《超现实树》解题报告

南京外国语学校 闫陈效

题目大意

对于常数 k, 称一个字符串为 "k-超现实数串", 如果其只包含字符 `{`, `|`, `}`, 且:

- 空串为 k-超现实数串;
- 如果 s, t 为 k-超现实数串, 那么 $s + t$ 为 k-超现实数串;
- 如果 $k + 1$ 个字符串 $s_1, s_2, \cdots s_{k+1}$ 都是 k-超现实数串, 那么 `{` $+ s_1 +$ `|` $+ s_2 +$ `|` $+ \cdots +$ `|` $+ s_{k+1} +$ `}` 为 k-超现实数串;
- k-超现实数串仅限于此.

给定一棵 n 个点的无根树, 每个点 i 上有一个字符 $a_i \in \{\texttt{\{}, \texttt{|}, \texttt{\}}\}$. 给定整数 m, 你需要对 $k = 0, 1, \cdots, m$ 分别求出树上有多少条路径对应字符串是 k-超现实数串.

数据范围

对于所有数据, 有 $2 \le n \le 10^5$, $0 \le m \le n - 2$, $a_i \in \{\texttt{\{}, \texttt{|}, \texttt{\}}\}$.

- **Subtask 1** (5 分): $n \le 4601$;
- **Subtask 2** (20 分): 对每条边 (x, y) 有 $y = x + 1$;
- **Subtask 3** (5 分): $a_i \ne \texttt{|}$, $m = 0$;
- **Subtask 4** (15 分): $m \le 3$;
- **Subtask 5** (25 分): $n \le 5 \times 10^4$;
- **Subtask 6** (30 分): 无特殊限制.

时间限制: 1s; 空间限制: 512MB.

解题过程

算法一

我会暴力! 考虑枚举 k 以及路径 (x, y), 检查这条路径对应的字符串是否是 k-超现实数串.

如何检查? 首先, 一个字符串是一个 k-超现实数串的充要条件显然是: 其所有不为竖线的字符组成的子序列是合法的括号序列, 并且直接属于每层匹配括号对 (而不属于其子代匹配括号对) 的竖线数量都为 k. 那么可以使用栈扫描进行括号匹配. 对于栈中的每个左括号, 考虑额外记录到目前为止直接属于该左括号层的竖线数量, 当扫描到右括号时, 还需检查栈顶是否为 k. 如此可以 $\mathrm{O}(n)$ 判断字符串是否 k-超现实树串. 总时间复杂度 $\mathrm{O}(n^4)$.

注意到栈的操作可以在 DFS 的过程中维护, 需要额外支持撤销操作, 这是容易的. 那么只要枚举 k 和路径起点 x, 从 x 开始 DFS 即可一次性得到每个 y 的判断结果. 总时间复杂度 $\mathrm{O}(n^3)$.

另一个观察是, 每个字符串只可能有唯一一个 k_0 使其是 k_0-超现实数串. 并且 "括号字符组成的子序列是合法括号序列" 的条件是与 k 无关的, 于是可以不必对每个 k 都进行扫描, 而是对每条路径只扫描一次, 记录遇到右括号时栈顶的数是否始终唯一, 如果不唯一肯定不是任何超现实数串, 否则设为 k_0, 其为 k_0-超现实数串. 总时间复杂度 $\mathrm{O}(n^2)$, 可以通过 Subtask 1, 期望得分 5 分.

算法二

我会树为链的情形!

依然从 "括号子序列是合法括号序列" 的条件入手. 在序列上, 可以观察到, 如果要合法地匹配, 则每个括号字符对应的匹配位置一定是唯一的.

定理 对于 $a_i = \texttt{(}$, 从 i 开始往后使用栈进行括号匹配, 特殊地若遇到右括号且栈为空则什么都不做, 设与 i 匹配的位置为 j (j 有可能不存在). 则: 任何包含 i 的子串要么不合法, 要么在其中 i 与 j 匹配. 对 $a_i = \texttt{)}$ 结论类似.

证明 只证 $a_i = \texttt{(}$ 的情况. 容易验证, 对于包含 i 的子串 $[l, r]$, 从 l 开始用栈匹配, 扫描到 i 之后的位置时, 要么已经判断不合法, 要么此时栈的顶端的连续的一段就等于从 i 开始扫描到当前位置的栈. 那么底部的那段并不产生影响. □

那么可以正反分别用栈扫一遍, 得到每个括号字符唯一能匹配的位置, 再筛选出互为对方匹配字符的位置二元组们.

我们称互相匹配的括号对为 "可能括号对", 不属于任何可能括号对的括号为 "不可能括号", 则任何包含不可能括号的子串都不合法. 容易证明, 所有可能括号对之间形成要么包含要么相离的关系, 可以对它们建一棵树, 那么合法括号串只可能是某个点的儿子序列的某个区间. 容易类似树形 DP 地预处理出树上每个节点对应的区间是否合法, 以及如果合法则是几-超现实数串. 然后对每个点的儿子序列做相等段划分, 容易统计答案. 总时间复杂度 $\mathrm{O}(n)$, 可以通过 Subtask 2, 结合算法一期望得分 25 分.

可以发现, 算法二中的结论在树上不再成立, 某个括号字符的匹配位置不一定唯一.

算法三

我会对每个 k 分别点分治!

点分治过程中, 在 x 为重心时, 我们需要预处理出对于每个点 y, 路径 (y, x) 作为字符串前缀以及路径 (x, y) 作为字符串后缀时, 在未知字符串另一半的情况下, 能留下的全部匹配信息, 并尽可能简洁地记录它. 正反两种情况是几乎一样的, 只要把左右括号反过来看待即可, 以下以路径 (x, y) 作为后缀为例.

依然考虑仿照算法一, 从 x 出发 DFS 并维护一个栈. 不同的是, 当遇到右括号和竖线时, 如果当前栈为空, 并不一定说明匹配失败, 因为其可能与未知的前缀进行匹配, 而这正是我们要记录的匹配信息; 相反, 在半条路径内自己匹配掉的括号对反而不用记录 (当然, 依然需要判断竖线数量是否为 k).

于是, 在栈底部的下面还需要分配一个保留位置, 用于记录与另半条路径匹配的竖线数量. 每次遇到右括号, 如果栈顶是保留位置, 则将其压入用于记录匹配信息的序列, 并清零保留位置. 这样, 每个点 y 都记录了一正一反两个序列表示匹配信息, 其中每个元素表示独属于从 x 往 y 方向第 i 个未匹配的右括号的竖线数量. 那么显然, 一个正序列和一个反序列能匹配成功, 当且仅当它们长度相等且对应位置之和都为 k.

肯定不能把序列们显式地存下来. 由于每个点的序列继承于其父亲, 可以快速计算哈希值, 其能成功匹配的对应序列的哈希值是容易计算的, 存到哈希表里查询即可. 也可以直接将哈希值排序做相等段划分, 总时间复杂度为大常数 $\mathrm{O}(mn \log n)$ 或小常数 $\mathrm{O}(mn \log^2 n)$, 可以通过 Subtask 3, 4, 结合算法一期望得分 25 分. 当然, 也可以使用 Trie 维护序列, 不再赘述.

算法四

我会只做一次点分治!

类似算法三对每个点记录正反两个序列. 不同的是, k 是任意的, 那么就不能在半条路径内部匹配时检查竖线数量是否为 k. 不过我们知道, 如果两次匹配得到的竖线数量不同, 那肯定不合法, 这就是说, k 要么是任意的, 要么是唯一的. 我们对每个点的正反两个方向再记录一个数, 如果其等于 -1 则表示 k 任意, 否则表示唯一的那个 k.

另一个不同是, 现在只知道两个序列能匹配成功仅当长度相等且对应位置之和都相等. 这可以对差分哈希, 从而划分出等价类. 每个等价类中, k 唯一的那些可以类似算法三直接查询然后贡献答案, 这样简单处理掉, 不再赘述. 以下假设所有都是 k 任意.

那么对于同一等价类里的任意一对正反序列, 它们都能贡献答案, 具体地, 设其第一个元素分别是 x, y, 则贡献到 $k = x + y$. 这等价于一个卷积. 对每个等价类做 NTT 显然不现实, 不过我们注意到以 x 为重心时, 参与卷积的非零元素只有 $\mathrm{O}(s)$ 个, 其中 s 是当前连通块大小. 那么可以根号分治, 令 $B = \mathrm{O}(\sqrt{s \log s})$, 对非零元素数量不超过 B 的等价类暴力, 否则 NTT, 这样两部分的复杂度都是 $\mathrm{O}(s\sqrt{s \log s})$, 达到平衡. 根据 Master 定

理, 外面套一层点分治后, 总时间复杂度是 $\mathrm{O}(n\sqrt{n\log n})$, 可以通过 Subtask 1, 5, 期望得分 30 分到 100 分.

算法五

我会正解! 考虑优化算法四.

定理 对于序列长度大于 1 的等价类, 其中本质不同的序列数量 (即参与卷积的非零元素数量) 为 $\mathrm{O}(\sqrt{s})$.

证明 考察该等价类中每种不同的序列中, 最靠近 x 的那个未匹配右括号所在节点 (序列第一个元素对应节点) u, 它们显然互不相同, 且无祖先后代关系. 考虑序列第二个元素为 t, 那么 u 子树内至少有 t 条竖线, 而这个 t 显然也对不同的序列不同. 这就转化为: 若干个互不相同的数之和不超过 s, 那么显然只有 $\mathrm{O}(\sqrt{s})$ 个. □

利用这个结论, 可以对所有长度大于 1 的等价类暴力, 只对长度等于 1 的唯一等价类做 NTT. 根据 Master 定理, 总时间复杂度 $\mathrm{O}(n\log^2 n + n\sqrt{n}) = \mathrm{O}(n\sqrt{n})$, 可以通过所有数据, 期望得分 100 分.

参考资料

[1] OI Wiki: 树分治

[2] OI Wiki: 快速数论变换