
下次一定

background

是个被用烂了的 idea 出的题目, 当然并不是搬运题或者改编题.
算是个半原创题吧, 所以不对题目解法和数据的正确性做保证

“我出了个经典 NPC 问题的模型, 帮助学弟学妹们学习搜索算法呢.”

“你这个原创题的假的吧, 我在 TOCS 上面查到了一个多项式的做法. 这玩意明明是 P 问题.”

“啊这...下次...下次一定...一定不会锅了.”

statement

spinach 出了一些训练题尝试组成模拟赛,

因为他的智力不足以掌握数组以外的数据结构, 所以题目被排成一行, 而组一个模拟赛需要一个非空的区间中的所有题目.

形式化地说, 给定 n , 有 n 个题目, 分别编号 $1, 2, 3 \dots n$, 一个模拟赛组织方案用 (l, r) 表示 $l \leq i \leq r$ 的 a_i 组成模拟赛, 满足 $1 \leq l \leq r \leq n$ (即一个闭区间).

明天就是第一场模拟赛了, spinach 正做着 “青年理论计算机科学家” 的白日梦, 在家里啃论文, 它突然意识到某些题目假了, 而某些题目正确性存疑.

具体来说, 对于编号为 i 的模拟赛, 用 a_i 表示其正确性, 如果是真的那么 $a_i = 0$, 假题有 $a_i = 1$, 存疑是 $a_i = 2$

他来不及剔除假题和存疑的题目, 打算蒙混过关!

如果模拟赛中有假题, 那么 OI 生涯就完蛋啦, 那么, 凑出一套看起来正确的题目, 样挨过去吧.

如果 $l \leq i \leq r$ 的 a_i 之和模 3 余 0 那么 (l, r) 组织的比赛就会看起来很正确.

形式化地说 $(\sum_{l \leq i \leq r} a_i) \equiv 0 \pmod{3}$ 则选择 (l, r) 是正确的.

现在他想问问你, 给定 L, R , 如果只能用 $L \leq l \leq r \leq R$ 的区间 (l, r) 那么有多少种模拟赛组织方案是看起来很正确的.

I/O

input

第一行两个正整数 n, m 表示题目数量和询问数量.

第二行, n 个空格分割的整数, 表示接下来 m 行, 每行两个满足 $1 \leq L \leq R \leq n$ 的正整数 L, R , 表示一个查询.

output

对于每次询问, 输出一个整数, 表示满足题目要求的, 看起来很正确的模拟赛数量.

example

case1

- input

```
1 5 3
2 0 0 0 0 0
3 1 5
4 3 4
5 4 4
```

- output

```
1 15
2 3
3 1
```

- explanation

惊了, 他居然除了 5 个正确的题目, 那么可以随便组题不用担心出锅啦.

第一次询问, 可以选 $(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 2) \dots (5, 5)$ 共 15 个.

后面两次询问同理.

case2

- input

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

- output

```
1 4
```

- explanation 可以用 $(1, 1), (1, 3), (2, 3), (3, 4)$

restrictions

- compiler flags: (备注:luogu 使用的 gnu 套件版本高于 NOI linux 的 4.8.4, 两者的版本均高于 devcpp 自带的远古 mingw 和 cena 自带的编译器)
- TL=2.5s (备注: 现在 NOIp 的评测机性能和 Luogu 高性能模式是接近的, 但是蔽校一机房的远古教师机已经跟不上时代里...请根据评测机实际性能对 TL 进行调整, 在 std 的 2 倍最大运行时间以上)
- ML=256MB

共 10 组数据, 不设置 subtask 模式评测, 而是传统的每个点分数相同, 得分互相独立.

- 对于前 30% 的数据, 有 $1 \leq n, m \leq 2000$
- 所有数据, 有 $1 \leq n, m \leq 2 \times 10^5$

solution

线段树, 考虑如何合并.

对于一个区间 $[l, r]$ 维护

- $pre[0/1/2]$ 表示和模 3 余 0, 1, 2 的前缀数量.
- $suf[0/1/2]$ 表示和模 3 余 0, 1, 2 的后缀数量.
- sum 表示和模 3 的结果.
- $cnt[0/1/2]$ 表示和模 3 余 0, 1, 2 的子区间数量.

对于一个 a_i , 可以轻松求出这些东西.

对于一个 A, B 从左到右拼接而成的 $A + B$ 有

-
- *pre* 两类, 一类只在 *A* 中, 一类是 *A* 整体拼接上 *B* 的前缀.
 - *suf* 类似 *pre*.
 - *sum* 直接相加.
 - *cnt* 一种是 *A, B* 的子区间, 另一种是 *A* 的 *suf* 拼接上 *B* 的 *pre*.

复杂度 $O(n + m \log n)$

还有一种非常简单的做法, 就是莫队, 是 $O(n\sqrt{n} + m)$ 的.

类似地维护 *pre, suf, cnt* 考虑在左边/右边加入/删除一个点的变化.

如果你的实现对于空区间会挂掉, 那么你应该使用一点基础技巧 (先 *l* 向左, *r* 向右进行扩大, 再考虑缩小区间的移动, 这样任意时刻你的区间都是非空的).

我推荐一直保持区间非空, 并加入一些特殊判定.