冲刺 CCF NOI2025 全国赛模拟题

梦熊集训–Day6

时间：2025 年 07 月 11 日 08:00 ～ 13:00

题目名称	九九	棋盘	数字
题目类型	传统型	传统型	传统型
目录	`nine`	`board`	`number`
可执行文件名	`nine`	`board`	`number`
输入文件名	`nine.in`	`board.in`	`number.in`
输出文件名	`nine.out`	`board.out`	`number.out`
每个测试点时限	1.0 秒	3.0 秒	2.0 秒
内存限制	1024 MiB	1024 MiB	1024 MiB
测试点数目	10	7	9
测试点是否等分	是	否	否

提交源程序文件名

对于 C++ 语言	`nine.cpp`	`board.cpp`	`number.cpp`

编译选项

对于 C++ 语言	`-O2 -std=c++14 -static`

注意事项（请仔细阅读）

1. 选手提交的源程序必须存放在以自己姓名命名的文件夹中，文件名称与对应试题英文名一致。
2. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
3. C++ 中函数 main() 的返回值类型必须是 int，值必须为 0。
4. **对于因未遵守以上规则对成绩造成的影响，相关申诉不予受理。**
5. 若无特殊说明，结果比较方式为**忽略行末空格、文末回车后的全文比较。**
6. 程序可使用的栈空间大小与该题内存空间限制一致。
7. 在终端中执行命令 ulimit -s unlimited 可将当前终端下的栈空间限制放大，但你使用的栈空间大小不应超过题目限制。
8. 每道题目所提交的**代码文件大小限制为 100KB。**
9. 若无特殊说明，输入文件与输出文件中同一行的相邻整数均使用一个空格分隔。
10. 输入文件中可能存在行末空格，请选手使用更完善的读入方式（例如 scanf 函数）避免出错。

11. 直接复制 PDF 题面中的多行样例，数据将带有行号，建议选手直接使用对应目录下的样例文件进行测试。
12. 使用 std::deque 等 STL 容器时，请注意其内存空间消耗。
13. 请务必使用题面中规定的的编译参数，保证你的程序在本机能够通过编译。此外**不允许在程序中手动开启其他编译选项**，一经发现，本题成绩以 0 分处理。
14. 评测时采用的机器配置为：Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz，内存 32GB。上述时限以此配置为准。

九九（nine）

【题目描述】

称所有字符的均为数字字符的串为数字串。

称一个数字串是 9 的倍数，即这个数字串按顺序写下后的数字是 9 的倍数。

现给出数字串长度 n 和 m 个限制，第 i 个限制要求数字串 s 的第 l_i 个字符到第 r_i 个字符构成的子串是 9 的倍数，求长度为 n 且满足所有 m 个限制的数字串 s 的数量对 10^9+7 取模后的结果。

【输入格式】

从文件 ***nine.in*** 中读入数据。

第一行两个整数 n, m，表示数字串长度和限制数量。

接下来 m 行，第 i 行两个整数 l_i, r_i，表示限制。

【输出格式】

输出到文件 ***nine.out*** 中。

一行一个整数表示答案。

【样例 1 输入】

```
4
2
1 2
2 4
```

【样例 1 输出】

```
136
```

【样例 1 解释】

第 2 个位置确定以后，就可以确定第一个位置和 3, 4 两个位置的和。对于每种情况的答案累加即可。

【样例 2】

见选手目录下的 ***nine/nine2.in*** 与 ***nine/nine2.ans***。

【样例 3】

见选手目录下的 ***nine/nine3.in*** 与 ***nine/nine3.ans***。

【样例 4】

见选手目录下的 ***nine/nine4.in*** 与 ***nine/nine4.ans***。

【数据范围】

对于所有数据，保证 $1 \le n \le 10^9$，$1 \le m \le 15$，$1 \le l_i \le r_i \le n$。

测试点编号	$m \le$	特殊限制
1	15	A
2, 3		B
4	3	无
5	5	
6	8	
7	10	
8, 9, 10	15	

特殊性质 A：满足 $n \le 8$；
特殊性质 B：满足 $r_i < l_{i+1}$。

棋盘（board）

【题目描述】

Berlandia 是一个由方格组成的无限大棋盘。行从下到上用递增的整数编号，列从左到右用递增的整数编号。令 (r, c) 表示第 r 行第 c 列的方格。如果两个不同方格至少有一个角接触，我们就称这两个格子相邻。这意味着格子是八连通的。

两个格子 (R_A, C_A) 与 (R_B, C_B) 之间的距离是欧几里得距离，也就是：

$$\sqrt{(R_A - R_B)^2 + (C_A - C_B)^2}$$

Berlandia 地区居住着 n 只熊。第 i 只熊居住在方格 (r_i, c_i) 处。同一方格中可以居住多只熊。

熊可以单独生活，但是有时也会相互靠近。当一只熊吼叫时，其他方格中所有的熊会立即移动到相邻的方格中离吼叫的熊最近的方格。可以证明有且仅有一个这样的方格（不会出现并列情况）。与吼叫的熊在同一方格的熊不会移动位置。

例如，考虑一对熊，一只在方格 $(2, 1)$，另一只在方格 $(4, 8)$。方格 $(2, 1)$ 中的熊吼叫会让另一只熊移向方格 $(3, 7)$ ，这两只熊最后相距 $\sqrt{(3-2)^2 + (7-1)^2} = \sqrt{37}$。

这些熊会按第一只，第二只，……，最后一只的顺序依次吼叫。除了一只叫 Limak 的熊，他太冷了以至于吼不出来，并且他也不能离开他所在的方格，可怜的 Limak。

但你不知道 Limak 是哪只熊，对于 1 到 n 的每一个 k，如果第 k 只熊是 Limak，请找出所有熊的最终位置。对于每种可能，输出所有熊的横纵坐标乘积之和就可以了。也就是说，假设在 $n-1$ 次吼叫后，第 i 只熊在 (r'_i, c'_i)，则输出：

$$\sum_{i=1}^{n} r'_i c'_i$$

【输入格式】

从文件 ***board.in*** 中读入数据。

输入第一行包括一个正整数 n $(2 \le n \le 2.5 \times 10^5)$，表示熊的数量。

接下来 n 行，每行两个整数 r_i, c_i $(1 \le r_i, c_i \le 10^6)$，第 i 行表示第 i 只熊的初始位置。

【输出格式】

输出到文件 ***board.out*** 中。

输出 n 行，第 k 行输出一个整数，表示假设 Limak 是第 k 只熊的话，最终所有熊所在行列之积的和。

【样例 1 输入】

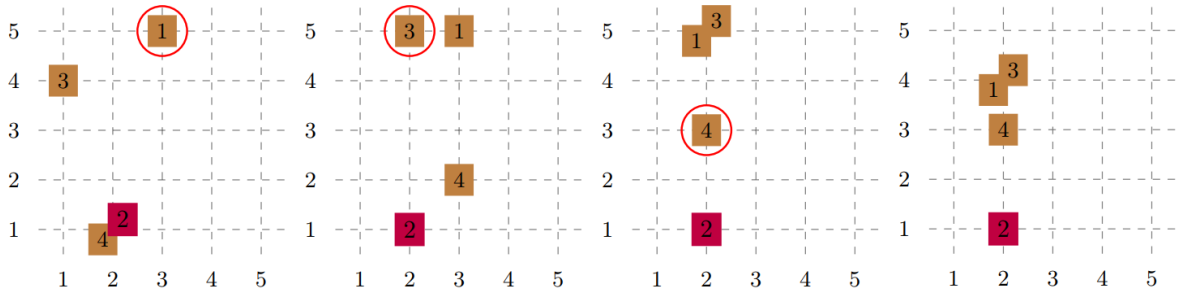
```
4
3 5
2 1
1 4
2 1
```

【样例 1 输出】

```
27
24
25
35
```

【样例 1 解释】

下图展示了 $k = 2$ 的情况，即熊的吼叫顺序为 $1, 3, 4$。红色圆圈表示吼叫的熊。最后乘积的和为 $2 \cdot 4 + 2 \cdot 1 + 2 \cdot 4 + 2 \cdot 3 = 24$。



【样例 2】

见选手目录下的 ***board/board2.in*** 与 ***board/board2.ans***。
该样例满足子任务 3 的限制条件。

【样例 3】

见选手目录下的 ***board/board3.in*** 与 ***board/board3.ans***。
该样例满足子任务 4 的限制条件。

【样例 4】

见选手目录下的 ***board/board4.in*** 与 ***board/board4.ans***。
该样例满足子任务 6 的限制条件。

【数据范围】

本题采用 Subtask 捆绑测试。
对于 100% 的数据，保证 $2 \le n \le 2.5 \times 10^5$，$1 \le r_i, c_i \le 10^6$。

子任务编号	$n \le$	特殊限制	分值
1	2	无	5
2	200		15
3	2000		20
4	2×10^5	A	15
5	2.5×10^5		5
6	10^5	无	20
7	2.5×10^5		20

特殊性质 A：满足 $c_i = 1$。

数字（number）

【题目描述】

给定两个长度为 $3N$ 的序列 A 和 B，它们都恰好包含 $1, 2, \ldots, N$ 中的每个整数各 3 次。换句话说，它们是 $(1,\ 1,\ 1,\ 2,\ 2,\ 2,\ \ldots,\ N,\ N,\ N)$ 的一种重新排列。

你可以对数列 A 任意次执行以下操作：

- 从 $1, 2, \ldots, N$ 中选一个数 x。A 中恰好有 3 个 x，将它们中**中间位置**的那个 x 移动到**整个序列的**开头或末尾。

具体地说，找到 $i,\ j,\ k$ 满足 $A_i = A_j = A_k = x$ 且 $i < j < k$，则 A_j 是**中间位置**的那个 x。具体参照样例解释。

请判断能否通过若干次操作把 A 变成 B。如果可以，请输出最少的操作次数；否则输出 -1。。

【输入格式】

从文件 ***number.in*** 中读入数据。

第一行一个正整数 N。

接下来一行 $3N$ 个正整数为 $A_1, A_2, ..., A_{3N}$。

接下来一行 $3N$ 个正整数为 $B_1, B_2, ..., B_{3N}$。

【输出格式】

输出到文件 ***number.out*** 中。

若可以变换，输出最小操作次数；无解输出 -1。

【样例 1 输入】

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

【样例 1 输出】

```
5
```

【样例 1 解释】

可以这样进行操作：

- 1 2 3 3 2 3 2 1 1：原序列。
- → 1 2 3 3 3 2 1 1 2：将原序列处于中间的 2，即 A_2、A_5、A_7 中的 A_5 移动到序列末尾。
- → 1 2 3 3 3 1 1 2 2：将上一步中间的 2 移动到序列末尾。
- → 1 2 3 3 3 1 2 2 1：将上一步中间的 1 移动到序列末尾。
- → 1 2 3 3 3 2 2 1 3：将上一步中间的 3 移动到序列末尾。
- → 3 1 2 3 1 2 2 1 3：将上一步中间的 3 移动到序列开头。

共进行 5 次操作即可变换成功。

【样例 2 输入】

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

【样例 2 输出】

```
6
```

【样例 3】

见选手目录下的 ***number/number3.in*** 与 ***number/number3.ans***。

【样例 4】

见选手目录下的 ***number/number4.in*** 与 ***number/number4.ans***。

【样例 5】

见选手目录下的 ***number/number5.in*** 与 ***number/number5.ans***。

【数据范围】

对于 100% 的数据，满足 $N \leq 100$。

子任务编号	N	特殊性质	分值
1	$=2$	无	5
1	$=3$		6
3	≤ 9		7
4	≤ 20		16
5	≤ 100	A	7
6		B	8
7		C	9
8		D	17
9		无	25

特殊性质 A：若有解，保证答案 ≤ 8；

特殊性质 B：保证存在一个最优解，使每种数字最多操作一次；

特殊性质 C：保证存在一个最优解，使每次操作都是加到开头；

特殊性质 D：保证一定有解。