第二届包子杯题目讲解

包包，fatalerror，核电站

2025 年 5 月 10 日

H. Dilworth's Theorem

题意

给定整数 n，构造一个 1 到 n 的排列 p，使得它的最长上升子序列和最长下降子序列的长度相等。

现场情况

首个提交：998@00:08
首个 AC：998@00:08
通过队伍：48

Hint 1: 注意到 $n = 2$ 是唯一无解的情况。

H. Dilworth's Theorem

Hint 1: 注意到 $n = 2$ 是唯一无解的情况。

Hint 2: 可以根据 n 的奇偶性分类讨论。

Fun fact: 不少队伍都给出了 n 为偶数时无解的错误结论。

解答： 签到题，有多种构造方法。

如果 n 为奇数，一种可行的构造为

$$n, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil + 1, 1, 2, \ldots, \left\lceil \frac{n}{2} \right\rceil$$

H. Dilworth's Theorem

解答： 签到题，有多种构造方法。

如果 n 为奇数，一种可行的构造为

$$n, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil + 1, 1, 2, \ldots, \left\lceil \frac{n}{2} \right\rceil$$

如果 n 为偶数，将上述方案的 n 与 $n-1$ 的位置交换即可。

总结： 构造题但一签。

I: Two Squared Equals Four

题意

给定平面上 n 个不同点。求在删除恰好一个点后，剩余的点最多能构成多少个正方形。

现场情况

首个提交：猪联璧合 @02:53
首个 AC：猪联璧合 @03:21
通过队伍：5

Hint 1: 需要计算每个顶点被多少个正方形共用。也就是说，需要计数有多少个正方形。

Hint 1: 需要计算每个顶点被多少个正方形共用。也就是说，需要计数有多少个正方形。

Hint 2: 显然需要通过 $O(n^2)$ 的复杂度枚举。如何枚举?

Hint 1: 需要计算每个顶点被多少个正方形共用。也就是说，需要计数有多少个正方形。

Hint 2: 显然需要通过 $O(n^2)$ 的复杂度枚举。如何枚举?

Hint 3: 能形成正方形的四个顶点应具有怎样的位置关系？如果枚举两个点，怎么确定一个正方形?

解答： 枚举对角线的两个顶点 $A(x_1, y_1), C(x_2, y_2)$，这样就可以确定正方形中心点 $O(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ 以及其余两个顶点

$$B(\frac{x_1+x_2}{2} - \frac{y_1-y_2}{2}, \frac{y_1+y_2}{2} + \frac{x1-x2}{2}), D(\frac{x_1+x_2}{2} + \frac{y_1-y_2}{2}, \frac{y_1+y_2}{2} - \frac{x1-x2}{2})$$

从而对每个顶点分别计数即可。复杂度 $O(n^2)$。

总结： 需要一点变形的几何题。

Fun fact: 尽管是出题人认为的二签，但赛时约前 3h 竟然无人尝试。

E: Another GCD

题意

维护一个正整数对 (v, w) 构成的多重集。需要支持添加，删除，查询当前多重集中满足 $\gcd(v, k) \neq 1$ 的数对里最大的 w。

现场情况

首个提交：Ray@00:17
首个 AC：Ray@00:17
通过队伍：37

E: Another GCD

Hint： 对于 $v \leq 5e5$，v 的不同质因子个数不会超过 6 个。

E: Another GCD

解答： 对于某个询问 k，满足 $\gcd(v,k)\neq 1$ 的 v 一定与 k 有至少一个相同的质因子。所以我们对于每个质数 p 都维护一个多重集 S_p，存所有满足 $p|v$ 的 w。对于一个询问 k，答案即为：

$$\max_{p|k}(\max(S_p))$$

删除和添加数对直接更新 v 所有质因子的多重集即可。复杂度 $O(V\log V+n(\log n+\log V))$。关于 $O(V\log V)$ 预处理，且 $O(\log V)$ 查询某个数质因子的实现可以参考 oiwiki 等。

注意枚举因子而非质因子会超时。

总结： 简单数论，三签。

J: HDZ Explosion

题意

给定一个 1 到 n 的排列 p。对于一个给定的 d，每次可以从 j 移动至 $p[j-d \ldots j+d]$ 中最大值的位置。求最小的 d，使得不论初始位置在哪，都最终可以到达 n 所在位置。

现场情况

首个提交：998@00:12
首个 AC：998@00:12
通过队伍：42

J: HDZ Explosion

Hint 1: 设 $p[\text{dest}] = n$，满足题意的情况下，所有位置 i 的下一步都会落在 $[i, \text{dest}]$ 或 $[\text{dest}, i]$。

证明： 不妨令 $i < \text{dest}$，若位置 i 的下一步落在位置 j $(i \le j)$，则在 $p[j-d \dots j-1]$ 中没有大于 p_j 的项。此时位置 j 的下一步也会使下标不减。由数学归纳法，$i, j, \dots$ 的移动链会使得下标单调不减。$i > \text{dest}$ 同理。故满足题意当且仅当所有位置均朝 dest 移动。

解法一（核电 & 包包）：

Hint 2: 这个问题的 d 是否满足单调性？如何证明？

解法一（核电 & 包包）：

Hint 2: 这个问题的 d 是否满足单调性？如何证明？

Answer 2: 满足的，兄弟，满足的。

证明： 当 $d = k$ 满足题意时，考察 $d = k + 1$。在每一次移动后，$d = k + 1$ 情况下的位置不会比 $d = k$ 情况下的位置离 dest 更远。数学归纳法可知，$d = k + 1$ 也满足题意。因此 d 的单调性成立。

解法一（核电 & 包包）：

Hint 2: 这个问题的 d 是否满足单调性？如何证明？

Answer 2: 满足的，兄弟，满足的。

证明： 当 $d = k$ 满足题意时，考察 $d = k + 1$。在每一次移动后，$d = k + 1$ 情况下的位置不会比 $d = k$ 情况下的位置离 dest 更远。数学归纳法可知，$d = k + 1$ 也满足题意。因此 d 的单调性成立。

Hint 3: 给定一个 d，如何判断在当前 d 下是否必定走到目的地？

J: HDZ Explosion

可用滑动窗口或 ST 表在 $O(n)$ 时间内验证每个点是否朝向目标移动。具体实现：

1. 将数组前后填充 d 个负无穷值。
2. 维护窗口大小为 $2d+1$ 的滑动窗口，记录每个位置的下一步。
3. 检查所有位置是否均朝 dest 方向移动。

最终算法为：二分答案 d + 滑动窗口/ST 表验证，时间复杂度 $O(n \log n)$。1e6 只给 1s 是为了卡掉双 log 解法。

解法二（fatalerror）：

Hint 2'： 从下标 i 跳到下标 j，则一定有 $p[i] < p[j]$。

解法二（fatalerror）：

Hint 2'： 从下标 i 跳到下标 j，则一定有 $p[i] < p[j]$。

可以一边倒序遍历 $x = n, n-1, \ldots, 1$ 所在位置，一边维护已经遍历过的下标。当遍历到 $x = p[i]$ 时，在已经遍历过的 $n, n-1, \ldots, x+1$ 的下标中查找到最近的 j 的距离 d，对所有 d 取最大值即可。复杂度 $O(n \log n)$。

实际实现中，`c++` 的 `set` 效率太低，为此可以改用线段树二分等方式。

解法二（fatalerror）：

Hint 2'： 从下标 i 跳到下标 j，则一定有 $p[i] < p[j]$。

可以一边倒序遍历 $x = n, n-1, \ldots, 1$ 所在位置，一边维护已经遍历过的下标。当遍历到 $x = p[i]$ 时，在已经遍历过的 $n, n-1, \ldots, x+1$ 的下标中查找到最近的 j 的距离 d，对所有 d 取最大值即可。复杂度 $O(n \log n)$。

实际实现中，`c++` 的 `set` 效率太低，为此可以改用线段树二分等方式。

在此基础上，还可以直接用单调栈处理每个 x 左侧和右侧更大值的位置。复杂度 $O(n)$。

总结： 四签。

K: Boring Tree

题意

给定一棵 n 个节点的树，树上有 k 个石子分散在各个节点上，同一节点可以有多个石子。每次操作可以将一个石子移动树上相邻节点，求最少操作数，使得所有石子都处在某条简单路径上。

现场情况

首个提交：积了一杜子德 @00:15
首个 AC：剑来 @00:49
通过队伍：13

K: Boring Tree

Hint 1: 简单版本：如果最终路径要求不能拐弯怎么做?

K: Boring Tree

Hint 1: 简单版本：如果最终路径要求不能拐弯怎么做?

Answer 1: 我们可以用树上动态规划的思想。定义 f_u 为：只考虑子树 u 中的节点和石子，且最终路径的端点从 u 开始，所需的最小操作数是多少。转移考虑路径下一段连接到 u 的哪个子节点。

K: Boring Tree

Hint 1: 简单版本：如果最终路径要求不能拐弯怎么做?

Answer 1: 我们可以用树上动态规划的思想。定义 f_u 为：只考虑子树 u 中的节点和石子，且最终路径的端点从 u 开始，所需的最小操作数是多少。转移考虑路径下一段连接到 u 的哪个子节点。

设路径下一段为 v，有转移：

$$f_u = \min_{v \in C_u} \left(f_v + \sum_{(i \in S_u) \wedge (i \notin S_v)} d(i,u) \cdot a_i \right)$$

其中 $d(u,v)$ 为 u 到 v 的简单路径的长度，a_u 表示节点 u 上的石子数量，S_u 代表子树 u 中的节点集合，C_u 代表 u 的子节点集合。最终答案即为：

$$\min_{1 \le u \le n} \left(f_u + \sum_{i \notin S_u} d(i,u) \cdot a_i \right)$$

K: Boring Tree

解答： 如果路径可以拐弯，问题其实还是类似的。定义 g_u 和 f_u 相同，只是路径的最浅节点为 u 且允许路径拐弯。注意到 g_u 实际上可以由 f 转移而来。唯一不同的是，我们可以选择两个子节点转移而不是一个。所以 g_u 的转移方程为：

$$g_u = \min_{(v,w\in C_u)\wedge(v\neq w)} \left(f_v + f_w + \sum_{(i\in S_u)\wedge(i\notin S_v)\wedge(i\notin S_w)} d(i,u)\cdot a_i \right)$$

最终答案也与 f 相似，即：

$$\min_{1\le u\le n} \left(\min(g_u, f_u) + \sum_{i\notin S_u} d(i,u)\cdot a_i \right)$$

K: Boring Tree

代码实现上依然有一些技巧。比如在计算 g_u 的时候，二重循环枚举 v 和 w 是会被菊花图卡成 $O(n^2)$ 的。转移方程中的求和项可以用欧拉序 + 前缀和等方法预处理，并在 dfs 的过程中动态计算。时间复杂度 $O(n)$ 或 $O(n \log n)$。

总结：需要一些细节的树形 dp。

G: Fatalerror: Implementation Failed

题意

交互题。评测机初始有一个隐藏的 64-bit 整数 n。每次可以选定某个 64-bit 整数 x，并询问 $n \oplus x$ 的置位数量。在 63 次询问内找到 n。

现场情况

首个提交：tomcat@00:20
首个 AC：Whispering Snowflakes@00:29
通过队伍：22

G: Fatalerror: Implementation Failed

记 $H = 64$，$f(n)$ 为 n 的置位数量。

Hint 1: 不考虑查询次数限制下，不难想到以下朴素算法：

首先查询 $f(n) = f(n \oplus 0) = C$。然后对 $0 \le i < H$，依次令 $x_i = 2^i$、即翻转 n 的第 i 位、查询 $f(n \oplus x_i) = c_i$，将 c_i 与 C 比较，若 $c_i < C$ 则第 i 位是 1，反之则是 0。这一方法显然需要 $1 + H = 65$ 次查询。

G: Fatalerror: Implementation Failed

记 $H = 64$，$f(n)$ 为 n 的置位数量。

Hint 1: 不考虑查询次数限制下，不难想到以下朴素算法：

首先查询 $f(n) = f(n \oplus 0) = C$。然后对 $0 \le i < H$，依次令 $x_i = 2^i$、即翻转 n 的第 i 位、查询 $f(n \oplus x_i) = c_i$，将 c_i 与 C 比较，若 $c_i < C$ 则第 i 位是 1，反之则是 0。这一方法显然需要 $1 + H = 65$ 次查询。

Hint 2: 在此基础上还可以把对 C 的查询优化掉，因为所有 c_i 的种类数最多有两种：

1. 如果 c_i 有两种，则结果只能是 $C \pm 1$，可以通过 c_i 之间的相对大小分辨数位是 1 或 0。
2. 如果 c_i 只有一种，则结果只能是 1 或 $H - 1$，从而分辨 n 的全部数位都是 1 或 0。

但优化后的方法依然需要 $H = 64$ 次查询。

G: Fatalerror: Implementation Failed

解法一：

依旧可以延续之前的思路，仅查询后 63 位，同样判断所有 c_i 的种类数：

1. 如果 c_i 有两种，则可以确定 C，从而通过 c_i 和 C 的大小关系确定后 63 位，并进一步结合 C 确定最高位。
2. 如果 c_i 只有一种，则取值仅可能是 $1, 2, 62, 63$，并且后 63 位应全都相同，简单分类讨论即可确定 n 的后 63 位和最高位。

解法二：

每次查询一个数位的效率较低，考虑相邻两个数位一组进行查询。具体地，第 i 次查询令 $x_i = 2^i + 2^{i+1}$，查询 n 的第 i 和 $i+1$ 位。这样总查询次数正好是 $H - 1 = 63$ 次。

每次查询的结果 $c_i = C + \{2, 0, -2\}$，分别对应相邻数位为 00、01/10、11。其中，两个 0 和两个 1 可以直接确定数位，一 0 一 1 则需要进一步处理。从已知的数位一步步推断出相邻的未知数位即可。

具体来说，所有查询的 c_i 的种类数有三种：

G: Fatalerror: Implementation Failed

1. 如果 c_i 有三种，则显然易知。

2. 如果 c_i 有两种，则结果可能是：
(1) $C+\{2,0\}$，相邻数位仅包含 00、01/10；
(2) $C+\{0,-2\}$，相邻数位仅包含 01/10、11。
于是可以对两种可能情况分别作试填，检查试填结果是否满足 $f(n)=C$ 即可。

3. 如果 c_i 仅有一种，则结果可能是：
(1) $C+2$，所有相邻数位全为 00；
(2) $C-2$，所有相邻数位全为 11；
(3) C，所有相邻数位全为 01/10。
但对于第三种情况，无法分辨 n 为 $(010101\ldots)_2$ 或 $(101010\ldots)_2$。

G: Fatalerror: Implementation Failed

1. 如果 c_i 有三种，则显然易知。

2. 如果 c_i 有两种，则结果可能是：
(1) $C+\{2,0\}$，相邻数位仅包含 00、01/10；
(2) $C+\{0,-2\}$，相邻数位仅包含 01/10、11。
于是可以对两种可能情况分别作试填，检查试填结果是否满足 $f(n)=C$ 即可。

3. 如果 c_i 仅有一种，则结果可能是：
(1) $C+2$，所有相邻数位全为 00；
(2) $C-2$，所有相邻数位全为 11；
(3) C，所有相邻数位全为 01/10。
但对于第三种情况，无法分辨 n 为 $(010101\ldots)_2$ 或 $(101010\ldots)_2$。

针对这一情况，可以考虑对查询数位的顺序进行随机打乱，这样可以让被查询数位按顺序为 $(010101\ldots)_2$ 或 $(101010\ldots)_2$ 的可能性尽可能低。

解法三：

回到最初已知 C 的情况。类似解法二、把多个数位的查询结合起来，同时查询 n 的 k 个数位。如果这 k 个位上恰好全是 0 或 1，那么就可以通过 1 次查询就确定 k 个位置；否则对其中一个数位执行朴素查询方法，然后重复进行以上过程，直到 k 个数位全部确定。

具体地，令 $k=2$、即所有数位两两一对，第 i 次查询令 $x_i = 2^{2i} + 2^{2i+1}$，查询 n 的第 $2i$ 和 $2i+1$ 位。如果均为 1 或 0，则可以一次确定两个数位；否则两个数位为一个 1 一个 0，就再查询其中一个数位。

G: Fatalerror: Implementation Failed

问题是对后半部分的查询，仅剩余 $63 - 1 - 32 = 30$ 次查询。如果按照 n 的原始数位顺序两两进行分组，很容易就会在 $(010101\ldots)_2$ 或 $(101010\ldots)_2$ 上超出次数限制。

G: Fatalerror: Implementation Failed

问题是对后半部分的查询，仅剩余 $63 - 1 - 32 = 30$ 次查询。如果按照 n 的原始数位顺序两两进行分组，很容易就会在 $(010101\ldots)_2$ 或 $(101010\ldots)_2$ 上超出次数限制。

既然不能确保这 2 个位相同，那就让这 2 个位不同的可能性尽可能低。考虑随机化，每次随机选择 2 个未查询的数位进行查询，让每次取出的两个数位都是一 1 一 0 的可能性尽可能低。

解法四：

直接对 n 异或一个随机数。具体地，在解法二或三的基础上，先生成一个随机数 RD，每次查询时令 $x_i := x_i \oplus RD$。

解法四：

直接对 n 异或一个随机数。具体地，在解法二或三的基础上，先生成一个随机数 RD，每次查询时令 $x_i := x_i \oplus RD$。

总结： 乱搞题，能过就是好方法。

Fun fact: 出题人甚至没想到有确定性算法。

G: Fatalerror: Implementation Failed

严格计算（以解法二为例）：

考虑由 $\frac{H}{2}$ 个 1、$\frac{H}{2}$ 个 0 构成的 n。不合法的情况发生在 $\frac{H}{2}$ 次取出的两个数位全是一 1 一 0，其概率为

$$\frac{\frac{H}{2}\cdot\frac{H}{2}}{C(H,2)}\cdot\frac{(\frac{H}{2}-1)\cdot(\frac{H}{2}-1)}{C(H-2,2)}\cdot\ldots\frac{1\cdot1}{C(2,2)}=\frac{((\frac{H}{2})!)^2\cdot2^{\frac{H}{2}}}{H!}\approx2\cdot10^{-9}$$

再考虑由 $\frac{H}{2}+1$ 个 1、$\frac{H}{2}-1$ 个 0 构成的 n。不合法的情况发生在 $\frac{H}{2}-1$ 次取出的两个数位全是一 1 一 0，其概率为

$$\frac{H}{2}\cdot\frac{C(\frac{H}{2}+1,2)}{C(H,2)}\cdot\frac{(\frac{H}{2}-1)\cdot(\frac{H}{2}-1)}{C(H-2,2)}\cdot\ldots\frac{1\cdot1}{C(2,2)}=\frac{(\frac{H}{2}+1)!\cdot(\frac{H}{2})!\cdot2^{\frac{H}{2}-1}}{H!}\approx4\cdot10^{-8}$$

其他 n 的情况下均不会发生错误。设发生错误概率为 ε，总测试用例数量为 N，则单次提交无法通过的概率

$$P=1-(1-\varepsilon)^N\approx\varepsilon N\approx10^{-4}$$

D: Why Does Every Baozii Cup Have a GCD Problem

题意

维护一个数组 a，支持：单点更新，区间更新 $a_i := \gcd(a_i, x)$，区间和查询。

现场情况

首个提交：cwwdka@00:30
首个 AC：998@03:30
通过队伍：6

D: Why Does Every Baozii Cup Have a GCD Problem

解答： 对于某两个正整数 x 和 y，由于 $\gcd(x, y)$ 一定是 x 的因子，所以 $\gcd(x, y)$ 要么等于 x，要么不大于 $\frac{x}{2}$。那么 x 最多会被 gcd 操作 $O(\log x)$ 次（执行 gcd 后有变化的次数）就会变成 1。
（相关主题：LogTrick）

D: Why Does Every Baozii Cup Have a GCD Problem

解答：对于某两个正整数 x 和 y，由于 $\gcd(x, y)$ 一定是 x 的因子，所以 $\gcd(x, y)$ 要么等于 x，要么不大于 $\frac{x}{2}$。那么 x 最多会被 gcd 操作 $O(\log x)$ 次（执行 gcd 后有变化的次数）就会变成 1。
（相关主题：LogTrick）

我们可以把吓人的区间 gcd 更新变成暴力的单点更新，而且最多有效操作次数是 $O((n + q) \log V)$ 级别的。

D: Why Does Every Baozii Cup Have a GCD Problem

解答：对于某两个正整数 x 和 y，由于 $\gcd(x, y)$ 一定是 x 的因子，所以 $\gcd(x, y)$ 要么等于 x，要么不大于 $\frac{x}{2}$。那么 x 最多会被 gcd 操作 $O(\log x)$ 次（执行 gcd 后有变化的次数）就会变成 1。
（相关主题：LogTrick）

我们可以把吓人的区间 gcd 更新变成暴力的单点更新，而且最多有效操作次数是 $O((n+q)\log V)$ 级别的。

对于某个区间 $[l, r]$，和正整数 x，操作二我们有以下的暴力更新：

1. 初始化 i 为 l。
2. 找到最小的 $j \geq i$ 且 $\gcd(a_j, x) \neq a_j$。
3. 如果不存在这样的 j 或者 $j > r$，退出循环。
   否则更新 $a_j := \gcd(a_j, x)$，$i := j+1$，并回到步骤 2。

D: Why Does Every Baozii Cup Have a GCD Problem

以上做法，最多会执行 $O((n+q)\log V)$ 次操作 2、3，现在问题变成如何快速执行步骤 2。

D: Why Does Every Baozii Cup Have a GCD Problem

以上做法，最多会执行 $O((n+q)\log V)$ 次操作 2、3，现在问题变成如何快速执行步骤 2。

考虑到对于某个正整数 x，$\gcd(a_i, x) \neq a_i$ 当且仅当 a_i 不是 x 的因子。步骤 2 于是等价于找到最小的 $j \geq i$ 且 $a_i, a_{i+1}, \ldots, a_{j-1}$ 都是 x 的因子。注意到一个区间里的数都是 x 的因子，等价于它们的 lcm 是 x 的因子。所以线段树记录区间 lcm，j 可以用线段树二分在 $O(\log n)$ 的时间找到。区间和用树状数组记录即可。最终复杂度 $O((n+q)\log n \log V)$。

（相关主题：均摊复杂度，势能线段树）

D: Why Does Every Baozii Cup Have a GCD Problem

以上做法，最多会执行 $O((n+q)\log V)$ 次操作 2、3，现在问题变成如何快速执行步骤 2。

考虑到对于某个正整数 x，$\gcd(a_i, x) \neq a_i$ 当且仅当 a_i 不是 x 的因子。步骤 2 于是等价于找到最小的 $j \geq i$ 且 $a_i, a_{i+1}, \ldots, a_{j-1}$ 都是 x 的因子。注意到一个区间里的数都是 x 的因子，等价于它们的 lcm 是 x 的因子。所以线段树记录区间 lcm，j 可以用线段树二分在 $O(\log n)$ 的时间找到。区间和用树状数组记录即可。最终复杂度 $O((n+q)\log n \log V)$。
（相关主题：均摊复杂度，势能线段树）

实现时还有一个细节，就是线段树节点无法记录区间的真实 lcm，需要对超过一定大小的 lcm 进行截断。

D: Why Does Every Baozii Cup Have a GCD Problem

以上做法，最多会执行 $O((n+q)\log V)$ 次操作 2、3，现在问题变成如何快速执行步骤 2。

考虑到对于某个正整数 x，$\gcd(a_i, x) \neq a_i$ 当且仅当 a_i 不是 x 的因子。步骤 2 于是等价于找到最小的 $j \geq i$ 且 $a_i, a_{i+1}, \ldots, a_{j-1}$ 都是 x 的因子。注意到一个区间里的数都是 x 的因子，等价于它们的 lcm 是 x 的因子。所以线段树记录区间 lcm，j 可以用线段树二分在 $O(\log n)$ 的时间找到。区间和用树状数组记录即可。最终复杂度 $O((n+q)\log n \log V)$。
（相关主题：均摊复杂度，势能线段树）

实现时还有一个细节，就是线段树节点无法记录区间的真实 lcm，需要对超过一定大小的 lcm 进行截断。

总结：码量不大，需要一点注意力，一看就是包包出的数据结构题。

Fun fact: 测题时候才发现出了原题：洛谷 P9989。

A: Beautiful Substrings

题意

定义一个字符串 s 为美丽的，当且仅当 s 可以被表示为 $t + t' + t$ 的形式，其中 t 为任意非空字符串，t' 为 t 的翻转。给定一个串 s，求出 s 的美丽子串的数量。

现场情况

首个提交：积了一杜子德 @00:18
首个 AC：积了一杜子德 @00:22
通过队伍：16

A: Beautiful Substrings

Hint 1: 注意到一个美丽串 s 其实是由两个偶数长度的回文串 $s_1 = t + t'$ 和 $s_2 = t' + t$ 组成的，并且这两个回文串都是偶数长度串、且回文臂长相等。

（前置知识：对于偶串，定义回文臂长 d_i 为最大的 j 使得 $s_{i-j}s_{i-j+1}\dots s_i s_{i+1}\dots s_{i+j+1}$ 为回文串。如：$s = \text{abbaab}$，则 $d = [0, 2, 0, 2, 0]$。）

A: Beautiful Substrings

Hint 1: 注意到一个美丽串 s 其实是由两个偶数长度的回文串 $s_1 = t + t'$ 和 $s_2 = t' + t$ 组成的，并且这两个回文串都是偶数长度串、且回文臂长相等。

（前置知识：对于偶串，定义回文臂长 d_i 为最大的 j 使得 $s_{i-j}s_{i-j+1}\dots s_i s_{i+1}\dots s_{i+j+1}$ 为回文串。如：$s = \text{abbaab}$，则 $d = [0, 2, 0, 2, 0]$。）

Hint 2: 利用 Manacher 算法等找到回文中心和臂长并不困难，但是满足什么条件的两个回文串才能组成一个美丽子串呢?

A: Beautiful Substrings

解答： 如果枚举右侧回文串 s_2 的回文中心 i 和回文臂长 d_i，则满足以下条件的 s_1 可以和 s_2 组成美丽子串：

以 i 为中心的回文串可以覆盖 j，同时以 j 为中心的回文串可以覆盖 i。
形式化地：s_1 的回文中心 j 满足 $i - d_i \leq j \leq i$，并且回文臂长 d_j 满足 $j + d_j \geq i$。

A: Beautiful Substrings

解答： 如果枚举右侧回文串 s_2 的回文中心 i 和回文臂长 d_i，则满足以下条件的 s_1 可以和 s_2 组成美丽子串：

以 i 为中心的回文串可以覆盖 j，同时以 j 为中心的回文串可以覆盖 i。
形式化地：s_1 的回文中心 j 满足 $i - d_i \le j \le i$，并且回文臂长 d_j 满足 $j + d_j \ge i$。

因此可以把 j 按照 $j + d_j$ 排序，从左到右枚举 i 的同时维护 j，计算区间 $[i - d_i, i]$ 内 j 的数量，这可以用树状数组解决。时间复杂度 $O(n \log n)$。解法并不唯一。

总结： 发掘美丽子串性质后，还需要明确维护什么东西、用什么维护。字符串 + 又是数据结构。

L: OR + AND

题意

对于一个数组 b，定义 $f(b)$ 为：

- 将 b 划分成两个不相交子集 S_1 和 S_2。
- $f(b)$ 为所有划分中，S_1 所有元素的 OR 与 S_2 所有元素的 AND 的和的最大值。

给定一个数组 a。回答 q 次询问，每次计算 $f([a_l, \ldots, a_r])$。

现场情况

首个提交：猪联璧合 @00:27
首个 AC：猪联璧合 @00:35
通过队伍：4

L: OR + AND

有一个显然的性质：S_2 只包含一个元素一定是最优的。这是因为如果 S_2 如果包含多于一个元素，从 S_2 中拿出一个元素放到 S_1 中，会使得 S_1 的 OR 增加或不变，且 S_2 的 AND 增加或不变，一定更优。

L: OR + AND

有一个显然的性质：S_2 只包含一个元素一定是最优的。这是因为如果 S_2 如果包含多于一个元素，从 S_2 中拿出一个元素放到 S_1 中，会使得 S_1 的 OR 增加或不变，且 S_2 的 AND 增加或不变，一定更优。

解法一：

但选择哪个元素放进 S_2 中并不是显然的。这里给出一个结论：放进 S_2 中的元素一定在数组 b 的前 $\log V$ 大的数中，其中 V 是值域范围，也就是前 30 大。

L: OR + AND

有一个显然的性质：S_2 只包含一个元素一定是最优的。这是因为如果 S_2 如果包含多于一个元素，从 S_2 中拿出一个元素放到 S_1 中，会使得 S_1 的 OR 增加或不变，且 S_2 的 AND 增加或不变，一定更优。

解法一：

但选择哪个元素放进 S_2 中并不是显然的。这里给出一个结论：放进 S_2 中的元素一定在数组 b 的前 $\log V$ 大的数中，其中 V 是值域范围，也就是前 30 大。

这样问题就变得简单了：用 ST 表支持询问某个子数组的前 30 大的数，枚举它们作为 S_2 中的元素，最后取最大的答案即可。复杂度 $O(n \log n \log V + q \log V)$。

Fun fact: 一开始是想放带 q 的双 log 过的，但某个 tester 写了个大常数做法，且这种卡双 log 有前科（详见牛客寒假营），最终决定卡掉。但后来又有 tester 写了小常数做法极限过题，再加上放宽了时限，最终还是放过了这种做法（见解法二）。

下面给出结论的严格证明：

对于数组 b，定义 c_k 为 b 中有多少数的第 k 位为 1。在划分子集时，对于每一位考虑两种情况：

- $c_k \leq 1$：无论如何划分，第 k 位对答案的贡献是不变的。
- $c_k > 1$：如果给 S_2 分了一个 1，答案会增加 2^k。

由于最大化低位的答案并不会使高位答案更优，我们可以从高位到低位贪心。维护一个集合 T 表示当前可能分给 S_2 的数。在枚举到第 i 位时，筛选出所有第 i 位为 1 的数。如果不存在则不更新集合。这样最后集合里剩下的数就是 S_2 的最优选。

L: OR + AND

注意到上面的贪心过程实际上和下面的构造是等价的：

定义一个整数 m：如果 $c_k > 1$ 则 m 的第 k 位为 1，否则为 0。将 b 中的所有数按照 $x \,\&\, m$ 排序后得到的最大数即为最优解，如果存在多个 $x \,\&\, m$ 相同的数，取最大的，记为 y。

注意到上面的贪心过程实际上和下面的构造是等价的：

定义一个整数 m：如果 $c_k > 1$ 则 m 的第 k 位为 1，否则为 0。将 b 中的所有数按照 $x \,\&\, m$ 排序后得到的最大数即为最优解，如果存在多个 $x \,\&\, m$ 相同的数，取最大的，记为 y。

由于如果只考虑 m 里的置位，y 是所有数中最大的，别的非最优选数如果想比 y 更大，只能从 m 的非置位上比它大。然而我们对于 m 的定义，规定了 m 的非置位在 b 中只能出现最多一次。所以，最多只会有 $\log V$ 个数比 y 大，结论得证。

总结：大胆猜吧。

解法二（fatalerror）：

对原数组 a，先预处理出第 k 位为 1 的所有下标、以及下一个第 k 位为 1 的下标。这样就可以快速判断区间内哪些下标的 1 只出现过一次，这些只出现过一次的下标数量不超过 $\log V$。

由于只需要求出区间内所有 x 和 m 作 AND 后的最大值（m 的定义同上），因此可以暂时去掉这些只出现过一次的 1，再计算区间最大值，最后还原下标的值。这可以通过单点更新、区间查询线段树实现，复杂度 $O(n \log V + q \log n \log V)$。

C: Large Graph

题意

数轴上有节点 $1, \ldots, n$。每次进行两种操作之一：在两个节点间连边；在一段区间内的所有节点之间连边。每次操作之后需要计算连通点对的数量。要求强制在线。

现场情况

首个提交：Quaternary Search@00:19
首个 AC：猪联璧合 @03:39
通过队伍：1

C: Large Graph

Hint 1: 显然需要维护连通块的大小和数量。

C: Large Graph

Hint 1: 显然需要维护连通块的大小和数量。

Hint 2: 如果 n 不大，直接模拟即可；如果 n 很大且允许离线的话，离散化后模拟即可。n 很大且强制在线时如何解决?

C: Large Graph

Hint 1: 显然需要维护连通块的大小和数量。

Hint 2: 如果 n 不大，直接模拟即可；如果 n 很大且允许离线的话，离散化后模拟即可。n 很大且强制在线时如何解决?

Hint 3.1: 如果不考虑操作 2、只有操作 1 的话，整个过程就是朴素并查集，维护连通块大小及计数即可。

Hint 3.2: 如果不考虑操作 1、只有操作 2 的话，整个过程就是区间合并，使用一个 `set` 等有序结构，$O(\log q)$ 地维护即可，需要一边合并区间一边计算区间长度。

C: Large Graph

Hint 1: 显然需要维护连通块的大小和数量。

Hint 2: 如果 n 不大，直接模拟即可；如果 n 很大且允许离线的话，离散化后模拟即可。n 很大且强制在线时如何解决?

Hint 3.1: 如果不考虑操作 2、只有操作 1 的话，整个过程就是朴素并查集，维护连通块大小及计数即可。

Hint 3.2: 如果不考虑操作 1、只有操作 2 的话，整个过程就是区间合并，使用一个 `set` 等有序结构，$O(\log q)$ 地维护即可，需要一边合并区间一边计算区间长度。

Hint 4: 如果再把操作 1 拆分成先插入两个长为 1 的区间、再对两个区间进行连通这两步，就只需要解决区间连通的问题了。

C: Large Graph

解答： 给每个区间指定一个区间内的点来代表这个区间，不妨令其为区间的左端点。在合并区间的过程中，同时维护这个代表性点的连通性。复杂度 $O(q \log q + q\alpha(q))$。

不妨举一例说明。以下变量中，$n = 7$，$q = 3$，seg 是维护有序区间的 `set`，dsu 是维护代表性点的连通性的并查集，sz 是维护数轴上连通区域大小的哈希表（注意：不是 dsu 中代表性点的连通块的大小）。

初始状态：

1 2 3 4 5 6 7
seg : []
dsu : {}
sz : {}

C: Large Graph

`query: 2 1 3`

$\underline{\mathbf{1}\ 2\ 3}\ 4\ 5\ 6\ 7$
$seg : [[1, 3]]$
$dsu : \{\{1\}\}$
$sz : \{1 : 3\}$

C: Large Graph

`query: 2 1 3`

$\underline{\mathbf{1}\ 2\ 3}\ 4\ 5\ 6\ 7$
$seg : [[1, 3]]$
$dsu : \{\{1\}\}$
$sz : \{1 : 3\}$

`query: 2 5 7`

$\underline{1\ 2\ 3}\ 4\ \underline{\mathbf{5}\ 6\ 7}$
$seg : [[1, 3], [5, 7]]$
$dsu : \{\{1\}, \{5\}\}$
$sz : \{1 : 3, 5 : 3\}$

C: Large Graph

`query: 1 2 6`

$\underline{1\ \mathbf{2}\ 3}\ 4\ \underline{5\ \mathbf{6}\ 7}$

$seg : [[1, 3], [5, 7], [2, 2], [6, 6]]$

$dsu : \{\{1\}, \{5\}, \{2\}, \{6\}\}$

$sz : \{1 : 3, 5 : 3, 2 : 1, 6 : 1\}$

C: Large Graph

`query: 1 2 6`

$1\ \underline{\mathbf{2}}\ 3\ 4\ 5\ \underline{\mathbf{6}}\ 7$

$seg : [[1,3],[5,7],[2,2],[6,6]]$
$\xrightarrow{\text{合并[1,3]和[2,2]}} [[1,3],[5,7],[6,6]]$

$dsu : \{\{1\},\{5\},\{2\},\{6\}\}$
$\xrightarrow{\text{连通1和2}} \{\{1,2\},\{5\},\{6\}\}$

$sz : \{1:3,5:3,2:1,6:1\}$
$\xrightarrow{\text{区间合并}} \{1:3,5:3,6:1\}$

C: Large Graph

`query: 1 2 6`

$\underline{1\ \mathbf{2}\ 3}\ 4\ \underline{5\ \mathbf{6}\ 7}$

$seg: [[1,3],[5,7],[2,2],[6,6]]$

$\xrightarrow{\text{合并[1,3]和[2,2]}} [[1,3],[5,7],[6,6]] \xrightarrow{\text{合并[5,7]和[6,6]}} [[1,3],[5,7]]$

$dsu: \{\{1\},\{5\},\{2\},\{6\}\}$

$\xrightarrow{\text{连通1和2}} \{\{1,2\},\{5\},\{6\}\} \xrightarrow{\text{连通5和6}} \{\{1,2\},\{5,6\}\}$

$sz: \{1:3,5:3,2:1,6:1\}$

$\xrightarrow{\text{区间合并}} \{1:3,5:3,6:1\} \xrightarrow{\text{区间合并}} \{1:3,5:3\}$

C: Large Graph

`query: 1 2 6`

1 **2** 3 4 5 **6** 7

seg : $[[1,3],[5,7],[2,2],[6,6]]$

$\xrightarrow{\text{合并[1,3]和[2,2]}}$ $[[1,3],[5,7],[6,6]]$ $\xrightarrow{\text{合并[5,7]和[6,6]}}$ $[[1,3],[5,7]]$

dsu : $\{\{1\},\{5\},\{2\},\{6\}\}$

$\xrightarrow{\text{连通1和2}}$ $\{\{1,2\},\{5\},\{6\}\}$ $\xrightarrow{\text{连通5和6}}$ $\{\{1,2\},\{5,6\}\}$ $\xrightarrow{\text{连通2和6}}$ $\{\{1,2,5,6\}\}$

sz : $\{1:3,5:3,2:1,6:1\}$

$\xrightarrow{\text{区间合并}}$ $\{1:3,5:3,6:1\}$ $\xrightarrow{\text{区间合并}}$ $\{1:3,5:3\}$ $\xrightarrow{\text{区间连通}}$ $\{1:6\}$

总结： 实现时需要很细心的 Implementation 题目。

B: Firefly's Favourite Problem

题意

给定正整数 N，设其十进制下的长度为 n。另给定一个非零数位 x。对于每个 $0 \le k \le n$，求出在不小于 N 的正整数中，有多少数的十进制表示中 x 的数量为 k。

现场情况

首个提交：slashTeen@00:21
首个 AC：Ray@01:01
通过队伍：1

B: Firefly's Favourite Problem

令 n 为 N 的位数，定义：

$$\mathrm{dp}_{i,j,t} = \left|\left\{ M\colon\ M \text{ 与} N \text{ 前} i \text{ 位的关系由} t \text{ 决定，且已选} j \text{ 个} x \right\}\right|,$$

其中

$$t = \begin{cases} 1, & \text{仍然“紧贴”上界（前} i \text{ 位都等于} N \text{ 的前} i \text{ 位），} \\ 0, & \text{已经松开（小于上界），后续任意.} \end{cases}$$

B: Firefly's Favourite Problem

转移：设当前处理到第 $i+1$ 位，上界数字为 $D = N_{i+1}$。枚举本位取值 $d \in \{0, 1, \ldots, 9\}$，

$$j' = j + [\, d = x \,].$$

$$\mathrm{dp}_{i+1,j',1} \mathrel{+}= \begin{cases} \mathrm{dp}_{i,j,1}, & d = D, \\ 0, & d < D; \end{cases}$$

$$\mathrm{dp}_{i+1,j',0} \mathrel{+}= \underbrace{\mathrm{dp}_{i,j,1}}_{d<D} + \underbrace{\sum_{d=0}^{9} \mathrm{dp}_{i,j,0}}_{\text{任何}d}.$$

最终

$$f(k) = \mathrm{dp}_{n,k,0} + \mathrm{dp}_{n,k,1}$$

。总时间 $O(n \cdot n \cdot 10)$。

B: Firefly's Favourite Problem

将 $\mathrm{dp}_{i,*,0}$ 与 $\mathrm{dp}_{i,*,1}$ 用生成函数封装：

$$T_i(z) = \sum_{j\ge 0} \mathrm{dp}_{i,j,1}\, z^j, \quad L_i(z) = \sum_{j\ge 0} \mathrm{dp}_{i,j,0}\, z^j.$$

增加一位 $D = N_{i+1}$ 后，状态向量 $\big(T_i(z), L_i(z)\big)^{\mathsf{T}}$ 乘以 2×2 多项式矩阵：

$$\begin{pmatrix} T_{i+1}(z) \\ L_{i+1}(z) \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{TT}(z) & \mathrm{TL}(z) \\ 0 & 9+z \end{pmatrix}}_{M_D(z)} \begin{pmatrix} T_i(z) \\ L_i(z) \end{pmatrix}.$$

其中

$$\mathrm{TT}(z) = \begin{cases} 1, & D \neq x, \\ z, & D = x, \end{cases} \qquad \mathrm{TL}(z) = A_0 + A_1\, z,$$

$$A_0 = \big|\{\, d' < D : d' \neq x\}\big|, \quad A_1 = \big|\{\, d' < D : d' = x\}\big|.$$

“loose→tight” 处于零块。

B: Firefly's Favourite Problem

将区间 $[1, n]$ 二分为左右两段，递归构造每段的转移矩阵 $M^{(\ell)}, M^{(r)} \in \mathbb{Z}[z]^{2\times 2}$。合并为整体转移

$$M = M^{(\ell)} \times M^{(r)}, \quad (M_{ij}(z)) = \sum_{k=0}^{1} M^{(\ell)}_{ik}(z) * M^{(r)}_{kj}(z),$$

其中 "$*$" 为多项式卷积（NTT），成本 $O(m \log m)$。

总共 $O(n)$ 个节点，树高 $\log n$，第 h 层处理多项式长度 $\frac{n}{2^h}$，

$$\sum_{h=0}^{\log n} 2^h \, O\Big(\tfrac{n}{2^h} \log \tfrac{n}{2^h}\Big) = O(n \log^2 n).$$

B: Firefly's Favourite Problem

根节点矩阵记为 $\big[R_{TT},\, R_{TL};\, 0,\, R_{LL}\big]$。初始 $\big(T_0(z)=1,\; L_0(z)=0\big)$,

$$T_n(z) = R_{TT}(z), \quad L_n(z) = R_{TL}(z).$$

合法正整数个数多项式为

$$F(z) = R_{TT}(z) + R_{TL}(z) \;-\; 1,$$

其中 "-1" 去掉 $M=0$。输出 $f(k) = [z^k]\, F(z)$ $(0 \le k \le n)$。

总结：半个科技题，转移并不是很难。

M: ABAB

题意

给定一个数组 a，求出 a 中 ABAB 型子序列的数量。

现场情况

首个提交：泥萌好 @01:54
首个 AC：泥萌好 @01:54
通过队伍：1

M: ABAB

解答： 首先有两种暴力做法。

方法一：我们可以枚举 ABAB 中的 A 为哪个元素。在确定了 A 的情况下，采取枚举第二个 A、维护左右的方法。

假设当前枚举的下标为 i，定义 l_x 为 $[a_1, \ldots, a_i]$ 中 x 的数量，r_x 为 $[a_{i+1}, \ldots, a_n]$ 中 x 的数量，c_x 为 $[a_1, \ldots, a_i]$ 中 Ax 子序列的数量。答案即为总贡献

$$S = \sum_i \sum_{a_i \neq A} c_{a_i} \cdot r_{a_i}$$

M: ABAB

当遍历到 i 时，记 $x = a_i$：

1. 更新 c_x：如果 $x = A$，则无需处理；否则 $c_x := c_x + l_A$。
2. 更新 l_x, r_x：$l_x := l_x + 1, r_x := r_x - 1$。
3. 更新 $c_x \cdot r_x$：如果 $x = A$，则无需处理；否则更新 x 对 S 的贡献为 $c_x \cdot r_x$，也就是说 S 是可以 $O(1)$ 更新的。
4. 更新 S。

这样从左扫到右即可求出单个 A 对答案的贡献。由于 a 中可能有 $O(n)$ 个不同元素，总复杂度为 $O(n^2)$。

M: ABAB

当遍历到 i 时，记 $x = a_i$：

1. 更新 c_x：如果 $x = A$，则无需处理；否则 $c_x := c_x + l_A$。
2. 更新 l_x, r_x：$l_x := l_x + 1, r_x := r_x - 1$。
3. 更新 $c_x \cdot r_x$：如果 $x = A$，则无需处理；否则更新 x 对 S 的贡献为 $c_x \cdot r_x$，也就是说 S 是可以 $O(1)$ 更新的。
4. 更新 S。

这样从左扫到右即可求出单个 A 对答案的贡献。由于 a 中可能有 $O(n)$ 个不同元素，总复杂度为 $O(n^2)$。

(Easy Version: ABC318E，本题只需要求子序列 ABA 的数量，可以体会这种维护前后缀乘积之和的方法。)

M: ABAB

方法二：依然是枚举 A 为哪个元素。考虑枚举第一个 A 和第三个 A 的位置，设为 i 和 j。这一对下标对答案的贡献为：

$$\sum_{B \neq A} \left(\sum_{p=i+1}^{j-1} [a_p = B] \right) \cdot \left(\sum_{p=j+1}^{n} [a_p = B] \right)$$

如果 B 在 a 中出现了 v_B 次，那这个方法的时间复杂度为 $O\left(n \sum v_B^2\right)$。

方法二：依然是枚举 A 为哪个元素。考虑枚举第一个 A 和第三个 A 的位置，设为 i 和 j。这一对下标对答案的贡献为：

$$\sum_{B\neq A}\left(\sum_{p=i+1}^{j-1}[a_p=B]\right)\cdot\left(\sum_{p=j+1}^{n}[a_p=B]\right)$$

如果 B 在 a 中出现了 v_B 次，那这个方法的时间复杂度为 $O\left(n\sum v_B^2\right)$。

注意每对下标的贡献类似于对数组中某个区间 $[l,r]$ 的询问，且 $[l,r]$ 转移到 $[l,r+1]$，$[l,r-1]$，$[l+1,r]$，$[l-1,r]$ 都是 $O(1)$ 的。这启发我们把所有询问全部离线下来跑莫队，复杂度为 $O\left(n\sqrt{q}\right)$，其中 $q\sim O(\sum v_B^2)$ 为询问个数。

M: ABAB

注意到第一种方法在 a 中不同元素数量不多的时候复杂度较优秀，而第二种方法在 a 中相同元素对数较少、或者说不同元素数量较多的时候复杂度较优秀。

因此考虑分治：固定一个阈值 B，对于出现次数大于 B 的元素使用第一种方法，否则使用第二种方法。

M: ABAB

接下来进行复杂度分析：B 取多大可以获得最优的时间复杂度?

M: ABAB

接下来进行复杂度分析：B 取多大可以获得最优的时间复杂度?

由于出现次数大于 B 的元素最多有 $\frac{n}{B}$ 个，所以第一部分的时间复杂度为 $O\left(\frac{n^2}{B}\right)$。

M: ABAB

接下来进行复杂度分析：B 取多大可以获得最优的时间复杂度?

由于出现次数大于 B 的元素最多有 $\frac{n}{B}$ 个，所以第一部分的时间复杂度为 $O\left(\frac{n^2}{B}\right)$。

第二部分的复杂度为 $O\left(n\sqrt{\sum v^2}\right) \leq O\left(n\sqrt{\sum Bv}\right) \leq O\left(n\sqrt{nB}\right)$

总体复杂度为：

$$O\left(\frac{n^2}{B} + n\sqrt{nB}\right)$$

M: ABAB

接下来进行复杂度分析：B 取多大可以获得最优的时间复杂度?

由于出现次数大于 B 的元素最多有 $\frac{n}{B}$ 个，所以第一部分的时间复杂度为 $O\left(\frac{n^2}{B}\right)$。

第二部分的复杂度为 $O\left(n\sqrt{\sum v^2}\right) \leq O\left(n\sqrt{\sum Bv}\right) \leq O\left(n\sqrt{nB}\right)$

总体复杂度为：

$$O\left(\frac{n^2}{B} + n\sqrt{nB}\right)$$

取 $B = \Theta(\sqrt[3]{n})$，可以得到 $O\left(n^{\frac{5}{3}}\right)$ 的最优复杂度。实现时由于常数原因，B 可能不会取理论最优值。

M: ABAB

以上并不是本题的唯一做法，三个出题人分别给出了三个分治解法：

- 包包：题解做法。
- error：用树状数组处理小块下的区间相交，$O(n\sqrt{n}\log n)$。
- 核电站：纯分类讨论，$O(n\sqrt{n})$。

不过由于常数和写法等原因，最终用时受复杂度影响有限。

总结：开放性题目。

F: WTF Another GCD

题意

维护一个多重集，其中都是 (v, w) 型的正整数对。需要支持添加，删除，查询当前多重集中满足 $\gcd(v, k) = 1$ 的数对里最大的 w.

现场情况

首个提交：Ray@00:16
首个 AC：-
通过队伍：0

F: WTF Another GCD

解答： 虽然和 E 的题面只差几个字，但难度并不是一个级别的。

在正式开始题解之前，需要一个前置知识：

对于一个多重集 S，定义 $c(i)$ 为 S 中 i 的倍数的个数。考虑一个整数 k，S 中存在与 k 互质的数的充要条件为：

$$f(k) = \sum_{d|k} c(d) \cdot \mu(d) > 0$$

其中 $\mu(x)$ 为莫比乌斯函数。

F: WTF Another GCD

考虑整体二分。

定义函数 $dfs(l, r, ops, qus)$ 为二分区间 $[l, r]$ 里的更新以及询问。首先按照 $mid = \frac{l+r}{2}$ 把所有更新分成两类：$w \le mid$ 和 $w > mid$ 的。接下来按照时间顺序处理所有询问，且只考虑第一类更新。

如果某个询问在只有第一类更新的时候，在 S 内有与它互质的数，则将其递归到 $dfs(mid+1, r, opsr, qusr)$，否则递归到 $dfs(l, mid, opsl, qusl)$。每个询问最多会被处理 $\log n$ 次，每次更新和询问需要 $O(d(n))$ 的时间，总复杂度 $O(n \log nd(n))$。

The End

希望各位在本次比赛中都有学习到知识!