Noip 专题训练：数据结构（二）

【注意事项

- 赛 不可以使用前已经有的代码。
- 不需要文件输入输出。
- 赛 训练采用 ioi赛 制。每道题最多提交 32 次。
- 测试环境为 Linux，编译命令为 -std=c++14,-O2。
- 祝选手好运。
- 赛 比由窗社（mado-soft）精神赞助！

【题目列表

英文名称	时间限制	空间限制	码长限制	题目类型	提交连接
外勤部	6s	500MB	20KB	传统	顺从
娜斯佳和哥伦比亚广播公司	4s	250MB	20KB	传统	顺从
安流沙和神经障碍	4s	250MB	20KB	传统	顺从
跳过阵列	8s	500MB	20KB	传统	顺从
Iqea	3s	250MB	20KB	传统	顺从

【赞助商广告

和泉妃爱：哥哥把很棒的女性把到手了。恭喜，哥哥！……

Pbo .D

【题目描述

让 成为 顶点上的一棵树。考虑一个最初等于 的图$_0$ 。您将得到 的更新序列，其中 -th 更新[illegible]两个不同的整数 和 -th 更新组成。

.

对于从 1 到 的每个 ，我们定义图形 如下：

- 如果$_{-1}$ 包含一条边 { , } ，则删除这条边，形成 。则删除这条边，形成 。
- 否则，将这条边加到$_{-1}$ 上，形成 。

形式上， ：=$_{-1}$ Δ {{ , }} 其中，Δ 表示对称差集(symmetric difference)。此外，可以保证 总是 的子图。换句话说，更新永远不会删除 的边。

考虑一个连通图 并在其上运行深度优先搜索。我们可以看到，树状边（即在遍历时通向尚未访问的顶点的边）构成了图 的生成树。对于一个特定的图来说，这棵生成树通常并不固定--它取决于起始顶点以及遍历每个顶点的相邻顶点的顺序。

如果对每个顶点的邻居进行排序，使得从 开始的深度优先搜索能产生 这棵生成树，我们就称顶点 为好顶点。对于从 1 到 的每个 ，找出并报告好顶点的数量。

【输入格式

第一行包含两个整数 和 (3 ≤ ≤ 2 · 10^5 , 1 ≤ ≤ 2 · 10^5) --分别是节点数和更新数。

接下来的每 - 1 行包含两个整数和 (1 ≤ , ≤ , ≠) - 由 中的一条边连接的顶点。可以保证这个图是一棵树。

接下来的每一行都包含两个整数和 (1 ≤ , ≤ , ≠) -- 添加或删除的边的端点。保证这条边不属于 。

【输出格式

每次更新后，打印一个整数 - 相应更新后好顶点的数量 。

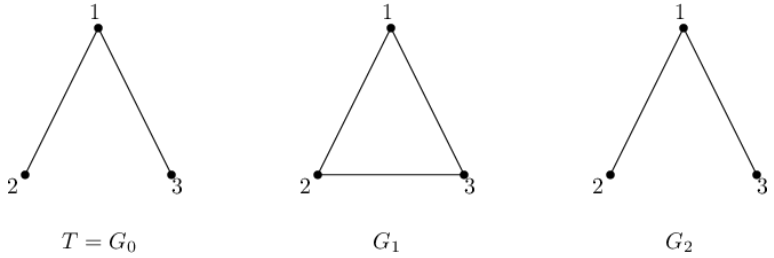
【测试样例

样例输入	样例输出
3 2 1 2 1 3 2 3 3 2	2 3

6 6 1 2 2 3 1 4 4 5 1 6 2 5 3 4 5 2 6 4 3 4 6 5	3 2 3 2 3 2

【样例解释

第一个样本如下图所示。



第一次更新后， 包含所有三条可能的边。DFS的结果如下：

- 假设起始顶点为 1。我们有两种选择来排列 1 的邻接点，要么是[2,3] ，要么是[3,2] 。
  - 如果我们选择前者，则会到达顶点 2。无论其邻居的排序如何，下一个被访问的顶点将是 3。因此，该DFS生成的生成树将包含边 {1,2} 和 {2,3} ，这不等于 。
  - 如果我们选择后者，就会得到一棵有边 {1,3} 和 {2,3} 的生成树。因此，无法对顶点的邻接排序，从而使DFS 产生 ，因此 1 不是一个好顶点。
- 假设起始顶点为 2。我们有两种遍历其邻居的选择。如果我们先访问 3，那么生成树将由边 {2,3} 和 {1,3} 组成，这不等于 。然而，如果我们先访问 1，那么我们只能从这里继续访问 3，生成树将由边 {1,2} 和 {1,3} 组成，等于 。因此，2 是一个好顶点。
- 从顶点 3 开始的情况与从顶点 2 开始的情况是对称的，因此 3 是一个好顶点。

第二次更新后，2 和 3 之间的边被删除，因此

= 。由此可见，DFS 生成的生成树总是等于 ，与起始顶点的选择无关。因此，答案是 3 。

在第二个样本中，相应查询后的良好顶点集合为

- {2,3,5}
- {3,5}
- {3,4,5}
- {4,5}
- {4,5,6}
- {5,6}

Pbo .NtN

【题目描述

娜斯佳是个很有竞争力的程序员，但她现在只是在学习。最近，Denis 告诉了她检查字符串括号序列是否正确的方法。没想到，娜斯佳提出了一个更复杂的问题，而Denis 却无法解决。你能解决这个问题吗?

给出一个字符串。它由 种成对的括号组成。每个括号的形式是 - 它是一个整数，使得 1 ≤ ≤ 。如果括号的形式是 ，那么:

- 如果 > 0 ，那么它就是 类型的开口括号。
- 如果 < 0 ，那么它就是-类型的闭合括号。因此，总共有 种类型的成对括号。

需要回答这些问题:

1. 将 位置的支架更换为 形式的支架。
2. 检查从 -th 到 -th 位置（包括）的子串是否为正确的括号序列。

回顾正确括号序列的定义：- 空序列是正确的。

- 如果 和 是两个正确的括号序列，那么它们的连接" "也是正确的括号序列。
- 如果是正确的括号序列，（ 1 ≤ ≤) 是一种括号类型，那么序列 " - "也是正确的括号序列。

【输入格式

第一行包含一个整数（ 1 ≤ ≤ 10^5) - 字符串长度和（ 1 ≤ ≤) - 成对括号的种类数。

第二行包含长度为 的字符串 - 整数 $,_{12},\ldots,$（ 1 ≤≤ 第三行包含单整数(1 ≤ ≤ 10^5) - 查询次数。
以下 行中的每一行都对查询进行了描述:

- 1 - 1 类型的查询 1 ≤ ≤ , 1 ≤ ≤) 。
- 2 - 2 类型的查询 1 ≤ ≤ ≤) 。

【输出格式

对于每个 2 类查询，如果查询的子串是正确的括号序列，则输出 "是"，否则输出 "否"。
所有字母均可以大小写显示。

【测试样例

样例输入	样例输出
2 1 1 -1 1 2 1 2	是
2 2 1 -2 1 2 1 2	没有
6 2 1 2 -2 -1 1 -1 3 2 1 6 2 1 4 2 2 5	是 是 否

2 2 -1 1	没有 是
样例输入	样例输出
(继续) 4 2 1 2 1 1 1 1 2 -1 2 1 2	

【样例 4 解释

在第四次测试中，最初字符串不是一个正确的括号序列，因此第一次查询的答案是 "否"。经过两次修改后，它将等于 " 1 -1 "，因此它是一个正确的括号序列，第四次查询的答案是 "是"。

Pbo .aPdh dN

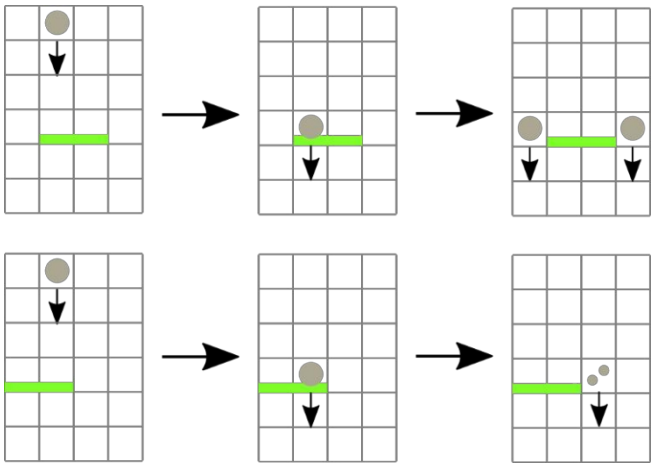
【题目描述

Andryusha 发现了一台令人费解的游戏机。这台游戏机是一个垂直调整的棋盘，被分成一个个正方形单元。棋盘从左到右有 列，从1 到 ，从下到上有 Λ 行，从 1 到 Λ。

此外，一些棋盘行中还存在障碍。总共有 个障碍物，其中 -th 占据了 行的 至 单元。安德柳莎清楚地记得，没有两个障碍物共用同一行。此外，没有一行完全被障碍物占据，也就是说，每一行中至少有一个单元是空闲的。

玩家可以从上方向机器的任何一列投掷弹珠。弹珠会向下坠落，直到遇到障碍物或跌穿棋盘底部。弹珠一旦遇到障碍物就会消失，但在同一障碍物的左侧和右侧又会有两颗弹珠取而代之。在障碍物位于棋盘边缘的情况下，两颗弹珠会出现在边缘对面的障碍物旁边。多个弹珠可以占据棋盘的同一位置，但不会妨碍彼此的移动。最终，所有弹珠都会从机器底部落下。

大理石与障碍物相互作用的实例。



奇怪的是，有时弹珠可以像自由细胞一样穿过障碍物。这是因为障碍物实际上是有生命的，当弹珠从很高的高度向它们飞来时，它们会感到害怕。更具体地说，如果一个弹珠从相对高度超过

(也就是说，它开始下落的高度严格高于 +)，那么障碍物就会避开弹珠。如果一颗弹珠从棋盘顶端抛出，则认为它出现的高度是(裔 + 1) 。

Andryusha 记得在每一列都扔过一次弹珠。请帮助他找出落在机器底部的弹珠总数。由于答案可能比较大，请打印出它的模数 $10^9 + 7$ 。

【输入格式

第一行包含三个整数：屹、 和 ($1 \le$ 屹 $\le 10^9$, $2 \le \le 10^5$, $0 \le \le 10^5$)

- 分别表示机器的行数、列数和障碍数。

接下来的 行描述了障碍。 其中第 1 行包含四个整数 , , , 和

($1 \le \le$ 括号 , $1 \le \le \le$, $1 \le \le 10^9$) - - 第 1 个障碍物的行索引、最左和最右列索引以及最大相对下落高度，使得障碍物不会躲避下落的弹珠。保证每行至少有一个空闲单元格，且所有 都是不同的。

【输出格式

打印一个整数--10 模问题的答案^9 + 7 。

【测试样例

样例输入	样例输出
10 5 1 3 2 3 10	7
10 5 2 3 1 3 10 5 3 5 10	16
10 5 2 3 1 3 7 5 3 5 10	14
10 15 4 7 3 9 5 6 4 10 1 1 1 4 10 4 11 11 20	53

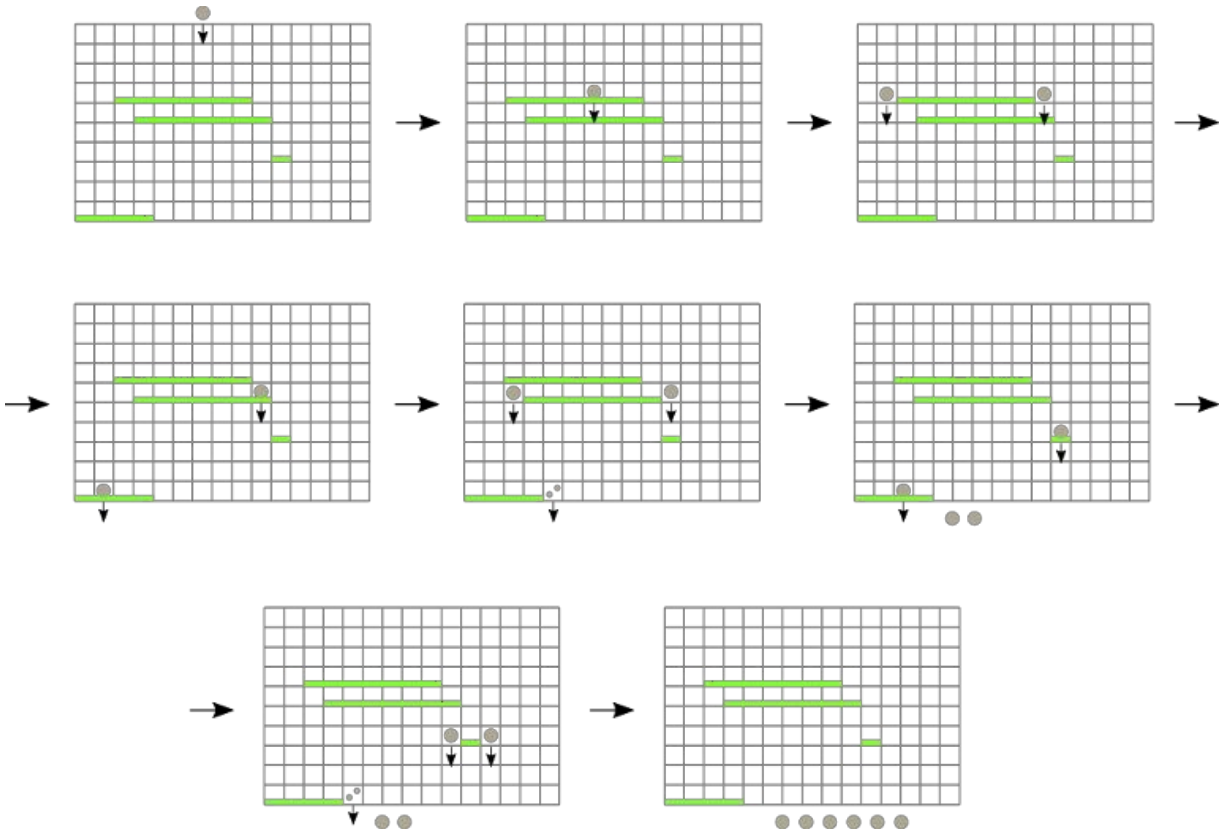
【样例解释

在第一个示例中，只有一个障碍：如果向第二列或第三列投掷一个弹珠，就会有两个弹珠出来，否则就只有一个。总答案为 7。

在第二个示例中，产生的弹珠数量按初始弹珠的索引列顺序为 2 ， 2 ， 4 ， 4 ， 4。

在第三个示例中，产生的弹珠数量分别为 1 ， 1 ， 4 ， 4 ， 4 。请注意，第一个障碍物可以避开从棋盘顶部掉落的弹珠，但不能避开从第二个障碍物掉落的弹珠。

在第四个示例中，得到的弹珠数是 2 ， 2 ， 6 ， 6 ， 6 ， 6 ， 6 ， 6 ， 1 ， 2 ， 1 ，1 ， 1 。下图显示了一颗弹珠被扔进第七列的情况。



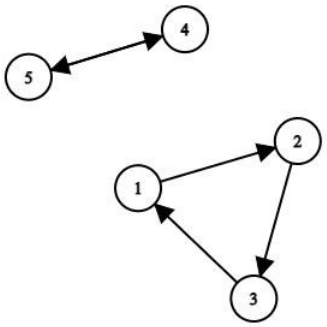
将弹珠扔进第七列的结果。

Pbo .

【题目描述

给你一个大小为 的整数数组 和一个大小为 的排列 。您将收到三种类型的 查询：

1. 对于给定的数字 和 ，计算数组 中从 到 。
2. 给你两个数字和 。让我们从 perutation 建立一个有向图：它有 顶点和 边 → 。让 成为该图中可从 到达的顶点集合。您应该添加到所有 中，以便在 .
3. 给您的索引是 和 。您应将 和 互换。



与排列对应的图形[2,3,1,5,4] 。

请处理所有查询并打印 1 类查询的答案。

【输入格式

第一行包含一个整数 （ 1 ≤ ≤ 2 · 10^5 ） - 数组和排列的大小。

第二行包含整数 ，12 ，…， （ - 10^8 ≤ ≤ 10 ）。第三行包含 个不同的整数 ，12 ，…， （ 1 ≤ ≤ ）.

第四行包含一个整数 - 查询次数（1 ≤ ≤ 2 · 10^5 ）。

接下来的 行包含查询说明。其中第 行以整数 （ 1 ≤ ≤3 ） 开头 - - 查询类型。

- 如果 = 1 ，那么 -th line 也包含两个整数 ，（ 1 ≤ ≤ ≤ ）。
- 如果 = 2 ，那么 -th line 也包含两个整数 ，（ 1 ≤ ≤ ，- 10 ≤^{8} ≤ ^{108} ）.
- 如果 = 3 ，那么 -th line 也包含两个整数 ，（ 1 ≤ ，≤ ）.

【输出格式

对于每个第一种类型的查询，打印一个整数--该查询的答案。

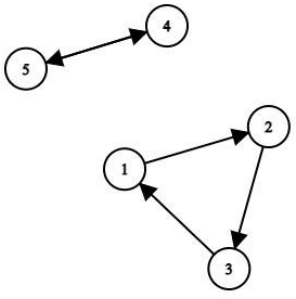
【测试样例

样例输入	样例输出

5 6 9 -5 3 0 2 3 1 5 4 6 1 1 5 2 1 1 1 1 5 3 1 5 2 1 -1 1 1 5	13 16 11
样例输入	样例输出
8 -15 52 -4 3 5 9 0 5 2 4 6 8 1 3 5 7 10 2 2 2 2 5 -1 1 1 8 1 1 5 1 5 8 3 1 6 2 1 50 1 1 8 2 6 -20 1 1 8	61 45 22 461 301
1 1 1 1 1 1 1	1

【样例解释

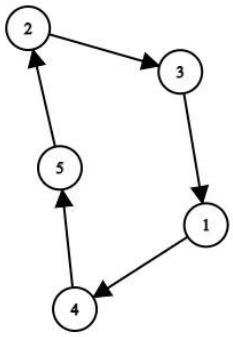
在第一个例子中



与初始排列相对应的图形

共有 6 项查询。

1. 1 至 5 段的总和为 $1 + 2 + 3 + 4 + 5 = 6 + 9 + (-5) + 3 + 0 = 13$ 。
2. 如果从 1 开始，我们可以查询到 {1,2,3} 。查询 后得到: [7,10, - 4,3,0] 。
3. 1 至 5 段的总和为 $1 + 2 + 3 + 4 + 5 = 6 + 9 + (-5) + 3 + 0 = 16$ 。
4. 查询后 =[4,3,1,5,2] 。



新排列对应的图形。

5. 如果从 2 开始，我们可以查询到{1,2,3,4,5}。之后的查询 为: [6,9, - 5,2, - 1 。] (

6. 1 至 5 段的总和为$_1$ +$_2$ +$_3$ +$_4$ +$_5$ = 6 + 9 +(-5) + 2 + -1) = 11 。

Pbo .q

【题目描述

Gridland 位于无限网格上，其形状由单元格组成。网格地的每个单元格都是一个城市。相邻单元中的两个城市通过长度为 1 的道路相连。通过道路可以从任何城市到达任何其他城市。两个城市之间的距离就是一个城市到另一个城市的道路总长度的最小值。只需使用不属于网格地的单元格，就可以从任何不属于网格地的单元格到达任何其他不属于网格地的单元格。换句话说，网格地是连通的，网格地的补集也是连通的。

目前，格瑞德兰还没有一座城市拥有 Iqea 名店。但 Iqea 有在格瑞德兰建造商店的伟大计划。为了方便顾客，Iqea 决定开发一款应用程序。通过这个应用程序，每个人都可以了解到离最近的 Iqea 的距离。您将开发这款应用程序。要求您处理两种类型的查询：

- 新的 Iqea 商店已在该市开业，坐标为 ,() ；
- 客户想知道离其所在城市最近的已开业 Iqea 商店的距离，该商店位于坐标为 ,() 的小区。

注意，顾客只能通过道路移动，在前往商店的途中绝对不能离开网格地带。

【输入格式

第一行包含一个整数 - 网格地中的城市数量 (1 ≤ ≤ 300000)。以下 行包含两个整数 和 - 包含 -th 的单元格坐标

(1 ≤ , ≤ 300000) 。没有两个单元重合。单元构成相连区域，该区域的补码也是相连的。

下面一行包含单个整数 - 查询次数 (0 ≤ ≤ 300000) 。 之后的行包含查询次数， -th line 包含三个整数 , 和 (∈ 1,2 , 1 ≤ ,≤300000).如果 = 1 ，则在坐标为 ,() 的城市开设了新的 Iqea 商店。该城市以前肯定没有 Iqea 商店。如果 = 2 ， 您要查找离坐标为 ,() 的城市最近的已开业 Iqea 商店的距离。可以保证在这两种类型的单元格查询中 ,() 都属于 Gridland。

【输出格式

对于每一个第二种类型的查询，输出一个整数--到最近的 Iqea 商店的最小距离。如果尚未开设 Iqea 商店，则输出-1。

【测试样例

样例输入	样例输出

7 1 2 1 3 2 3 3 1 3 2 3 3 4 2 5 2 3 2 1 4 2 2 1 2 1 3 3 2 2 3	-1 5 1
样例输入	样例输出
6 1 1 1 2 1 3 2 1 2 2 3 1 4 1 3 1 2 1 2 1 2 2 2 1 3	3 2