省选模拟赛

zzq

2018.03.29

题目名称	修序列 1	修序列 2	修墙
源程序文件名	var.c/cpp/pas	hidden.cpp	wall.c/cpp/pas
输入文件名	var.in	N/A	wall.in
输出文件名	var.out	N/A	wall.out
时间限制	1s	1s	2s
是否捆绑测试	是	是	是
内存限制	1GB	1GB	1GB
是否有部分分	否	是	否
题目类型	传统	交互	传统
是否有附加文件	否	是	是
编译开关	-O2 -std=c++11	-O2 -std=c++11	-O2 -std=c++11

注意：0.**今天代码长度限制为 10kb，请务必注意。**

1. 评测时的栈空间大小不做单独限制, 但使用的总空间大小不能超过内存限制。

2. 今天题目很简单，请大家不要假，认真做题。

3.AK 了不要 D 出题人，没 AK 也不要 D 出题人。

4. 今天在 zzq 的电脑上评测，所有题目时限均为 std 运行最大点用时两倍以上。

5. 今天可能会送出一份神秘礼物，条件见第二题题面。

Problem A. 修序列 1(var.c/cpp/pas)

Input file:	var.in
Output file:	var.out
Time limit:	1 seconds
Memory limit:	1 gigabytes

$w \times h$ 是一位神仙。

一天，他打算出一套 noip 普及组模拟。他想了五秒钟，就想到了一个绝妙的签到题：给一个长度为 n 的数列，你需要算出它的所有子区间中不同方差的种类数，$1 \leq n \leq 500$。为了照顾没学过方差的普及组选手，他还在题目中给出了方差的定义：对于一个序列 $a_1, a_2 ... a_n$，记它的平均数 $\overline{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$，则方差为 $\frac{1}{n}\sum_{i=1}^{n}(a_i - \overline{a})^2$。

zzq 菜鸡看到了 $w \times h$ 的这道题，打算搬到良心模拟赛里当第三题。他看了看 $w \times h$ 生成的数据，发现每个点的 out 依次是 12 34 56 78 233 666 2333 6666 23333 66666 ，感觉十分疑惑。因为 zzq 水平不行，所以于是 zzq 希望让你也帮帮他生成指定 out 的数据。

一句话题意：你需要构造一个长度不超过 500 的任意序列，它的非空子区间有恰好 k 种不同的方差。由于 zzq 懒得写高精度，你构造的序列中的每个数需要在 $[0, 2333333]$ 范围内。

Input

一行一个正整数 k。

Output

第一行一个正整数 n，表示你构造的序列的长度。你需要保证 $1 \leq n \leq 500$。

第二行 n 个用空格分割的非负整数，依序给出你构造的序列中的元素，每个元素需要在 $[0, 2333333]$ 范围内。

Examples

var.in	var.out
3	3 2 3 3

Notes

子任务编号	子任务分值	$k \leq$
1	10	20
2	10	100
3	20	500
4	30	10000
5	30	100000

Problem B. 修数列 2(hidden.cpp)

Input file:	stdin
Output file:	stdout
Time limit:	1 seconds
Memory limit:	1 gigabytes

这是一道交互题，本题仅支持 C++。

$w \times h$ 是一位神仙。

愚人节就要到了，为了让大家感受到愚人节的乐趣，$w \times h$ 设置了一个神秘交互库来测试大家的智商。

交互库中藏匿了一个长度为 n 的 01 串，你每次可以给交互库一个**长度不超过 n**的 01 序列，交互库会告诉你这个序列是不是给定串的子序列，你需要猜出那个串。

$w \times h$ 认为用调用交互库次数来评分太老套了，他决定设置传新的评分方式，于是你可以调用不超过 500000 次交互库，你的得分将由你给交互库的 01 序列的长度最大值决定。如果这个值 $\leq \lfloor n/2 \rfloor + 1$ 你会获得满分，具体评分方式详见 Notes 部分。

Interactor Notes

选手目录中有 `hidden.hpp` 、`sample.cpp` 两个文件，以下是详细说明，如果懒得看的话你也可以直接参照 `sample.cpp` 编写代码。

你需要在你代码的开头 `#include "hidden.hpp"` 来与交互库交互，你不应该实现 main 函数或试图进行任何文件输入输出，你只需要实现一个函数：`std::vector<int> guess(int n)`，它接受 n，返回一个长度为 n 的`std::vector` 表示你猜出的 01 序列。

你可以使用交互库提供的一个函数：`bool is_subsequence(std::vector<int> s)`，这个函数接受一个长度不超过 n 的由 01 组成的`std::vector s`，返回这个序列是否是原串的子序列。

样例交互库（下发的`hidden.hpp`）会从标准输入作为字符串读入 01 序列，把它的长度作为 n。

Notes

子任务编号	子任务分值	$n \in$
1	30	$[7, 10]$
2	70	$[90, 100]$

交互库不会占用超过 0.25s 时间和 64MB 内存。

交互库是 non-adaptive 的，即交互库在调用你的 guess 函数之前就会生成好 01 串，不会随着你的询问而改变。

每个 subtask 包含若干数据点，你的得分为这些数据点上得分的最小值。

对于每个数据点，如果你的程序进行了不合法的调用（长度超过 n、传输了非 01 序列、调用了超过 500000 次）或没有猜出正确的 01 序列，你会得到 0 分。

否则，设你询问的序列长度最大值为 a，$b = \lfloor n/2 \rfloor + 1$，该数据点满分为 x（30 或 70）。

若 $a \leq b$，你会获得 x 分。否则，你会获得 $\lfloor 0.9x \times 0.32^{a/b-1} \rfloor$ 分。

本题数据点数较多，希望大家不要交运行时间过长的代码。（例如如果你不会 subtask 2，建议特判一下并退出）

Examples

stdin	stdout
0000100101	Well done! You made aaa queries with max length bbb. You can get ccc of full score.

样例输出仅供参考。请注意这个样例是对样例交互库而言的，真实评测时使用的交互库可能与下发的有所不同。

Bonus

本题满分的选手中最后一个 subtask 最后一个点调用交互库次数最少的选手将获得神秘礼物一份，如果有多个满足条件的选手取（lemon 显示的）三题总运行时间最短的一位。

Problem C. 修墙 (wall.c/cpp/pas)

Input file:	`wall.in`
Output file:	`wall.out`
Time limit:	1 seconds
Memory limit:	1 gigabytes

$w \times h$ 是一位神仙。

众所周知，近日长城修得越来越高，`across the Great Wall, reach every corner in the world` 也变得越来越困难。

$w \times h$ 想要分析一下某个版本墙的特征。在地图上，土地可以大致用一个无限大的黑白二维矩阵表示，其中用户为白格，墙为黑格。由于墙很高，两个用户能够互相通信当且仅当在网格上这两个白格能够只经过四联通的白格相互到达。

经过一番细致的研究，$w \times h$ 发现这个地图可以由如下过程迭代构造。

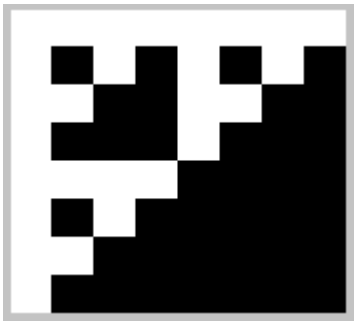
地图上一开始只有一个白格。

一次迭代中，假设原来的地图为 A，那么新的地图为

```
AA
AB
```

其中 B 为一个与 A 边长相同且全为黑格的正方形矩阵。

例如三次迭代之后，我们得到了这样一个矩阵（W 表示白格，B 表示黑格）：



经过无限次迭代之后，我们就得到了土地对应的黑白矩阵。

无限大的地图，分析起来过于困难。于是 $w \times h$ 每次会截出一个子矩阵，他想要知道这个子矩阵中的用户在墙的阻隔下组成了多少个联通块，联通块定义为极大的能够互相通信的用户集合。

由于一次询问不足以分析，$w \times h$ 需要进行 q 次询问。

Input

第一行一个正整数 q。

接下来 q 行每行四个正整数 x_1, y_1, x_2, y_2 （$1 \le x_1 \le x_2, 1 \le y_1 \le y_2$），表示询问以第 x_1 行第 y_1 列，第 x_2 行第 y_2 列为两对角的矩形内用户形成的联通块数。

Output

q 行，每行一个正整数，表示对应矩形内的联通块数量。

Examples

wall.in	wall.out
4 1 1 5 5 2 2 3 6 4 2 7 5 2 2 2 2	1 3 3 0

需要注意的是如果矩形内没有用户应该输出 0。

Notes

子任务编号	子任务分值	$q \leq$	$x_2, y_2 \leq$
1	20	300	300
2	20	3000	3000
3	30	100000	10^9
4	30	1000000	10^9

由于输入输出量较大，选手目录下有`io.cpp`，选手可以任意使用。