Solution

2023 年 11 月 2 日

1 tree

首先考虑确定一个根的做法：贪心地优先递归叶子节点标号最小的那个子树。

接着考虑如何确定根，首先以标号最小的那个叶子为根，向下贪心地找子树中“最小叶子节点标号”最大的那个点。

根据确定根后的贪心做法，除去这个子树外的点顺序都已经被确定了，因为这个子树中的叶子节点更大，于是就是考虑当前这个子树的根节点作为根更优还是作为路径上的一个点更优。即比较“最小叶子节点标号”和 u 本身的大小关系，若其大于 u 则继续递归。

2 young

本题中要求 $\mathbf{min}\{|a_i-a_j|,|b_i-b_j|\}$ ，通过 $a+b=\mathbf{max}(a,b)+\mathbf{min}(a,b)$ 转换可得。

考虑把 (a_i,b_i) 二元组视作二维平面上的整点，两个点 (i,j) 之间的产生的贡献实际上等于两个整点之间的切比雪夫距离，即 $\mathbf{max}\{|a_i-a_j|,|b_i-b_j|\}$ 。

通过旋转坐标系可以实现切比雪夫距离转曼哈顿距离：令 $p_i=\frac{a_i+b_i}{2},q_i=\frac{a_i-b_i}{2}$ ，则 $\mathbf{max}\{|a_i-a_j|,|b_i-b_j|\}=|p_i-p_j|+|q_i-q_j|$。

上式可以通过拆绝对值符号证明。

$$\begin{aligned}\mathbf{min}\{|a_i-a_j|,|b_i-b_j|\}&=|a_i-a_j|+|b_i-b_j|-\mathbf{max}\{|a_i-a_j|,|b_i-b_j|\}\\&=|a_i-a_j|+|b_i-b_j|-(|p_i-p_j|+|q_i-q_j|)\end{aligned}$$

接下来只需要解决 4 个形如 $\sum_{i\in\text{subtree }u}\sum_{j\in\text{subtree }u}|a_i-a_j|$ 的问题即可。

注意到，如果我们用可持久化线段树维护出关于 a_i 的有序数组，当我们插入一个新的数字 a_j 的时候，我们只需要知道小于 a_j 的元素的和，以及小于 a_j 的元素的数量，就可以知道 a_j 对新的答案产生的贡献，大于 a_j 的部分同理。

可以通过树上启发式合并的方式，对于每一个点维护出子树的答案。

时间复杂度为 $O(n\log^2 n)$。如果实现细节极其优秀，有可能通过。

注意到在启发式合并的过程中一些性质被我们浪费了，当我们维护出一个关于 a_i 的有序数组时，答案可以分治的去维护。

假设我们当前维护出的数组为 a ，长度为 n ，取中点 $\mathrm{m}=\frac{n+1}{2}$ 。那么对于 $a[1,n]$ 的答案可以由以下式子表示：

$$\begin{aligned}\text{ans }a[1,n]=&\text{ans }a[1,m]+\text{ans }a[m+1,n]\\&+\text{sum }a[m+1,n]\times\text{cnt }a[1,m]\\&-\text{sum }a[1,m]\times\text{cnt }a[m+1,n]\end{aligned}$$

于是，我们可以通过线段树合并的方式维护出有序数组 a ，在合并的时候按如上方式更新出答案即可。时间复杂度为 $O(n\log n)$ 。

3 subsequence

考虑求一个串的本质不同的子序列个数：dp_i 表示贪心匹配最后会匹配到原串中的第 i 个位置的方案数，则有：

$$dp_i = \sum_{j=\text{last}_i}^{i-1} dp_j$$

其中 last_i 表示上一个与第 i 个位置字符相同的位置，不存在则记为 0。前缀和优化可以使这部分时间复杂度为 $O(n)$。

对于本题我们发现，如果长度不同则一定是长度长的子序列权值大。那么我们只需要分两部分处理：

- 第 1 种是选出的两个子序列长度不同，等价于对每个字符串统计给定长度的子序列个数，只要给之前的 dp 加一维长度就可以做到 $O(n^2 + m^2)$。
- 第 2 种情况是选出的两个子序列长度相同，那么考虑枚举两个子序列相同部分的前缀，$f_{i,j}$ 表示在第 1 个串中匹配到了位置 i，在第 2 个串中匹配到了位置 j 的方案数，其中每一项都可以由一个矩形内的 f 值转移过来，二维前缀和优化后时间复杂度可以做到 $O(nm)$。

之后就是枚举第 1 个不同的位置，即枚举 i, j，且 $A_i > B_j$，则前面相等的前缀数同样也可以由 f 的一个矩阵转移。然后考虑在后面填上长度相等的后缀，这部分的做法和求 f 是类似的，区别在于要倒着算且不需要每一位都对应相等。时间复杂度：$O((n+m)^2)$。

4 palindrome

首先我们考虑一种会算重的算法，定义 $R(n)$ 为回文串或者两个回文串连接的方案数：

$$R(n) = \sum_{i=0}^{n-1} k^{\lceil \frac{i}{2} \rceil} k^{\lceil \frac{n-i}{2} \rceil}$$

这时候类似 “abaaba” 之类的串会被计算多次。

有一个结论，当回文串 s 有一个周期 p（对于所有 $i + p < |s|$, $s_i = s_{i+p}$），并且 $|s|$ 可以被 p 整除时，一个回文串有两个或更多的连接计划。计划的数量为 $|s|/p$（p 是 s 的最小周期长度）。

因此，我们定义 D(n) 为只有一种连接计划的回文串或字符串的数量，例如”caba”。我们需要减去算重的字符串数量。根据上述结论，我们只需枚举 n 的所有因子 l（除了 n 本身），并使用以下公式计算 $D(n)$:

$$D(n) = R(n) - \sum_{l|n \wedge l<n} \frac{n}{l} D(l)$$

最后，答案为 $\sum_{i=1}^{n} \sum_{l|i} D(l)$。时间复杂度为 $O(\sqrt{n})$ 或 $O(n \log n)$。