

2024 年全国青少年信息学奥林匹克联赛

时间：2024年11月19日 7:50-12:20

题目名称	3 idots	冒泡排序	多重集	简单题
题目类型	传统型	传统型	传统型	传统型
目录	string	sort	set	easy
可执行文件名	string	sort	set	easy
输入文件名	string.in	sort.in	set.in	easy.in
输出文件名	string.out	sort.out	set.out	easy.out
每个测试点时限	0.5 秒	1.0 秒	3.0 秒	2.0 秒
内存限制	256 MB	256 MB	512 MB	512 MB
测试点数目	2	10	20	10
测试点是否等分	否	是	是	是

提交源程序文件名

对于 C++ 语言	string.cpp	sort.cpp	set.cpp	easy.cpp
-----------	------------	----------	---------	----------

编译选项

对于 C++ 语言	-lm -O2 -std=c++14
-----------	--------------------

注意事项与提醒（请选手务必仔细阅读）

0. 选手可以直接提交源程序至 becoder.com.cn 上的对应比赛。
1. 选手提交的源程序必须存放在已建立好的，且带有**样例文件**和**下发文件**的文件夹中，文件夹名称与对应试题英文名一致。
2. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
3. C++ 中函数 `main()` 的返回值类型必须是 `int`，值必须为 0。
4. **对于因未遵守以上规则对成绩造成的影响，相关申诉不予受理。**
5. 若无特殊说明，结果比较方式为**忽略行末空格、文末回车后的全文比较**。
6. 程序可使用的栈空间大小与该题内存空间限制一致。
7. 在终端中执行命令 `ulimit -s unlimited` 可将当前终端下的栈空间限制放大，但你使用的栈空间大小不应超过题目限制。
8. 每道题目所提交的**代码文件大小限制为 100KB**。
9. 若无特殊说明，输入文件与输出文件中同一行的相邻整数均使用一个空格分隔。
10. 输入文件中可能存在行末空格，请选手使用更完善的读入方式（例如 `scanf` 函数）避免出错。

11. 直接复制 PDF 题面中的多行样例，数据将带有行号，建议选手直接使用对应目录下的样例文件进行测试。

12. 使用 `std::deque` 等 STL 容器时，请注意其内存空间消耗。

13. 请务必使用题面中规定的的编译参数，保证你的程序在本机能够通过编译。此外不允许在程序中手动开启其他编译选项，一经发现，本题成绩以 0 分处理。

3 idots (string)

【题目描述】

We fretted about the morrow, He simply reveled in today。

给定一个字符串 S ，先将字符串 S 复制一次，得到字符串 T ，然后在 T 中插入一个字符，得到字符串 U 。

给出字符串 U ，重新构造出字符串 S 。

所有字符串只包含大写英文字母。

【输入格式】

从文件 *string.in* 中读入数据。

第一行一个整数 N ，表示字符串 U 的长度。

第二行一个长度为 N 的字符串，表示字符串 U 。

【输出格式】

输出到文件 *string.out* 中。

一行一个字符串，表示字符串 S 。如果字符串无法按照上述方法构造出来，输出'NOT POSSIBLE'；如果字符串 S 不唯一，输出'NOT UNIQUE'。

【样例 1 输入】

```
1 7
2 ABXCABC
```

【样例 1 输出】

```
1 ABC
```

【样例 2 输入】

```
1 6
2 ABCDEF
```

【样例 2 输出】

```
1 NOT POSSIBLE
```

【样例 3 输入】

```
1 9
2 ABABABABA
```

【样例 2 输出】

```
1 NOT UNIQUE
```

【数据范围】

本题采用捆绑测试。

子任务一 (35 分): $2 \leq N \leq 2001$ 。

子任务二 (65 分): $2 \leq N \leq 2000001$ 。

冒泡排序 (sort)

【题目描述】

下面是一段实现冒泡排序算法的 C++ 代码：

```
1 for (int i=1; i<n; i++)  
2     for (int j=1; j<=n-i; j++)  
3         if (a[j]>a[j+1])  
4             swap(a[j], a[j+1]);
```

其中待排序的 a 数组是一个 $1 \dots n$ 的排列，「swap」函数将交换数组中对应位置的值。

对于给定的数组 a 以及给定的非负整数 k ，使用这段代码执行了正好 k 次「swap」操作之后数组 a 中元素的值会是什么样的呢？

【输入格式】

从文件 *sort.in* 中读入数据。

第 1 行包含空格隔开的一个正整数 n 和一个非负整数 k ；

第 2 行包含 n 个空格隔开的互不相同的正整数，表示初始时 a 数组中的排列。

【输出格式】

输出到文件 *sort.out* 中。

若在执行完整个代码之后执行「swap」的次数仍不够 k ，那么输出一个字符串 'Impossible!' (不含引号)，否则按顺序输出执行「swap」操作 k 次之后数组 a 的每一个元素，用空格隔开。

【样例 1 输入】

```
1 1 1  
2 1
```

【样例 1 输出】

```
1 Impossible!
```

【样例 2 输入】

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

【样例 2 输出】

1 3 4 2 1 5

【样例 3】

见附加文件中的 sort/ex_sort3.in 与 sort/ex_sort3.ans。

【数据范围】

对于全部数据, $n \leq 10^6, k \leq 10^{12}$ 。

对于 10% 的数据, $n \leq 10, k \leq 10^{12}$ 。

对于 20% 的数据, $n \leq 5000, k \leq 10^{12}$ 。

对于 30% 的数据, $n \leq 20000, k \leq 10^{12}$ 。

对于 40% 的数据, $n \leq 10^6, k \leq 10^{12}$ 。

对于 60% 的数据, $n \leq 10^6, k \leq 2 \times 10^6$ 。

对于 80% 的数据, $n \leq 10^5, k \leq 10^{12}$ 。

对于 100% 的数据, $n \leq 10^6, k \leq 10^{12}$ 。

多重集 (set)

【题目描述】

有两个初始为空的多重集 A, B ，其中每个元素都有两个属性 a, b 。

有 Q 次修改操作，每次修改会对两个多重集中的一个进行一次插入或者删除。

每次操作完成后，你需要求 $\max(a_x + a_y, b_x + b_y)$ 的最小值，其中 $x \in A, y \in B$ 。

【输入格式】

从文件 `set.in` 中读入数据。

第一行一个整数 Q ，表示修改次数。

接下来 Q 行，每行包含四个整数 opt, d, a, b ，其中 $opt = 0/1$ 分别表示删除/插入一个元素， $d = 0/1$ 分别表示 A 集合/ B 集合， a, b 描述这个元素的两个属性。

【输出格式】

输出到文件 `set.out` 中。

对于每个操作，你都需要输出操作后的 $\max(a_x + a_y, b_x + b_y)$ 的最小值。特别地，如果 A 集合或者 B 集合为空，那么输出 -1 。

【样例 1 输入】

```
1 6
2 1 0 5 10
3 1 1 100 23
4 1 1 1 45
5 1 1 22 33
6 1 0 2 3
7 0 1 22 33
```

【样例 1 输出】

```
1 -1
2 105
3 55
4 43
5 36
6 48
```

【样例 1 解释】

第一次操作, $A = \{\{5, 10\}\}, B = \emptyset$, 所以输出 -1 ;

第二次操作, $A = \{\{5, 10\}\}, B = \{\{100, 23\}\}$, 此时方案是唯一的, 所以输出 $\max\{5 + 100, 10 + 23\} = 105$;

第三次操作, $A = \{\{5, 10\}\}, B = \{\{100, 23\}, \{1, 45\}\}$, 此时选择 $x = \{5, 10\}, y = \{1, 45\}$, 所以输出 55 ;

第四次操作, $A = \{\{5, 10\}\}, B = \{\{100, 23\}, \{1, 45\}, \{22, 33\}\}$, 此时选择 $x = \{5, 10\}, y = \{22, 33\}$, 所以输出 43 ;

第五次操作, $A = \{\{5, 10\}, \{2, 3\}\}, B = \{\{100, 23\}, \{1, 45\}, \{22, 33\}\}$, 此时选择 $x = \{2, 3\}, y = \{22, 33\}$, 所以输出 36 ;

第六次操作, $A = \{\{5, 10\}, \{2, 3\}\}, B = \{\{100, 23\}, \{1, 45\}\}$, 此时选择 $x = \{2, 3\}, y = \{1, 45\}$, 所以输出 48 。

【样例 2 输入】

```
1 4
2 1 1 1000000000 1
3 1 0 2 1000000000
4 1 0 500 123456780
5 1 1000000000 1
```

【样例 2 输出】

```
1 -1
2 10000000002
3 10000000002
4 -1
```

【样例 3】

见附加文件中的 `set/ex_set3.in` 与 `set/ex_set3.ans`。

【样例 4】

见附加文件中的 `set/ex_set4.in` 与 `set/ex_set4.ans`。

【数据范围】

对于 10% 的数据, $Q \leq 600$ 。

对于 20% 的数据, $Q \leq 5000$ 。

对于 40% 的数据, $Q \leq 200000$ 。

对于另外 20% 的数据, 没有删除操作。

对于 100% 的数据, $1 \leq Q \leq 10^6$, $1 \leq a, b \leq 10^9$ 。

简单题 (easy)

【题目描述】

给出 n 个数字，每次询问一个区间 $[l, r]$ ，对这个区间内部的数进行操作，每次可以合并相邻两个数 x, y ，即把 x, y 删除后，在原来的位置上留下 $x + 2y$ ，最后只剩下一个数，问这个数的最大值。

答案对 $10^9 + 7$ 取模。
每次询问独立。

【输入格式】

从文件 *easy.in* 中读入数据。
第一行给出两个整数 n, q ，分别表示序列长度和询问个数。
第二行给出 n 个整数 $a_1, a_2 \dots a_n$
接下来的 q 行给出两个整数 l_i, r_i ，表示一个为 $[l_i, r_i]$ 的询问。

【输出格式】

输出到文件 *easy.out* 中。
 q 行每行一个整数，分别表示第 i 个询问的答案。

【样例 1 输入】

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

【样例 1 输出】

```
1 17
2 5
3 8
```

【样例 2 输入】

```
1 20 20
2 -1 0 1 1 -1 1 1 -1 1 -1 -1 1 0 1 -1 0 1 1 1 -1
3 7 12
4 7 16
```

```
5 17 17
6 3 14
7 8 12
8 3 16
9 6 20
10 4 12
11 16 19
12 7 13
13 19 20
14 8 17
15 16 19
16 8 20
17 20 20
18 10 10
19 12 13
20 11 17
21 12 13
22 13 19
```

【样例 2 输出】

```
1 11
2 137
3 1
4 2231
5 5
6 2229
7 14101
8 91
9 14
10 11
11 1000000006
12 453
13 14
14 3523
15 1000000006
16 1000000006
```

17 1
18 57
19 1
20 110

【数据范围】

对于 20% 的数据, $n \leq 10, q = 1$ 。

对于 30% 的数据, $n \leq 100000, r - l \leq 10$ 。

对于 50% 的数据, $n \leq 100000, r - l \leq 50$ 。

对于这 50% 的数据, $|a_i| \leq 1$ 。

对于另外 20% 的数据, $q = 1$ 。

对于 100% 的数据, $1 \leq n, q \leq 10^5, -10^9 \leq a_i \leq 10^9, 1 \leq l_i \leq r_i \leq n$ 。