

# 总k次方差

时间限制: 1.0 秒

空间限制: 512 MB



## 题目描述

对于一个数列 $a_1, a_2 \dots a_n$ , 我们定义它的总 $k$ 次方差为:

$$\sum_{i=1}^n \sum_{j=1}^n |a_i - a_j|^k$$

现在, 输入数列 $a_i$ , 求它的总 $k$ 次方差。

## 输入格式

从标准输入读入数据。

输入的第一行包含两个整数 $n, k$ , 如题意所述。

输入的第二行包含数组 $a_1, a_2 \dots a_n$ , 如题意所述。

## 输出格式

输出到标准输出。

输出一个整数, 表示输入数列的总 $k$ 次方差。

为了避免输出规模过大, 我们要求你输出这个答案除以998244353的余数。

## 样例输入

```
3 1
1 7 2
```

## 样例输出

```
24
```

## 子任务

对于50的数据,  $k = 1, n \leq 100, 0 \leq a_i \leq 1000$ ;

对于另外50的数据,  $k = 1, n \leq 100000, 0 \leq a_i \leq 100000$ ;

语言及编译选项信息

#	名称	编译器	额外参数	代码长度限制 (B)
0	g++ with std11	g++	-O2 -std=c++11 -DONLINE_JUDGE	65536
1	g++	g++	-O2 -DONLINE_JUDGE	65536
2	gcc with std11	gcc	-O2 -std=c11 -DONLINE_JUDGE	65536
3	gcc	gcc	-O2 -DONLINE_JUDGE	65536
4	java	javac		65536
5	python	python		65536
6	python3	python3		65536

递交历史

#	状态	时间
34628	<div>Accepted</div> (/#!/contest/66/detail/34628)	06:33:48 PM <div>有效递交</div>
34589	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34589)	06:07:05 PM

1

递交答案 (剩余次数: 30)

语言和编译选项

g++ with std11

1

递交评测

文件请拖入编辑器中，或

上传文件

# 考试

时间限制：1.0 秒

空间限制：512 MiB



## 题目描述

在这一学期一共有  $n$  次考试，考试科目有 4 种，分别为计原、网原、信原和操统。

每次考哪一科是不定的，但通过总结以往考试经验，发现了这样几个规律：

- 1.如果这次考的是计原，那么下一次一定会考网原；
- 2.如果这次考的是信原，那么下一次一定会考操统；
- 3.如果这次考的是网原，那么下一次要么考计原，要么考操统；
- 4.如果这次考的是操统，那么下一次要么考网原，要么考信原。

已经知道，本学期的第一次考试科目为计原。打算拟定一个可以应对所有可能情况的应考复习计划。问整个学期有多少种可能的考试科目安排满足以上规律。

## 输入格式

从标准输入读入数据。

一个正整数  $n$ ，代表本学期总的考试次数。输入数据保证  $n \leq 10000$ 。

## 输出格式

输出到标准输出。

一个正整数，表示符合规律的科目安排方案的总数。考虑到这个结果可能会很大，因此你只需要输出它 mod 7654321 的值即可。

## 样例输入

5

## 样例输出

5

语言及编译选项信息

#	名称	编译器	额外参数	代码长度限制 (B)
0	g++ with std11	g++	-O2 -std=c++11 -DONLINE_JUDGE	65536
1	g++	g++	-O2 -DONLINE_JUDGE	65536
2	gcc with std11	gcc	-O2 -std=c11 -DONLINE_JUDGE	65536
3	gcc	gcc	-O2 -DONLINE_JUDGE	65536
4	java	javac		65536
5	python	python		65536
6	python3	python3		65536

递交历史

#	状态	时间
34788	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34788)	08:49:27 PM <div>有效递交</div>
34785	<div>Compilation Error</div> (/#!/contest/66/detail/34785)	08:48:07 PM
34780	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34780)	08:46:17 PM
34777	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34777)	08:45:36 PM
34761	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34761)	08:33:28 PM
34755	<div>Wrong Answer</div> (/#!/contest/66/detail/34755)	08:31:58 PM
34660	<div>Time Limit Exceeded</div> (/#!/contest/66/detail/34660)	07:03:42 PM

1

递交答案 (剩余次数: 26)

语言和编译选项

g++ with std11

递交评测

文件请拖入编辑器中，或

上传文件

# 租约机制

时间限制：2.0 秒

空间限制：512 MB

相关文件：题目目录



## 题目描述

鸽鸽用一台土豆机器搭建了自己的博客，一开始只有几百人访问，一台土豆机器的单机系统工作非常流畅。现在有几百万人访问，一台土豆机器无法满足性能需求，给用户的体验很差。鸽鸽准备用多台土豆机器搭建一个分布式土豆系统来提高性能。

不同的用户实际访问的机器是不同的，但是效果必须得是相同。鸽鸽搭建系统时，面临的第一个问题就是，某台机器修改数据后，如何让所有的机器也获取到最新数据。一个朴素的想法，每台机器都存储一份完整的数据，任意一台机器更新数据时都将最新的数据广播给其他机器，让所有机器同时更新。但是这个办法很难保证所有机器都同时更新数据。可能第一台机器更新了数据，第二台机器还没来得及更新数据，在这期间有两个用户分别通过第一台和第二台机器来访问该数据。在这种场景中，两个用户得到的数据就会不一样，一个是新数据，一个是旧数据，这就造成了数据不一致。同时这种方法需要广播数据，性能也较低。鸽鸽想了想，决定只让一台机器作为中心节点可以更新数据，其他机器作为辅助节点可以访问但是无法更新数据。

鸽鸽的这个分布式系统中，有一个中心节点，存储、维护数据。系统中的其他节点通过访问中心节点读取、修改其上的数据。除了中心节点，其他节点初始化是没有数据的。当用户访问到某个节点，请求读取数据时，如果它没有数据就会从中心节点读取数据，然后返回给用户。如果每次用户访问数据，都要从中心节点读取数据，性能比单机系统还差。所以从中心节点获取数据的同时，会获取一个租约时间，**如果当前时间不超过（不大于）租约，就将租约和数据都保存下来**。下次用户访问时，如果**当前时间没有超过租约（租约没到期）**，就**直接返回数据**；否则，先清空现有数据，再**从中心节点读取数据**。聪明的你，肯定想问，如果租约没到期直接返回数据，这个时候中心节点已经更新了数据，不也会出现数据不一致吗？你能想到这个问题，鸽鸽也能想到。所以中心节点在所有节点租约到期前，不会更新数据。中心节点会阻塞所有更新请求，将这些更新请求都按时间顺序保存下来，等**所有租约全部到期后（大于最晚租约的最小时间）**，再一一更新。聪明的你，肯定又想问，那如果一直有其他节点从中心节点获取更晚的租约，那不是会一直无法更新数据？你能想到这个问题，鸽鸽也能想到。所以中心节点在处理完所有更新数据请求前，发送给任何节点的租约只会是**当前已有的最晚租约**，不会更晚。当用户访问某个节点，请求更新数据时，它会把这个请求转发给中心节点。因为**只有中心节点才能更新数据，其他节点只是缓存中心节点的数据**。鸽鸽给自己的这个方法取了个名字，叫租约协议。

再叙述一遍租约协议的读写流程：

- 用户读数据：判断数据是否已经存在本地且租约未到期。是，直接返回本地的数据；否，向中心节点发送读取数据和租约请求，读取成功后返回数据，如果租约没到期，则在本地缓存数据。
- 用户写数据：被访问节点向中心节点发出写数据请求。中心节点收到写数据请求后，阻塞缓存下来，同时所有新的读数据请求只发放已有最晚的租约。中心节点等待所有租约到期后，一一处理缓存的写请求。

鸽鸽的分布式土豆系统非常神奇，保证时间都是一样的，网络也是永远健康良好的，其他的因素也都是正常可靠的。

给定  $n$  台机器，编号从 1 到  $n$ 。其中，1 号机器是中心节点。时间从 0 开始，租约时长为  $k$ 。假设在  $a$  时间，向中心节点发出读请求，如果没有需要处理的写请求时，中心节点返回的租约一定是  $a + k$ ，否则返回已有的最晚租约。中心节点处理一个写请求的时间为  $d$ 。假设在  $a$  时间一共有  $D$  个写请求需要处理，在  $a + Dd$  时间处理完成。**读操作会立即返回结果，写操作有执行延迟。**

现在有  $m$  个用户请求， $R\ t\ x$  表示在  $t$  时间向  $x$  节点发出读数据请求， $W\ t\ x$  表示在  $t$  时间向  $x$  节点发出写数据请求。对于每个  $R\ t\ x$  请求，输出  $x$  对应的操作： $RWB$  表示从中心节点读取数据和租约，再保存在本地，然后返回给用户； $RB$  表示从中心节点读取数据和租约，发现租约到期，不保存在本地，然后返回给用户； $B$  表示直接从本地读取数据返回给用户。**保证对于同一时间，可能会有多个读请求，但是只会有一个写请求。如果同一时间，同时有读写请求，先处理写请求。保证用户只会直接访问非中心节点。**

## 输入格式

从标准输入读入数据。

输入的第一行包含四个正整数  $n, m, k, d$ ，保证  $n \leq 10^5$ ， $m \leq 10^6$ ， $k \leq 10^2$ ， $d \leq 10^3$ 。

输入的接下来  $m$  行，每行一个字符、两个整数，用空格隔开，一定是  $R\ t\ x$  或者  $W\ t\ x$ 。保证操作是按时间顺序，前面任意操作的时间一定不超过后面任意操作的时间。保证所有时间  $t$  不超过  $10^9$ 。

## 输出格式

输出到标准输出。

输出若干行，每行一个字符串表示对应读操作的响应。

## 样例1输入

```
3 22 10 3
R 0 2
R 0 3
R 1 2
R 1 3
R 10 2
R 10 3
R 11 2
W 12 2
R 20 3
R 21 3
R 24 2
R 24 3
R 25 2
R 25 3
W 40 2
R 43 2
R 43 3
W 43 2
R 46 2
R 46 3
R 56 2
R 56 3
```

## 样例1输出



RWB  
 RWB  
 B  
 B  
 B  
 B  
 RWB  
 RWB  
 B  
 RB  
 RB  
 RWB  
 RWB  
 RB  
 RB  
 RWB  
 RWB  
 B  
 B

## 样例1解释

有 3 个节点，22 个操作，租约时长为 10，处理一个写操作时间  $d$  为 3。

0 时间，2 和 3 节点数据为空，用户读数据，需要从中心节点获取数据和租约，租约为  $0 + 10 = 10$ ，当前时间未超过租约，所以需要执行 RWB 操作。

1 时间，2 和 3 节点缓存了数据，租约为 10，没有超过租约，用户读数据，直接返回，所以需要执行 B 操作。

10 时间，2 和 3 节点缓存了数据，租约为 10，没有超过租约，用户读数据，直接返回，所以需要执行 B 操作。

11 时间，2 节点缓存了数据，租约为 10，超过租约，用户读数据，需要重新从中心节点获取数据和租约，租约为  $11 + 10 = 21$ ，当前时间未超过租约，所以需要执行 RWB 操作。

12 时间，2 节点向中心节点发送写请求，因为还有租约未超时，所以被阻塞保存下来，将在时间 22 进行。

20 时间，3 节点缓存了数据，租约为 10，超过租约，用户读数据，需要重新从中心节点获取数据和租约，因为有写请求未处理完，所以租约为已有最晚租约 21，当前时间未超过租约，所以需要执行 RWB 操作。

21 时间，3 节点缓存了数据，租约为 21，没有超过租约，用户读数据，直接返回，所以需要执行 B 操作。

22 时间，最晚租约为 21，所有租约过期，中心节点开始处理写请求。

24 时间，2 和 3 节点缓存了数据，租约为 21，超过租约，用户读数据，需要重新从中心节点获取数据和租约，因为有写请求未处理完，所以租约为已有最晚租约 21，当前时间超过租约，所以需要执行 RB 操作。

25 时间，2 和 3 节点缓存了数据，租约为 21，超过租约，用户读数据，需要重新从中心节点获取数据和租约，因为所有写请求已处理完，所以租约为  $25 + 10 = 35$ ，当前时间超过租约，所以需要执行 RWB 操作。

40 时间，2 节点向中心节点发送写请求，最晚租约为 35，所有租约过期，中心节点开始处理写请求。

43 时间，2 节点向中心节点发送写请求，最晚租约为 35，所有租约过期，中心节点开始处理写请求。

43 时间，2 和 3 节点缓存了数据，租约为 35，超过租约，用户读数据，需要重新从中心节点获取数据和租约，因为有写请求未处理完，所以租约为已有最晚租约 35，当前时间超过租约，所以需要执行 RB 操作。

46 时间，2 和 3 节点缓存了数据，租约为 35，超过租约，用户读数据，需要重新从中心节点获取数据和租约，因为所有写请求已处理完，所以租约为  $46 + 10 = 56$ ，当前时间超过租约，所以需要执行 RWB 操作。

56 时间，2 和 3 节点缓存了数据，租约为 56，没有超过租约，用户读数据，直接返回，所以需要执行 B 操作。

子任务

测试点编号	$n \leq$	$m \leq$	$k \leq$	$d \leq$	操作说明	分数
1	2	$10^2$	$10^2$	$10^3$	无	10
2	2	$10^2$	$10^2$	$10^3$	无	10
3	$10^2$	$10^3$	$10^2$	$10^3$	中心节点修改数据期间不会有用户访问	10
4	$10^2$	$10^3$	$10^2$	$10^3$	中心节点修改数据期间不会有用户访问	10
5	$10^3$	$10^4$	$10^2$	$10^3$	没有修改请求	10
6	$10^3$	$10^4$	$10^2$	$10^3$	没有修改请求	10
7	$10^4$	$10^5$	$10^2$	$10^3$	无	10
8	$10^4$	$10^5$	$10^2$	$10^3$	无	10
9	$10^5$	$10^6$	$10^2$	$10^3$	无	10
10	$10^5$	$10^6$	$10^2$	$10^3$	无	10

语言及编译选项信息

#	名称	编译器	额外参数	代码长度限制 (B)
0	g++ with std14	g++	-O2 -std=c++14 -DONLINE_JUDGE	65536
1	g++ with std98	g++	-O2 -std=c++98 -DONLINE_JUDGE	65536
2	gcc with std11	gcc	-O2 -std=c11 -DONLINE_JUDGE	65536
3	gcc with std99	gcc	-O2 -std=c99 -DONLINE_JUDGE	65536

#	名称	编译器	额外参数	代码长度限制 (B)
4	java	javac		65536
5	python	python		65536
6	python3	python3		65536

递交历史

#	状态	时间
34796	<div>Wrong Answer</div> <a href="#">(&lt;/#!&gt;/contest/66/detail/34796)</a>	08:54:13 PM <div>有效递交</div>
34791	<div>Wrong Answer</div> <a href="#">(&lt;/#!&gt;/contest/66/detail/34791)</a>	08:52:19 PM
34772	<div>Wrong Answer</div> <a href="#">(&lt;/#!&gt;/contest/66/detail/34772)</a>	08:38:40 PM
34738	<div>Wrong Answer</div> <a href="#">(&lt;/#!&gt;/contest/66/detail/34738)</a>	08:18:56 PM
34728	<div>Wrong Answer</div> <a href="#">(&lt;/#!&gt;/contest/66/detail/34728)</a>	08:07:49 PM

1

递交答案 (剩余次数: 27)

语言和编译选项

g++ with std14

1

递交评测

文件请拖入编辑器中，或

上传文件