

## NOIP 模拟赛

Cmd2001 2019.01.31

(请参赛选手务必仔细阅读本页内容)

## 一. 题目概况:

|         |                        |                          |                                |
|---------|------------------------|--------------------------|--------------------------------|
| 题目名称    | 块                      | 拓展 Xor                   | 弦与切                            |
| 文件名称    | block                  | extxor                   | sinandcos                      |
| 输入文件名   | block.in               | extxor.in                | sinandcos.in                   |
| 输出文件名   | block.out              | extxor.out               | sinandcos.out                  |
| 时间限制    | 2s                     | 1s                       | 2s                             |
| 空间限制    | 256mb                  | 256mb                    | 256mb                          |
| 测试点数目   | 20                     | 20                       | 20                             |
| 单个测试点分值 | 5                      | 5                        | 5                              |
| 题目类型    | 传统型                    | 传统型                      | 传统型                            |
| 是否有 SPJ | 否                      | 否                        | 否                              |
| 编译命令    | g++ block.cpp -o block | g++ extxor.cpp -o extxor | g++ sinandcos.cpp -o sinandcos |

## 二. 测试环境:

## 硬件环境:

*i5 6400 @ 2.70Ghz with 8Gib DDR4 SDRAM*

## 软件环境:

*Lemon 1.20 @ Windows 10 LTSC 2019 X86\_64**MinGW GCC 8.3.0 X86\_64*

本次测试不开启O2优化, 不开启C++11。

## 块 (block)

### 【问题描述】

「いては散り、散っては咲く桜のように 巡り巡る縁に誘われ

(开了又谢、谢了又开的樱花一样，循环的因缘之约)

来世できっとまた、お逢いしましょう

(来生，我们一定会再相逢)」

——《拜啓、御前様》

传说在世界的尽头有一片巨大的樱花林，这里的每一棵樱树都会开出不同颜色的樱花。

而每一棵樱树开出的樱花，一定能构成一个联通的区块。

这里我们定义：两个块相邻当且仅当这两个块有且仅有一个维度的坐标的差值为1。

现在，我们拿到了这片樱花林的地图，上面标注了每个格子樱花的花色。

我们需要一个程序去计算，在这片樱花林里，至少有多少棵樱树。

当然，由于世界内核的崩坏，这个世界的地图不一定是二维的。所以，你的程序需要处理更加复杂的情况。



### 【输入描述】

第一行一个整数 $n$ ，表示地图的维数。

接下来一行 $n$ 个整数 $a_1, a_2, \dots, a_n$ ，表示每一维的大小。

接下来 $\prod_{i=1}^n a_i$ 个整数，按照从第 $n$ 维度到第1维度依次递增循环的顺序给出地图。

即：地图按照以下伪代码给出：

```
for( $i_1$  from 1 to  $a_1$ )
  for( $i_2$  from 1 to  $a_2$ )
    ...
    for( $i_n$  from 1 to  $a_n$ )
      print(map( $i_1, i_2, \dots, i_n$ ))
```

### 【输出描述】

一行一个整数，表示所求答案。

### 【样例输入】

[样例 1]

```
1
20
3 2 3 1 0 4 1 0 2 3 3 4 3 1 3 0 4 4 4 3
```

[样例 2]

```
2
3 3
0 1 1
```

0 1 2

2 1 2

[样例 3]

见下发文件`block\block_ex3.in`

**【 样例输出 】**

[样例 1]

17

[样例 2]

4

[样例 3]

见下发文件`block\block_ex3.ans`

**【 数据范围及提示 】**

对于30%的数据,  $n \leq 2$ ;

对于另外20%的数据,  $n \leq 100'000, \prod_{i=1}^n a_i \leq 1000$ ;

对于100%的数据,  $n \leq 100'000, \prod_{i=1}^n a_i \leq 262'144, 0 \leq \text{地图颜色} < 256$ 。

## 拓展 Xor (extxor)

### 【问题描述】

「世界を忘れたくないならば  
(你不愿忘却这个世界的话)  
君が選んだ哀しい物語を消し  
(我会将你所选择的悲伤故事消去)  
記憶を書き換えたのは  
(改写的记忆便是)  
独りぼっちの君への贈り物  
(我赠与孤身一人的你的礼物)」  
——《Planetia》



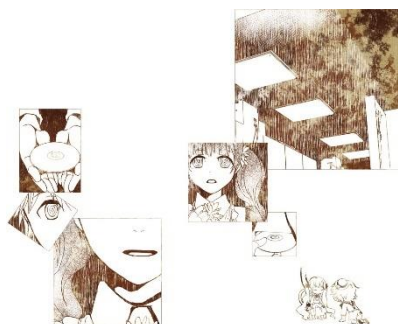
要想改写记忆的话，我们必须知道 Lanjerd 的系统密码。

对此，我们只有这样的线索：

有 $n$ 个巨大的整数和一个进制 $p$ 。

无限使用这些整数，进行 $p$ 进制下的不进位加法。

通过这种组合能拼凑出的最大数字，就是 Lanjerd 的密码了。



### 【输入描述】

第一行两个整数 $n, p$ ，表示数字个数和进制。

接下来 $n$ 行，每行一个十进制整数 $a_i$ ，表示一个给出的数字。

数字不保证没有前导零。

### 【输出描述】

一行一个十进制整数 $ans$ ，表示所求的答案。

你不可以输出前导零。

### 【样例输入】

[样例 1]

5 2

76

121

8

978

910

[样例 2]

5 3

803

87

760

771

9

[样例 3]

见下发文件`extxor\extxor_ex3.in`

**【 样例输出 】**

[样例 1]

1023

[样例 2]

1698

[样例 3]

见下发文件`extxor\extxor_ex3.ans`

**【 数据范围及提示 】**

对于15%的数据,  $n \leq 20, p = 2, a_i \leq 10^{18}$ ;

对于30%的数据,  $n \leq 20, p = 2$ ;

对于50%的数据,  $n \leq 200, p = 2$ ;

对于100%的数据,  $n \leq 200, p < 15, a_i \leq 10^{100}$  。

保证 $p$ 为质数。

## 弦与切 (sinandcos)

### 【问题描述】

「If I'm a sine wave

(如果我是一条正弦波)

Then you can sit on all my

(那么请你坐上)

TANGENTS

(我的切线)」

——《world.execute (me);》

而我们今天的问题，就和这正弦函数有关。

现在我们有一个长度为 $n$ 的行向量(数

组?) $a_1, a_2, \dots, a_n$ 。

我会给出一个区间 $l, r$ ，你需要求出其正弦值的

和。(即： $\sum_{i=l}^r \sin(a_i)$ )。

众所周知， $\sin$ 求导是 $\cos$ (即： $(\sin(x))' = \cos(x)$ )。

所以我还会给出一个区间 $l, r$ ，让你求出其正弦函数的切线的斜率(其实就是余弦函数)的和(即： $\sum_{i=l}^r \cos(a_i)$ )。

当然，只有这两种操作还是过于无聊，所以有时我还会给出一个区间 $l, r$ ，让其中的每个数都加上一个浮点数 $x$ 。

### 【输入描述】

第一行两个整数 $n, m$ ，表示行向量长度和操作次数。

接下来一行 $n$ 个浮点数 $a_1, a_2, \dots, a_n$ ，表示行向量的初始值。

接下来 $m$ 行每行三个整数 $o_i, l_i, r_i$ ，表示第 $i$ 个操作的类型和区间的左右边界。

如果 $o_i = 1$ ，表示你需要要求出区间 $l_i, r_i$ 的正弦函数的切线的斜率(其实就是余弦函数)的和。

如果 $o_i = 2$ ，表示你需要要求出区间 $l_i, r_i$ 的正弦函数的和。

如果 $o_i = 3$ ，那么该行还会跟一个浮点数 $x_i$ ，表示你需要对区间 $l_i, r_i$ 中的每一个数都加上 $x_i$ 。  
(注意这里的区间都是闭区间 的说)

### 【输出描述】

对于每个询问操作，输出一行一个浮点数 $ans_i$ ，表示询问的答案。

输出请采用浮点形式输出，保留三位小数，整数部分为0或小数位数不足的用0补齐，你的答案必须和标准答案完全相同才能得分。

(对于C++选手，可以用 `printf("%.3Lf", (long double)ans);` 进行输出)

### 【样例输入】

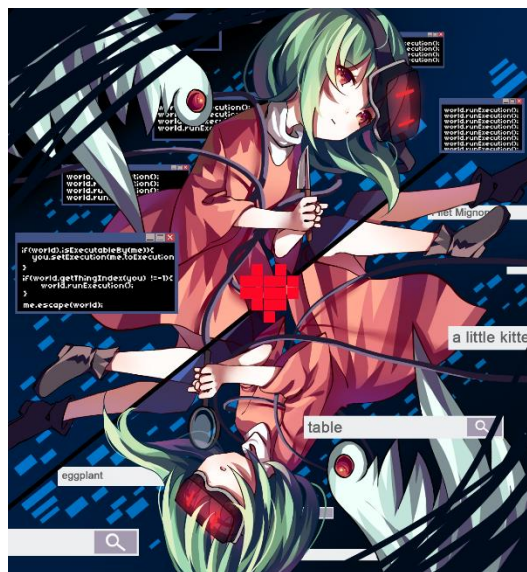
[样例 1]

3 3

0 -2 -6

3 1 2 -7

2 1 2



1 2 3

[样例 2]

7 3

0.123 -19.002 -57.507 34.434 -80.886 -17.115 69.843

2 2 7

3 2 7 16.341

1 1 3

[样例 3]

见下发文件`sinandcos\sinandcos_ex3.in`

### 【 样例输出 】

[样例 1]

-1.069

0.049

[样例 2]

1.519

-0.842

[样例 3]

见下发文件`sinandcos\sinandcos_ex3.ans`

### 【 数据范围及提示 】

对于30%的数据， $n, m \leq 5'000$ ；

对于另外20%的数据，不包含修改操作；

对于100%的数据， $n, m \leq 300'000$ 。

输入中所有的浮点数大小均小于 $10^{18}$ ，且以三位小数形式给出。