2025 ICPC 国际大学生程序设计竞赛亚洲区域赛（上海站）

清华大学

November 26, 2025

概况

预期难度：$\mathrm{D} < \mathrm{GH} < \mathrm{BIJ} < \mathrm{AE} < \mathrm{KLM} < \mathrm{CF}$
实际难度：$\mathrm{DH} < \mathrm{AGJ} < \mathrm{BI} < \mathrm{EK} < \mathrm{FLM} < \mathrm{C}$

D

第一位有 01? 三种情况，0 的情况和 1 的情况分别对应一个 $n-1$ 维的子问题，可以递归处理，递归处理后可以得到一个 3^{n-1} 大小的数组，表示子问题的答案。? 的情况可以看成 0 和 1 的结果相加。我们把两个 3^{n-1} 的数组拼起来，然后把他们对位相加的和拼在后面即可得到当前问题的答案。

时间复杂度 $\sum 2^i 3^{n-i} = O(3^n)$。

G

首先证明存在组数不超过 2 的最优分组方式：若最优分组方式有至少 3 组，且答案的某一位为 1，则所有组价值的对应位均为 1，此时选择其中任意三组，将前两组中的所有数合并到最后一组，则易得答案的对应位依然为 1，不会使答案变劣。

1. 最高位为 1 的数字数量为奇数：将所有数分为一组即可。
2. 最高位为 1 的数字数量为偶数：需要将所有数分为两组。此时原问题等价于最大异或子集，使用贪心线性基即可解决。

H

结论：

将所有出现次数为偶数的数两两配对，每一对选一个数，假设这些数的 XOR 为 X。

Case 1: 若未配对的数 $= 0$ 个，先手必胜当且仅当 $S = 0$。

Case 2: 若未配对的数 $= 2$ 个，先手必胜当且仅当这两个数中有一个为 X。

Case 3: 若未配对的数 > 2 个，后手必胜。

H

证明：

Case 1: 先后手都可以强制使得每一对中恰好一人一个，所以 $S = 0$ 时先手可以执行这个策略，其余情况后手可以执行这个策略。

Case 2: 若有 X 则先手直接拿。然后转化为 Case 1 的情况。

Case 3: 后手可以直接强制已配对的数一人一个，对于其余的数，当只剩下三个时，其中至多有一个可以使先手获胜，可以直接拿。

B

若图为二分图，因此走出来的环一定为偶数个点，若所有点点权的异或和不为 0 就寄了。

若不为二分图，可以先尝试走一个奇环将所有点点权的异或和变为 0。之后，找出一棵生成树进行 dfs，如果当某个点出栈后这个点点权是 1，就再走一圈这个点和父亲之间的边将他变为 0。因为树是二分图，所以走出来的一定是偶环，过程结束后异或和一定保持为 0，从而保证了根的权值为 0。

仅考虑保留下的数，容易发现如果一个数被修改，则修改为前一个数一定不劣。

可以得到这样一个 DP：

$f_i = \min(f_j + (a_i \oplus a_j) + (j - i - 1)C)$。

直接做是 $O(n^2)$ 的，一个优化方法是对准这个交叉项 $a_i \oplus a_j$。令 g_x 表示 $a_i = x$ 的所有 i 的 $f_i - (i+1)C$ 的最小值。转移时我们用 $f_i + k - (i+1)C$ 转移到所有 $g_{a_i \oplus k}$，其中 $0 \le k < 2^9$，然后从所有 $g_{a_i \oplus (2^9 k)} + 2^9 k + iC$ 转移到 f_i。

令 V 为值域，总复杂度为 $O(n\sqrt{V})$，由于本题的代价的特殊性，可能存在其他分析性质得出的相同复杂度的做法。

J

可以发现一个结论：如果某个路径在答案里，它的所有前缀都在答案里，因此我们可以通过在 trie 树上遍历枚举答案（根据结论，使用到的 trie 点数是 $\mathcal{O}(k)$ 的）。

于是我们初始把每条边加入 trie，按照字典序遍历整棵 trie，每当我们输出完该结点的答案，就往后走一步扩展其后继（可以发现是加新的叶子，字典序一定更大）。

这样直接模拟可能会扩展出 $\mathcal{O}(nk)$ 个状态，并不能通过。我们对 trie 二度化，每次扩展时只扩展字典序最小的后继，以及其前驱的下一个后继，可以发现这种扩展方式仍不重不漏，但此时只会扩展出 $\mathcal{O}(n+k)$ 个状态，复杂度 $\mathcal{O}(n+k)$（默认边权为常数）。

A

Sol1:

先用 $\log_2(n)$ 次问出深度。

然后维护剩余点集的重心，每次问重心的重儿子，由于是二叉树，这个 size 一定 $> \frac{1}{3}n$，所以每次一定能删去 $\frac{1}{3}$ 的点，总次数 $\log_{1.5}(n)$。

理论次数 $\log_2(n) + \log_{1.5}(n) \leq 40$。但仔细想想发现两部分不可能同时跑满，第二部分由于深度相同的限制也跑不满，最终比较精细的实现能做到 20 次以内。

A

Sol2:

这是一个随机化做法。

每次随机一个可能是答案的点，然后找一个 d，使得尽可能把剩余的点分成两份相同大小的。

实测 $20+$ 次。一个直觉上的证明是，假如修改成先问出深度，那么期望确实是 $O(\log n)$ 的，有点类似随机链剖分，很大概率能问到重儿子，然后所有块都不是很大。

E

本题在验题中出现了 5 个做法。分别为：模拟，压位 SA，哈希 + 随机化 + 线段树剪枝，哈希 + 分类讨论，哈希 + 分块。除了最后一个做法复杂都正确，并且都能通过。

E

考虑增量构造。本质上是要找到一个 $p_i < i$ 使得 $popcount(G_{p_i} \oplus G_i) \leq k$。考虑到这个题目的性质。令 p_i 为 i 的父亲，那么会有一个树。满足每个点和每个父亲的对称差不会太大。

于是可以考虑在原树上 dfs，时刻维护和 G_i 的对称差，每次递归和回溯的时候只有 $O(k)$ 个位置会变化，因此只需要考虑这些位置的改变。因此总时间复杂度 $O(n^2k)$。

K

三种做法 $\mathcal{O}(n\sqrt{n\log n}), \mathcal{O}(n\sqrt{n}), \mathcal{O}(n\log^3 n)$，出题人各实现了一份分别是 2.5s / 4.2s / 1.6s，开 6s 希望不会有选手被卡常（同时也希望能卡掉暴力）。

K

分块，每次操作散块都可以暴力处理，故只需考虑整块。

前面的元素对该块的影响只有块前所有元素的最小值 u，最大值 v，该块 $[l, r]$ 的贡献即 $\sum_{i=l}^{r} \min(\min_{j=l}^{i} a_j, u) \times \max(\max_{j=l}^{i} a_j, v)$.

由于前缀最小值、前缀最大值都单调，对 u chkmin 的一定是一段前缀，对 v chkmax 的同样是一段前缀，分段点可以通过在前缀最小值、最大值数组上 lower bound 求得。故该求和可以分为三段（同时被 u, v 更新，仅被一侧更新，两者都无法更新），每一段的贡献都可以在预处理一些前缀和后 $\mathcal{O}(1)$ 求出。

K

$\log$ 的瓶颈仅在于 lower bound，由于只有 $\mathcal{O}(n)$ 次块的重构，我们把所有可能的块都拿出来，再把对应的 lower bound 查询提前基数排序后挂在上面，然后重新跑一遍上述算法，就可以把 lower bound 改为归并排序，这样时空复杂度均为 $\mathcal{O}(n\sqrt{n})$，空间复杂度过大无法通过。

实际上我们也可以做到 $\mathcal{O}(n)$ 空间，根号重构就行了，因为这样每次就只用保存 $\mathcal{O}(\sqrt{n})$ 个块，常数不小，应该也可以轻松通过。

K

线段树，结点 $[l, r]$ 上维护这些信息 $\sum_{i=l}^{r}(\min_{j=l}^{i} a_j) \times (\max_{j=l}^{i} a_j), \sum_{i=l}^{r}(\min_{j=l}^{i} a_j), \sum_{i=l}^{r}(\max_{j=l}^{i} a_j)$.

后两者是经典的单侧递归线段树维护的问题，前者也可以类似处理，我们设计函数 ‘compute(nd,l,r,u,v)‘（线段树结点 nd 对应的区间即 $[l, r]$），计算 $\sum_{i=l}^{r} \min(\min_{j=l}^{i} a_j, u) \times \max(\max_{j=l}^{i} a_j, v)$。

K

我们假设左子树的最小值、最大值为 p, q，我们有：

- $u < p, v \geq q$：左子树直接被推平，只用在右子树继续 `compute`；
- $u \geq p, v < q$：手上的 u, v 对右子树无影响，可以通过预处理的“左对右影响”快速计算右子树答案，只用在左子树继续 `compute`；
- $u < p, v < q$：左子树最小值被推平为 u，递归到计算 $\sum_{i=l}^{r} \max\left(\max_{j=l}^{i} a_j, v\right)$，右子树继续 `compute`；
- $u \geq p, v \geq q$：同理左子树计算 $\sum_{i=l}^{r} \min\left(\min_{j=l}^{i} a_j, v\right)$，右子树继续 `compute`。

K

修改和查询都是线段树模板，调用 $\mathcal{O}(\log n)$ 次 `compute`，每次 `compute` 遍历 $\mathcal{O}(\log n)$ 个结点，每个结点也可能花费 $\mathcal{O}(\log n)$ 的时间计算前缀最小值/最大值之和，复杂度 $\mathcal{O}(n \log^3 n)$。

L

先考察一个序列 A 能否变成另一个序列 B。

考察最大和最小的相对关系，我们可以判断最初是否需要一次交换操作。

接下来我们选择一个位置断开。显然要求左右两段中，两个序列其中的元素集必须相同。

我们钦定每次断开必须选择最靠左的能断的位置。

即假如选择了位置 x 断，必须满足 $\forall 1 \le y < x, \{A_1, \cdots, A_y\} \neq \{B_1, \cdots, B_y\}$。这等价于位置 $\{1, 2, \cdots, x-1\}$ 不能被断开。

在我们做一次交换后，设最大最小的位置分别为 $p < q$。我们将区间 $\{p, p+1, \cdots, q-1\}$ 重新设置为可以断开。

L

于是有这样的 dp：设 $f_{l,r,s,t}$ 表示 $[l,r]$ 这个区间，元素变化的情况是 s（没变化/最大的变成最小/最小的变成最大），且不能被断开的元素集合为 t。我们枚举是否交换，再枚举断开的位置。

可以证明 t 只有 $O(1)$ 种情况。即：

1. 全都不能断
2. 全都可以断
3. 最大值右边的位置不能断
4. 最小值右边的位置不能断

复杂度 $O(n^3)$ 可以通过。

M

对于一个固定的树结构，一个点如果有俩叶子儿子，则俩叶子权值必须不同，若权值相同，则我们不关心根到当前节点的异或和。一个点如果有一个叶子儿子，则我们同样不关心根到当前点的异或和，一定有一种方案的填写使得合法。

因此在合法的前提下，方案数为 2^{cnt_0}，其中 cnt_0 为两个儿子都不是叶子的节点个数。类似的，也可以写成 2^{cnt_2-1}，cnt_2 为两个儿子都是叶子的节点个数（此时已经满足了合法的限制，因此这些点的两个叶子儿子权值都不同）。

M

考虑容斥，选择一些相邻相同的点令他们合并，并乘上 $(-1)^p$ 的系数。注意到 2^{cnt_2} 可以看成随便钦定一个 cnt_2 的子集，其他随便选。则我们可以随便选任意一个子集使得他们相邻，若相等则乘 -1 的系数，否则乘 1 的系数，然后剩下的随意合并。注意到随意合并即卡塔兰数。于是我们只需计算有多少选出 i 个相邻的方案。于是我们可以在 $O(n\log^2 n)$ 的时间内在用分治算法完成此题。

C

关键性质：排序排列和排序 01 串类似。

可以把 $> \frac{n}{2}$ 的视为 1，$\leq \frac{n}{2}$ 的视为 0。如果我们能排序这个 01 串，那我们在排序之后执行一次 FakeSort(1,n)，就可以排序排列。

先考虑 n 足够大的 Case（事实上，$n \geq 8$）。唯一无解的 Case 是 01010101 这一类 01 交错的串。这时所有长度为偶数的区间都有一半 0 一半 1，排序都是无效的。

C

我们考虑一下这一个操作对 01 串的影响，若其中 $0,1$ 的个数差为 $2k$。假如我们先执行一次 FakeSort(1,n)，那所有 $0,1$ 内部都是有序的，我们实际上是把将后 k 个 0 移到前 k 个 1 的位置。

于是我们可以得到一个排序 01 串的方法，假如排序区间为 l,r，可以有一个函数 Sort(l,r)。若 0 比一多，我们可以一直 FakeSort(1,n)，Fakesort(l,r)，然后删去末尾连续的偶数个 1。

可以发现每做一轮这个操作，01 的个数差会翻倍，所以 $\log$ 次可以完成排序。

C

一个做法是这样的，找到一个前缀和绝对值为 2 且包含至少两个 0 的前缀 i，执行 Sort(1,i)，然后执行 Sort(3,n)。

如果没有前缀和绝对值为 2 的，则一定有 1 和 -1，可以找其中最远的一对排一个序。可以造一个前缀和绝对值为 2 的。($n = 6$ 时不一定找得到，这个 case 在样例里有所体现)。

调用一次 Sort 函数需要 $2 \log n$ 次，上述做法需要 $4 \log n + O(1)$ 次。

C

一个细节更少的写法是，找到一个前缀和绝对值为 2 的前缀，若 $i \le n/2$ 则 Sort(i+1,n)，然后 Sort(1,n-2)，否则按照原做法，这样可以避免第一次 Sort 区间过短带来的问题。并且仔细分析可以发现第一次排序之后一定有很长的连续段，大小至少 $n/4$，删掉这样第二次 Sort 开始时 k 就很大，实际次数是 $2\log n + O(1)$。题目开 114 是为了放可能存在的不同做法。

F

这个题目是被削过一版的，原题是：

平面上有 n 个线段，每个线段都过 x 轴或 y 轴，线段相交是联通，请你把每个连通块求出来。

$n \leq 2 \times 10^5$。

本题是这个原题分治后需要求的东西。

F

对于每个询问分别考虑 $x < x_0$ 和 $x > x_0$ 的两类线段。

因此，考虑对 x 扫描线两次，然后维护凸包，查询在凸包上二分即可。

要特殊处理坐标为 0 的情况。

时间复杂度 $O((n+q)\log n)$。