

冲刺 CCF NOIP2024 模拟试题

时间：2024 年 11 月 28 日 08:00 ~ 12:30

题目名称	再种花	梦相遇	双人卦	无向图
题目类型	传统型	传统型	传统型	传统型
目录	plant	meet	divining	graph
可执行文件名	plant	meet	divining	graph
输入文件名	plant.in	meet.in	divining.in	graph.in
输出文件名	plant.out	meet.out	divining.out	graph.out
每个测试点时限	1.0 秒	4.0 秒	5.0 秒	1.5 秒
内存限制	512 MiB	1024 MiB	1024 MiB	256 MiB
测试点数目	20	20	27	20
测试点是否等分	是	是	否	是

提交源程序文件名

对于 C++ 语言	plant.cpp	meet.cpp	divining.cpp	graph.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`。
- 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）。
- 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。
- 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。
- 直接复制 PDF 题面中的多行样例，数据将带有行号，并且某些字符可能无法正常显示，建议选手直接使用对应目录下的样例文件进行测试。
- 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。

再种花 (plant)

【题目描述】

小 C 决定在他的花园里种出 CCF 字样的图案，因此他想知道有多少种种花的方案；不幸的是，花园中有一些土坑，这些位置无法种花，因此他希望你能帮助他解决这个问题。

花园可以看作有 $n \times m$ 个位置的网格图，从上到下分别为第 1 到第 n 行，从左到右分别为第 1 列到第 m 列，其中每个位置有可能是土坑，也有可能不是，可以用 $a_{i,j} = 1$ 表示第 i 行第 j 列这个位置有土坑，否则用 $a_{i,j} = 0$ 表示这个位置没土坑。

一种种花方案被称为**合法的**，如果存在 $x_1, x_2 \in [1, n]$ ，以及 $y_1, y_2 \in [1, m]$ ，满足 $x_1 \leq x_2$ ，并且 $y_1 \leq y_2$ ，使得第 x_1 到第 x_2 行的第 y_1 到第 y_2 列都**不为土坑**，且只在上述这些位置上种花。

现在小 C 想知道，给定 n, m 以及表示每个位置是否为土坑的值 $\{a_{i,j}\}$ ，**合法**种花方案有多少种可能？由于答案可能非常之大，你只需要输出其对 998244353 取模的结果即可。

【输入格式】

从文件 `plant.in` 中读入数据。

第一行包含两个整数 T, id ，分别表示数据组数和测试点编号。如果数据为样例则保证 $id = 0$ 。

接下来一共 T 组数据，在每组数据中：

第一行包含四个整数 n, m ，其中 n, m 分别表示花园的行数、列数。

接下来 n 行，每行包含一个长度为 m 且仅包含 0 和 1 的字符串，其中第 i 个串的第 j 个字符表示 $a_{i,j}$ ，即花园里的第 i 行第 j 列是不是一个土坑。

【输出格式】

输出到文件 `plant.out` 中。

你需要对每一组数据输出一行用一个非负整数，表示方案数对 998244353 取模后的结果。

【样例 1 输入】

```
1 1 0
2 4 3
3 001
4 010
```

```
5 000
6 000
```

【样例 1 输出】

```
1 30
```

【样例 2】

见选手目录下的 *plant/plant2.in* 与 *plant/plant2.ans*。
该样例数据满足测试点 3 的性质。

【样例 3】

见选手目录下的 *plant/plant3.in* 与 *plant/plant3.ans*。
该样例数据满足测试点 6 的性质。

【样例 4】

见选手目录下的 *plant/plant4.in* 与 *plant/plant4.ans*。
该样例数据满足测试点 8, 9 的性质。

【样例 5】

见选手目录下的 *plant/plant5.in* 与 *plant/plant5.ans*。
该样例数据满足测试点 12, 13, 14 的性质。

【样例 6】

见选手目录下的 *plant/plant6.in* 与 *plant/plant6.ans*。
该样例数据满足测试点 17, 18 的性质。

【样例 7】

见选手目录下的 *plant/plant7.in* 与 *plant/plant7.ans*。
该样例数据满足测试点 21, 22, 23 的性质。

【样例 8】

见选手目录下的 *plant/plant8.in* 与 *plant/plant8.ans*。

该样例数据满足测试点 25 的性质。

【数据范围】

对于所有数据，保证： $1 \leq T \leq 5$ ， $1 \leq n, m \leq 10^3$ ， $a_{i,j} \in \{0, 1\}$ 。

测试点编号	n	m	特殊性质
1	$= 1$	$= 1$	无
2	$= 3$	$= 2$	
3	$= 4$		
4	≤ 1000	$= 1$	
5		$= 2$	
6	≤ 10	≤ 10	
7	≤ 20	≤ 20	
8,9	≤ 30	≤ 30	
10,11	≤ 50	≤ 50	
12,13,14	≤ 100	≤ 100	
15,16	≤ 200	≤ 200	
17,18	≤ 300	≤ 300	
19,20	≤ 500	≤ 500	
21,22,23	≤ 1000	≤ 1000	A
24			B
25			无

特殊性质 A：保证 $a_{i,j} = 0$ ；

特殊性质 B：保证 $a_{i,j} = 1$ 。

梦相遇（meet）

【题目描述】

如何在梦里找寻他的踪迹？小 X 有一个可以任意变化大小的正方形梦境搜索仪，他希望利用这个科技实现一场相遇。

小 X 的梦境是一个 n 行 m 列的方格平面，仪器可以在这个平面上沿水平或竖直方向移动。仪器移动过程中覆盖过的地方，即可被认为是搜索过的。

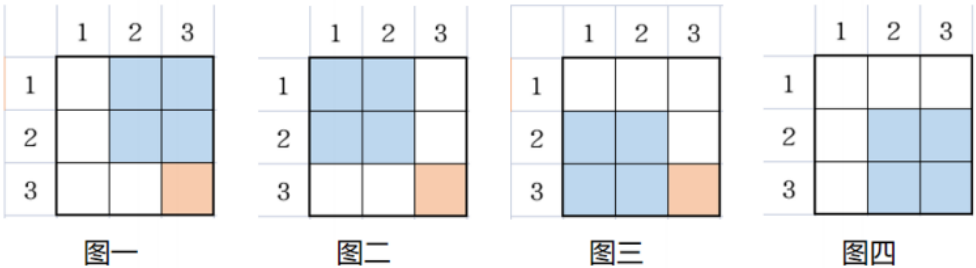
为了保证梦境的完整性，小 X 选择了 k 个方格作为禁行区，仪器移动过程中，不能覆盖到这 k 个方格。

小 X 可以任意选择一个 1 到 10^{100} 内的正整数 x ，让这个仪器变形为一个占地 x 行 x 列的正方体。然后他会选择一个位置放置这个仪器，并按照他喜欢的方式操纵这个仪器沿水平或竖直方向移动。

小 X 认为，仪器的尺寸越大，搜索的效率就越高。因此，请你帮他找到一个最大的正整数 x ，使得存在一种操纵仪器的方式，满足下列两个条件：

- 仪器移动的过程中，它不会覆盖到禁行区内任意一个方格的任意一个部分；
- 仪器移动有限时间后，所有非禁行区的方格均是被搜索过的。

下面四张图展示了一个 3 行 3 列的梦境，其中 (3,3) 是唯一的一个禁行区（用橙色表示）。有一个占地 2 行 2 列的仪器放置在梦境中（用浅蓝色表示）。



- 仪器可以从图一的位置向左移动至图二的位置，再向下移动至图三的位置。在移动结束后，所有非禁行区的方格均是被搜索过的。
- 仪器不能从图一的位置向下移动至图四的位置，因为这样移动会导致仪器覆盖 (3,3) 这个禁行区。

由于梦境总是在变化，所以你需要处理多组数据。

【输入格式】

从文件 `meet.in` 中读入数据。

本题的单个测试点中有 **多组测试数据**。

第一行两个正整数 id, T ，表示测试点编号和数据组数。

接下来是 T 组测试数据。

每组的第一个正整数 n, m, k ，表示梦境的尺寸以及禁行区的数量。

接下来的 k 行每行两个正整数 a_i, b_i ，表示 (a_i, b_i) 代表的方格是一个禁行区。

【输出格式】

输出到文件 `meet.out` 中。

对于每组数据，输出一行一个整数，表示最大的、满足条件的正整数 x 。若不存在满足条件的正整数，请输出 -1。

【样例 1 输入】

```
1 1 4
2 3 3 1
3 3 3
4 3 5 4
5 1 1
6 1 2
7 1 5
8 3 3
9 3 5 5
10 1 1
11 1 2
12 1 5
13 2 4
14 3 3
15 6 6 6
16 1 1
17 1 2
18 2 1
19 5 6
20 6 5
21 6 6
```

【样例 1 输出】

1	2
2	1
3	-1
4	3

【样例 1 解释】

第一组数据描述的梦境和题目描述部分的图例一致。 $x = 2$ 时，可以按照题目描述部分描述的方式移动。

第二组数据描述的梦境如下图所示。

	1	2	3	4	5
1					
2					
3					

第三组数据描述的梦境如下图所示。 $(3, 1)$ 和 $(3, 5)$ 所在的两个连通块互相不能到达，故不存在一个合法的正整数 x 。

	1	2	3	4	5
1					
2					
3					

第四组数据描述的梦境如下图所示。

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

【样例 2】

见选手目录下的 *meet/meet2.in* 与 *meet/meet2.ans*。
该样例满足测试点 9, 10 的限制。

【样例 3】

见选手目录下的 *meet/meet3.in* 与 *meet/meet3.ans*。
该样例满足测试点 1 ~ 3 的限制。

【样例 4】

见选手目录下的 *meet/meet4.in* 与 *meet/meet4.ans*。
该样例满足测试点 8 的限制。

【样例 5】

见选手目录下的 *meet/meet5.in* 与 *meet/meet5.ans*。
该样例满足测试点 12 的限制。

【样例 6】

见选手目录下的 *meet/meet6.in* 与 *meet/meet6.ans*。
该样例满足测试点 18 ~ 20 的限制。

【数据范围】

下面设 $S = \sum(n \times m)$ 。

对于 100% 的数据, 保证 $1 \leq n, m \leq 3 \times 10^6$, $0 \leq k < n \times m \leq 3 \times 10^6$, $1 \leq T \leq S \leq 10^7$, $\sum k \leq 3 \times 10^6$ 。

测试点编号	$S \leq$	特殊性质
1 ~ 3	500	无
4 ~ 6	3000	
7	2×10^5	A
8		B
9, 10		无
11	10^6	A
12		C
13, 14		无
15, 16	10^7	B
17		C
18 ~ 20		无

特殊性质 A：保证 $k \leq 1$ 。

特殊性质 B：保证 $n \leq 3$ 。

特殊性质 C：保证在同一个测试点中， a_i 全部相等。

请注意常数因子对程序效率的影响。

双人卦 (divining)

【题目描述】

“合卦”是一种双人卦，由一个标有 $1 \sim n$ 的盘面和编号为 $1 \sim n$ 的 n 根卦签组成。卜卦时，两人会按顺序依次执行下列操作：

1. 将 n 根卦签均匀打乱并随机分成两堆。假设两堆中的卦签根数分别为 x 和 $n-x$ ，其中 $x \in [0, n]$ 。
2. 第一个人选择卦签数量为 x 的一堆，并将这一堆中的所有卦签按任意顺序放置在盘面的 $1 \sim x$ 号位置上。
3. 第二个人将剩的那一堆中的所有卦签按任意顺序放置在盘面的 $x+1 \sim n$ 号位置上。
4. 卦象可以用一个数 B 来表示。具体地， B 是盘面的所有位置上，卦签编号与位置编号的差的绝对值之和。形式化地，设盘面上位置 i 处放置的卦签编号为 p_i ，有 $B = \sum_{i=1}^n |i - p_i|$ 。

小 X 和小 Y 正在卜卦。小 X 先将签数为 x 的那一堆卦签摆在了盘面上 $1 \sim x$ 的位置上，其中第 i 个位置放置的卦签编号为 p_i 。

通过阅读古书，小 Y 认为卦象为 b 是最好的。小 Y 希望知道，是否存在一种放置剩下的 $n-x$ 根卦签的方式，使得最终的卦象为 b 。

小 X 有一个幸运数字 s 。下面的问题中将会用到这个数字。

本题有两类测试点。

- 在第一类测试点中， $sid = 1$ 。此时你只需要判断是否存在至少 s 种放置剩下的 $n-x$ 根卦签的方式，使得最终的卦象为 b 即可。
- 在第二类测试点中， $sid = 2$ 。你需要判断是否存在至少 s 种放置剩下的 $n-x$ 根卦签的方式，使得最终的卦象为 b 。若存在，你需要找到放置结束后，满足卦象为 b 的所有盘面 p 中，字典序第 s 小的盘面。

由于小 X 和小 Y 占卜了很多次，所以你需要处理多组不同的数据。

【输入格式】

从文件 `divining.in` 中读入数据。

本题的单个测试点中有多组测试数据。

第一行三个整数 id, sid, T ，表示测试点编号、测试点种类和数据组数。

接下来是 T 组测试数据。

每组的 **第一行** 四个非负整数 n, b, s, x ，表示盘面大小、目标卦象、小 X 的幸运数字和小 X 放置的卦签数量。

第二行 x 个正整数，第 i 个正整数 p_i 表示小 X 在盘面上位置 i 处放置的卦签编号。

【输出格式】

输出到文件 *divining.out* 中。

对于每组数据，按照下面的要求进行输出：

- $sid = 1$ 时：输出一行一个整数。若存在，则输出 1；否则输出 -1。
- $sid = 2$ 时：若存在，则输出两行，第一行一个整数 1，第二行一个长度为 n 的排列 q ，表示字典序第 s 小的合法最终盘面。若不存在，输出一行一个整数 -1 即可。

【样例 1 输入】

```
1 12 2 5
2 3 2 3 0
3 3 2 1 1
4 2
5 3 2 1 2
6 2 3
7 3 4 2 0
8 3 4 2 1
9 3
```

【样例 1 输出】

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

【样例 1 解释】

当 $n = 3$ 时，所有可能的最终盘面有以下的、按字典序从小到大排序的 6 种：

- $(1, 2, 3), B = 0$ 。

- $(1, 3, 2), B = 2$ 。
- $(2, 1, 3), B = 2$ 。
- $(2, 3, 1), B = 4$ 。
- $(3, 1, 2), B = 4$ 。
- $(3, 2, 1), B = 4$ 。

下面给出每组数据的解释：

- 对第一组数据， $B = 2$ 的最终盘面只有 2 个，不足 3 个。
- 对第二组数据，满足 $p_1 = 2$ 且 $B = 2$ 的最终盘面共有 1 个，字典序第 1 小的为 $(2, 1, 3)$ 。
- 对第三组数据，没有满足 $p_1 = 2, p_2 = 3$ 且 $B = 2$ 的最终盘面。
- 对第四组数据，满足 $B = 4$ 的最终盘面共有 3 个，字典序第 2 小的为 $(3, 1, 2)$ 。
- 对第五组数据，满足 $p_1 = 3$ 且 $B = 4$ 的最终盘面共有 2 个，字典序第 2 小的为 $(3, 2, 1)$ 。

【样例 2】

见选手目录下的 *divining/divining2.in* 与 *divining/divining2.ans*。

该样例与测试点 4 限制相同。

【样例 3】

见选手目录下的 *divining/divining3.in* 与 *divining/divining3.ans*。

该样例与测试点 9 ~ 11 限制相同。

【样例 4】

见选手目录下的 *divining/divining4.in* 与 *divining/divining4.ans*。

该样例与测试点 12 ~ 13 限制相同。

【样例 5】

见选手目录下的 *divining/divining5.in* 与 *divining/divining5.ans*。

该样例与测试点 14 ~ 15 限制相同。

【样例 6】

见选手目录下的 *divining/divining6.in* 与 *divining/divining6.ans*。

该样例与测试点 16 ~ 17 限制相同。

【样例 7】

见选手目录下的 *divining/divining7.in* 与 *divining/divining7.ans*。
该样例与测试点 20 ~ 23 限制相同。

【样例 8】

见选手目录下的 *divining/divining8.in* 与 *divining/divining8.ans*。
该样例与测试点 24 ~ 27 限制相同。

【数据范围】

下面设 $S = \sum n$ 。

对于 100% 的数据，保证 $1 \leq n \leq 28$ ， $1 \leq S \leq 100$ ， $0 \leq b \leq n^2$ ， $1 \leq s \leq 10^{18}$ ， $0 \leq x \leq n$ 。保证 $1 \leq p_i \leq n$ 且 p_i 互不相同。

本题的各个测试点不等分。

测试点编号	$sid =$	$n \leq$	$S \leq$	特殊性质	单个测试点分值	总分值
1	1	9	10^2	无	3	3
2 ~ 3		14			4	8
4		28		A,B	4	4
5 ~ 6				A	4	8
7 ~ 8				B	7	14
9 ~ 11				无	5	15
12 ~ 13	2	9			4	8
14 ~ 15		14			4	8
16 ~ 17		17			2	4
18 ~ 19		22			2	4
20 ~ 23		25			3	12
24 ~ 27		28			3	12

特殊性质 A：保证 $s = 1$ 。
特殊性质 B：保证 $x = 0$ 。
请注意常数因子对程序效率的影响。

无向图 (graph)

【题目描述】

小 D 有一个 n 个点的简单无向图，初始没有边。

有 m 次加边操作，每次给定 a_i, b_i, c_i, d_i ，对所有满足 $a_i \leq u \leq b_i, c_i \leq v \leq d_i$ 的点 (u, v) 连上一条边。当所有加边操作结束后，小 D 会把自环去掉，相同的重边只保留一条，这样就变成简单无向图啦。

现在小 D 需要在此基础上加任意多条边（也可以不加）使图变成一个所有点的度数都为偶数的简单无向图（不能添加重边自环和已有的边），问方案数对 998244353 取模的余数，若添加的边的集合不同则两个方案不同。

聪明的小 D 很快就解决了，但他想考考你，于是有按响了你家的门铃。

【输入格式】

从文件 `graph.in` 中读入数据。

第一行两个整数 n, m ，代表无向图点的个数与加边操作次数。

接下来 m 行，每行 4 个整数 a_i, b_i, c_i, d_i ，代表一次加边操作。

【输出格式】

输出到文件 `graph.out` 中。

输出仅一行一个整数表示答案方案数。

【样例 1 输入】

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

【样例 1 输出】

```
1 2
```

【样例 1 解释】

加完边后的图有边 $(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5)$ 。

我们不加边或再加边 $(3, 4), (4, 5), (3, 5)$ ，共两种方案使得所有点的度数均为偶数。

【样例 2 输入】

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

【样例 2 输出】

```
1 2
```

【样例 3】

见选手目录下的 *graph/graph3.in* 与 *graph/graph3.ans*。
该样例数据满足测试点 3 ~ 4 的限制。

【样例 4】

见选手目录下的 *graph/graph4.in* 与 *graph/graph4.ans*。
该样例数据满足测试点 5 ~ 10 的限制。

【样例 5】

见选手目录下的 *graph/graph5.in* 与 *graph/graph5.ans*。
该样例数据满足测试点 11 ~ 14 的限制。

【样例 6】

见选手目录下的 *graph/graph6.in* 与 *graph/graph6.ans*。
该样例数据满足测试点 15 ~ 20 的限制。

【数据范围】

对于所有数据，满足 $1 \leq n, m \leq 10^5, 1 \leq a_i, b_i, c_i, d_i \leq n, a_i \leq b_i, c_i \leq d_i$ 。

测试点编号	$n \leq$	m	特殊性质
1 ~ 2	7	≤ 100	A
3 ~ 4	10^5	$= 0$	无
5 ~ 10			A
11 ~ 14	8×10^3	$\leq 10^5$	无
15 ~ 20	10^5		无

性质 A: $a_i = b_i, c_i = d_i$ 。