WC2018 模拟赛 题解

Niro BC

January 27, 2018

Problem A : 诗歌 (poem)

Subtask1 : 19 分

直接枚举 i,j,k 并判断，时间复杂度 $O(N^3)$。

Subtask1,2 : 19+22=41 分

记下每个高度出现的位置，枚举 i,j，并判断 $2H_j - H_i$ 所在的位置是否大于 j，时间复杂度 $O(N^2)$。

100 分

从左往右枚举 j，对于当前 j，判断此时是否存在 (i, k) 使得 $H_i + H_k = 2H_j$ 且 $i < j < k$。

怎么判断呢？如果存在 (i, k) 满足 $H_i + H_k = 2H_j$ 且 i 和 k 在 j 的不同侧，则发现一组可行的 (i, j, k) 或 (k, j, i)。那么我们在从左往右枚举 j 的过程中，维护一个长度为 N 的 01 序列 s，$s_x = 1$ 表示高度为 x 的山峰在位置 j 的左侧。如果此时 s 的以第 H_j 个字符为中心的长度为 $2min(H_j - 1, N - H_j) + 1$ 的子串（即以 H_j 为中心的极长子串）不是回文串，则存在 (i, k) 使得 H_i 与 H_k 关于 H_j 对称，且 i 和 k 在 j 的不同侧（$s_{H_i} \neq s_{H_k}$），那就找到了一个可行的三元组 (i, j, k) 或 (k, j, i)。

判断 01 序列的子串是否为回文串可以用线段树维护 01 子串的 hash 值和子串翻转后的 hash 值，总时间复杂度 $O(N \log N)$。

Problem B：猫咪 (cat)

Subtask1：5 分

区间 DP。

设 $F_{L,R}$ 为当 $S_1 = T_{L...R}$ 时 K 的最大值

$$F_{L,R} = \max_{T_{l...r}\text{是}T_{L...R}\text{的双子串}} F_{l,r} + 1$$

由于 S_1 只需是 T 的子串，输出 $\max_{1\le L\le R\le N} F_{L,R}$ 即可。

时间复杂度是个关于 N 的多项式，几次不重要，范围 $N \le 50$，只要是正常写法都能过。

Subtask1,2：5+24=29 分

定义一个字符串 s 的 border 为最长的不是其本身的既是其前缀又是其后缀的字符串，如 $border(\text{'abazaba'}) = \text{'aba'}$，$border(\text{'abcabcab'}) = \text{'abcab'}$。

我们可以强制对于所有 $2 \le i \le K$，$S_i = border(S_{i-1})$。这样不会更劣，因为我们可以从 S_{K-1} 至 S_1 不断缩短每一个串，使得所有 S_{i+1} 都是 S_i 的 border，于是我们便得又到了一组 K 相同的符合强制条件的方案。

同学们应该都知道 KMP 算法，KMP 算法可以在 $O(N)$ 的时间内找到一个长度为 N 的字符串的每个前缀的 border。

我们定义一个字符串的深度为

$$dep(s) = \begin{cases} 0, & s \text{ 是空串} \\ dep(border(s)) + 1, & s \text{ 不是空串} \end{cases}$$

显然，$K = dep(S_1)$，又因为 S_1 是 T 的子串，我们要求的 $\max\{K\} = \max_{1\le L\le R\le N} dep(T_{L...R})$。

我们可以对 T 的 N 个后缀各跑一遍 KMP，这样就求出了每个后缀的每个前缀（即所有子串）的 border，自然也求出了它们的深度。由于跑了 N 遍 KMP，时间复杂度为 $O(N^2)$。

Subtask3：17 分

这个 Subtask3，你只要会了 Subtask2 就肯定能想到啦，特殊限制就是说，你只需考虑 S_1 的左端点是 1 的情况，即只用对 T 跑 1 次 KMP，时间复杂度 $O(N)$。

100 分

我们定义一个字符串 s 是有用的，当且仅当它满足下列 2 个条件之一：

1. s 是单个字符。
2. $border(s)$ 是有用的，并且 $border(s)$ 仅在 s 中以前缀和后缀出现恰好 2 次，比如 'abaca' 和 'abaxabayaba' 就不是有用的。

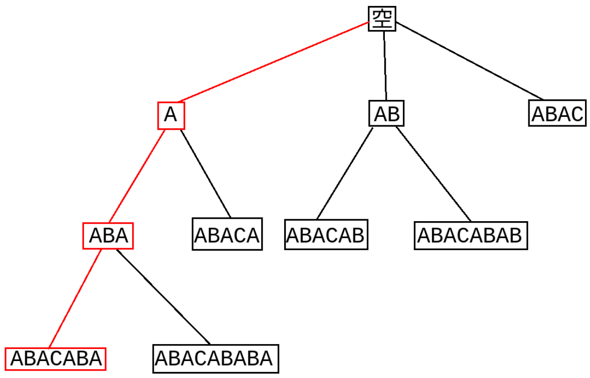
对于一个不是有用的字符串 s，一定存在一个长度小于 s 的有用的 s 的子串 s' 使得 $dep(s) = dep(s')$，所以答案为

$$\max_{s\text{为}T\text{的子串且}s\text{有用}} dep(s)$$

接下来，我们要找到 T 的所有本质不同的有用的子串并求出它们的深度。别怕，接下来会证明本质不同的有用的子串恰好为 N 个。

直接做这个不好做，我们考虑给定一个字符串 A，问 A 的哪些前缀是有用的。设 $fail_i$ 为 $border(A_{1...i})$ 的长度，如果对于一个前缀 $A_{1...i}$，存在 $j < i$ 使得 $fail_j = fail_i$，那么 $A_{1...i}$ 就一定不是有用的。因为 $border(A_{1...i}) = A_{1...fail_i} = A_{j-fail_i+1...j} = A_{i-fail_i+1...i}$，$border(A_{1...i})$ 在 $A_{1...i}$ 中出现了至少 3 次，不符合我们对"有用"的定义。

举个例子吧，ABACABABA 这个字符串，建出一棵 fail 树（节点 i 的父亲是 $fail_i$ 的树，每个节点代表一个前缀），如下图：



可以发现，从根节点（空）开始，每次朝当前点最短的儿子走，直到叶子，途经所有前缀是有用的（本例子中是 A，ABA，ABACABA），其他前缀都是没用的。另外，有用的前缀中，有且只有那个叶子（最长的有用的前缀，本例子中是 ABACABA）不是 $A_{2...|A|}$ 的子串。其他的有用的前缀，由于需要是更长的有用的前缀的后缀，就必须在后面的 $A_{2...|A|}$ 中出现。

这给我们一开始的问题"找出 T 的所有本质不同的有用的子串"带来了启发。我们从右往左枚举要找的有用的串的左端点 L，寻找 $T_{L...N}$ 的前缀中有哪些有用的串。有用的串的 border 一定是 $T_{L+1...N}$ 的子串，而我们是从右往左枚举左端点的，这意味着 $T_{L...N}$ 的有用的前缀的 border（有用的串的 border 也是有用的）都已经被我们找到过。找到已发现过的有用的串中最长的是 $T_{L...N}$ 的前缀的串，设其在 $T_{L+1...N}$ 中第一次出现为 $T_{x...y}$，这样我们就发现了一个新的有用的串 $T_{L...y}$，这是 $T_{L...N}$ 的前缀中最长的有用的串，也是唯一一个**新发现**（唯一一个不是 $T_{L+1...N}$ 的子串的有用的串，不与任何已发现的有用的串相同）的串。特殊地，如果不存在一个已经发现的有用的串是 $T_{L...N}$ 的前缀，T_L 单个字符就是一个新发现的有用的串。由于在每个左端点处，我们都会新发现**恰好** 1 个有用的串，所以本质不同的有用的串的数量恰好为 N。

在从右往左枚举左端点的过程中，需要进行的操作有：

1. "发现"一个有用的子串。
2. 找到已发现的最长的是 $T_{L...N}$ 的前缀的子串，求出其在 $T_{L+1...N}$ 中第一次出现的位置。

这些操作可以用**后缀数组上二分 + 线段树**或**后缀树上的子树/链赋值查询**（还是要做子树/链操作，还是要用线段树，哈哈哈哈哈哈）来完成，时间复杂度 $O(N \log N)$。

Problem C：花朵 (flowers)

Subtask1,2：7+15=22 分

送分点。

定义 $size(p)$ 为点 p 的子树中（包括 p）点的个数。

设 $F_{i,j}$，$G_{i,j}$（$0 \le j \le size(i)$）分别代表在 i 的子树中恰好取了 j 个城市，没有一条边的两个端点同时被取，点 i 本身是否取（F 取 G 不取）的所有方案的取的点权的积的和。

设多项式 $F_i = \sum_{j=0}^{size(i)} F_{i,j}x^j$，那么有转移

$$F_i = B_i x \prod_{v \in child(i)} G_i$$

$$G_i = \prod_{v \in child(i)} (F_i + G_i)$$

注意到将两个度数为 D_1 和 D_2 的多项式暴力相乘的复杂度为 $O(D_1D_2)$ 而非 $O((D_1+D_2)^2)$，所以在计算 F_p 和 G_p 时花费的时间正比于以 p 为 LCA 的点对的个数，所以总时间复杂度为 $O(N^2)$。

我为什么要弄两档 500 和 4000 的部分分？因为有的同学可能没有发现自己写的 $O(N^3)$ 做法其实是 $O(N^2)$ 而不敢交题，导致此题爆零，善良又温暖的出题人不希望这样的情况发生。

Subtask3：15 分

和 Subtask1,2 一样的做法，唯一的不同就是你不须记下 F_p 和 G_p 中 $> M$ 次的项，总时间复杂度 $O(NM^2)$。

Subtask4：18 分

数据是一条链。

对于链上的一段区间，我们需要求出“左右都不选”，“左选右不选”，“左不选右选”，“左右都选”对应的多项式（多项式的第 j 项代表选出恰好 j 个点时所有合法方案的点权的积的和）。那可以分治求出左右两半各自的四个多项式并合并，合并的过程可以用 FFT 做 $O(1)$ 次多项式乘法，时间复杂度 $T(N) = O(N \log N) + 2T(N/2) = O(N \log^2 N)$。

Subtask5：20 分

数据是一朵菊花。

当选中心时，别的点都不能选，仅在 $M = 1$ 时特殊考虑。

当不选中心时，问题转化成了“在 $N-1$ 个数中任选 M 个数，求所有选的方案的选中数字的积的和。”，相当于要求 $\prod_{i=2}^{N}(x+B_i)$ 这个多项式的 M 次项。

依旧考虑分治，多项式 $\prod_{i=L}^{R}(x+B_i)$ 可由 $\prod_{i=L}^{mid}(x+B_i)$ 和 $\prod_{i=mid+1}^{R}(x+B_i)$ 用 FFT 相乘得到。时间复杂度 $T(N) = O(N \log N) + 2T(N/2) = O(N \log^2 N)$。

100 分

把 Subtask4,5 的做法恰当组合一下，就是正解啦。

将这棵树轻重链剖分。

对于一个是根节点或是其父亲的轻儿子的点 p，从它开始往下形成了一条重链。我们希望对于所有这样的 p，求出多项式 F_p 与 G_p（F_p 与 G_p 的定义同 Subtask1,2 的暴力 DP）。

设以 p 为顶的重链的各个点的点号从上往下依次是 $U_1, U_2, \ldots, U_K$（$U_1 = p$，U_K 是叶子），设多项式

$$f_i = B_i x \prod_{v \in child(U_i) \text{且} v \neq U_{i+1}} G_v$$

$$g_i = \prod_{v \in child(U_i) \text{且} v \neq U_{i+1}} (F_v + G_v)$$

f_i 和 g_i 可以用类似 Subtask5 的分治求出,这部分时间复杂度 $O(size(p) \log^2 size(p))$。

现在问题已经变成了 Subtask4 的链上问题了，对于 $1 \leq i \leq K$，i 可以为答案乘上 f_i 或 g_i，并且不能有两个相邻的 i 和 $i+1$（$1 \leq i < K$）同时取了 f_i 和 f_{i+1}，F_p 为取 f_1 时可能的多项式的积的和，G_p 为取 g_1 时可能的多项式的积的和，也可以用类似 Subtask4 的分治求出，时间复杂度 $O(size(p) \log^2 size(p))$。

由于轻重链剖分，$\sum size(p) = O(N \log N)$，总时间复杂度 $O(N \log^3 N)$。