

CCF 全国青少年信息学奥林匹克联赛

CCF NOIP 2021

时间：2021 年 11 月 20 日 08:30 ~ 13:00

题目名称	报数	数列	方差	棋局
题目类型	传统型	传统型	传统型	传统型
目录	number	sequence	variance	chess
可执行文件名	number	sequence	variance	chess
输入文件名	number.in	sequence.in	variance.in	chess.in
输出文件名	number.out	sequence.out	variance.out	chess.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	4.0 秒
内存限制	512 MiB	512 MiB	512 MiB	1024 MiB
子任务数目	10	20	25	25
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	number.cpp	sequence.cpp	variance.cpp	chess.cpp
对于 C 语言	number.c	sequence.c	variance.c	chess.c
对于 Pascal 语言	number.pas	sequence.pas	variance.pas	chess.pas

编译选项

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

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/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。
3. 提交的程序代码文件的放置位置请参考各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 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）。
6. 程序可使用的栈空间内存限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Inter(R) Core(TM) i7-8700K CPU @3.70GHz，内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以此为准。

报数 (number)

【题目描述】

报数游戏是一个广为流传的休闲小游戏。参加游戏的每个人要按一定顺序轮流报数，但如果下一个报的数是 7 的倍数，或十进制表示中含有数字 7，就必须跳过这个数，否则就输掉了游戏。

在一个风和日丽的下午，刚刚结束 *SPC20nn* 比赛的小 r 和小 z 闲得无聊玩起了这个报数游戏。但在只有两个人玩的情况下计算起来还是比较容易的，因此他们玩了很久也没分出胜负。此时小 z 灵光一闪，决定把这个游戏加强：任何一个十进制中含有数字 7 的数，它的所有倍数都不能报出来！

形式化地，设 $p(x)$ 表示 x 的十进制表示中是否含有数字 7，若含有则 $p(x) = 1$ ，否则 $p(x) = 0$ 。则一个正整数 x 不能被报出，当且仅当存在正整数 y 和 z ，使得 $x = yz$ 且 $p(y) = 1$ 。

例如，如果小 r 报出了 6，由于 7 不能报，所以小 z 下一个需要报 8；如果小 r 报出了 33，则由于 $34 = 17 \times 2$, $35 = 7 \times 5$ 都不能报，小 z 下一个需要报出 36；如果小 r 报出了 69，由于 $70 \sim 79$ 的数都含有 7，小 z 下一个需要报出 80 才行。

现在小 r 的上一个数报出了 x ，小 z 想快速算出他下一个数要报多少，不过他很快就发现这个游戏可比原版的游戏难算多了，于是他需要你的帮助。当然，如果小 r 报出的 x 本身是不能报出的，你也要快速反应过来小 r 输了才行。

由于小 r 和小 z 玩了很长时间游戏，你也需要回答小 z 的很多个问题。

【输入格式】

从文件 *number.in* 中读入数据。

第 1 行，一个正整数 T 表示小 z 询问的数量。

接下来 T 行，每行 1 个正整数 x ，表示这一次小 r 报出的数。

【输出格式】

输出到文件 *number.out* 中。

输出共 T 行，每行一个整数，如果小 r 这一次报出的数是不能报出的，输出 -1 ，否则输出小 z 下一次报出的数是多少。

【样例 1 输入】

```
1 4
2 6
3 33
```

```
4 69  
5 300
```

【样例 1 输出】

```
1 8  
2 36  
3 80  
4 -1
```

【样例 1 解释】

这一组样例的前 3 次询问在题目描述中已有解释。

对于第 4 次询问，由于 $300 = 75 \times 4$ ，而 75 中含有 7，所以小 r 直接输掉了游戏。

【样例 2 输入】

```
1 5  
2 90  
3 99  
4 106  
5 114  
6 169
```

【样例 2 输出】

```
1 92  
2 100  
3 109  
4 -1  
5 180
```

【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*。

【数据范围】

对于 10% 的数据, $T \leq 10, x \leq 100$ 。

对于 30% 的数据, $T \leq 100, x \leq 1000$ 。

对于 50% 的数据, $T \leq 1000, x \leq 10000$ 。

对于 70% 的数据, $T \leq 10000, x \leq 2 \times 10^5$ 。

对于 100% 的数据, $T \leq 2 \times 10^5, x \leq 10^7$ 。

数列 (sequence)

【题目描述】

给定整数 n, m, k , 和一个长度为 $m + 1$ 的正整数数组 v_0, v_1, \dots, v_m 。

对于一个长度为 n , 下标从 1 开始且每个元素均不超过 m 的非负整数序列 $\{a_i\}$, 我们定义它的权值为 $v_{a_1} \times v_{a_2} \times \dots \times v_{a_n}$ 。

当这样的序列 $\{a_i\}$ 满足整数 $S = 2^{a_1} + 2^{a_2} + \dots + 2^{a_n}$ 的二进制表示中 1 的个数不超过 k 时, 我们认为 $\{a_i\}$ 是一个合法序列。

计算所有合法序列 $\{a_i\}$ 的权值和对 998244353 取模的结果。

【输入格式】

从文件 *sequence.in* 中读入数据。

输入的一行是三个整数 n, m, k 。

第二行 $m + 1$ 个整数, 分别是 v_0, v_1, \dots, v_m 。

【输出格式】

输出到文件 *sequence.out* 中。

仅一行一个整数, 表示所有合法序列的权值和对 998244353 取模的结果。

【样例输入】

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

【样例输出】

```
1 40
```

【样例 1 解释】

由于 $k = 1$, 而且由 $n \leq S \leq n \times 2^m$ 知道 $5 \leq S \leq 10$, 合法的 S 只有一种可能: $S = 8$, 这要求 a 中必须有 2 个 0 和 3 个 1, 于是有 $\binom{5}{2} = 10$ 种可能的序列, 每种序列的贡献都是 $v_0^2 v_1^3 = 4$, 权值和为 $10 \times 4 = 40$ 。

【样例 2】

见选手目录下的 *sequence/sequence2.in* 与 *sequence/sequence2.ans*。

【数据范围】

对所有测试点保证 $1 \leq k \leq n \leq 30, 0 \leq m \leq 100, 1 \leq v_i < 998244353$ 。

测试点	n	k	m	
1 ~ 4	= 8	$\leq n$	= 9	
5 ~ 7	= 30		= 7	
8 ~ 10			= 12	
11 ~ 13	= 5	= 1	= 100	
14 ~ 15		$\leq n$	= 50	
16	= 15		= 100	
17 ~ 18	= 30		= 30	
19 ~ 20			= 100	

方差 (variance)

【题目描述】

给定长度为 n 的非严格递增正整数数列 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。每次可以进行的操作是：任意选择一个正整数 $1 < i < n$ ，将 a_i 变为 $a_{i-1} + a_{i+1} - a_i$ 。求在若干次操作之后，该数列的方差最小值是多少。请输出最小值乘以 n^2 的结果。

其中方差的定义为：数列中每个数与平均值的差的平方的平均值。更形式化地说，方差的定义为 $D = \frac{1}{n} \sum_{i=1}^n (a_i - \bar{a})^2$ ，其中 $\bar{a} = \frac{1}{n} \sum_{i=1}^n a_i$ 。

【输入格式】

从文件 *variance.in* 中读入数据。

输入的第一行包含一个正整数 n ，保证 $n \leq 10^4$ 。

输入的第二行有 n 个正整数，其中第 i 个数字表示 a_i 的值。数据保证 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。

【输出格式】

输出到文件 *variance.out* 中。

输出仅一行，包含一个非负整数，表示你所求的方差的最小值的 n^2 倍。

【样例 1 输入】

```
1 4
2 1 2 4 6
```

【样例 1 输出】

```
1 52
```

【样例 1 解释】

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，第一次操作得到的数列有 $(1, 3, 4, 6)$ ，第二次操作得到的新的数列有 $(1, 3, 5, 6)$ 。之后无法得到新的数列。

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，平均值为 $\frac{13}{4}$ ，方差为 $\frac{1}{4}((1 - \frac{13}{4})^2 + (2 - \frac{13}{4})^2 + (4 - \frac{13}{4})^2 + (6 - \frac{13}{4})^2) = \frac{59}{16}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 4, 6)$ ，平均值为 $\frac{7}{2}$ ，方差为 $\frac{1}{4}((1 - \frac{7}{2})^2 + (3 - \frac{7}{2})^2 + (4 - \frac{7}{2})^2 + (6 - \frac{7}{2})^2) = \frac{13}{4}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 5, 6)$, 平均值为 $\frac{15}{4}$, 方差为 $\frac{1}{4}((1 - \frac{15}{4})^2 + (3 - \frac{15}{4})^2 + (5 - \frac{15}{4})^2 + (6 - \frac{15}{4})^2) = \frac{59}{16}$ 。

【样例 2】

见选手目录下的 *variance/variance2.in* 与 *variance/variance2.ans*。

【样例 3】

见选手目录下的 *variance/variance3.in* 与 *variance/variance3.ans*。

【样例 4】

见选手目录下的 *variance/variance4.in* 与 *variance/variance4.ans*。

【数据范围】

测试点编号	$n \leq$	$a_i \leq$
1 ~ 3	4	10
4 ~ 5	10	40
6 ~ 8	15	20
9 ~ 12	20	300
13 ~ 15	50	70
16 ~ 18	100	40
19 ~ 22	400	600
23 ~ 25	10000	50

对于所有的数据, 保证 $n \leq 10000$, $a_i \leq 600$ 。

棋局 (chess)

【题目背景】

在输了一晚上的麻将之后，小 z 和小 c 卸掉了手机上的所有牌类游戏。不过这怎么可能阻挡得了他们上课颓废的决心呢？现在他们的目光盯在了棋类游戏上，但他们两个除了天天下飞行器以外，几乎所有棋类游戏都只懂个大概规则。

“既然我们都会玩但只能玩一点点，不如我们自己搞个缝合怪出来吧！”

于是，在他们的精心脑洞之下，一个融合了围棋、象棋与军棋的奇妙游戏诞生了……

【题目描述】

游戏在一张长 n 行宽 m 列的网格形棋盘上进行，棋子落在网格的交叉点上。我们不妨记左上角的交叉点的坐标为 $(1, 1)$ ，右下角的交叉点坐标为 (n, m) 。

棋子分为黑白两色，对局双方各执一方棋子。

每个棋子除了颜色以外还有等级，不妨设 col_i 为棋子 i 的颜色， lv_i 为棋子 i 的等级。另外，棋盘上的网格线共有 4 种状态，对于第 i 条网格线，设其状态为 opt_i 。

轮到每方下棋时，他可以选择棋盘上的一个己方棋子沿网格线进行移动到另一个交叉点，称为走子。形式化定义走子的过程如下：选择一个坐标序列 $(x_0, y_0), (x_1, y_1), \dots, (x_k, y_k)$ ，其中 k 是任意选定的正整数， (x_0, y_0) 是棋子初始的位置， (x_k, y_k) 是棋子最终走到的位置，需要满足：

- 对于任意 $i = 0, 1, \dots, k - 1$ ，坐标 (x_i, y_i) 和 (x_{i+1}, y_{i+1}) 之间必须有网格线直接相连，也就是说走子必须沿着网格线走；
- 对于任意 $i \neq j$ ，必须有 $(x_i, y_i) \neq (x_j, y_j)$ ，也就是说走子过程中不能经过重复位置，特别地 $(x_0, y_0) \neq (x_k, y_k)$ ，也就是说不能原地不动（或走回原地）。
- 对于任意 $i = 1, \dots, k - 1$ ，坐标 (x_i, y_i) 上必须没有棋子，也就是说走子时不能越过已有的棋子。
- 若 (x_k, y_k) 上没有棋子，称为普通走子，否则称为吃子。在吃子过程中，设正在走的棋子颜色为 col_1 ，等级为 lv_1 ，被吃的棋子颜色为 col_2 ，等级为 lv_2 ，则必须满足 $col_1 \neq col_2$, $lv_1 \geq lv_2$ ，换句话说只能吃与自己颜色不同，且等级不高于自己等级的棋子。

需要注意的是，由上述定义可以得出，不允许棋子在吃子后继续向前走。

网格线的状态含义如下所述：

- 如果 $opt_i = 0$ ，代表此路不通，走子时不能经过这条网格线。
- 如果 $opt_i = 1$ ，代表这条网格线是一条“普通道路”，每次走子时棋子最多只能经过 1 条普通道路。
- 如果 $opt_i = 2$ ，代表这条网格线是一条“直行道路”，每次走子时棋子可以经过任意条直行道路，但只能一直沿横向或一直沿纵向走，不能转弯。如沿直行道路从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(1, 3)$ 是可以的，但是从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(2, 2)$ 不行。

- 如果 $opt_i = 3$ ，代表这条网格线是一条“互通道路”，每次走子时棋子可以经过任意条互通道路，且中途可任意转弯。

同时规定在一次走子过程中，棋子经过的网格线的状态必须全部相同，比如从 $(1, 1)$ 经过直行道路走到 $(1, 2)$ 再经过互通道路走到 $(1, 3)$ 是不允许的。

至于如何判断胜负等其他细节，与本题无关，故略去。

小 z 和小 c 开发出这款棋类游戏后，为了提升水平，想了一个训练的策略：一开始棋盘是空的，然后小 c 会每次往棋盘的某个空交叉点上放一枚棋子，小 z 需要快速计算出：若选择这枚新放上的棋子进行一次走子，棋盘上一共有多少个位置是能被走到的？注意，因为这只是思维训练，他们并不会真的走这枚棋子。

可怜的小 z 发现他的计算力不足以算出这个问题，只好向你求助。

【输入格式】

从文件 **chess.in** 中读入数据。

每个测试点由多组数据组成。

第一行：一个正整数 T 表示数据组数。

对于每组数据：

第一行：3 个正整数 n, m, q ，分别表示棋盘的行数、列数和游戏的轮数。

接下来 n 行，每行为一个长 $m - 1$ 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i, j + 1)$ 的网格线状态。

接下来 $n - 1$ 行，每行为一个长 m 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i + 1, j)$ 的网格线状态。

接下来 q 行，每行 4 个非负整数 col_i, lv_i, x_i, y_i ，表示在第 i 轮有一枚颜色为 col_i ，等级为 lv_i 的棋子放在了交叉点 (x_i, y_i) 上。其中 $col_i = 0$ 表示黑子， $col_i = 1$ 表示白子。保证之前交叉点 (x_i, y_i) 上没有棋子。

【输出格式】

输出到文件 **chess.out** 中。

对于每组数据输出 q 行，每行一个非负整数，表示第 i 枚棋子放置后能走到的交叉点数量。

【样例 1 输入】

```
1 1
2 3 3 5
3 13
4 22
5 23
```

```
6 010
7 233
8 0 1 2 3
9 1 2 2 1
10 1 3 1 2
11 0 2 3 2
12 1 3 2 2
```

【样例 1 输出】

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

【样例 1 解释】

放置棋子 1 后，它能走到的位置为 $(2, 1), (2, 2), (3, 2), (3, 3)$ 。

放置棋子 2 后，它能走到的位置为 $(2, 2), (2, 3), (3, 1)$ 。

放置棋子 3 后，它能走到的位置为 $(1, 1), (1, 3), (2, 2)$ 。

放置棋子 4 后，它能走到的位置为 $(2, 2), (3, 1), (3, 3)$ 。

放置棋子 5 后，它能走到的位置为 $(2, 3), (3, 2)$ 。

【样例 2 输入】

```
1 2
2 2 3 4
3 22
4 33
5 123
6 0 2 1 2
7 0 1 2 1
8 1 2 1 3
9 0 3 2 2
10 3 2 3
11 3
```

```

12 1
13 3
14 32
15 32
16 0 2 1 2
17 1 2 3 2
18 0 1 2 2

```

【样例 2 输出】

```

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

```

【样例 3】

见选手目录下的 *chess/chess3.in* 与 *chess/chess3.ans*。

【样例 4】

见选手目录下的 *chess/chess4.in* 与 *chess/chess4.ans*。

【数据范围】

测试点编号	$n \times m \leq$	$q \leq$	特殊性质
1 ~ 2	100	50	无
3 ~ 6	5000	2000	
7 ~ 8			不存在“直行道路”与“互通道路”
9 ~ 11			不存在“互通道路”
12 ~ 14			不存在“直行道路”
15 ~ 16			$lv_i = i$
17 ~ 18			$lv_i = q - i + 1$
19 ~ 21		2000	$n, m \leq 1000$
22 ~ 25		10^5	无

对于 100% 的数据， $1 \leq T \leq 5$ ， $2 \leq n, m \leq 10^5$ ， $4 \leq n \times m \leq 2 \times 10^5$ ， $1 \leq q \leq \min\{10^5, n \times m\}$ ， $1 \leq lv_i \leq q$ ， $1 \leq x_i \leq n$ ， $1 \leq y_i \leq m$ ， $col_i \in \{0, 1\}$ 。

注：由于本题输入输出规模较大，建议使用较为快速的输入输出方式。

CCF 全国青少年信息学奥林匹克联赛

CCF NOIP 2021

时间：2021 年 11 月 20 日 08:30 ~ 13:00

题目名称	报数	数列	方差	棋局
题目类型	传统型	传统型	传统型	传统型
目录	number	sequence	variance	chess
可执行文件名	number	sequence	variance	chess
输入文件名	number.in	sequence.in	variance.in	chess.in
输出文件名	number.out	sequence.out	variance.out	chess.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	4.0 秒
内存限制	512 MiB	512 MiB	512 MiB	1024 MiB
子任务数目	10	20	25	25
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	number.cpp	sequence.cpp	variance.cpp	chess.cpp
对于 C 语言	number.c	sequence.c	variance.c	chess.c
对于 Pascal 语言	number.pas	sequence.pas	variance.pas	chess.pas

编译选项

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

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/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。
3. 提交的程序代码文件的放置位置请参考各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 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）。
6. 程序可使用的栈空间内存限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Inter(R) Core(TM) i7-8700K CPU @3.70GHz，内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以此为准。

CCF 全国青少年信息学奥林匹克联赛

CCF NOIP 2021

时间：2021 年 11 月 20 日 08:30 ~ 13:00

题目名称	报数	数列	方差	棋局
题目类型	传统型	传统型	传统型	传统型
目录	number	sequence	variance	chess
可执行文件名	number	sequence	variance	chess
输入文件名	number.in	sequence.in	variance.in	chess.in
输出文件名	number.out	sequence.out	variance.out	chess.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	4.0 秒
内存限制	512 MiB	512 MiB	512 MiB	1024 MiB
子任务数目	10	20	25	25
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	number.cpp	sequence.cpp	variance.cpp	chess.cpp
对于 C 语言	number.c	sequence.c	variance.c	chess.c
对于 Pascal 语言	number.pas	sequence.pas	variance.pas	chess.pas

编译选项

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

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/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。
3. 提交的程序代码文件的放置位置请参考各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 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）。
6. 程序可使用的栈空间内存限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Inter(R) Core(TM) i7-8700K CPU @3.70GHz，内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以此为准。

报数 (number)

【题目描述】

报数游戏是一个广为流传的休闲小游戏。参加游戏的每个人要按一定顺序轮流报数，但如果下一个报的数是 7 的倍数，或十进制表示中含有数字 7，就必须跳过这个数，否则就输掉了游戏。

在一个风和日丽的下午，刚刚结束 *SPC20nn* 比赛的小 r 和小 z 闲得无聊玩起了这个报数游戏。但在只有两个人玩的情况下计算起来还是比较容易的，因此他们玩了很久也没分出胜负。此时小 z 灵光一闪，决定把这个游戏加强：任何一个十进制中含有数字 7 的数，它的所有倍数都不能报出来！

形式化地，设 $p(x)$ 表示 x 的十进制表示中是否含有数字 7，若含有则 $p(x) = 1$ ，否则 $p(x) = 0$ 。则一个正整数 x 不能被报出，当且仅当存在正整数 y 和 z ，使得 $x = yz$ 且 $p(y) = 1$ 。

例如，如果小 r 报出了 6，由于 7 不能报，所以小 z 下一个需要报 8；如果小 r 报出了 33，则由于 $34 = 17 \times 2$, $35 = 7 \times 5$ 都不能报，小 z 下一个需要报出 36；如果小 r 报出了 69，由于 $70 \sim 79$ 的数都含有 7，小 z 下一个需要报出 80 才行。

现在小 r 的上一个数报出了 x ，小 z 想快速算出他下一个数要报多少，不过他很快就发现这个游戏可比原版的游戏难算多了，于是他需要你的帮助。当然，如果小 r 报出的 x 本身是不能报出的，你也要快速反应过来小 r 输了才行。

由于小 r 和小 z 玩了很长时间游戏，你也需要回答小 z 的很多个问题。

【输入格式】

从文件 *number.in* 中读入数据。

第 1 行，一个正整数 T 表示小 z 询问的数量。

接下来 T 行，每行 1 个正整数 x ，表示这一次小 r 报出的数。

【输出格式】

输出到文件 *number.out* 中。

输出共 T 行，每行一个整数，如果小 r 这一次报出的数是不能报出的，输出 -1 ，否则输出小 z 下一次报出的数是多少。

【样例 1 输入】

```
1 4
2 6
3 33
```

```
4 69  
5 300
```

【样例 1 输出】

```
1 8  
2 36  
3 80  
4 -1
```

【样例 1 解释】

这一组样例的前 3 次询问在题目描述中已有解释。

对于第 4 次询问，由于 $300 = 75 \times 4$ ，而 75 中含有 7，所以小 r 直接输掉了游戏。

【样例 2 输入】

```
1 5  
2 90  
3 99  
4 106  
5 114  
6 169
```

【样例 2 输出】

```
1 92  
2 100  
3 109  
4 -1  
5 180
```

【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*。

【数据范围】

对于 10% 的数据, $T \leq 10, x \leq 100$ 。

对于 30% 的数据, $T \leq 100, x \leq 1000$ 。

对于 50% 的数据, $T \leq 1000, x \leq 10000$ 。

对于 70% 的数据, $T \leq 10000, x \leq 2 \times 10^5$ 。

对于 100% 的数据, $T \leq 2 \times 10^5, x \leq 10^7$ 。

数列 (sequence)

【题目描述】

给定整数 n, m, k , 和一个长度为 $m + 1$ 的正整数数组 v_0, v_1, \dots, v_m 。

对于一个长度为 n , 下标从 1 开始且每个元素均不超过 m 的非负整数序列 $\{a_i\}$, 我们定义它的权值为 $v_{a_1} \times v_{a_2} \times \dots \times v_{a_n}$ 。

当这样的序列 $\{a_i\}$ 满足整数 $S = 2^{a_1} + 2^{a_2} + \dots + 2^{a_n}$ 的二进制表示中 1 的个数不超过 k 时, 我们认为 $\{a_i\}$ 是一个合法序列。

计算所有合法序列 $\{a_i\}$ 的权值和对 998244353 取模的结果。

【输入格式】

从文件 *sequence.in* 中读入数据。

输入的一行是三个整数 n, m, k 。

第二行 $m + 1$ 个整数, 分别是 v_0, v_1, \dots, v_m 。

【输出格式】

输出到文件 *sequence.out* 中。

仅一行一个整数, 表示所有合法序列的权值和对 998244353 取模的结果。

【样例输入】

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

【样例输出】

```
1 40
```

【样例 1 解释】

由于 $k = 1$, 而且由 $n \leq S \leq n \times 2^m$ 知道 $5 \leq S \leq 10$, 合法的 S 只有一种可能: $S = 8$, 这要求 a 中必须有 2 个 0 和 3 个 1, 于是有 $\binom{5}{2} = 10$ 种可能的序列, 每种序列的贡献都是 $v_0^2 v_1^3 = 4$, 权值和为 $10 \times 4 = 40$ 。

【样例 2】

见选手目录下的 *sequence/sequence2.in* 与 *sequence/sequence2.ans*。

【数据范围】

对所有测试点保证 $1 \leq k \leq n \leq 30, 0 \leq m \leq 100, 1 \leq v_i < 998244353$ 。

测试点	n	k	m	
1 ~ 4	= 8	$\leq n$	= 9	
5 ~ 7	= 30		= 7	
8 ~ 10			= 12	
11 ~ 13	= 5	= 1	= 100	
14 ~ 15		$\leq n$	= 50	
16	= 15		= 100	
17 ~ 18	= 30		= 30	
19 ~ 20			= 100	

方差 (variance)

【题目描述】

给定长度为 n 的非严格递增正整数数列 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。每次可以进行的操作是：任意选择一个正整数 $1 < i < n$ ，将 a_i 变为 $a_{i-1} + a_{i+1} - a_i$ 。求在若干次操作之后，该数列的方差最小值是多少。请输出最小值乘以 n^2 的结果。

其中方差的定义为：数列中每个数与平均值的差的平方的平均值。更形式化地说，方差的定义为 $D = \frac{1}{n} \sum_{i=1}^n (a_i - \bar{a})^2$ ，其中 $\bar{a} = \frac{1}{n} \sum_{i=1}^n a_i$ 。

【输入格式】

从文件 *variance.in* 中读入数据。

输入的第一行包含一个正整数 n ，保证 $n \leq 10^4$ 。

输入的第二行有 n 个正整数，其中第 i 个数字表示 a_i 的值。数据保证 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。

【输出格式】

输出到文件 *variance.out* 中。

输出仅一行，包含一个非负整数，表示你所求的方差的最小值的 n^2 倍。

【样例 1 输入】

```
1 4
2 1 2 4 6
```

【样例 1 输出】

```
1 52
```

【样例 1 解释】

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，第一次操作得到的数列有 $(1, 3, 4, 6)$ ，第二次操作得到的新的数列有 $(1, 3, 5, 6)$ 。之后无法得到新的数列。

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，平均值为 $\frac{13}{4}$ ，方差为 $\frac{1}{4}((1 - \frac{13}{4})^2 + (2 - \frac{13}{4})^2 + (4 - \frac{13}{4})^2 + (6 - \frac{13}{4})^2) = \frac{59}{16}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 4, 6)$ ，平均值为 $\frac{7}{2}$ ，方差为 $\frac{1}{4}((1 - \frac{7}{2})^2 + (3 - \frac{7}{2})^2 + (4 - \frac{7}{2})^2 + (6 - \frac{7}{2})^2) = \frac{13}{4}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 5, 6)$, 平均值为 $\frac{15}{4}$, 方差为 $\frac{1}{4}((1 - \frac{15}{4})^2 + (3 - \frac{15}{4})^2 + (5 - \frac{15}{4})^2 + (6 - \frac{15}{4})^2) = \frac{59}{16}$ 。

【样例 2】

见选手目录下的 *variance/variance2.in* 与 *variance/variance2.ans*。

【样例 3】

见选手目录下的 *variance/variance3.in* 与 *variance/variance3.ans*。

【样例 4】

见选手目录下的 *variance/variance4.in* 与 *variance/variance4.ans*。

【数据范围】

测试点编号	$n \leq$	$a_i \leq$
1 ~ 3	4	10
4 ~ 5	10	40
6 ~ 8	15	20
9 ~ 12	20	300
13 ~ 15	50	70
16 ~ 18	100	40
19 ~ 22	400	600
23 ~ 25	10000	50

对于所有的数据, 保证 $n \leq 10000$, $a_i \leq 600$ 。

棋局 (chess)

【题目背景】

在输了一晚上的麻将之后，小 z 和小 c 卸掉了手机上的所有牌类游戏。不过这怎么可能阻挡得了他们上课颓废的决心呢？现在他们的目光盯在了棋类游戏上，但他们两个除了天天下飞行器以外，几乎所有棋类游戏都只懂个大概规则。

“既然我们都会玩但只能玩一点点，不如我们自己搞个缝合怪出来吧！”

于是，在他们的精心脑洞之下，一个融合了围棋、象棋与军棋的奇妙游戏诞生了……

【题目描述】

游戏在一张长 n 行宽 m 列的网格形棋盘上进行，棋子落在网格的交叉点上。我们不妨记左上角的交叉点的坐标为 $(1, 1)$ ，右下角的交叉点坐标为 (n, m) 。

棋子分为黑白两色，对局双方各执一方棋子。

每个棋子除了颜色以外还有等级，不妨设 col_i 为棋子 i 的颜色， lv_i 为棋子 i 的等级。另外，棋盘上的网格线共有 4 种状态，对于第 i 条网格线，设其状态为 opt_i 。

轮到每方下棋时，他可以选择棋盘上的一个己方棋子沿网格线进行移动到另一个交叉点，称为走子。形式化定义走子的过程如下：选择一个坐标序列 $(x_0, y_0), (x_1, y_1), \dots, (x_k, y_k)$ ，其中 k 是任意选定的正整数， (x_0, y_0) 是棋子初始的位置， (x_k, y_k) 是棋子最终走到的位置，需要满足：

- 对于任意 $i = 0, 1, \dots, k - 1$ ，坐标 (x_i, y_i) 和 (x_{i+1}, y_{i+1}) 之间必须有网格线直接相连，也就是说走子必须沿着网格线走；
- 对于任意 $i \neq j$ ，必须有 $(x_i, y_i) \neq (x_j, y_j)$ ，也就是说走子过程中不能经过重复位置，特别地 $(x_0, y_0) \neq (x_k, y_k)$ ，也就是说不能原地不动（或走回原地）。
- 对于任意 $i = 1, \dots, k - 1$ ，坐标 (x_i, y_i) 上必须没有棋子，也就是说走子时不能越过已有的棋子。
- 若 (x_k, y_k) 上没有棋子，称为普通走子，否则称为吃子。在吃子过程中，设正在走的棋子颜色为 col_1 ，等级为 lv_1 ，被吃的棋子颜色为 col_2 ，等级为 lv_2 ，则必须满足 $col_1 \neq col_2$, $lv_1 \geq lv_2$ ，换句话说只能吃与自己颜色不同，且等级不高于自己等级的棋子。

需要注意的是，由上述定义可以得出，不允许棋子在吃子后继续向前走。

网格线的状态含义如下所述：

- 如果 $opt_i = 0$ ，代表此路不通，走子时不能经过这条网格线。
- 如果 $opt_i = 1$ ，代表这条网格线是一条“普通道路”，每次走子时棋子最多只能经过 1 条普通道路。
- 如果 $opt_i = 2$ ，代表这条网格线是一条“直行道路”，每次走子时棋子可以经过任意条直行道路，但只能一直沿横向或一直沿纵向走，不能转弯。如沿直行道路从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(1, 3)$ 是可以的，但是从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(2, 2)$ 不行。

- 如果 $opt_i = 3$ ，代表这条网格线是一条“互通道路”，每次走子时棋子可以经过任意条互通道路，且中途可任意转弯。

同时规定在一次走子过程中，棋子经过的网格线的状态必须全部相同，比如从 $(1, 1)$ 经过直行道路走到 $(1, 2)$ 再经过互通道路走到 $(1, 3)$ 是不允许的。

至于如何判断胜负等其他细节，与本题无关，故略去。

小 z 和小 c 开发出这款棋类游戏后，为了提升水平，想了一个训练的策略：一开始棋盘是空的，然后小 c 会每次往棋盘的某个空交叉点上放一枚棋子，小 z 需要快速计算出：若选择这枚新放上的棋子进行一次走子，棋盘上一共有多少个位置是能被走到的？注意，因为这只是思维训练，他们并不会真的走这枚棋子。

可怜的小 z 发现他的计算力不足以算出这个问题，只好向你求助。

【输入格式】

从文件 **chess.in** 中读入数据。

每个测试点由多组数据组成。

第一行：一个正整数 T 表示数据组数。

对于每组数据：

第一行：3 个正整数 n, m, q ，分别表示棋盘的行数、列数和游戏的轮数。

接下来 n 行，每行为一个长 $m - 1$ 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i, j + 1)$ 的网格线状态。

接下来 $n - 1$ 行，每行为一个长 m 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i + 1, j)$ 的网格线状态。

接下来 q 行，每行 4 个非负整数 col_i, lv_i, x_i, y_i ，表示在第 i 轮有一枚颜色为 col_i ，等级为 lv_i 的棋子放在了交叉点 (x_i, y_i) 上。其中 $col_i = 0$ 表示黑子， $col_i = 1$ 表示白子。保证之前交叉点 (x_i, y_i) 上没有棋子。

【输出格式】

输出到文件 **chess.out** 中。

对于每组数据输出 q 行，每行一个非负整数，表示第 i 枚棋子放置后能走到的交叉点数量。

【样例 1 输入】

```
1 1
2 3 3 5
3 13
4 22
5 23
```

```
6 010
7 233
8 0 1 2 3
9 1 2 2 1
10 1 3 1 2
11 0 2 3 2
12 1 3 2 2
```

【样例 1 输出】

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

【样例 1 解释】

放置棋子 1 后，它能走到的位置为 $(2, 1), (2, 2), (3, 2), (3, 3)$ 。

放置棋子 2 后，它能走到的位置为 $(2, 2), (2, 3), (3, 1)$ 。

放置棋子 3 后，它能走到的位置为 $(1, 1), (1, 3), (2, 2)$ 。

放置棋子 4 后，它能走到的位置为 $(2, 2), (3, 1), (3, 3)$ 。

放置棋子 5 后，它能走到的位置为 $(2, 3), (3, 2)$ 。

【样例 2 输入】

```
1 2
2 2 3 4
3 22
4 33
5 123
6 0 2 1 2
7 0 1 2 1
8 1 2 1 3
9 0 3 2 2
10 3 2 3
11 3
```

```

12 1
13 3
14 32
15 32
16 0 2 1 2
17 1 2 3 2
18 0 1 2 2

```

【样例 2 输出】

```

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

```

【样例 3】

见选手目录下的 *chess/chess3.in* 与 *chess/chess3.ans*。

【样例 4】

见选手目录下的 *chess/chess4.in* 与 *chess/chess4.ans*。

【数据范围】

测试点编号	$n \times m \leq$	$q \leq$	特殊性质
1 ~ 2	100	50	无
3 ~ 6	5000	2000	
7 ~ 8			不存在“直行道路”与“互通道路”
9 ~ 11			不存在“互通道路”
12 ~ 14			不存在“直行道路”
15 ~ 16			$lv_i = i$
17 ~ 18			$lv_i = q - i + 1$
19 ~ 21		2000	$n, m \leq 1000$
22 ~ 25		10^5	无

对于 100% 的数据， $1 \leq T \leq 5$ ， $2 \leq n, m \leq 10^5$ ， $4 \leq n \times m \leq 2 \times 10^5$ ， $1 \leq q \leq \min\{10^5, n \times m\}$ ， $1 \leq lv_i \leq q$ ， $1 \leq x_i \leq n$ ， $1 \leq y_i \leq m$ ， $col_i \in \{0, 1\}$ 。

注：由于本题输入输出规模较大，建议使用较为快速的输入输出方式。

报数 (number)

【题目描述】

报数游戏是一个广为流传的休闲小游戏。参加游戏的每个人要按一定顺序轮流报数，但如果下一个报的数是 7 的倍数，或十进制表示中含有数字 7，就必须跳过这个数，否则就输掉了游戏。

在一个风和日丽的下午，刚刚结束 *SPC20nn* 比赛的小 r 和小 z 闲得无聊玩起了这个报数游戏。但在只有两个人玩的情况下计算起来还是比较容易的，因此他们玩了很久也没分出胜负。此时小 z 灵光一闪，决定把这个游戏加强：任何一个十进制中含有数字 7 的数，它的所有倍数都不能报出来！

形式化地，设 $p(x)$ 表示 x 的十进制表示中是否含有数字 7，若含有则 $p(x) = 1$ ，否则 $p(x) = 0$ 。则一个正整数 x 不能被报出，当且仅当存在正整数 y 和 z ，使得 $x = yz$ 且 $p(y) = 1$ 。

例如，如果小 r 报出了 6，由于 7 不能报，所以小 z 下一个需要报 8；如果小 r 报出了 33，则由于 $34 = 17 \times 2$, $35 = 7 \times 5$ 都不能报，小 z 下一个需要报出 36；如果小 r 报出了 69，由于 $70 \sim 79$ 的数都含有 7，小 z 下一个需要报出 80 才行。

现在小 r 的上一个数报出了 x ，小 z 想快速算出他下一个数要报多少，不过他很快就发现这个游戏可比原版的游戏难算多了，于是他需要你的帮助。当然，如果小 r 报出的 x 本身是不能报出的，你也要快速反应过来小 r 输了才行。

由于小 r 和小 z 玩了很长时间游戏，你也需要回答小 z 的很多个问题。

【输入格式】

从文件 *number.in* 中读入数据。

第 1 行，一个正整数 T 表示小 z 询问的数量。

接下来 T 行，每行 1 个正整数 x ，表示这一次小 r 报出的数。

【输出格式】

输出到文件 *number.out* 中。

输出共 T 行，每行一个整数，如果小 r 这一次报出的数是不能报出的，输出 -1 ，否则输出小 z 下一次报出的数是多少。

【样例 1 输入】

```
1 4
2 6
3 33
```

```
4 69  
5 300
```

【样例 1 输出】

```
1 8  
2 36  
3 80  
4 -1
```

【样例 1 解释】

这一组样例的前 3 次询问在题目描述中已有解释。

对于第 4 次询问，由于 $300 = 75 \times 4$ ，而 75 中含有 7，所以小 r 直接输掉了游戏。

【样例 2 输入】

```
1 5  
2 90  
3 99  
4 106  
5 114  
6 169
```

【样例 2 输出】

```
1 92  
2 100  
3 109  
4 -1  
5 180
```

【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*。

【数据范围】

对于 10% 的数据, $T \leq 10, x \leq 100$ 。

对于 30% 的数据, $T \leq 100, x \leq 1000$ 。

对于 50% 的数据, $T \leq 1000, x \leq 10000$ 。

对于 70% 的数据, $T \leq 10000, x \leq 2 \times 10^5$ 。

对于 100% 的数据, $T \leq 2 \times 10^5, x \leq 10^7$ 。

数列 (sequence)

【题目描述】

给定整数 n, m, k , 和一个长度为 $m + 1$ 的正整数数组 v_0, v_1, \dots, v_m 。

对于一个长度为 n , 下标从 1 开始且每个元素均不超过 m 的非负整数序列 $\{a_i\}$, 我们定义它的权值为 $v_{a_1} \times v_{a_2} \times \dots \times v_{a_n}$ 。

当这样的序列 $\{a_i\}$ 满足整数 $S = 2^{a_1} + 2^{a_2} + \dots + 2^{a_n}$ 的二进制表示中 1 的个数不超过 k 时, 我们认为 $\{a_i\}$ 是一个合法序列。

计算所有合法序列 $\{a_i\}$ 的权值和对 998244353 取模的结果。

【输入格式】

从文件 *sequence.in* 中读入数据。

输入的一行是三个整数 n, m, k 。

第二行 $m + 1$ 个整数, 分别是 v_0, v_1, \dots, v_m 。

【输出格式】

输出到文件 *sequence.out* 中。

仅一行一个整数, 表示所有合法序列的权值和对 998244353 取模的结果。

【样例输入】

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

【样例输出】

```
1 40
```

【样例 1 解释】

由于 $k = 1$, 而且由 $n \leq S \leq n \times 2^m$ 知道 $5 \leq S \leq 10$, 合法的 S 只有一种可能: $S = 8$, 这要求 a 中必须有 2 个 0 和 3 个 1, 于是有 $\binom{5}{2} = 10$ 种可能的序列, 每种序列的贡献都是 $v_0^2 v_1^3 = 4$, 权值和为 $10 \times 4 = 40$ 。

【样例 2】

见选手目录下的 *sequence/sequence2.in* 与 *sequence/sequence2.ans*。

【数据范围】

对所有测试点保证 $1 \leq k \leq n \leq 30, 0 \leq m \leq 100, 1 \leq v_i < 998244353$ 。

测试点	n	k	m	
1 ~ 4	= 8	$\leq n$	= 9	
5 ~ 7	= 30		= 7	
8 ~ 10			= 12	
11 ~ 13	= 5	= 1	= 100	
14 ~ 15		$\leq n$	= 50	
16	= 15		= 100	
17 ~ 18	= 30		= 30	
19 ~ 20			= 100	

方差 (variance)

【题目描述】

给定长度为 n 的非严格递增正整数数列 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。每次可以进行的操作是：任意选择一个正整数 $1 < i < n$ ，将 a_i 变为 $a_{i-1} + a_{i+1} - a_i$ 。求在若干次操作之后，该数列的方差最小值是多少。请输出最小值乘以 n^2 的结果。

其中方差的定义为：数列中每个数与平均值的差的平方的平均值。更形式化地说，方差的定义为 $D = \frac{1}{n} \sum_{i=1}^n (a_i - \bar{a})^2$ ，其中 $\bar{a} = \frac{1}{n} \sum_{i=1}^n a_i$ 。

【输入格式】

从文件 *variance.in* 中读入数据。

输入的第一行包含一个正整数 n ，保证 $n \leq 10^4$ 。

输入的第二行有 n 个正整数，其中第 i 个数字表示 a_i 的值。数据保证 $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ 。

【输出格式】

输出到文件 *variance.out* 中。

输出仅一行，包含一个非负整数，表示你所求的方差的最小值的 n^2 倍。

【样例 1 输入】

```
1 4
2 1 2 4 6
```

【样例 1 输出】

```
1 52
```

【样例 1 解释】

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，第一次操作得到的数列有 $(1, 3, 4, 6)$ ，第二次操作得到的新的数列有 $(1, 3, 5, 6)$ 。之后无法得到新的数列。

对于 $(a_1, a_2, a_3, a_4) = (1, 2, 4, 6)$ ，平均值为 $\frac{13}{4}$ ，方差为 $\frac{1}{4}((1 - \frac{13}{4})^2 + (2 - \frac{13}{4})^2 + (4 - \frac{13}{4})^2 + (6 - \frac{13}{4})^2) = \frac{59}{16}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 4, 6)$ ，平均值为 $\frac{7}{2}$ ，方差为 $\frac{1}{4}((1 - \frac{7}{2})^2 + (3 - \frac{7}{2})^2 + (4 - \frac{7}{2})^2 + (6 - \frac{7}{2})^2) = \frac{13}{4}$ 。

对于 $(a_1, a_2, a_3, a_4) = (1, 3, 5, 6)$, 平均值为 $\frac{15}{4}$, 方差为 $\frac{1}{4}((1 - \frac{15}{4})^2 + (3 - \frac{15}{4})^2 + (5 - \frac{15}{4})^2 + (6 - \frac{15}{4})^2) = \frac{59}{16}$ 。

【样例 2】

见选手目录下的 *variance/variance2.in* 与 *variance/variance2.ans*。

【样例 3】

见选手目录下的 *variance/variance3.in* 与 *variance/variance3.ans*。

【样例 4】

见选手目录下的 *variance/variance4.in* 与 *variance/variance4.ans*。

【数据范围】

测试点编号	$n \leq$	$a_i \leq$
1 ~ 3	4	10
4 ~ 5	10	40
6 ~ 8	15	20
9 ~ 12	20	300
13 ~ 15	50	70
16 ~ 18	100	40
19 ~ 22	400	600
23 ~ 25	10000	50

对于所有的数据, 保证 $n \leq 10000$, $a_i \leq 600$ 。

棋局 (chess)

【题目背景】

在输了一晚上的麻将之后，小 z 和小 c 卸掉了手机上的所有牌类游戏。不过这怎么可能阻挡得了他们上课颓废的决心呢？现在他们的目光盯在了棋类游戏上，但他们两个除了天天下飞行器以外，几乎所有棋类游戏都只懂个大概规则。

“既然我们都会玩但只能玩一点点，不如我们自己搞个缝合怪出来吧！”

于是，在他们的精心脑洞之下，一个融合了围棋、象棋与军棋的奇妙游戏诞生了……

【题目描述】

游戏在一张长 n 行宽 m 列的网格形棋盘上进行，棋子落在网格的交叉点上。我们不妨记左上角的交叉点的坐标为 $(1, 1)$ ，右下角的交叉点坐标为 (n, m) 。

棋子分为黑白两色，对局双方各执一方棋子。

每个棋子除了颜色以外还有等级，不妨设 col_i 为棋子 i 的颜色， lv_i 为棋子 i 的等级。另外，棋盘上的网格线共有 4 种状态，对于第 i 条网格线，设其状态为 opt_i 。

轮到每方下棋时，他可以选择棋盘上的一个己方棋子沿网格线进行移动到另一个交叉点，称为走子。形式化定义走子的过程如下：选择一个坐标序列 $(x_0, y_0), (x_1, y_1), \dots, (x_k, y_k)$ ，其中 k 是任意选定的正整数， (x_0, y_0) 是棋子初始的位置， (x_k, y_k) 是棋子最终走到的位置，需要满足：

- 对于任意 $i = 0, 1, \dots, k - 1$ ，坐标 (x_i, y_i) 和 (x_{i+1}, y_{i+1}) 之间必须有网格线直接相连，也就是说走子必须沿着网格线走；
- 对于任意 $i \neq j$ ，必须有 $(x_i, y_i) \neq (x_j, y_j)$ ，也就是说走子过程中不能经过重复位置，特别地 $(x_0, y_0) \neq (x_k, y_k)$ ，也就是说不能原地不动（或走回原地）。
- 对于任意 $i = 1, \dots, k - 1$ ，坐标 (x_i, y_i) 上必须没有棋子，也就是说走子时不能越过已有的棋子。
- 若 (x_k, y_k) 上没有棋子，称为普通走子，否则称为吃子。在吃子过程中，设正在走的棋子颜色为 col_1 ，等级为 lv_1 ，被吃的棋子颜色为 col_2 ，等级为 lv_2 ，则必须满足 $col_1 \neq col_2$, $lv_1 \geq lv_2$ ，换句话说只能吃与自己颜色不同，且等级不高于自己等级的棋子。

需要注意的是，由上述定义可以得出，不允许棋子在吃子后继续向前走。

网格线的状态含义如下所述：

- 如果 $opt_i = 0$ ，代表此路不通，走子时不能经过这条网格线。
- 如果 $opt_i = 1$ ，代表这条网格线是一条“普通道路”，每次走子时棋子最多只能经过 1 条普通道路。
- 如果 $opt_i = 2$ ，代表这条网格线是一条“直行道路”，每次走子时棋子可以经过任意条直行道路，但只能一直沿横向或一直沿纵向走，不能转弯。如沿直行道路从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(1, 3)$ 是可以的，但是从 $(1, 1)$ 经过 $(1, 2)$ 走到 $(2, 2)$ 不行。

- 如果 $opt_i = 3$ ，代表这条网格线是一条“互通道路”，每次走子时棋子可以经过任意条互通道路，且中途可任意转弯。

同时规定在一次走子过程中，棋子经过的网格线的状态必须全部相同，比如从 $(1, 1)$ 经过直行道路走到 $(1, 2)$ 再经过互通道路走到 $(1, 3)$ 是不允许的。

至于如何判断胜负等其他细节，与本题无关，故略去。

小 z 和小 c 开发出这款棋类游戏后，为了提升水平，想了一个训练的策略：一开始棋盘是空的，然后小 c 会每次往棋盘的某个空交叉点上放一枚棋子，小 z 需要快速计算出：若选择这枚新放上的棋子进行一次走子，棋盘上一共有多少个位置是能被走到的？注意，因为这只是思维训练，他们并不会真的走这枚棋子。

可怜的小 z 发现他的计算力不足以算出这个问题，只好向你求助。

【输入格式】

从文件 **chess.in** 中读入数据。

每个测试点由多组数据组成。

第一行：一个正整数 T 表示数据组数。

对于每组数据：

第一行：3 个正整数 n, m, q ，分别表示棋盘的行数、列数和游戏的轮数。

接下来 n 行，每行为一个长 $m - 1$ 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i, j + 1)$ 的网格线状态。

接下来 $n - 1$ 行，每行为一个长 m 的字符串，每个字符为 0、1、2、3 中的一个，第 i 行第 j 个字符表示交叉点 (i, j) 连向交叉点 $(i + 1, j)$ 的网格线状态。

接下来 q 行，每行 4 个非负整数 col_i, lv_i, x_i, y_i ，表示在第 i 轮有一枚颜色为 col_i ，等级为 lv_i 的棋子放在了交叉点 (x_i, y_i) 上。其中 $col_i = 0$ 表示黑子， $col_i = 1$ 表示白子。保证之前交叉点 (x_i, y_i) 上没有棋子。

【输出格式】

输出到文件 **chess.out** 中。

对于每组数据输出 q 行，每行一个非负整数，表示第 i 枚棋子放置后能走到的交叉点数量。

【样例 1 输入】

```
1 1
2 3 3 5
3 13
4 22
5 23
```

```
6 010
7 233
8 0 1 2 3
9 1 2 2 1
10 1 3 1 2
11 0 2 3 2
12 1 3 2 2
```

【样例 1 输出】

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

【样例 1 解释】

放置棋子 1 后，它能走到的位置为 $(2, 1), (2, 2), (3, 2), (3, 3)$ 。

放置棋子 2 后，它能走到的位置为 $(2, 2), (2, 3), (3, 1)$ 。

放置棋子 3 后，它能走到的位置为 $(1, 1), (1, 3), (2, 2)$ 。

放置棋子 4 后，它能走到的位置为 $(2, 2), (3, 1), (3, 3)$ 。

放置棋子 5 后，它能走到的位置为 $(2, 3), (3, 2)$ 。

【样例 2 输入】

```
1 2
2 2 3 4
3 22
4 33
5 123
6 0 2 1 2
7 0 1 2 1
8 1 2 1 3
9 0 3 2 2
10 3 2 3
11 3
```

```

12 1
13 3
14 32
15 32
16 0 2 1 2
17 1 2 3 2
18 0 1 2 2

```

【样例 2 输出】

```

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

```

【样例 3】

见选手目录下的 *chess/chess3.in* 与 *chess/chess3.ans*。

【样例 4】

见选手目录下的 *chess/chess4.in* 与 *chess/chess4.ans*。

【数据范围】

测试点编号	$n \times m \leq$	$q \leq$	特殊性质
1 ~ 2	100	50	无
3 ~ 6	5000	2000	
7 ~ 8			不存在“直行道路”与“互通道路”
9 ~ 11			不存在“互通道路”
12 ~ 14			不存在“直行道路”
15 ~ 16			$lv_i = i$
17 ~ 18			$lv_i = q - i + 1$
19 ~ 21		2000	$n, m \leq 1000$
22 ~ 25		10^5	无

对于 100% 的数据， $1 \leq T \leq 5$ ， $2 \leq n, m \leq 10^5$ ， $4 \leq n \times m \leq 2 \times 10^5$ ， $1 \leq q \leq \min\{10^5, n \times m\}$ ， $1 \leq lv_i \leq q$ ， $1 \leq x_i \leq n$ ， $1 \leq y_i \leq m$ ， $col_i \in \{0, 1\}$ 。

注：由于本题输入输出规模较大，建议使用较为快速的输入输出方式。