

第二十届北京航空航天大学程序设计竞赛 预赛

题目解析

A 志愿时长

按照题目模拟即可。日期之间的比较实际上可视为字符串之间按字典序比较, 依此实现代码会带来些许便利。

B MAX? MEX?

记可重集 $C_x = \{a_i = x\}$ 。考虑枚举 $\max A = \text{mex } B = k$, 显然地有 $\{0, 1, \dots, k-1\} \subseteq B$, $k \notin B$ 。这表明 C_k 只能划分进 A , $C_{>k}$ 只能划分进 B 。而对于任意 $0 \leq x \leq k$, 由于至少有一个 $a_i = x$ 需要划分进 B , 方案数为 $2^{|C_x|} - 1$ 。注意到 $k = \text{mex } B \leq |B| = n$, 答案即是:

$$\sum_{0 \leq k \leq n} [|C_k| > 0] \prod_{0 \leq x < k} (2^{|C_x|} - 1)$$

对于 $0 \leq x \leq n$ 记录其在给定整数中出现的次数, 并预处理 2 的次幂, 即可在 $O(n)$ 内计算答案。

C Toxel 与 City Walk

已知总共只有 $4 \cdot (t/2 - 1)$ 种不同的矩形, 而且每个矩形在非坐标轴走的路径都是互不相交的。因此, 我们只需要考虑怎么排列这些矩形的顺序, 使得坐标轴上走过的长度最大即可。

我们考虑这样一种构造, 前 4 天依次走这些点构成的矩形: $(1, t/2 - 1)$, $(-1, -t/2 + 1)$, $(t/2 - 1, 1)$, $(-t/2 + 1, -1)$ 。这样前 4 天在坐标轴上新增的长度就分别是 $t/2$, $t/2$, $t/2 - 2$, $t/2 - 2$ 。之后坐标轴都被走过了, 不会再增加长度。在这个基础上加上非坐标轴上的长度 $\min(n, 4 \cdot (t/2 - 1)) \cdot t/2$ 即可得到答案。

时间复杂度 $O(1)$ 。

D Ruler

注意到 $2\sqrt{n} < 2048$, 可以联想到分块取数。设 $B = \lfloor \sqrt{n} \rfloor$, 只需让集合包含 $\{1, 2, \dots, B-1, B, 2B, \dots, (B-1)B, B^2, n\}$ 即可。需要特殊关注 n 为完全平方数的实现, 避免重复。

E 蛇语破译

首先假如 s 原本就有 **she**, 那么:

- 第一遍可以考虑将其中一个 **she** 变成 **she** (其实就是整个串没变)。
- 第二遍可以数一下 **she** 的数量, 如果大于等于 2 个, 那么保持原样输出即可。

原本的串没有 **she** 的时候, 事实上有很多种不同的做法。这里介绍两种。

解法一:

- 第一遍将第一个单词的第二次出现替换成 **she**。
- 第二遍将 **she** 替换成第一个单词。

解法二:

- 第一遍将出现次数最少的单词 (如有多个, 任选其一即可) 替换成 **she**。
- 第二遍将 **she** 替换成此时除 **she** 外出现次数最小的单词。这个单词一定是唯一的。

时间复杂度 $O(n)$ 。

F 蛇球

设颜色 i 对应球的个数为 A_i (包括初始球), 总球数 $S = \sum_{i=1}^n A_i$ 。记状态为当前球颜色 i , 并记从状态 i 到结束需要的期望次数为 E_i 。考虑枚举下一次拿球的颜色 k , 若 $k = i$ 则直接结束, 否则转移到状态 k , 则有:

$$\begin{aligned} E_i &= 1 + \sum_{\substack{1 \leq k \leq n \\ k \neq i}} \frac{A_k}{S-1} E_k \\ &= 1 - \frac{A_i}{S-1} E_i + \sum_{1 \leq k \leq n} \frac{A_k}{S-1} E_k \end{aligned}$$

设 $D = \sum_{k=1}^n \frac{A_k}{S-1} E_k$ 则有:

$$\begin{aligned} E_i &= \frac{1+D}{1+\frac{A_i}{S-1}} \\ \Rightarrow D &= \sum_{k=1}^n \frac{A_k}{S-1} \cdot \frac{1+D}{1+\frac{A_k}{S-1}} \\ \Rightarrow \frac{D}{1+D} &= 1 - \frac{1}{1+D} = \sum_{k=1}^n \frac{A_k}{S-1+A_k} \end{aligned}$$

右侧式子所有量已知, 可以在 $O(n \log n)$ 时间内求出 D , 进而求出开始时手持颜色为 k 的球时的解 E_k 。

总时间复杂度 $O(n \log n)$ 。

Fun Fact: 我们无法得知此处的 More 是否同时有“我们需要更多的蛇!”的含义。

G Astral Quantization

由于 $H_n \sim \ln n + \gamma$, 实际上有 $H_{2 \times 10^5} < 13$ 。注意到 $\max\{a_i\}$ 是答案的下界, 因此 $H_{|S|} \cdot \gcd(S)$ 取得最大值时必有 $\gcd(S) \geq \frac{\max\{a_i\}}{13}$ 。可知 $\gcd(S) \in \{\frac{a_i}{k} \in \mathbb{Z} : 1 \leq k \leq 13\}$ 中, 统计其中各个整数出现的次数, 选择对应的最大量化值即是答案。时间复杂度 $O(n \log n + 13^2 n)$ 。

H 铸剑

先考虑在一条链上时的该问题。

我们可以从起点往终点遍历, 并维护三个变量: 当前遇到的最大品质的剑身 A , 当前遇到的最大品质的剑柄 B , 和当前能打造并洗炼出最高品质的剑的品质 R 。

我们假设在动态规划过程到第 i 个点时, 三个变量的值分别为 A_i, B_i, R_i , 那么转移方程可以写成:

$$\begin{aligned} A_i &= \max(A_{i-1}, a_i) \\ B_i &= \max(B_{i-1}, b_i) \\ R_i &= \max(R_{i-1} + c_i, A_{i-1} + b_i, a_i + B_{i-1}) \end{aligned}$$

可以发现, 转移方程仅仅使用了 $\max$ 和 $+$ 运算。而且, 这个式子是线性的, 于是可以想到与线性变换相关的工具——矩阵。

然后, 我们发现, $+$ 和 $\max$ 均满足结合律, 且 $+$ 对 $\max$ 满足分配律!

那么以 $+, \max$ 替换 $\times, +$ 进行广义矩阵乘法, 我们就实现了利用矩阵来维护状态转移方程。具体地, 可以使用矩阵

$$M_i = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ a_i & 0 & -\infty & -\infty \\ b_i & -\infty & 0 & -\infty \\ -\infty & b_i & a_i & c_i \end{bmatrix}$$

来维护在 i 点处的转移方程。初始状态的矩阵为:

$$S = \begin{bmatrix} 0 \\ -\infty \\ -\infty \\ k \end{bmatrix}$$

其中 k 是初始携带的剑的品质。

那么, 答案便是:

$$\left(\left(\prod_i M_i \right) S \right)_{4,1}$$

(由于矩阵乘法不满足交换律, 需注意矩阵摆放的顺序从右到左为 u 到 v)

回到树上, 我们要求的问题转化为了树上一段路径的积。这是一个经典问题, 可以考虑树剖后使用线段树维护 (注意由于矩阵乘法不满足交换律, 需要维护两侧的积), 这个算法是 $O(n \log n + q \log^2 n)$ 的。

如何优化?

注意到, 矩阵中有许多没必要记录的常数。而由前面的分析, 解决该问题最重要的性质是整体的结合律。因此, 只要我们的合并运算满足结合律, 这个问题就能正确解决。故我们可以只存个位数个变量, 从而优化掉矩阵乘法所带来的大常数。

由于该题不用实现修改操作, 我们也可以使用倍增维护这些有限个数的变量, 这进一步将算法优化到 $O(n \log n + q \log n)$ 。

本题的核心在于对题中转移式整体的结合律的应用, 因此, 以上三种做法均可以在时限内通过。

Fun Fact: 本题的 5.5s 时限本来是为了放过测试时近 5s 的 4×4 矩阵做法的, 但是由于这个评测机过于快, 实际上在赛时放过了 5×5 的超大矩阵。从题目考察目的上讲, 这确实也是可以通过的正确解法。

I Euler

对于一张联通无向图, 若度数为奇数的点数 $k = 0$ 或 $k = 2$, 可以一笔画完; 若度数为奇数的点数 $k \geq 4$, 则需要 $\frac{k}{2}$ 笔画完。证明可以参考[此文档](#)。对于每次询问, 只需求出删边后图中所有联通块的 $\max\{\frac{k}{2}, 1\}$ 之和即可。如果仅有加边操作, 可以使用并查集方便地维护每个联通块的答案。

对于本题中的删边操作, 可以离线下来转化为加边操作, 利用分治算法和可撤销并查集计算答案。如果一条边在第 $q_1, q_2, \dots, q_k$ 次询问中被删除, 就相当于在 $[1, q_1 - 1], [q_1 + 1, q_2 - 1] \dots, [q_k + 1, q]$ 次询问中被包含。使用线段树进行区间插入, 总共只会插入 $(\sum_{i=1}^q k_i) \log q$ 条边。插入完毕后, 在线段树上递归求解答案, 每遍历到一个结点, 加入覆盖当前结点区间的所有边, 递归左右子节点, 结束后撤销加边操作。为了让并查集支持撤销, 不能路径压缩, 需要保存子树大小按秩合并。总时间复杂度为 $O((\sum_{i=1}^q k_i) \log q \log n)$ 。

本题也可以不使用线段树, 直接对询问分治, 只需在递归一侧时加入所有该侧询问未删除, 另一侧询问删除的边即可。

J GGEZ

树上颜色问题直接考虑树上莫队。把树用欧拉序摊平成序列之后, u 到 v 的路径上点, 就对应于欧拉序上 u 到 v 区间内所有只出现了一次的点。这个限制再加上区间数颜色, 自然考虑用莫队处理: 维护点的出现次数、出现 1 次的点的颜色桶、颜色桶的频次桶。莫队需要挪动 $O(n\sqrt{n})$ 次指针, 回答询问需要 $O(q)$ 次查询颜色桶的频次桶的区间和。于是直接再拼上一个值域分块来维护颜色桶的频次桶, 支持 $O(1)$ 的修改和 $O(\sqrt{n})$ 的查询, 就能做到总复杂度单根号 $O((n + q)\sqrt{n})$ 。

由于实在不想放过 $O((n + q)\sqrt{n} \log n)$ 的做法, 所以时限设的比较紧, 不过依然是 std 的三倍。如果确实被卡常了, 可以尝试调整一下块长或者加奇偶块排序优化。

K Forgotten

解法 1: 点分治

使用点分治计算每条路径的贡献。一条端点为 x, y 的跨过分治中心的路径, 会让路径上每条边的答案与 $t_x \oplus t_y$ 取 $\max$ 。对于当前分治中心, 建一棵 01-trie 树, 插入所在连通块的全部点权。单独考虑与分治中心相连的每棵子树, 先将子树内点权从 trie 中删除, 然后递归子树, 对于每个结点在 trie 上查询跨过分治中心路径端点的异或和最大值, 自底向上传递最大值, 更新每条边的答案。递归更新完, 再将子树内点权加回 trie 树。每个结点至多只会被 $\log n$ 个分治中心访问, 每次访问仅含常数次 trie 树操作, 时间复杂度为 $O(n \log n \log \max\{t\})$ 。

解法 2: 树链剖分 + dsu on tree

类似本场比赛的 I 题, 对每条重链分治求解即可, 时间复杂度为 $O(n \log^2 n \log \max\{t\})$, 感兴趣的同学可以思考实现。

L Serval 与 Toxel 与 popcount

对于所有满足前两个条件的数组, 我们首先考虑条件 3 中最小的异或和是多少。需要注意到: 不同位之间是不可区分的, 因此我们只需关注异或结果中的 1 的个数 (即 popcount)。最小的答案必然满足所有 1 都集中在最低的若干位上。

此外, 还可以发现所有可能的 popcount 取值构成一个连续区间, 但是只能取其中的奇数或偶数。这一点通过归纳法很容易证明。

接下来, 我们尝试构造字典序最小的解。设 cur_1 是刚刚计算出的最小异或和。我们尝试为 $a_1, a_2, \dots, a_n$ 赋值, 使得:

$$\text{cur}_1 \oplus a_1 \oplus a_2 \oplus \dots \oplus a_n = 0$$

一个自然的想法是逐个确定 a_i 的值。但是需要注意, 在给 a_i 赋值时, 必须确保余下的 $a_{i+1}, \dots, a_n$ 仍然可以取某些值使得最终异或和为 0。在所有可能的 a_i 中, 我们需要取最小的那个。

假设我们已确定 $a_1, \dots, a_{i-1}$, 记当前前缀异或和为:

$$\text{cur}_i = \text{cur}_1 \oplus a_1 \oplus \dots \oplus a_{i-1}$$

假设我们已经知道后缀 $a_{i+1} \oplus \dots \oplus a_n$ 的 popcount 可取区间为 $[L_i, R_i]$, 且为与 L_i (或 R_i) 同奇偶性的任意整数。并且这些 1 可以分布在任意位上 (即位置无限制)。

换言之, 我们只需保证 $\text{cur}_i \oplus a_i$ 的 popcount 落在 $[L_i, R_i]$ 内。

进一步思考, 设当前 cur_i 的某位为 0, 我们将 a_i 在该位置为 1 时, 会使异或结果该位变为 1, 从而 popcount 增加 1; 反之, 若 cur_i 某位为 1, 将 a_i 在该位置为 1 会使异或结果该位变为 0, popcount 减少 1。因此, 若我们在 cur_i 目前为 0 的位中选了 x 个位置置为 1 (即 a_i 在这些位置为 1), 则异或后的 popcount 为:

$$\text{popcount}(\text{cur}_i) + 2x - p_i$$

该值必须落在 $[L_i, R_i]$ 内, 且同时需满足约束:

- $0 \leq x \leq p_i$

- $x \leq m - \text{popcount}(\text{cur}_i)$
- $p_i - x \leq \text{popcount}(\text{cur}_i)$

通过以上不等式可以解出 x 可以取值的区间。

进一步, 我们发现一个有趣的性质: 在固定的 p_i 下, a_i 的值关于 x 是一个凸函数 (证明留作练习)。此外显而易见的是: 当 a_i 的所有 1 都集中在最低若干位时, 其数值最小。因此不需要使用三分等复杂的做法, 只需要优先尝试 “最低位全为 1” 这种解。若求得 popcount 不在 $[L_i, R_i]$ 内, 则取可行区间的端点 (即最小或最大允许的 x) 即可。

我们可以用一棵线段树来维护 cur 。

- 假如 a_i 直接在最低位就满足要求, 那么线段树只需要支持一下区间异或。
- 否则这个操作一定是对最低若干位异或 1, 再将中间某一个区间全部置 0 或置 1。读者可以自己手画几个例子加深理解。

时间复杂度为 $O(n \log m + m)$ 。

M Popotnik -The Traveller of Ljubljana-

Fun Fact: This problem was proposed as Div. 2D but was rejected for being “not so interesting”.

首先分连通块考虑:

- 若连通块中仅包含一个字符为 a 的结点, 那么这个连通块仅能贡献长度为 1 的字符串 a ;
- 若连通块中包含两个字符分别为 a 、 b 的结点 (当然, 字符可以相同), 那么这个连通块可贡献两个长度为 l 的字符串 $abab\dots$ 与 $baba\dots$;
- 若连通块中包含多于两个结点, 那么这个连通块可贡献至少 $2^{l/2}$ 个长度为 l 的字符串。

基于此观察, 可贪心地确定答案的前缀, 再利用动态规划根据字典序次序确定答案最后 $2 \log k$ 个字符。具体地:

- 除去末 $2 \log k$ 个字符, 前缀必然为某两个字符 i 、 j 的循环;
- 若 $l \leq 2 \log k$, 直接动态规划即可;
- 若不然, 按字典序从小到大枚举 i 、 j :
 - 若存在一条边连接字符分别为 i 、 j 的结点, 且这条边所在连通块有多于两个结点, 则前缀为这两个字符的循环;
 - 若边存在但所在连通块仅有两个结点, 将其计入对字典序次序的贡献。
- 标记 i 、 j 循环 $l - 2 \log k$ 次后所在的结点, 从这些结点出发动态规划;
- 特殊地, 若多于一条连接字符分别为 i 、 j 的结点的边形成连通块, 可知这个连通块可贡献至少 $2^{l/2}$ 个长度为 l 的字符串 $ijij\dots$, 即可直接确定答案。

实现时可取 $2 \log k \geq 128$ 。时间复杂度 $O(\sum |\Sigma|^2 + l + (n + m) \log k)$ 。