2017 年全国青少年信息学奥林匹克竞赛

上海市代表队选拔　第一试

比赛时间：2017 年 4 月 22 日 13:00 ~ 17:30

题目名称	期末考试	相逢是问候	组合数问题
题目类型	传统型	传统型	传统型
目录	`exam`	`verbinden`	`problem`
可执行文件名	`exam`	`verbinden`	`problem`
输入文件名	`exam.in`	`verbinden.in`	`problem.in`
输出文件名	`exam.out`	`verbinden.out`	`problem.out`
每个测试点时限	1.0 秒	2.0 秒	1.0 秒
内存限制	512 MB	512 MB	512 MB
测试点数目	20	20	20
每个测试点分值	5	5	5

提交源程序文件名

对于 C++ 语言	`exam.cpp`	`verbinden.cpp`	`problem.cpp`
对于 C 语言	`exam.c`	`verbinden.c`	`problem.c`
对于 Pascal 语言	`exam.pas`	`verbinden.pas`	`problem.pas`

编译选项

对于 C++ 语言	`-lm -O2`	`-lm -O2`	`-lm`
对于 C 语言	`-lm -O2`	`-lm -O2`	`-lm`
对于 Pascal 语言	`-O2`	`-O2`	

注意事项：

1. 文件名（文件夹名、程序名和输入输出文件名）必须使用英文小写，不得带有空白符。
2. 源代码应当放置在选手目录下各题目的目录下，必须为每道题目单独建立目录。
3. 除非特殊说明，结果比较方式均为忽略行末空格及文末回车的全文比较。
4. C/C++ 函数 main() 的返回值类型必须是 int，程序正常结束时的返回值是 0。
5. 文件读写结束后必须关闭文件。
6. 评测在 NOI Linux 系统下进行，测试数据使用 Linux 换行符 `\n`。

期末考试（exam）

【问题描述】

有 n 位同学，每位同学都参加了全部的 m 门课程的期末考试，都在焦急的等待成绩的公布。

第 i 位同学希望在第 t_i 天或之前得知**所有**课程的成绩。如果在第 t_i 天，有至少一门课程的成绩没有公布，他就会等待最后公布成绩的课程公布成绩，每等待一天就会产生 C 不愉快度。

对于第 i 门课程，按照原本的计划，会在第 b_i 天公布成绩。

有如下两种操作可以调整公布成绩的时间：

1. 将负责课程 X 的部分老师调整到课程 Y，调整之后公布课程 X 成绩的时间推迟一天，公布课程 Y 成绩的时间提前一天；每次操作产生 A 不愉快度。
2. 增加一部分老师负责学科 Z，这将导致学科 Z 的出成绩时间提前一天；每次操作产生 B 不愉快度。

上面两种操作中的参数 X, Y, Z 均可任意指定，每种操作均可以执行多次，每次执行时都可以重新指定参数。

现在希望你通过合理的操作，使得最后总的不愉快度之和最小，输出最小的不愉快度之和即可。

【输入格式】

从文件 $\boldsymbol{exam.in}$ 中读入数据。

第一行三个非负整数 A, B, C ，描述三种不愉快度，详见【问题描述】;

第二行两个正整数 $n, m (1 \le n, m \le 10^5)$ ，分别表示学生的数量和课程的数量；

第三行 n 个正整数 t_i ，表示每个学生希望的公布成绩的时间；

第四行 m 个正整数 b_i ，表示按照原本的计划，每门课程公布成绩的时间。

【输出格式】

输出到文件 $\boldsymbol{exam.out}$ 中。

输出一行一个整数，表示最小的不愉快度之和。

【样例 1 输入】

```
100 100 2
4 5
5 1 2 3
1 1 2 3 3
```

【样例 1 输出】

```
6
```

【样例 1 说明】

由于调整操作产生的不愉快度太大，所以在本例中最好的方案是不进行调整; 全部 5 的门课程中，最慢的在第 3 天出成绩;

同学 1 希望在第 5 天或之前出成绩，所以不会产生不愉快度;

同学 2 希望在第 1 天或之前出成绩，产生的不愉快度为 $(3-1)*2=4$;

同学 3 希望在第 2 天或之前出成绩，产生的不愉快度为 $(3-2)*2=2$;

同学 4 希望在第 3 天或之前出成绩，所以不会产生不愉快度;

不愉快度之和为 $4+2=6$ 。

【样例 2 输入】

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

【样例 2 输出】

```
33
```

【样例 3】

见选手目录下的 ***exam/exam3.in*** 与 ***exam/exam3.ans***。

【数据范围和约定】

测试点	n,m,t_i,b_i	A,B,C
1,2	$1 \le n,m,t_i,b_i \le 2,000$	$A=10^9; B=10^9; 0 \le C \le 10^2$
3,4		$0 \le A,C \le 10^2; B=10^9$
5,6,7,8		$0 \le B \le A \le 10^2; 0 \le C \le 10^2$
9,10,11,12		$0 \le A,B,C \le 10^2$
13,14	$1 \le n,m,t_i,b_i \le 10^5$	$0 \le A,B \le 10^5; C=10^{16}$
15,16,17,18,19,20		$0 \le A,B,C \le 10^5$

相逢是问候（verbinden）

【问题描述】

Informatik verbindet dich und mich.

信息将你我连结。

B 君希望以维护一个长度为 n 的数组，这个数组的下标为从 1 到 n 的正整数。

一共有 m 个操作，可以分为两种：

0 l r 表示将第 l 个到第 r 个数（$a_l, a_{l+1}, \ldots, a_r$）中的每一个数 a_i 替换为 c^{a_i}，即 c 的 a_i 次方，其中 c 是输入的一个常数，也就是执行赋值

$$a_i = c^{a_i}$$

1 l r 求第 l 个到第 r 个数的和，也就是输出：

$$\sum_{i=l}^{r} a_i$$

因为这个结果可能会很大，所以你只需要输出结果 $\bmod p$ 的值即可。

【输入格式】

从文件 ***verbinden.in*** 中读入数据。

第一行有三个整数 n, m, p, c，所有整数含义见问题描述。

接下来一行 n 个整数，表示 a 数组的初始值。

接下来 m 行，每行三个整数，其中第一个整数表示了操作的类型。

如果是 `0` 的话，表示这是一个修改操作，操作的参数为 l, r。

如果是 `1` 的话，表示这是一个询问操作，操作的参数为 l, r。

【输出格式】

输出到文件 ***verbinden.out*** 中。

对于每个询问操作，输出一行，包括一个整数表示答案 $\bmod p$ 的值。

【样例 1 输入】

```
4 4 7 2
1 2 3 4
0 1 4
1 2 4
0 1 4
1 1 3
```

【样例 1 输出】

```
0
3
```

【样例 2 输入】

```
1 40 19910626 2
0
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
```

```
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
0 1 1
1 1 1
```

【样例 2 输出】

```
1
2
4
16
65536
11418102
18325590
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
13700558
```

【样例 3】

见选手目录下的 ***verbinden/verbinden3.in*** 与 ***verbinden/verbinden3.ans***。

【子任务】

- 对于 0% 的测试点，和样例一模一样；
- 对于另外 10% 的测试点，没有修改；
- 对于另外 20% 的测试点，每次修改操作只会修改一个位置（也就是 $l = r$），并且每个位置至多被修改一次；
- 对于另外 10% 的测试点，$p = 2$；
- 对于另外 10% 的测试点，$p = 3$；
- 对于另外 10% 的测试点，$p = 4$；
- 对于另外 20% 的测试点，$n \le 100, m \le 100$；
- 对于 100% 的测试点，$1 \le n \le 50000, 1 \le m \le 50000, 1 \le p \le 100000000, 0 < c < p, 0 \le a_i < p$。

组合数问题（problem）

【问题描述】

组合数 C_n^m 表示的是从 n 个互不相同的物品中选出 m 个物品的方案数。举个例子，从 $(1,2,3)$ 三个物品中选择两个物品可以有 $(1,2),(1,3),(2,3)$ 这三种选择方法。根据组合数的定义，我们可以给出计算组合数 C_n^m 的一般公式：

$$C_n^m = \frac{n!}{m!(n-m)!}$$

其中 $n! = 1 \times 2 \times \cdots \times n$。（特别的，当 $n = 0$ 时，$n! = 1$ ，当 $m > n$ 时，$C_n^m = 0$）

小葱在 NOIP 的时候学习了 C_i^j 和 k 的倍数关系，现在他想更进一步，研究更多关于组合数的性质。小葱发现，C_i^j 是否是 k 的倍数，取决于 $C_i^j \bmod k$ 是否等于 0，这个神奇的性质引发了小葱对 mod 运算（取余数）的兴趣。现在小葱选择了是四个整数 n,p,k,r，小葱现在希望知道

$$\left(\sum_{i=0}^{\infty} C_{nk}^{ik+r}\right) \bmod p$$

即

$$\left(C_{nk}^{r} + C_{nk}^{k+r} + C_{nk}^{2k+r} + \cdots + C_{nk}^{(n-1)k+r} + C_{nk}^{nk+r} + \cdots\right) \bmod p$$

的值。

【输入格式】

从文件 ***problem.in*** 中读入数据。

第一行有四个整数 n,p,k,r，所有整数含义见问题描述。

【输出格式】

输出到文件 ***problem.out*** 中。

一行一个整数代表答案。

【样例 1 输入】

```
2 10007 2 0
```

【样例 1 输出】

```
8
```

【样例 1 说明】

$C_4^0 + C_4^2 + C_4^4 + \cdots = 1 + 6 + 1 = 8$

【样例 2 输入】

```
20 10007 20 0
```

【样例 2 输出】

```
176
```

【子任务】

- 对于 30% 的测试点，$1 \le n, k \le 30$，p 是质数；
- 对于另外 5% 的测试点，$p = 2$；
- 对于另外 5% 的测试点，$k = 1$；
- 对于另外 10% 的测试点，$k = 2$；
- 对于另外 15% 的测试点，$1 \le n \le 10^3, 1 \le k \le 50$，$p$ 是质数；
- 对于另外 15% 的测试点，$1 \le n \times k \le 10^6$，$p$ 是质数；
- 对于另外 10% 的测试点，$1 \le n \le 10^9, 1 \le k \le 50$，$p$ 是质数；
- 对于 100% 的测试点，$1 \le n \le 10^9, 0 \le r < k \le 50, 2 \le p \le 2^{30} - 1$。