20221026

题目名称	座位	几何	定点数	异或
题目类型	传统型	传统型	传统型	传统型
每个测试点时限	C/C++ 1 秒, 其他语言 2 秒	C/C++ 1 秒, 其他语言 2 秒	C/C++ 1 秒, 其他语言 2 秒	C/C++ 1 秒, 其他语言 2 秒
内存限制	C/C++ 128MB, 其他语言 256MB	C/C++ 128MB, 其他语言 256MB	C/C++ 128MB, 其他语言 256MB	C/C++ 128MB, 其他语言 256MB
子任务数目	10	40	40	20
测试点是否等分	是	是	是	是

1.座位

【题目描述】

受疫情影响，大家在就餐的时候需要隔位就坐。

给定一个长度为n的01串，1表示已经有人在此就坐并开始就餐，0表示无人就坐。

为最大化就餐人数，你需要在保证隔位就坐的前提下尽可能多的安排座位。

请你计算一下有多少种方案满足要求。

【输入格式】

第一行，包含一个正整数n。

第二行，一个长度为n的01串，表示目前的就坐情况。

【输出格式】

共一行，包含一个非负整数，表示方案数。

由于方案数可能会很大，你需要输出方案数对998244353取模后的值。

【样例 1 输入】

3

010

【样例 1 输出】

1

【样例 1 说明】

虽然无法分配座位，但是这仍然是一种方案。

【样例 2 输入】

4

1001

【样例 2 输出】

1

【样例 2 说明】

显然中间的空座位无论如何分配都不能保证间隔就做。

【样例 3 输入】

5

10001

【样例 3 输出】

1

【样例 3 说明】

给第三个位置分配一个人就坐，可以最大化用餐人数。

【样例 4 输入】

6

100001

【样例 4 输出】

2

【样例 4 说明】

给第三个位置或者第四个位置分配一个人就坐，可以最大化用餐人数，方案数为 2。

【样例 5 输入】

3

111

【样例 5 输出】

0

【样例 5 说明】

目前的就坐情况已经不满足间隔就坐要求，方案数为 0。

【样例 6 输入】

20

00000001000000010000

【样例 6 输出】

4

【样例 7 输入】

50

00000000010000000000000001000000000000100000001000

【样例 7 输出】

60

【样例 8 输入】

100

00000000010000000000000001000000000000100000001000000000000100000

00100000001000000000000100000001000

【样例 8 输出】

2160

【数据范围】

对于100%的数据：$3 \le n \le 300000$。

本题共有10个测试点，部分测试点满足以下性质:

测试点1：$n \leq 15$。

测试点2：字符串至多有15个位置为0，其余位置为1。

测试点3：字符串只包含0。

测试点4：字符串只包含1。

测试点5：字符串只包含1个1。

测试点6：字符串只包含2个1。

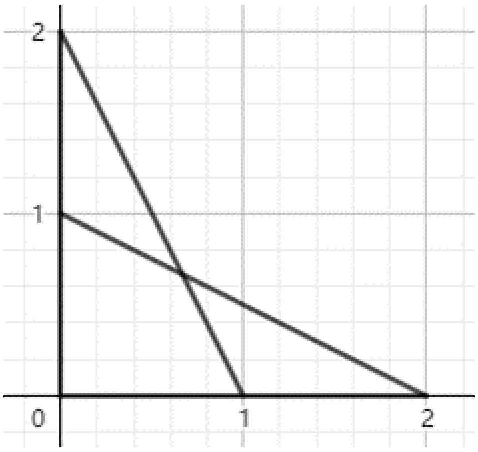
2.几何

【题目描述】

相信平面几何对你来说一定不陌生。现在，请你拿出纸和笔来，尝试解决这样一道几何题。

给定一个整数g，在第一象限内画出所有端点分别在两个坐标轴上，且端点横纵坐标和为g的所有线段，请你尝试计算出这些线段与坐标轴围成的图形中包含的三角形个数。

例如，当g为3时，你将画出$(0,2)(1,0)$与$(0,1)(2,0)$两条线段，这些线段与坐标轴围成的图形如图所示。图中共有4个三角形，其顶点坐标分别为$(0,2)(0,0)(1,0)$、$(0,2)(0,1)(2/3,2/3)$、$(1,0)(0,0)(2,0)$、$(2/3,2/3)(1,0)(2,0)$。



【输入格式】

共一行，为一个整数g，含义如题目描述所述。

【输出格式】

共一行，表示你所求解的三角形数量。

由于这个数量可能会很大，你只需要输出数量对998244353取模后的值。

【样例 1 输入】

2

【样例 1 输出】

1

【样例 2 输入】

3

【样例 2 输出】

4

【样例 2 说明】

样例解释请见问题描述部分。

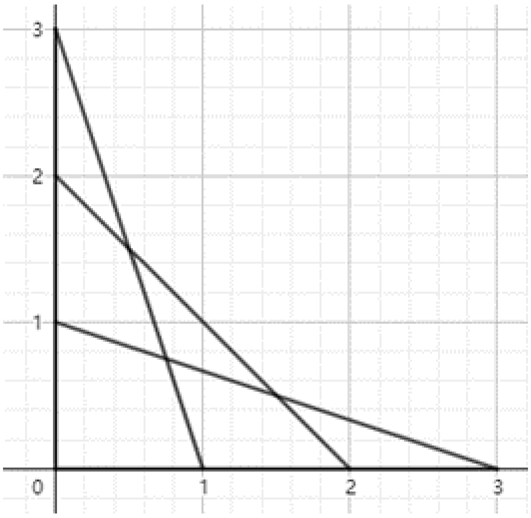
【样例 3 输入】

4

【样例 3 输出】

10

【样例 3 说明】



【样例 4 输入】

5

【样例 4 输出】

20

【样例 5 输入】

996

【样例 5 输出】

164674490

【样例 6 输入】

999996

【样例 6 输出】

608861885

【样例 7 输入】

999999996

【样例 7 输出】

536236838

【数据范围】

对于10%的数据：$g \le 5$。

对于20%的数据：$g \le 10$。

对于40%的数据：$g \le 10^3$。

对于70%的数据：$g \le 10^6$。

对于100%的数据：$2 \le g \le 10^9$。

可能会用到的公式：$(n+1)^3 - n^3 = 3n^2 + 3n + 1$

3.定点数

【题目描述】

计算机在处理大多数问题时，都免不了与小数打交道。对于“确切”的计算机来说，“不确切”的小数自然而然显得格格不入。科学家们提出了两种方式来解决这个棘手的问题——定点数与浮点数，而我们今天要讨论的主题便是定点数。

顾名思义，在采用定点数表示小数时，小数点隐含在某一个固定的位置上。例如，现有一个8位二进制定点数，规定整数部分占4位、小数部分占4位，则二进制定点数10101000（带小数点时表示为1010.1000），即为十进制下的10.5。

阅读了上面介绍，相信聪明的你一定明白了定点数的概念，能够灵活运用这一数据类型了。那么，这里有一道关于定点数的问题，赶快来试试看吧！

给定两个十进制小数p,q和一个算符（+、 − 、 ∗ 、/，分别表示加、减、乘、除），请你分别将他们转换为32位二进制定点数，并尝试通过定点数运算得到运算结果。

为方便计算，本题中所有数据的类型均为无符号型（包括输入的十进制小数、运算结果的最终表示等）。运算时，本题规定一个32位二进制定点数的整数部分占24位，小数部分占8位。出现精度损失时，最小化因精度损失带来的误差（也即，若定点数p和q为两个相邻的定点数、待转换的定点数为x，当$p \leq x < (p+q)/2$时，你需要输出p，否则你需要输出q）。输出时，你需要将定点数转换为十进制

小数形式输出，并保留至小数点后8位。

【输入格式】

第一行包含两个十进制小数p和q。

第二行包含一个算符op 。

【输出格式】

第一行包含两个十进制小数u和v，表示转换结果。

第二行包含一个十进制小数r，表示运算结果。

【样例 1 输入】

1.0 2.5

+

【样例 1 输出】

1.00000000 2.50000000

3.50000000

【样例 1 说明】

p 的值为 1.0，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0001.0000\ 0000$，也即十进制表示下的$1.0$。

q 的值为 2.5，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0010.1000\ 0000$，也即十进制表示下的$2.5$。

两个定点数求和，结果为$0000\ 0000\ 0000\ 0000\ 0000\ 0011.1000\ 0000$，也即十进制表示下的$3.5$。

【样例 2 输入】

0.03125 0.03125

*

【样例 2 输出】

0.03125000 0.03125000

0.00000000

【样例 2 说明】

p 和 q 的值为 0.03125，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0000.0000\ 1000$，也即十进制表示下的$0.03125$。

两个定点数相乘，结果为$0000\ 0000\ 0000\ 0000\ 0000\ 0000.0000\ 0000\ (1000)$，此时乘积结果已经超出了定点数所能表示的范围（上述示例用括号补充出了正确的运算结果），因此在十进制表示下为0.0。

【样例 3 输入】

0.5 1.0

-

【样例 3 输出】

0.50000000 1.00000000

16777215.50000000

【样例 3 说明】

p 的值为 0.5，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0000.1000\ 0000$，也即十进制表示下的 0.5。

q 的值为 1.0，转换为 32 位二进制定点数为 0000 0000 0000 0000 0000 0001.0000 0000，也即十进制表示下的1.0。

两个定点数作差，结果为1111 1111 1111 1111 1111 1111.1000 0000，也即十进制表示下的16777215.5。

【样例 4 输入】

8388608.0 0.25

/

【样例 4 输出】

8388608.00000000 0.25000000

0.00000000

【样例 4 说明】

p 的值为 8388608.0，转换为 32 位二进制定点数为 1000 0000 0000 0000 0000 0000.0000 0000，也即十进制表示下的 8388608.00000000。

q 的值为 0.25，转换为 32 位二进制定点数为 0000 0000 0000 0000 0000 0000.0100 0000，也即十进制表示下的0.25。

两个定点数作商，结果为(0010) 0000 0000 0000 0000 0000 0000.0000 0000，此时乘积结果已经超出了定点数所能表示的范围（上述示例用括号补充出了正确

的运算结果)，因此在十进制表示下为0.0。

【样例 5 输入】

0.3 1.2

+

【样例 5 输出】

0.30078125 1.19921875

1.50000000

【样例 5 说明】

p 的值为 0.3，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0000.0100\ 1101$，也即十进制表示下的$0.30078125$。

q 的值为 1.2，转换为 32 位二进制定点数为 $0000\ 0000\ 0000\ 0000\ 0000\ 0001.0011\ 0011$，也即十进制表示下的$1.19921875$。

两个定点数求和，结果为$0000\ 0000\ 0000\ 0000\ 0000\ 0001.1000\ 0000$，也即十进制表示下的$1.5$。

【样例 6 输入】

1.2 0.3

-

【样例 6 输出】

1.19921875 0.30078125

0.89843750

【样例 6 说明】

p 的值为 1.2，转换为 32 位二进制定点数为 0000 0000 0000 0000 0000 0001.0011 0011，也即十进制表示下的1.19921875。

q 的值为 0.3，转换为 32 位二进制定点数为 0000 0000 0000 0000 0000 0000.0100 1101，也即十进制表示下的0.30078125。

两个定点数作差，结果为0000 0000 0000 0000 0000 0000.1110 0110，也即十进制表示下的0.89843750。

【数据范围】

对于全部测试数据，25%的数据运算为加法，25%的数据运算为减法，25%的数据运算为乘法，25%的数据运算为除法。保证除法运算时，小数转换为定点数后不为0。

特别的，有至少10%的数据在类型转换时不会出现精度损失，另有至少20%的数据计算时不会出现下溢（即样例2出现的情况），另有至少20%的数据计算时不会出现上溢（即样例4出现的情况）。

4.异或

【题目描述】

异或是数学运算的一种。在计算机科学中，如同其他位运算一样，异或运算的定义为：对于任意两个整数a和b的每一位，当且仅当a和b在这一位上的值不同，运算结果在这一位为1，否则为0。

例如，3异或5的结果为6。根据上述运算法则，3的四位二进制表示为0011，5的四位二进制表示为0101。对于最高位和最低位，3和5在每一位上的值相同，所以结果为0；对于其余两位，3和5在每一位的值不同，所以结果为1。因此，0011异或0101的结果为0110，也即十进制下的6。

阅读了上面介绍，相信聪明的你一定学会了异或运算的法则，能够灵活运用这一运算了。那么，这里有一道关于异或运算的问题，赶快来试试看吧！

现有一个长度为n的数组（下标为0到$n-1$），其初始值均为0。

你需要对其进行m次操作，每次操作需要以从l到r的每个数字异或p作为操作下标（若操作下标超过数组容量大小，则对这一下标的操作可以跳过），对数组这些下标的元素异或q。

最后，你需要依次输出这个数组每个元素的值（保证数组的每个元素在这m次操作过程中均在32位无符号整数范围内）。

【输入格式】

第一行包含两个整数n和m。

接下来有m行，每行四个整数l，r，p，q，含义见题目描述。

【输出格式】

只有一行，包含 n 个非负整数，即在 m 次操作后数组每个元素的值。

【样例 1 输入】

4 2

0 2 0 3

1 3 0 5

【样例 1 输出】

3 6 6 5

【样例 1 说明】

第一次操作，操作下标为$0,1,2$，数组变为：$3,3,3,0$

第二次操作，操作下标为$1,2,3$，数组变为：$3,6,6,5$

【样例 2 输入】

4 2

0 2 0 3

1 3 1 5

【样例 2 输出】

6 3 6 5

【样例 2 说明】

第一次操作，操作下标为$0,1,2$，数组变为：$3,3,3,0$

第二次操作，操作下标为$0,3,2$，数组变为：$6,3,6,5$

【样例 3 输入】

4 2

0 2 1 3

1 3 1 5

【样例 3 输出】

6 3 5 6

【样例 3 说明】

第一次操作，操作下标为$1,0,3$，数组变为：$3,3,0,3$

第二次操作，操作下标为$0,3,2$，数组变为：$6,3,5,6$

【数据范围】

对于所有的测试点，满足$1 \le n,m \le 2^{18}$，$0 \le l \le r < n$，保证数组的每个元素在这m次操作过程中均在32位无符号整数范围内。

请选手注意由于输入输出带来的时间开销对题目时间限制的影响。

<table>
<tr><th>测试点编号</th><th>n</th><th>m</th><th>特殊性质</th></tr>
<tr><td>1</td><td rowspan="2">$n \le 2^4$</td><td rowspan="2">$m \le 2^4$</td><td rowspan="2">无特殊性质</td></tr>
<tr><td>2</td></tr>
<tr><td>3</td><td rowspan="4">$n \le 2^{10}$</td><td rowspan="2">$m \le 2^{10}$</td><td>$log_2(l)$为整数，$log_2(r+1) = log_2(l) + 1$</td></tr>
<tr><td>4</td><td>无特殊性质</td></tr>
<tr><td>5</td><td rowspan="2">$m \le 2^{18}$</td><td>$p = 0$</td></tr>
<tr><td>6</td><td>无特殊性质</td></tr>
<tr><td>7</td><td rowspan="14">$n \le 2^{18}$</td><td rowspan="2">$m \le 2^{10}$</td><td>$p = 0$</td></tr>
<tr><td>8</td><td>无特殊性质</td></tr>
<tr><td>9</td><td rowspan="12">$m \le 2^{18}$</td><td rowspan="2">$p = 0$</td></tr>
<tr><td>10</td></tr>
<tr><td>11</td><td rowspan="4">$log_2(l)$为整数，$log_2(r+1) = log_2(l) + 1$</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td></tr>
<tr><td>14</td></tr>
<tr><td>15</td><td rowspan="4">数组的每个元素在这 m 次操作过程中均在 32 位有符号非负整数范围内</td></tr>
<tr><td>16</td></tr>
<tr><td>17</td></tr>
<tr><td>18</td></tr>
<tr><td>19</td><td rowspan="2">无特殊性质</td></tr>
<tr><td>20</td></tr>
</table>