语言	-lm	-O2 -lm	-O2 -lm
对于 Pascal 语言		-O2	-O2

面试（interview）

【题目描述】

生活在在外星球 X 上的小 Z 想要找一些小朋友组成一个舞蹈团，于是他在网上发布了信息，一共有 n 个人报名面试。

面试必须按照报名的顺序依次进行。小 Z 可以选择在面试完若干小朋友以后，在所有已经面试过的小朋友中进行任意顺序的挑选，以组合成一个舞蹈团。

虽然说是小朋友，但是外星球 X 上的生态环境和地球上的不太一样，这些小朋友的身高可能相差很大。小 Z 希望组建的这个舞蹈团要求至少有 m 个小朋友，并且这些小朋友的最高身高和最低身高之差不能超过 k 个长度单位。

现在知道了这些小朋友的身高信息，问小 Z 至少要面试多少小朋友才能在已经面试过的小朋友中选出不少于 m 个组成舞蹈团。

【输入格式】

从文件 ***interview.in*** 中读入数据。

第一行 3 个整数 n, m, k，意义见题面描述；$1 \le m \le n \le 10^5; 0 \le k \le 10^5$；

第二行 n 个整数，第 i 个数 h_i 表示第 i 个报名面试的小朋友的身高，$1 \le h_i \le 10^5$。

【输出格式】

输出到文件 ***interview.out*** 中。

如果可以选出舞蹈团，输出至少要面试多少人；否则输出 `impossible`。

【样例 1 输入】

```
6 3 5
170 169 175 171 180 175
```

【样例 1 输出】

```
4
```

【样例 1 解释】

当面试了前 4 个小朋友之后，这些小朋友的身高分别为 $170, 169, 175, 171$，可选出身高为 $170, 175, 171$ 的小朋友组成舞蹈团，故只用面试 4 个小朋友即可。

【样例 2 输入】

```
6 4 5
170 169 175 171 180 175
```

【样例 2 输出】

```
6
```

【样例 2 解释】

在这个样例中，小 Z 需要面试所有小朋友，才能选出身高为 $170, 175, 171, 175$ 的小朋友组成舞蹈团。

【样例 3 输入】

```
6 5 5
170 169 175 171 180 175
```

【样例 3 输出】

```
impossible
```

【样例 4】

见选手目录下的 ***interview/interview4.in*** 与 ***interview/interview4.ans***。

【子任务】

本题目一共 20 个测试点，所有测试点均不开启 **O2** 优化。

测试点编号	n, m	h_i, k
1,2	$1 \le m \le n \le 100$	$k = 0; 1 \le h_i \le 100$
3,4	$1 \le m \le n \le 2 \times 10^3$	$0 \le k \le 50; 1 \le h_i \le 100$
5,6,7,8		$0 \le k \le 100; 1 \le h_i \le 5 \times 10^3$
9,10,11,12		$0 \le k \le 5 \times 10^3; 1 \le h_i \le 5 \times 10^3$
13,14	$1 \le m \le n \le 2 \times 10^3$	$0 \le k \le 10^5; 1 \le h_i \le 10^5$
15,16	$1 \le m \le n \le 10^5$	$0 \le k \le 100; 1 \le h_i \le 10^5$
17,18,19,20	$1 \le m \le n \le 10^5$	$0 \le k \le 10^5; 1 \le h_i \le 10^5$

扫雷（mine）

【题目描述】

扫雷（minesweeper）是一个有趣的单人益智类游戏，游戏目标是在最短的时间内根据棋盘上的提示信息，找出所有非雷方块，同时避免踩到地雷。随着桌面操作系统 Windows 的流行，其自带的扫雷游戏也因为有趣的玩法、精致的画面受到大家的欢迎。

小 L 的电脑上曾经也有一个扫雷游戏，它和主流的扫雷游戏基本相似，但是有一些不同的地方，具体介绍如下：

游戏开始时，玩家可以看到 $N \times M$ 个整齐排列的空白方块，玩家须根据棋盘已有的信息，运用逻辑推理来推断哪些方块含或不含地雷。

1. 玩家可以用鼠标左键点击空白方块，表示推断这个方块没有地雷，尝试探明它。
   - 如果玩家点开没有地雷的方块，会有一个数字显现其上，这个数字代表着八连通的相邻方块有多少颗地雷（至多为 8）
   - 如果这个方块八连通的方块中没有地雷（也即，方块显示的数字为 0），则系统会自动帮玩家点开它相邻的方块，这个过程可能会引起连锁反应。
   - 如果玩家点开有地雷的方块，则游戏结束，玩家失败。
2. 玩家可在推测有地雷的方块上点鼠标右键，表示放置旗帜来标明地雷的位置；在有旗帜的方块上再次点击右键，会使旗帜消失，成为空白的方块。在已标明旗帜的方块点击左键，方块不会有任何的变动。若在游戏进行中错置旗帜，可以用右键来改变方块状态。
3. 玩家可以在一个已探明的方块上同时点击左键及右键。此时，如果方块相邻的 8 个方块放置旗帜的数目与方块上的数字相同，那么周围未探明的方块就会自动打开。然而，玩家若错置旗帜位置，此动作可能会打开真正藏有地雷的方块，导致游戏失败。不过这样的点击动作可加快游戏速度以便得到高分。

然而，年代久远，小 L 已经找不到当年陪他度过十年求学时光的扫雷游戏了，于是他找到了精通编程的你，希望你能帮他写一个简单的扫雷游戏，帮助他回忆那些快乐时光。

具体来说，你的程序应该读入一个地雷布置图。然后读入用户的每一次游戏操作，并在每次操作后给用户以反馈，帮助用户进行游戏。

【输入格式】

从文件 ***mine.in*** 中读入数据。

约定：我们用坐标 (x,y) 表示棋盘第 x 行、第 y 列的方块。

第一行用空格隔开的两个整数 n,m，表示棋盘的规模。

接下来 n 行，每行一个长为 m 的字符串，描述棋盘，其中第 i 行的第 j 个字符表示棋盘的方块 (i,j)。为 `*` 表示方块里有一个地雷，为 `.` 表示方块是安全的。

接下来每一行按时间顺序描述每一次用户操作，直到文件结束。每一行的格式如下：

1. 首先读入一个字符串，表示这次操作的内容：
   - Flag：表示右键点击某个方块，插上／撤销一面旗帜。
   - Sweep：表示左键点击某个方块，判断这个方块没有地雷，要探明之。
   - DSweep：表示左右键同时点击某个方块，尝试探明与它相邻的方块。
   - Quit：表示放弃本局游戏并退出。
2. 若操作不为 Quit，则之后有空格隔开的两个整数 x,y，表示这次操作的坐标为 (x,y)，保证 $1 \le x \le n$, $1 \le y \le m$。

输入数据保证存在有且仅有一次 Quit 操作。

【输出格式】

输出到文件 ***mine.out*** 中。

对每一次操作，向标准输出打印一行或多行，表示此次操作的反馈。具体格式如下：

1. 若读入了 Quit，忽略之后的所有输入，结束本局游戏，输出结束信息（见第 8 条）。
2. 对 Flag 操作：
   - 如果对应方块已经被探明，输出一行 `swept`。
   - 如果对应方块未被探明，插上旗帜，输出一行 `success`。
   - 如果对应方块上有旗帜，清除之，输出一行 `cancelled`。
3. 对 Sweep 操作：
   - 如果对应方块已经被探明，输出一行 `swept`。
   - 如果对应方块上有旗帜，输出一行 `flagged`。
   - 如果对应方块未被探明，进行扫雷过程，根据扫雷的结果，输出反馈信息（见第 5 6 条）。
4. 对 DSweep 操作：
   - 如果对应方块未被探明，输出一行 `not swept`。
   - 如果对应方块数字为 0、或者它八连通的方块的旗帜数不等于方块显示的数，输出一行 `failed`。

- 否则，对方块八连通的每个空白方块进行扫雷过程，所有扫雷过程结束之后，根据扫雷的结果，输出反馈信息（见第 6 7 条）。

5. 扫雷过程，假设要对 (x,y) 进行扫雷：

- 如果 (x,y) 为地雷，扫雷失败。输出一行 `boom`。接着，忽略之后的所有输入，结束本局游戏，输出结束信息（见第 8 条）。
- 否则，标记这个方块为“已探明”，令这个方块显示它相邻的方块的地雷总数。如果它相邻的方块不存在地雷，则自动对它相邻的没有探明的方块进行扫雷（此时，清除它的相邻方块上的旗帜信息），这个过程可能会引起连锁反应。

6. 对 Sweep 操作，在扫雷过程成功结束之后输出扫雷反馈；对 DSweep 操作，在所有的扫雷过程（可能是 0 次）成功结束之后输出扫雷反馈，格式如下：

- 如果没有任何新方块被探明（可能在 DSweep 时发生），输出一行：`no cell detected`。
- 否则，设有 *num_of_cells* 个新方块被探明，首先输出一行：`NUM_OF_CELLS cell(s) detected`，其中 `NUM_OF_CELLS` 应该输出本次操作探明的方块数，请注意括号的输出。
- 接下来 *num_of_cells* 行，将所有新探明的方块按照所在行为第一关键字，所在列为第二关键字，从小到大排序输出，每一行输出空格隔开的三个整数 x,y,c，其中 x,y 表示方块的坐标，c 表示方块上显示的数字。

7. 若某次 Sweep / DSweep 操作结束之后，所有没有地雷的方块均被探明，忽略之后的所有输入，结束本局游戏，输出结束信息（见第 8 条）。

8. 结束信息的输出格式：

- 首先，输出游戏胜负情况：
- 若所有没有地雷的方块均被探明，输出一行：`finish`；
- 若踩到雷而结束游戏，输出一行：`game over`；
- 若因为 Quit 而结束游戏，输出一行：`give up`。
- 之后，计算玩家使用的行动次数 *total_step*，每次成功／不成功的 Flag, Sweep, DSweep 均视为一次行动，Quit 不算一次行动，输出一行：`total step TOTAL_STEP`，其中 `TOTAL_STEP` 应该输出行动次数。

注意：请特别注意各项输出的拼写和空格，否则将可能导致程序错误直至零分。

【样例 1 输入】

```
3 3
...
..*
...
Sweep 1 1
```

```
DSweep 1 2
Flag 1 3
Flag 2 3
DSweep 1 2
Sweep 1 3
Flag 1 1
DSweep 1 3
Flag 1 3
DSweep 1 2
DSweep 1 2
Sweep 3 3
Quit
```

【样例 1 输出】

```
6 cell(s) detected
1 1 0
1 2 1
2 1 0
2 2 1
3 1 0
3 2 1
failed
success
success
failed
flagged
swept
not swept
cancelled
1 cell(s) detected
1 3 1
no cell detected
1 cell(s) detected
3 3 1
finish
total step 12
```

【样例 1 解释】

第一组数据展示了一个在简单的 3×3 棋盘上进行的游戏过程，样例输出中展示了上文提到的绝大部分输出信息。

【样例 2】

见选手目录下的 ***mine/mine2.in*** 与 ***mine/mine2.ans***。

【样例 2 解释】

第二组数据展示了一种因为错误的 Flag 操作和 DSweep 操作而导致游戏失败的情况。

【样例 3】

见选手目录下的 ***mine/mine3.in*** 与 ***mine/mine3.ans***。

【样例 3 解释】

第三组数据展示了一种因为 Quit 操作而结束游戏的情况，注意，当游戏结束之后，你的程序应该输出结束信息，并忽略之后的所有操作。

【子任务】

共有 20 个测试点，每个测试点满分为 5 分。

我们令 n, m 表示棋盘的规模，q 表示输入的操作次数，有以下约定：

测试点	n	m	q	性质
1 ~ 2	≤ 10	≤ 10	≤ 60	A
3 ~ 4	≤ 10	≤ 10	≤ 60	B
5 ~ 6	≤ 10	≤ 10	≤ 60	无
7 ~ 8	$= 1$	≤ 1000	≤ 1000	A
9 ~ 10	$= 1$	≤ 1000	≤ 1000	B
11 ~ 12	$= 1$	≤ 1000	≤ 1000	无
13 ~ 14	≤ 300	≤ 300	≤ 8000	A
15 ~ 16	≤ 300	≤ 300	≤ 8000	B
17 ~ 19	≤ 300	≤ 300	≤ 8000	无
20	≤ 1000	≤ 1000	≤ 60000	无

性质 A：保证只有 Sweep 操作和 Quit 操作。

性质 B：保证没有 DSweep 操作。

注意：对于规模较大的数据，请不要使用过于缓慢的输出方式。

多项式求和（polynomial）

【问题描述】

小 K 最近刚刚习得了一种非常酷炫的多项式求和技巧，可以对某几类特殊的多项式进行运算。

非常不幸的是，小 K 发现老师在布置作业时抄错了数据，导致一道题并不能用刚学的方法来解，于是希望你能帮忙写一个程序跑一跑。

给出一个 m 阶多项式

$$f(x)=\sum_{i=0}^{m} b_i x^i$$

对给定的正整数 a ，求

$$S(n)=\sum_{k=0}^{n} a^k f(k)$$

由于这个数可能比较大，所以你只需计算 $S(n)$ 对 10^9+7 取模后的值（即计算除以 10^9+7 后的余数）。

【输入格式】

从文件 ***polynomial.in*** 中读入数据。

第一行包含三个整数 n,m,a。

第二行包含 $m+1$ 个整数，$b_0,b_1,\ldots,b_m$ 描述给定多项式的系数。

对于所有数据，$1\le a,b_i\le 10^9$。

【输出格式】

输出到文件 ***polynomial.out*** 中。

输出一行一个数，表示 $S(n)$ 对 10^9+7 取模后的结果。

【样例 1 输入】

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

【样例 1 输出】

```
9658
```

【样例 1 解释】

$f(x)=1+x+x^2$，故 $f(0)=1, f(1)=3, f(2)=7, f(3)=13, f(4)=21, f(5)=31$。

$f(0)+3f(1)+9f(2)+27f(3)+81f(4)+243f(5)=1+3*3+9*7+27*13+81*21+243*31=9658$。

【样例 2 输入】

```
100 3 233
1 2 3 4
```

【样例 2 输出】

```
994811687
```

【样例 3 输入】

```
20170314 10 11037
1 2 3 4 5 6 7 8 9 10 11
```

【样例 3 输出】

```
133604769
```

【子任务】

测试点	n	m	a
$1\sim 2$	≤ 1000	≤ 10	$\le 10^9$
3	$\le 10^9$	$=1$	$=1$
4	$\le 10^9$	$=2$	$=1$
5	$\le 10^9$	$=3$	$\le 10^9$
6	$\le 10^9$	$=5$	$=1$
$7\sim 8$	$\le 10^9$	≤ 20	$=1$
9	$\le 10^9$	≤ 50	$\le 10^9$
10	$\le 10^9$	≤ 100	$\le 10^9$