

全国青少年信息学奥林匹克联赛

NOIP2024 模拟赛

时间：2024年11月16日 8:00~12:30

题目名称	棋盘游戏	挂分	移球游戏	逃出迷宫
题目类型	传统题	传统题	传统题	传统题
目录	distorted	fate	abstruse	dilemma
可执行文件名	distorted	fate	abstruse	dilemma
输入文件名	distorted.in	fate.in	abstruse.in	dilemma.in
输出文件名	distorted.out	fate.out	abstruse.out	dilemma.out
每个测试点时限	1.0 秒	1.0 秒	2.0 秒	2.0 秒
内存限制	512 MiB	512 MiB	512 MiB	2048 MiB
子任务数目	25	14	10	9
子任务是否等分	是	否	否	否

提交源程序文件名

对于 C++ 语言	distorted.cpp	fate.cpp	abstruse.cpp	dilemma.cpp
-----------	---------------	----------	--------------	-------------

编译选项

对于 C++ 语言	-O2 -std=c++14 -static
-----------	------------------------

【注意事项（请仔细阅读）】

- 文件名（程序名和输入输出文件名）必须使用英文小写。
- C++ 中函数 `main()` 的返回值类型必须是 `int`，程序正常结束时的返回值必须是 `0`。
- 因违反以上两点而出现的错误或问题，申诉时一律不予受理。
- 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
- 若无特殊说明，输入一行内的内容均使用空格进行分隔。
- 选手提交的程序源文件必须不大于 100KB。
- 程序可使用的栈空间内存限制与题目的内存限制一致。
- 只提供 Linux 格式附加样例文件。
- 禁止在源代码中改变编译器参数（如使用 `#pragma` 命令），禁止使用系统结构相关指令（如内联汇编）和其他可能造成不公平的方法。
- 如果您水平吊打出题人，或者见过原题等导致您光速 AK 了，请不要大声喧哗。

棋盘游戏 (distorted)

【题目背景】

题目背景被季风吹走了，快使用魔法手杖重塑时光，找回丢失的题目背景！

【题目描述】

小 t 有一个 $n \times n$ 的棋盘，其中 n 一定是奇数，小 z 在这个棋盘的每个格子上放入了一个填有正整数的卡片。

他们邀请你一起和他进行一个游戏：现在你可以取走若干个卡片，当你取走某个格子上的卡片时，会以这个格子为中心生成一个 $n \times n$ 的矩形，并覆盖在这个棋盘上。当棋盘上的所有格子都被生成的矩形覆盖时，你会获得一个用于疏通马桶和用来解决电车问题的马桶搋子。

你想要获得这个马桶搋子，并且你希望，在能够获得马桶搋子的同时，最小化你取出的卡片上的数字总和。

【输入格式】

从文件 *distorted.in* 中读入数据。

本题单测试点内有多组测试数据。

第一行一个正整数 T ，表示数据组数。

对于每组数据，第一行一个正整数 n ，表示棋盘的大小。

接下来 n 行，每行 n 个正整数，第 i 行 j 列格子上的卡片上的数。

【输出格式】

输出到文件 *distorted.out* 中。

对于每组数据，输出一行，表示在获得马桶搋子的前提下最小的取出卡片上的数字的总和。

【样例 1 输入】

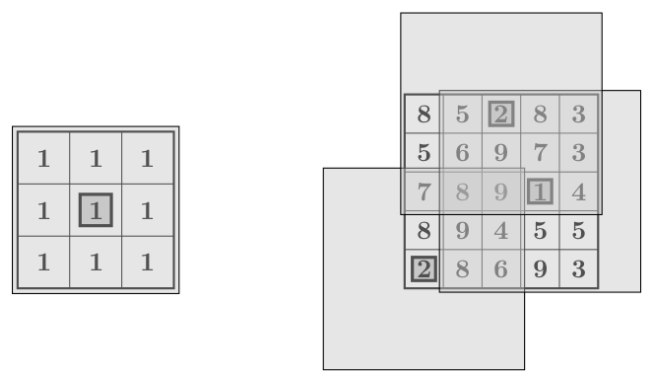
```
2
3
1 1 1
1 1 1
1 1 1
5
8 5 2 8 3
5 6 9 7 3
7 8 9 1 4
8 9 4 5 5
2 8 6 9 3
```

【样例 1 输出】

```
1
5
```

【样例 1 解释】

下面的图展示了两组数据取卡片的最优方案。



【样例 2】

见选手目录下的 *distorted/distorted2.in* 和 *distorted/distorted2.ans*。

【数据范围】

测试点编号	$n \leq$	$\sum n^2 \leq$
1 ~ 2	1	10
3	3	50
4 ~ 5	5	100
6 ~ 7	7	100
8 ~ 10	9	500
11 ~ 15	15	1000
16 ~ 18	49	10000
19 ~ 21	99	100000
22 ~ 25	499	500000

对于所有数据，保证： $1 \leq n \leq 499, 1 \leq \sum n^2 \leq 500000$ ， n 是奇数，每个卡片上的数为 $[1, 10^9]$ 中的正整数。

挂分 (fate)

【题目背景】

在上个题的题目背景中，你获得了一个马桶搋子。

从现在开始，你可以将用这个马桶搋子剑指出题人，在比赛结束时，如果你觉得这场是马桶，那么你可以拉动它，让电车驶向出题人。不然的话，电车就会自然地驶向曾经的各位马桶豪杰。

这就是出题人的命运啊！一场比赛后出题人的生死，完全掌握在各位坐题人的手中，至于这是不是咎由自取，那就成为一个谜题了。

【题目描述】

回调时间线，那个时候出题人小 t 还只是一个普普通通的打模拟赛选手，而不是现在的毒瘤出题人。

机房里有 n 个人，编号为 $1 \sim n$ ，小 t 的编号为 x 。

现在举行了一场模拟赛，第 i 个人预计得分为 $a_i (a_i \geq 0)$ 分，由于编号是按选手实力排的，因此对于编号分别为 i, j 的选手，如果 $i < j$ 则有 $a_i \leq a_j$ 。

由于挂分是模拟赛的一大特色，因此每个人的最终得分不一定是 a_i 分，设每个人挂了 $b_i (b_i \leq a_i)$ 分，则每个人的最终得分为 $a_i - b_i$ 分。

但是现在出现了一个神秘现象，由于小 t 的存在，挂分并不是一件不确定的事，具体地说，小 t 总是不会挂分（也就是 $b_x = 0$ ），而从小 t 两侧开始，靠近小 t 的人挂的分会由以多到少的形式出现。形式化地，若 $i < j < x$ ，则 $b_i \leq b_j$ ，若 $x < j < i$ ，则也有 $b_i \leq b_j$ ，而对于 $i < x < j$ 的情况， b_i, b_j 的大小关系不确定。

更加神奇的是，在挂完分之后，每个人的排名不会发生变化！也就是对于 $i < j$ ，有 $a_i - b_i \leq a_j - b_j$ 。

你对这个现象感到很神奇，因此你想知道，有多少种本质不同的挂分方式满足上面的情况，称两种挂分方式本质不同，当且仅当存在一个人，在这两种挂分方式中挂的分不同，由于答案很大，因此你只要输出答案对 998244353 取模的结果。

【输入格式】

从文件 `fate.in` 中读入数据。

第一行一个正整数 n ，表示机房人数。

第二行 n 个自然数 a_i ，表示每个人的预计得分。

第三行一个正整数 x ，表示小 t 的位置。

【输出格式】

输出到文件 `fate.out` 中。

一行一个自然数，表示答案对 998244353 取模的结果。

【样例 1 输入】

```
3
1 2 3
2
```

【样例 1 输出】

```
4
```

【样例 1 解释】

可能的挂分情况有 $\{0, 0, 0\}, \{1, 0, 0\}, \{0, 0, 1\}, \{1, 0, 1\}$ 。

注意 $\{1, 0, 2\}$ 不是合法的挂分情况，因为第三个人挂了 2 分后，得分低于第二个人。

注意 $\{1, 1, 1\}$ 不是合法的挂分情况，因为第 $x = 2$ 个人不能挂分。

注意 $\{2, 2, 2\}$ 不是合法的挂分情况，因为第一个人不能挂掉比自己原本更高的分。

【样例 2 输入】

```
5
1 1 2 3 5
4
```

【样例 2 输出】

```
12
```

【样例 3】

见选手目录下的 *fate/fate3.in* 和 *fate/fate3.ans*。

【样例 4】

见选手目录下的 *fate/fate4.in* 和 *fate/fate4.ans*。

【数据范围】

本题捆绑测试。

子任务编号	$n \leq$	$a_i \leq$	$x =$	得分
1	9	9	—	4
2	18	18	9	8
3	200	200	1	4
4	3000	3000	1	8
5	3000	10^6	1	8
6	3×10^5	10^8	1	12
7	200	200	n	4
8	3000	3000	n	8
9	3000	10^6	n	8
10	3×10^5	10^8	n	12
11	200	200	—	4
12	3000	3000	—	4
13	3000	10^6	—	4
14	3×10^5	10^8	—	12

其中 $x = -$ 的部分表示不对 x 有特殊限制。

对于所有数据，保证： $2 \leq n \leq 3 \times 10^5, 0 \leq a_i \leq 10^8, 1 \leq x \leq n$ 。

移球游戏 (abstruse)

【题目背景】

《移球游戏》是 NOIP 2020 竞赛中的一道构造题，这道题目深刻考察了选手的思维能力和解决问题的综合能力。题目要求选手在给定的条件下，通过设计和实现一种算法或策略，来完成特定的任务。这不仅仅是对算法技能的考验，更是对选手逻辑推理能力、创新思维以及问题分析能力的全面检验。选手需要运用数学和计算机科学中的相关知识，合理规划解决方案，并考虑多种可能的情况和优化策略，从而展示出他们在面对复杂问题时的系统性思维和创造力。

《移球游戏》是 NOIP 2020 竞赛中的一道挑战性极高的构造题，旨在深度考察选手的综合思维能力。题目设置不仅要求选手具备扎实的基础算法知识，还需要他们能够进行深入的分析 and 设计。选手必须根据题目提供的复杂条件，设计出有效的算法来解决问题，同时要考虑到各种可能的优化策略和边界情况。这道题目考验了选手的逻辑推理能力、创新思维以及将理论应用于实际问题的能力，从而全面检验他们的数学建模和程序设计能力，是对他们能力的终极挑战。

但是题目标题并不是移球游戏，这是为什么呢？

【题目描述】

有一个 n 行 S 列的网格，其中 S 一定是 2 的正整数幂次，网格的每个格子上都有一个颜色在 $1 \sim T$ 的小球。

你可以进行如下操作若干次：

- 选择位于同一行的两个球，交换他们的位置。

你希望在进行若干次操作后，最后的网格能够五彩斑斓，因此你希望，对于某种颜色和任意两列，这个颜色的球在这两列的出现次数差不超过 1。

求一种操作方案，你只需要输出操作后的网格即可。

数据保证，一定存在一种合法的操作方案。

【输入格式】

从 `abstruse.in` 中读入数据。

第一行两个正整数 n, S, T ，表示网格的行，列，以及球的颜色种数。

接下来 n 行，每行 S 个在 $1 \sim T$ 范围内的正整数，表示网格对应位置内球的颜色。

【输出格式】

输出到文件 `abstruse.out` 中。

输出 n 行，每行 S 个 $1 \sim T$ 的正整数，表示最终网格对应每个位置的球的颜色。

本题采用 Special Judge 评测，只要你的方案合法即判定为正确。

【样例 1 输入】

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

【样例 1 输出】

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

【样例 2 输入】

```
3 4 3
2 3 2 2
2 3 3 2
2 2 3 2
```

【样例 2 输出】

```
2 2 2 3
3 2 3 2
2 3 2 2
```

【样例 3】

以下所有样例均不下发答案文件。

见选手目录下 *abstruse/abstruse3.in*。

该样例满足子任务 2 的限制。

【样例 4】

见选手目录下 *abstruse/abstruse4.in*。

该样例满足子任务 3 的限制。

【样例 5】

见选手目录下 *abstruse/abstruse5.in*。

该样例满足子任务 5 的限制。

【样例 6】

见选手目录下 `abstruse/abstruse6.in`。

该样例满足子任务 6 的限制。

【数据范围】

本题捆绑测试。

子任务编号	限制	特殊性质	得分
1	$S = 2, n, T \leq 20$	—	10
2	$S = 2$	AB	15
3	$S = 2$	A	5
4	$S = 2$	—	5
5	$nS \leq 10^4$	AB	15
6	$nS \leq 10^4$	A	5
7	$nS \leq 10^4$	—	5
8	$nS \leq 5 \times 10^5$	AB	20
9	$nS \leq 5 \times 10^5$	A	10
10	$nS \leq 5 \times 10^5$	—	10

注意对于子任务 3,4,6,7,9,10，它们都依赖与自己编号减一的子任务。

特殊性质 A: $T \leq n$ 。

特殊性质 B: 对于所有颜色，每种颜色的小球个数都能整除 S 。

对于所有数据，保证： $1 \leq n, S, T \leq 10^5, nS \leq 5 \times 10^5$ ，存在一个正整数 k 使得 $S = 2^k$ 。

【提示】

本题下放了 Special Judge，你可以编译下发的 `checker.cpp` 然后在命令行中运行：

```
checker.exe <input_file> <output_file> <output_file>
```

（windows 环境下）

或者

```
./checker <input_file> <output_file> <output_file>
```

（linux 环境下）

来检验你的输出是否正确。

其中 `<input_file>` 为输入文件，`<output_file>` 为你的输出文件。

逃出迷宫 (dilemma)

【题目背景】

请选手注意部分数据结构的寻址常数对效率的影响。

【题目描述】

你被困在一个迷宫之中。

迷宫可以用一个长度为 n 且值域在 $[1, n]$ 的序列 $a_{1..n}$ 描述。

你在迷宫的一个区间 $[l, r]$ 上，现在你试图逃出迷宫。

你可以在迷宫中进行一次跳跃，这会使 $l \leftarrow \min(a_l, a_{l+1}, \dots, a_{r-1}, a_r)$, $r \leftarrow \max(a_l, a_{l+1}, \dots, a_{r-1}, a_r)$ ，注意这两个赋值操作是同时进行的。

如果任何时候 $l = 1, r = n$ ，则说明你逃出了迷宫。

有 q 次询问，第 i 次询问如果你从区间 $[l_i, r_i]$ 开始，你需要进行多少次跳跃才能逃出迷宫，特别的，如果你无法逃出迷宫请报告。

【输入格式】

第一行两个正整数 n, q ，表示迷宫对应的序列长度和询问数。

接下来一行 n 个正整数，表示序列 $a_{1..n}$ 。

接下来 q 行，每行两个正整数 l_i, r_i ，表示第 i 次询问你所在的区间。

【输出格式】

对于每次询问，如果能逃出迷宫，则输出逃出迷宫所需的次数，否则输出 -1 。

【样例 1 输入】

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

【样例 1 输出】

-1
0
1
2
3
4

【样例 1 解释】

给出每次询问的逃离迷宫的过程：

- 询问 1：容易验证其不能逃出迷宫。
- 询问 2：不需要跳跃即可逃出迷宫。
- 询问 3： $[1, 4] \rightarrow [1, 5]$ 。
- 询问 4： $[3, 5] \rightarrow [1, 4] \rightarrow [1, 5]$ 。
- 询问 5： $[4, 5] \rightarrow [1, 3] \rightarrow [2, 5] \rightarrow [1, 5]$ 。
- 询问 6： $[2, 3] \rightarrow [4, 5] \rightarrow [1, 3] \rightarrow [2, 5] \rightarrow [1, 5]$ 。

【样例 2,3,4,5,6,7】

见选手目录下的 *dilemma/dilemma2,3,4,5,6.in* 和 *dilemma/dilemma2,3,4,5,6.ans*。

其中样例 6 满足特殊性质 B。

【数据范围】

本题捆绑测试。

子任务编号	$n \leq$	$q \leq$	特殊性质	得分
1	100	100	—	8
2	3000	2000	—	8
3	800	10^5	—	8
4	3000	10^5	—	12
5	20000	10^5	—	12
6	10^5	10^5	A	8
7	10^5	10^5	B	8
8	10^5	10^5	C	12
9	10^5	10^5	—	24

特殊性质 A：保证 $a_{1\dots n}$ 在所有长度为 n 的排列中均匀随机生成，所有询问在所有满足 $l \leq r$ 的二元组中均匀生成。

特殊性质 B：保证 $r_i - l_i = 1$ 。

特殊性质 C：保证所有询问给出的区间交的结果非空。

对于所有数据，保证： $1 \leq n, q \leq 10^5$ ， $1 \leq a_i \leq n$ ， $1 \leq l_i \leq r_i \leq n$ 。

【提示】

请选手注意部分数据结构的寻址常数对效率的影响。