# 2019 全国青少年信息学奥林匹克竞赛

# 联合省代表队选拔

## 第一试

**时间：**2019 年 4 月 6 日 08:00 ~ 12:30

| 题目名称 | 异或粽子 | 字符串问题 | 骗分过样例 |
|---|---|---|---|
| 题目类型 | 传统型 | 传统型 | 传统型 |
| 目录 | xor | string | software |
| 可执行文件名 | xor | string | software |
| 输入文件名 | xor.in | string.in | software.in |
| 输出文件名 | xor.out | string.out | software.out |
| 每个测试点时限 | 1.0 秒 | 4.0 秒 | 3.0 秒 |
| 内存限制 | 1 GB | 1 GB | 512 MB |
| 测试点/包数目 | 20 | 10 | 16 |
| 测试点是否等分 | 是 | 是 | 否 |

提交源程序文件名

| | | | |
|---|---|---|---|
| 对于 C++ 语言 | xor.cpp | string.cpp | software.cpp |
| 对于 C 语言 | xor.c | string.c | software.c |
| 对于 Pascal 语言 | xor.pas | string.pas | software.pas |

编译选项

| | |
|---|---|
| 对于 C++ 语言 | -O2 -std=c++11 |
| 对于 C 语言 | -O2 -std=c11 |
| 对于 Pascal 语言 | -O2 |

# 异或粽子（xor）

## 【题目描述】

小粽是一个喜欢吃粽子的好孩子。今天她在家里自己做起了粽子。

小粽面前有 $n$ 种互不相同的粽子馅儿，小粽将它们摆放为了一排，并从左至右编号为 1 到 $n$。第 $i$ 种馅儿具有一个非负整数的属性值 $a_i$。每种馅儿的数量都足够多，即小粽不会因为缺少原料而做不出想要的粽子。小粽准备用这些馅儿来做出 $k$ 个粽子。

小粽的做法是：选两个整数数 $l, r$，满足 $1 \le l \le r \le n$，将编号在 $[l, r]$ 范围内的所有馅儿混合做成一个粽子，所得的粽子的美味度为这些粽子的属性值的异或和。（异或就是我们常说的 xor 运算，即 C/C++ 中的 ^ 运算符或 Pascal 中的 xor 运算符）

小粽想品尝不同口味的粽子，因此它不希望用同样的馅儿的集合做出一个以上的粽子。

小粽希望她做出的所有粽子的美味度之和最大。请你帮她求出这个值吧！

## 【输入格式】

从文件 ***xor.in*** 中读入数据。

第一行两个正整数 $n, k$，表示馅儿的数量，以及小粽打算做出的粽子的数量。

接下来一行为 $n$ 个非负整数，第 $i$ 个数为 $a_i$，表示第 $i$ 个粽子的属性值。

对于所有的输入数据都满足：$1 \le n \le 5 \times 10^5, 1 \le k \le \min\left\{\frac{n(n-1)}{2}, 2 \times 10^5\right\}, 0 \le a_i \le 4,294,967,295$。

## 【输出格式】

输出到文件 ***xor.out*** 中。

输出一行一个整数，表示小粽可以做出的粽子的美味度之和的最大值。

## 【样例 1 输入】

```
3 2
1 2 3
```

## 【样例 1 输出】

```
6
```

## 【样例 1 解释】

小粽可以选取 $[3,3], [1,2]$ 两个区间的馅儿做成粽子，两个粽子的美味度都为 3，和为 6。可以证明不存在比这更优的方案。

**【样例 2】**

见选手目录下的 ***xor/xor2.in*** 与 ***xor/xor2.ans***。

**【子任务】**

| 测试点 | $n$ | $k$ |
|---|---|---|
| 1,2,3,4,5,6,7,8 | $\leq 10^3$ | $\leq 10^3$ |
| 9,10,11,12 | $\leq 5 \times 10^5$ | |
| 13,14,15,16 | $\leq 10^3$ | $\leq 2 \times 10^5$ |
| 17,18,19,20 | $\leq 5 \times 10^5$ | |

# 字符串问题（string）

## 【题目背景】

Yazid 和 Tiffany 喜欢字符串问题。在这里，我们将给你介绍一些关于字符串的基本概念。

对于一个字符串 $S$，我们定义 $|S|$ 表示 $S$ 的长度。

接着，我们定义该串的子串 $S\,(L,R)$ 表示由 $S$ 中从左往右数，第 $L$ 个字符到第 $R$ 个字符依次连接形成的字符串，特别地，如果 $L<1$ 或 $R>|S|$ 或 $L>R$，则 $S\,(L,R)$ 表示空串。

我们说两个字符串相等，当且仅当它们的长度相等，且从左至右各位上的字符依次相同。

我们说一个字符串 $T$ 是 $S$ 的**前缀**，当且仅当 $S\,(1,|T|)=T$。

两个字符串 $S,T$ 相加 $S+T$ 表示的是在 $S$ 后紧挨着写下 $T$ 得到的长度为 $|S|+|T|$ 的字符串。

## 【题目描述】

现有一个字符串 $S$。

Tiffany 将从中划出 $n_a$ 个子串作为 A 类串，第 $i$ 个（$1\le i\le n_a$）为 $A_i=S\,(la_i,ra_i)$。

类似地，Yazid 将划出 $n_b$ 个子串作为 B 类串，第 $i$ 个（$1\le i\le n_b$）为 $B_i=S\,(lb_i,rb_i)$。

现额外给定 $m$ 组支配关系，每组支配关系 $(x,y)$ 描述了第 $x$ 个 A 类串**支配**第 $y$ 个 B 类串。

求一个**长度最大**的目标串 $T$，使得存在一个串 $T$ 的分割 $T=t_1+t_2+\cdots+t_k$（$k\ge 0$）满足：

- 分割中的每个串 $t_i$ 均为 A 类串：即存在一个与其相等的 A 类串，不妨假设其为 $t_i=A_{id_i}$。
- 对于分割中所有相邻的串 $t_i,t_{i+1}$（$1\le i<k$），都有存在一个 $A_{id_i}$ 支配的 B 类串，使得该 B 类串为 $t_{i+1}$ 的前缀。

方便起见，你只需要输出这个最大的长度即可。

特别地，如果存在无限长的目标串（即对于任意一个正整数 $n$，都存在一个满足限制的长度超过 $n$ 的串），请输出 $-1$。

## 【输入格式】

从文件 ***string.in*** 中读入数据。

单个测试点中包含多组数据，输入的第一行包含一个非负整数 $T$ 表示数据组数。接下来依次描述每组数据，对于每组数据：

- 第 1 行一个只包含小写字母的字符串 $S$。
- 第 2 行一个非负整数 $n_a$，表示 A 类串的数目。接下来 $n_a$ 行，每行 2 个用空格隔开的整数。
  - 这部分中第 $i$ 行的两个数分别为 $la_i, ra_i$，描述第 $i$ 个 A 类串。
  - 保证 $1 \le la_i \le ra_i \le |S|$。
- 接下来一行一个非负整数 $n_b$，表示 B 类串的数目。接下来 $n_b$ 行，每行 2 个用空格隔开的整数。
  - 这部分中第 $i$ 行的两个数分别为 $lb_i, rb_i$，描述第 $i$ 个 B 类串。
  - 保证 $1 \le lb_i \le rb_i \le |S|$。
- 接下来一行一个非负整数 $m$，表示支配关系的组数。接下来 $m$ 行，每行 2 个用空格隔开的整数。
  - 这部分中每行的两个整数 $x, y$，描述一对 $(x, y)$ 的支配关系，具体意义见【题目描述】。
  - 保证 $1 \le x \le n_a$，$1 \le y \le n_b$。保证所有支配关系两两不同，即不存在两组支配关系的 $x, y$ 均相同。

## 【输出格式】

输出到文件 ***string.out*** 中。
依次输出每组数据的答案，对于每组数据：

- 一行一个整数表示最大串长。特别地，如果满足限制的串可以是无限长的，则请输出 $-1$。

## 【样例 1 输入】

```
3
abaaaba
2
4 7
1 3
1
3 4
1
2 1
abaaaba
2
```

```
4 7
1 3
1
7 7
1
2 1
abbaabbaab
4
1 5
4 7
6 9
8 10
3
1 6
10 10
4 6
5
1 2
1 3
2 1
3 3
4 1
```

【样例 1 输出】

```
7
-1
13
```

【样例 1 解释】

对于第 1 组数据，A 类串有 aaba 与 aba，B 类串有 aa，且 $A_2$ 支配 $B_1$。我们可以找到串 abaaaba，它可以拆分成 $A_2 + A_1$，且 $A_1$ 包含由 $A_2$ 所支配的 $B_1$ 作为前缀。可以证明不存在长度更大的满足限制的串。

对于第 2 组数据，与第 1 组数据唯一不同的是，唯一的 B 类串为 a。容易证明存在无限长的满足限制的串。

对于第 3 组数据，容易证明 abbaabbaaaabb 是最长的满足限制的串。

**【样例 2】**

见选手目录下的 ***string/string2.in*** 与 ***string/string2.ans***。

**【样例 3】**

见选手目录下的 ***string/string3.in*** 与 ***string/string3.ans***。

**【样例 3 解释】**

这个测试点满足【子任务】中提到的 $|A_i| \geq |B_j|$ 的限制。

**【子任务】**

<table>
<tr><th>$n_a$</th><th>$n_b$</th><th>$|S|$</th><th>测试点</th><th>$m$</th><th>$|A_i| \geq |B_j|$</th><th>其他限制</th></tr>
<tr><td>$\leq 100$</td><td>$\leq 100$</td><td>$\leq 10^4$</td><td>1</td><td>$\leq 10^4$</td><td rowspan="6">保证</td><td>保证所有 $|A_i|, |B_j| \leq 100$</td></tr>
<tr><td>$\leq 1000$</td><td>$\leq 1000$</td><td rowspan="5">$\leq 2 \times 10^5$</td><td>2~3</td><td>$\leq 2 \times 10^5$</td><td rowspan="2">无</td></tr>
<tr><td>$= 1$</td><td rowspan="4">$\leq 2 \times 10^5$</td><td>4</td><td>$= n_b$</td></tr>
<tr><td rowspan="3">$\leq 2 \times 10^5$</td><td>5~6</td><td rowspan="3">$\leq 2 \times 10^5$</td><td>保证所有 $ra_i + 1 = la_{i+1}$</td></tr>
<tr><td>7~8</td><td rowspan="2">无</td></tr>
<tr><td>9~10</td><td>不保证</td></tr>
</table>

为了方便你的阅读，我们把**测试点编号**放在了表格的中间，请你注意这一点。

表格中的 $|A_i| \geq |B_j|$ 指的是**任意** B 类串的长度不超过**任意** A 类串的长度。

对于所有测试点，保证：$T \leq 100$，且对于测试点内所有数据，$|S|, n_a, n_b, m$ 的**总和**分别不会超过**该测试点中对应**的**单组数据的限制**的 10 倍。比如，对于第 1 组测试点，就有 $\sum n_a \leq 10 \times 100 = 1000$ 等。特别地，我们规定对于测试点 4，有 $T \leq 10$。

对于所有测试点中的每一组数据，保证：$1 \leq |S| \leq 2 \times 10^5$，$n_a, n_b \leq 2 \times 10^5$，$m \leq 2 \times 10^5$

**【提示】**

十二省联考命题组温馨提醒您：

**数据千万条，清空第一条。**

**多测不清空，爆零两行泪。**

# 骗分过样例（software）

这是一道**传统题**。

**【题目背景】**

“我的程序需要完成什么功能呀？……”

“我也不知道……”

“啊？那我怎么写呀……”

“已经有人给你写好测试了，只要你通过这些测试就可以了……”

“啊？……”

“所有的**最终测试数据**都在题目目录下，**请做好备份，避免误删！**”

“这……”

“哦，我还可以把输入格式告诉你……不过都有完整的数据了，知道输入格式可能也没太大用处吧……”

**【输入格式】**

从文件 ***software.in*** 中读入数据。

第一行输入一个字符串，表示需要运行的软件功能编号。两个编号越相似，说明对应的两个功能的算法越接近。

接下来根据功能的不同，可能有任意长度的输入，详见每个功能的文档。

**【输出格式】**

输出到文件 ***software.out*** 中。

详见每个功能的文档。

**【子任务】**

“‘每个功能的文档’在哪里呀？”

“我也没有，就像我没有题目描述一样……”

“好吧……那我是不是打表就可以了呀……”

“**代码长度限制是 102400 字节**（100KB），超过此限制会导致该题获得零分，直接打肯定是不行的！不过，需要的话倒是可以稍微打一些小的表……”

“唔……”

“另外，我们会给你的程序对于每个测试点分别评分，求和后得到总分。按照传统的规矩，每个测试点正确得满分，错误得 0 分。**每个测试点的分值不全相同，测试点的分值、顺序与难度没有必然联系**。”

| 测试点 | 功能编号 | 分值 |
|---|---|---|
| 1 | 1_998244353 | 4 |
| 2 | | 4 |
| 3 | | 4 |
| 4 | 1? | 7 |
| 5 | 1?+ | 9 |
| 6 | 1wa_998244353 | 6 |
| 7 | | 7 |
| 8 | 2p | 4 |
| 9 | | 6 |
| 10 | | 8 |
| 11 | 2u | 5 |
| 12 | | 6 |
| 13 | | 9 |
| 14 | 2g | 5 |
| 15 | | 7 |
| 16 | 2g? | 9 |

**【提示】**

在你使用 C/C++ 的 `int` 类型时，如果发生了溢出，比较可能的情况是按照模 $2^{32}$ 同余的前提下，在 `int` 范围内取一个合理的值。例如在计算 $2147483647 + 2$ 时，较有可能会得到 $-2147483647$。

然而，C/C++ 标准将这种情况归类为“未定义行为”。当你的程序试图计算会溢出的 `int` 运算时，除了上述结果外，编译器还可能会让你的程序在此时计算出错误结果、死循环、运行错误等，这也是符合 C/C++ 标准的。

如果你的程序希望利用 `int` 的自然溢出的特性，请转换为 `unsigned` 类型运算。例如将 `a + b` 改写为 `(int) ((unsigned) a + (unsigned) b)`，以避免出现不预期的错误。