# 范围查询(Range)

**Descriptioin**

Let S be a set of n integral points on the x-axis. For each given interval [a, b], you are asked to count the points lying inside.

**Input**

The first line contains two integers: n (size of S) and m (the number of queries).

The second line enumerates all the n points in S.

Each of the following m lines consists of two integers a and b and defines an query interval [a, b].

**Output**

The number of points in S lying inside each of the m query intervals.

**Example**

Input

5 2

1 3 7 9 11

4 6

7 12

Output

0

3

**Restrictions**

0 <= n, m <= 5 \* 10^5

For each query interval [a, b], it is guaranteed that a <= b.

Points in S are distinct from each other.

Coordinates of each point as well as the query interval boundaries a and b are non-negative integers not greater than 10^7.

Time: 2 sec

Memory: 256 MB

**描述**

数轴上有n个点，对于任一闭区间 [a, b]，试计算落在其内的点数。

**输入**

第一行包括两个整数：点的总数n，查询的次数m。

第二行包含n个数，为各个点的坐标。

以下m行，各包含两个整数：查询区间的左、右边界a和b。

**输出**

对每次查询，输出落在闭区间[a, b]内点的个数。

**样例**

见英文题面

**限制**

0 ≤ n, m ≤ 5×105

对于次查询的区间[a, b]，都有a ≤ b

各点的坐标互异

各点的坐标、查询区间的边界a、b，均为不超过10^7的非负整数

时间：2 sec

内存：256 MB

# 祖玛(Zuma)

**Description**

Let's play the game Zuma!

There are a sequence of beads on a track at the right beginning. All the beads are colored but no three adjacent ones are allowed to be with a same color. You can then insert beads one by one into the sequence. Once three (or more) beads with a same color become adjacent due to an insertion, they will vanish immediately.



Note that it is possible for such a case to happen for more than once for a single insertion. You can't insert the next bead until all the eliminations have been done.

Given both the initial sequence and the insertion series, you are now asked by the fans to provide a playback tool for replaying their games. In other words, the sequence of beads after all possible eliminations as a result of each insertion should be calculated.

**Input**

The first line gives the initial bead sequence. Namely, it is a string of capital letters from 'A' to 'Z', where different letters correspond to beads with different colors.

The second line just consists of a single interger n, i.e., the number of insertions.

The following n lines tell all the insertions in turn. Each contains an integer k and a capital letter Σ, giving the rank and the color of the next bead to be inserted respectively. Specifically, k ranges from 0 to m when there are currently m beads on the track.

**Output**

n lines of capital letters, i.e., the evolutionary history of the bead sequence.

Specially, "-" stands for an empty sequence.

**Example**

Input

ACCBA

5

1 B

0 A

2 B

4 C

0 A

Output

ABCCBA

AABCCBA

AABBCCBA

-

A

**Restrictions**

0 <= n <= 10^4

0 <= length of the initial sequence <= 10^4

Time: 2 sec

Memory: 256 MB

**Hints**

List

**描述**

祖玛是一款曾经风靡全球的游戏，其玩法是：在一条轨道上初始排列着若干个彩色珠子，其中任意三个相邻的珠子不会完全同色。此后，你可以发射珠子到轨 道上并加入原有序列中。一旦有三个或更多同色的珠子变成相邻，它们就会立即消失。这类消除现象可能会连锁式发生，其间你将暂时不能发射珠子。

开发商最近准备为玩家写一个游戏过程的回放工具。他们已经在游戏内完成了过程记录的功能，而回放功能的实现则委托你来完成。

游戏过程的记录中，首先是轨道上初始的珠子序列，然后是玩家接下来所做的一系列操作。你的任务是，在各次操作之后及时计算出新的珠子序列。

**输入**

第一行是一个由大写字母'A'~'Z'组成的字符串，表示轨道上初始的珠子序列，不同的字母表示不同的颜色。

第二行是一个数字n，表示整个回放过程共有n次操作。

接下来的n行依次对应于各次操作。每次操作由一个数字k和一个大写字母Σ描述，以空格分隔。其中，Σ为新珠子的颜色。若插入前共有m颗珠子，则k ∈ [0, m]表示新珠子嵌入之后（尚未发生消除之前）在轨道上的位序。

**输出**

输出共n行，依次给出各次操作（及可能随即发生的消除现象）之后轨道上的珠子序列。

如果轨道上已没有珠子，则以“-”表示。

**样例**

见英文题面

**限制**

0 ≤ n ≤ 10^4

0 ≤ 初始珠子数量 ≤ 10^4

时间：2 sec

内存：256 MB

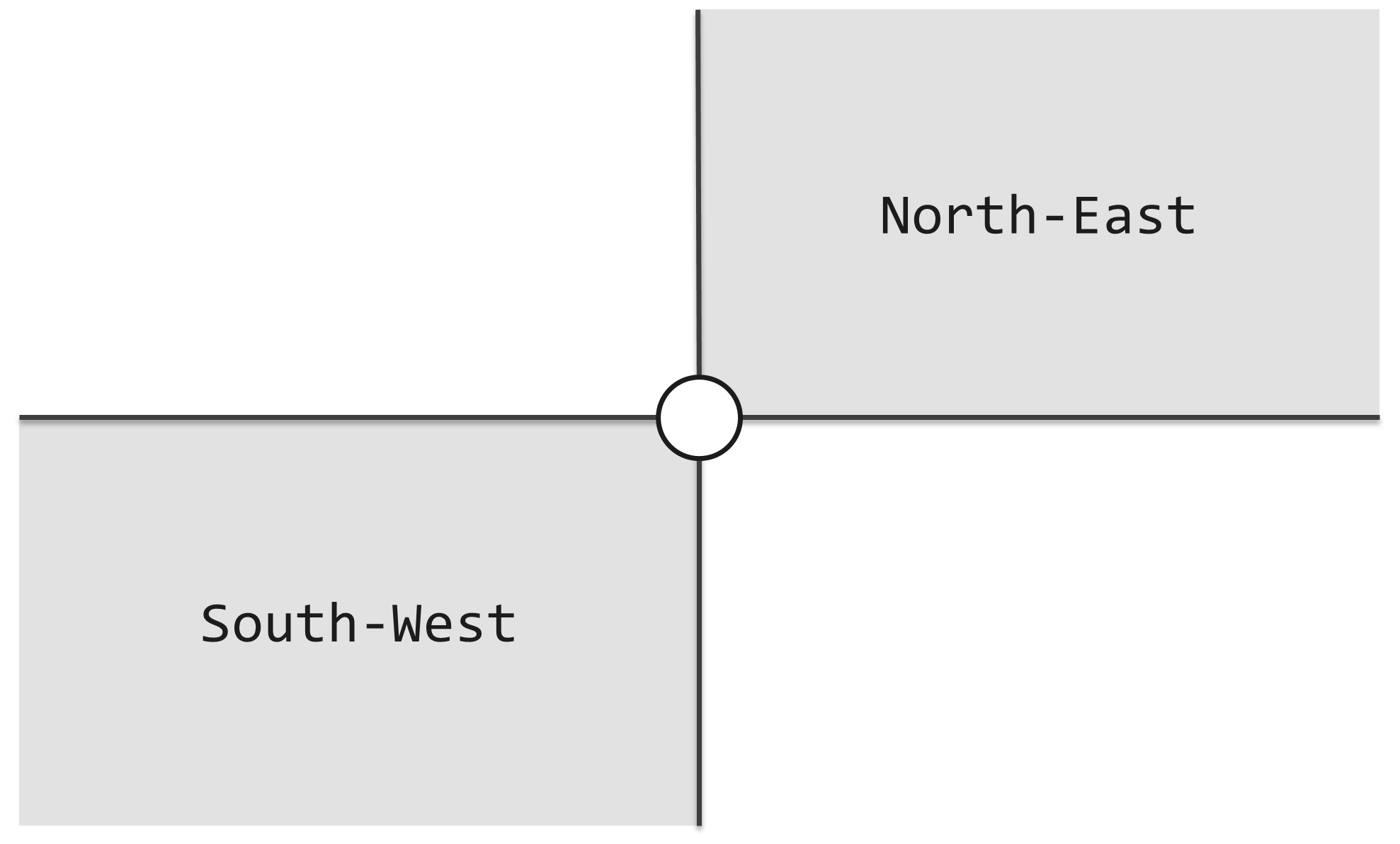
**提示**

列表

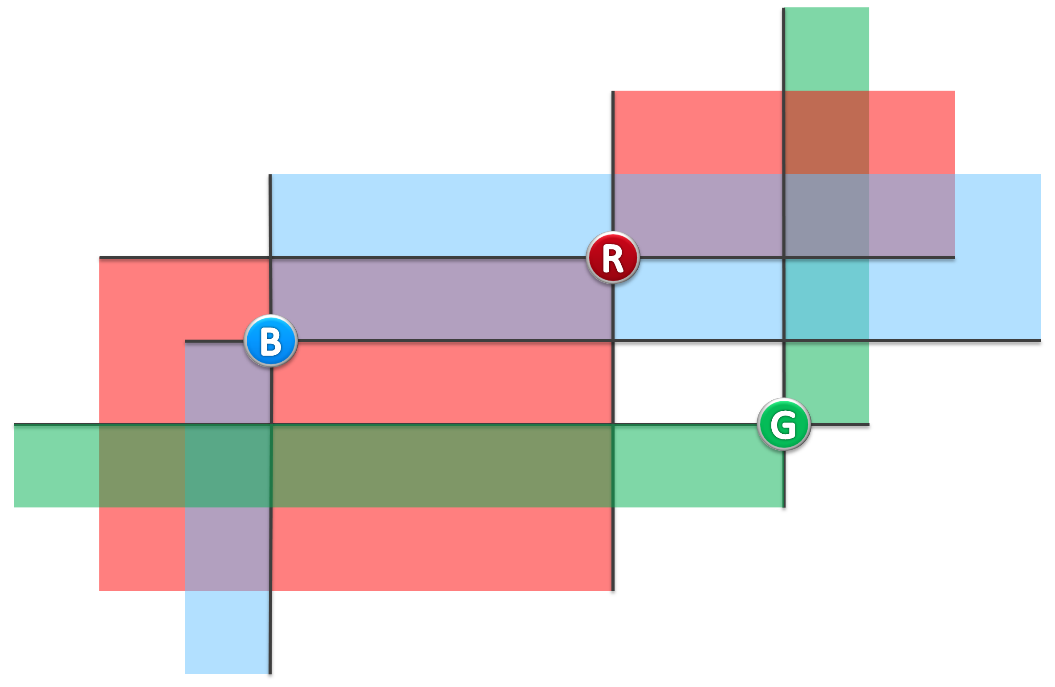
# 灯塔(LightHouse)

**Description**

As shown in the following figure, If another lighthouse is in gray area, they can beacon each other.



For example, in following figure, (B, R) is a pair of lighthouse which can beacon each other, while (B, G), (R, G) are NOT.



**Input**

1st line: N

2nd ~ (N + 1)th line: each line is X Y, means a lighthouse is on the point (X, Y).

**Output**

How many pairs of lighthourses can beacon each other

( For every lighthouses, X coordinates won't be the same , Y coordinates won't be the same )

**Example**

Input

3

2 2

4 3

5 1

Output

1

**Restrictions**

For 90% test cases: 1 <= n <= 3 \* 105

For 95% test cases: 1 <= n <= 106

For all test cases: 1 <= n <= 4 \* 106

For every lighthouses, X coordinates won't be the same , Y coordinates won't be the same.

1 <= x, y <= 10^8

Time: 2 sec

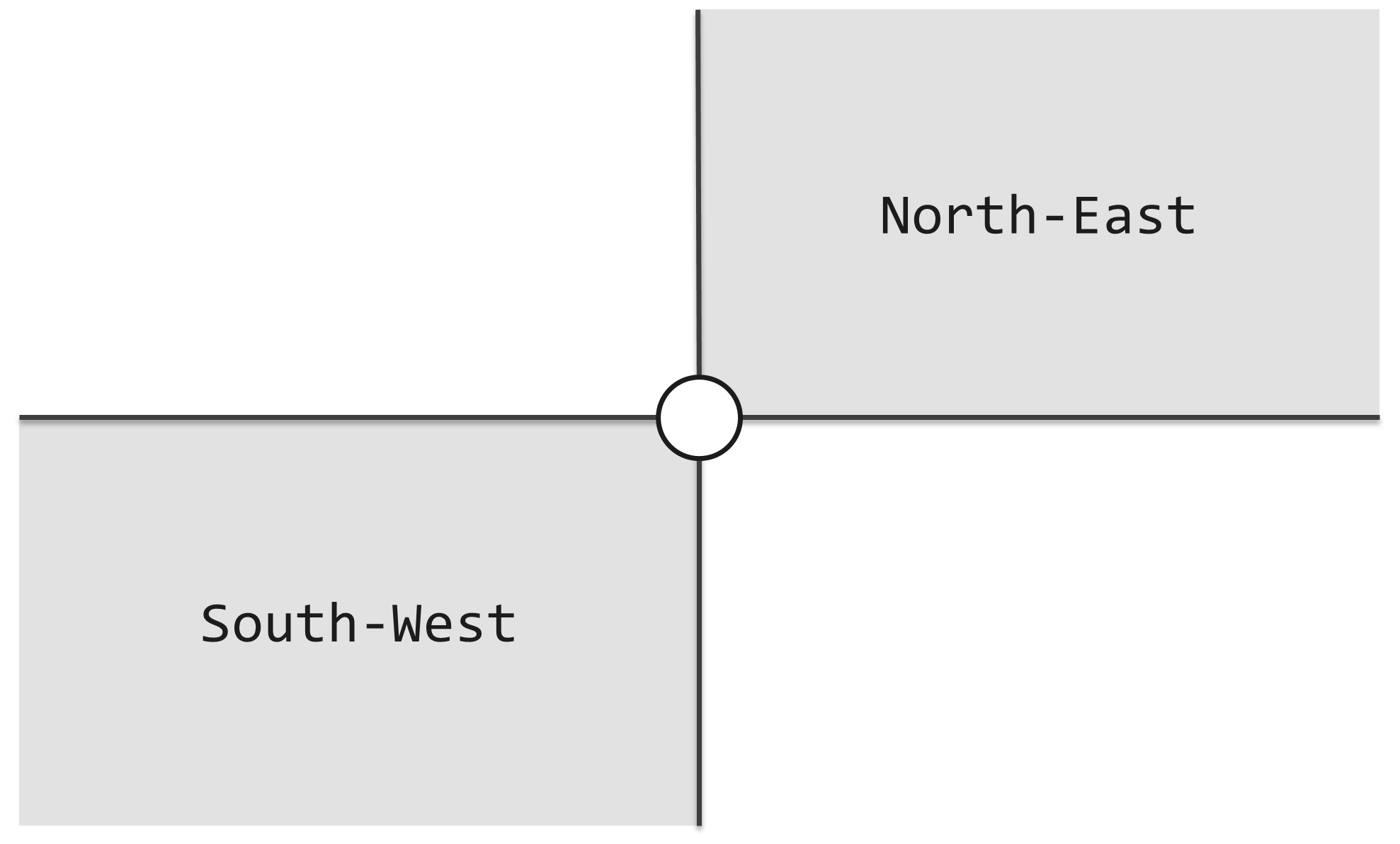
Memory: 256 MB

**Hints**

The range of **int** is usually [-231, 231 - 1], it may be too small.

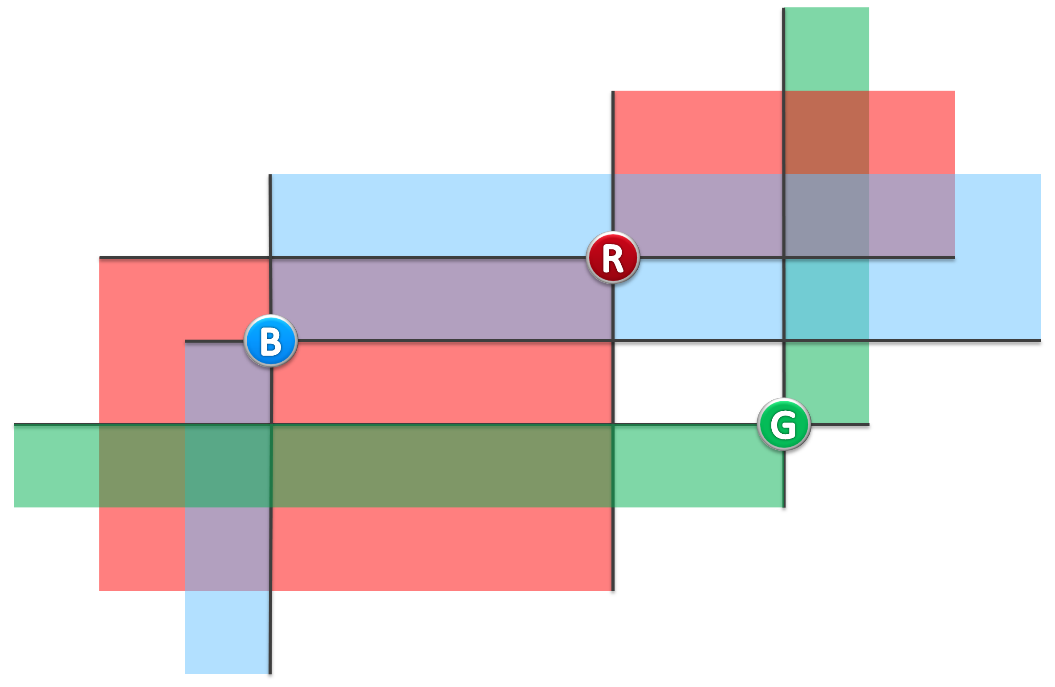
**描述**

海上有许多灯塔，为过路船只照明。



（图一）

如图一所示，每个灯塔都配有一盏探照灯，照亮其东北、西南两个对顶的直角区域。探照灯的功率之大，足以覆盖任何距离。灯塔本身是如此之小，可以假定它们不会彼此遮挡。



（图二）

若灯塔A、B均在对方的照亮范围内，则称它们能够照亮彼此。比如在图二的实例中，蓝、红灯塔可照亮彼此，蓝、绿灯塔则不是，红、绿灯塔也不是。

现在，对于任何一组给定的灯塔，请计算出其中有多少对灯塔能够照亮彼此。

**输入**

共n+1行。

第1行为1个整数n，表示灯塔的总数。

第2到n+1行每行包含2个整数x, y，分别表示各灯塔的横、纵坐标。

**输出**

1个整数，表示可照亮彼此的灯塔对的数量。

**样例**

见英文题面

**限制**

对于90%的测例：1 ≤ n ≤ 3×105

对于95%的测例：1 ≤ n ≤ 106

全部测例：1 ≤ n ≤ 4×106

灯塔的坐标x, y是整数，且不同灯塔的x, y坐标均互异

1 ≤ x, y ≤ 10^8

时间：2 sec

内存：256 MB

**提示**

注意机器中整型变量的范围，C/C++中的int类型通常被编译成32位整数，其范围为[-231, 231 - 1]，不一定足够容纳本题的输出。

# THU2015 spring 1-1 Team

### ****描述****

　　教练员A、B和C将要从编号为1到n的队员中挑选自己的队员。为公平起见，每个教练都根据自己的喜好程度将队员排序；你负责根据以下规则为他们分配队员。

　　你拿到的数据是a、b、c三个数组，表示三个教练对队员的喜好程度排序，每个数组都是数字1到n的一个排列，**下标越小**表示教练越喜欢该队员。你的分组规则是，从还未被分配的队员中找一个教练A最喜欢的队员分到A组；然后，在未分配的队员中分配教练B最喜欢的队员到B组；然后是教练C；再是教练A、B......依次类推直到所有队员分配完毕。

　　现在队员k希望知道自己被分配给哪位教练，请你来告诉他。

### ****输入****

　　共5行。

　　第1行包含一个整数n。

　　第2至4行依次是数组a、b和c，每行都是整数[1, n]的一个排列。

　　第5行包含一个整数k。

### ****输出****

　　仅一个字符，A、B或C，表示队员k被分配给哪位教练。

#### ****输入样例1****

3

1 2 3

1 2 3

1 2 3

3

#### ****输出样例1****

C

#### ****输入样例2****

5

1 2 3 4 5

1 3 5 4 2

5 4 3 2 1

4

#### ****输出样例2****

B

### ****限制****

1 <= n <= 500,000

1 <= k <= n

时间：1 sec

空间：256 MB

### ****提示****

#### ****一级提示****

　　大体上，1 sec内，O(n)的算法可以通过n = 10,000,000规模的数据，O(nlogn)通过500,000规模，O(n^2)通过5,000规模。

　　本题等一些复杂度是O(n)的题目受限于scanf(“%d”, ...)的读入速度，但又不希望通过读取二进制文件等不直观的方式增加同学们的负担。

# THU2015 spring 1-2 Company

### ****描述****

　　公司有n个员工，编号1 ~ n。员工数量众多，需要你为他们编写一个管理系统。

　　员工上班时都要登录管理系统登记一个code，离开要从管理系统上注销，员工也可以随时更新自己的code。到了下班时间，所有员工都会自动注销。公司管理人员随时都可能想要知道有多少员工上班，以及任一员工登记的code。

### ****输入****

　　第一行两个整数n、m。接下来m行，每行都是以下内容之一：

I a c //Log In：员工a登录，code为c。若a已登录，则将code更新为c

O a //Log Out：编号为a的员工注销。若a未登录，则注销无效

C //Close：到下班时间，所有员工注销

N //Number：询问有多少员工正在上班

Q a //Query：询问员工a的code（若未登录或已注销，视为-1）

### ****输出****

　　一个整数，是所有询问（N、Q）的答案之和（int表示，自动溢出，不用管溢出部分）

#### ****输入样例****

10 8

I 1 2

I 1 4

Q 1

C

I 2 4

O 2

Q 2

N

#### ****输出样例****

3

//“Q 1”答案是4，“Q 2”答案是-1，“N”答案是0，所有答案之和为3

### ****限制****

　　n <= 10,000,000

　　m <= 1,000,000

　　1<=a<=n

　　code为[0, 2^31)的整数

　　空间限制：256 MB

　　时间限制：2 sec

### ****提示****

#### ****一级提示****

　　测试数据中，每种操作都可能频繁出现，或很少出现。因此，你的算法需要应对所有可能的情况。

===\*\*一级提示\*\*===

　　Bitmap / 常数时间初始化的数组

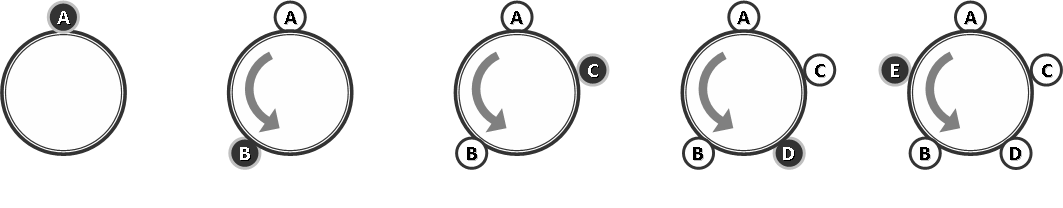
　　Bitmap: 习题解析[2-34]，实例参考[2-35]，[2-36]

# THU2015 spring 1-3 Interview

### ****描述****

　　某公司在对应聘者做过一轮笔试之后，从中选出n 人继续进行面试，每位应聘者被分配了一个整数ID。

　　为公平起见，组织者决定利用会议室外的圆桌，按以下方法“随机”确定面试顺序：第一个到达的应聘者在圆桌周围任意选择一个位置坐下；此后到达的 每位应聘者都从前一应聘者出发，沿逆时针方向围圆桌走过m 人（前一应聘者算作走过的第1 人，同一人可能经过多次），并紧邻第m 人右侧就座；所有应聘者到齐后，从最后到达者出发，绕圆桌以顺时针方向为序进行面试。



　　这里假定应聘者到达的时刻互异，且相对的就坐位置确定后，左、右两人之间总能插入一把椅子。

　　试编写一个程序，确定面试顺序。

### ****输入****

　　共2行。

　　第1行包含两个整数， n和m。

　　第2行包含n个整数，表示先后到达的n个应聘者的ID。

### ****输出****

　　共1行。以空格分隔的n个整数，分别表示顺次进行面试的应聘者的ID。

#### ****输入样例****

5 3

6 7 8 9 10

#### ****输出样例****

10 6 8 9 7

### ****限制****

　　1 ≤ n ≤ 10^3

　　1 ≤ m ≤ 2\*n

　　输入的ID保证在int类型的范围内。

　　必须使用链表实现。

　　时间：1 sec

　　空间：256 MB

### ****提示****

#### ****一级提示****

　　循环链表

# THU2015 spring 1-4 Hospital

### ****描述****

　　有n个村落分布在一条公路的沿线，为方便村民们就医，现打算在公路沿线修建一所医院。

　　由于各村落人数、年龄分布不同等原因，每个村落对医院的需求程度也不同。这里需求程度以一个非负的权值w表示：w越大，就医需求越大。对于一个 选址方案，Σwi\*di作为衡量其合理性的指标，其中wi和di分别表示第i个村落的权值和距离医院的距离，该指标越小，对应的选址就越优。请给出医院修 建的最优位置。

### ****输入****

　　共n+1行。

　　第一行包含一个整数n；

　　以下共n行，每行包含两个整数：第一个为对应村落的一维坐标xi，第二个为对应村落的权值wi。

### ****输出****

　　共两行。

　　第一行为医院的最优坐标。

　　第二行为对应的最小权值，即医院选择最优坐标时的Σwi\*di。

　　注意，如果有多个坐标权值均为最小值，请给出这些坐标中的最小者。

#### ****输入样例****

3

3 5

7 11

5 6

#### ****输出样例****

5

32

### ****限制****

　　1 <= n <= 300,000

　　村落的地址坐标是[0, 32767]的整数。

　　村落的权值是[1, 10^8]的整数。

　　时间：1 sec

　　空间：256 MB

### ****提示****

#### ****一级提示****

===\*\*一级提示\*\*===

　　可以在预处理时将坐标相同的村落合并。