《在路上》命题报告

重庆市巴蜀中学 郭雨豪

2022 年 10 月 31 日

目录

1 题目大意

这是一道交互题。

有一棵 n 个点，但形态未知的树，你需要找到它的重心，保证 n 为奇数，此时重心一定唯一。

你每次可以询问三个点 (x, y, z)，若其中存在一个点在另外两个点的最短路径上，则交互器会返回这个点，否则交互器会返回 0。

每个测试点内部有 t 组测试数据，你需要在总询问次数不超过 M 的情况下求出每一组数据的答案。

2 数据范围及限制

<table>
<tr><th>子任务编号</th><th>子任务分数</th><th>t</th><th>n</th><th>M</th><th>特殊性质</th></tr>
<tr><td>1</td><td>2</td><td>$=100$</td><td>$=3$</td><td>$=100$</td><td rowspan="2">无</td></tr>
<tr><td>2</td><td>8</td><td>$=10$</td><td rowspan="2">$=999$</td><td rowspan="2">$=10^7$</td></tr>
<tr><td>3</td><td>12</td><td rowspan="3">$=100$</td><td>A</td></tr>
<tr><td>4</td><td>17</td><td>无</td></tr>
<tr><td>5</td><td>18</td><td>$=49999$</td><td>$=2.5\times10^7$</td><td>A</td></tr>
<tr><td>6</td><td>12</td><td>$=9999$</td><td rowspan="2">$=4\times10^7$</td><td rowspan="3">无</td></tr>
<tr><td>7</td><td>19</td><td>$=50$</td><td rowspan="2">$=29999$</td></tr>
<tr><td>8</td><td>12</td><td>$=100$</td><td>$=5\times10^7$</td></tr>
</table>

特殊性质 A：保证每个点的度数不超过 2。

保证每个子任务的测试点数量不超过 20。

时间限制 $4s$，空间限制 $512MB$。

保证交互库使用的时间不超过 $2s$，空间不超过 $64MB$。

3 解题过程

3.1 简单做法

3.1.1 n=3

直接询问三个点，返回值即为重心，期望得分 2。

3.1.2 暴力做法

将 1 视为树根，那么询问 $1, x, y$ 可以得到 x, y 是否为祖先后代关系，同时得到其中的深度较小的一个，通过 $O(n^2)$ 次操作可以还原整棵树，期望得分 10。

也有很多其它做法。

3.2 链

3.2.1 初步分析

询问可以看做返回三个数中最靠中间的一个数，这个操作得到的信息量是严格劣于知道三个数的相对顺序的。如果想要还原整个序列等价于排序，但 $O(n)$ 次操作内是不能进行排序的，也就是说，在此档部分分以及之后的正解中，是不可能在 $O(n)$ 次还原整棵树的。

以下对链的讨论中将问题视为：有一个未知排列，每次你可以询问三个位置上的数的中位数。

3.2.2 做法

可以考虑用 nth_element 的方法，如果这样做需要找到一种比较两个数大小的方式。

一个想法是将这两个点与 1 同时询问，返回值即为这两个数中的较小数。

找到 1 可以用如下的方法：将 $1 \sim n$ 依次加入一个集合，当集合大小 $= 3$ 时查询集合的中位数，然后删掉其中的中位数，容易发现最后集合中剩下的两个数分别为 $1, n$，由于查询的是中位数所以具体用 1 还是 n 不会影响答案。

这个做法可以通过，具体的常数可以见题解的最后一页，结合之前的做法期望得分 40。

当然也可以利用每次询问三个数的性质，随机两个数，将序列分为三份，这样会更优秀一些。

3.3 树

3.3.1 初步分析

链上我们采取了分治来解决问题，而树上的分治在形态未知是是困难的。注意到重心的性质：每一个子树的大小不超过 $n/2$，于是我们考虑随机，每次随机两个点 x, y，之后我们假设重心在 $x \to y$ 的最短路上，这样的在树形态最坏（例如链）的情况下期望两次可以成功，之后将其视为一个类似的序列问题。

3.3.2 判定

先考虑一个简单很多的问题，怎么判定一个点是否是重心？

考虑到重心的性质：所有子树的大小不超过 $\frac{n}{2}$。但是直接询问三个点是不容易得出每个点属于哪个子树的。

又考虑到若不存在重心，则有一个子树的大小 $> \frac{n}{2}$，此时可以考虑找绝对众数，使用摩尔投票，虽然我们不能通过询问得到每一个点属于哪个子树，但我们可以通过询问得出每一个点是否与上一个点属于同一个子树，最后我们再询问一次可以得到每个点是否和摩尔投票的结果属于同一子树。

3.3.3 暴力做法

可以用链的方法用 $O(n \log n)$ 的时间将树上这条路径的具体顺序求出。由于我们假设重心在链上，我们只需要知道剩余的点在链上的位置，由于链上已经有序可以考虑二分答案，同样是 $O(n \log n)$ 的复杂度，由于最开始随机了路径，这个做法的总期望为 $O(n \log n)$ 的，结合之前的算法期望得分 $57 \sim 69$。

3.3.4 优化

同样考虑 nth_element，但直接做是有问题的，现在不能直接随机一个点，因为每个位置还有一个子树，询问时会带一个额外的代价，复杂度退化为 $O(n \log n)$（考虑一棵 X 形状的树）。如果想要带权随机需要定位点在序列上的位置，可以考虑每次单独枚举或者提前排序，同样也是 $O(n \log n)$ 的复杂度。带权随机之后我们还需要判断之后需要保留其中的哪一边，我们可以通过询问得出每个点是属于链上左侧或者链上右侧或是当前点的链外

部分，如果三个方向都不超过 $n/2$ 则该位置为重心，如果左右超过 $n/2$，那么可以递归其中一侧，否则进行一次判定，如果不成功说明重心不在链上。整个做法次数为 $O(n\log n)$ 的，期望得分 69。

3.3.5 再优化

注意到上述算法的一个上界在于排序，如果解决排序整道题可以做到线性。

可以考虑先定位，再排序，于是可以先随机 $O(\sqrt{n})$ 个点，将这些点单独排序，进行上述的过程后得到了重心所属的区间，然后再次找出区间内的点排序，最后定位到重心，这样做已经达到了 $O(n)$ 的询问次数，不过常数可能略大。

该做法和其它一些类似的线性但常数过大的做法可以得到 $88\sim100$ 分。

3.3.6 正解

注意到上述算法的优化在于，我们需要排序，但不需要完全的顺序，只需要一部分元素的顺序。

一种方法是将 nth_element 的结构记录下来，也就是维护链上的一个区间 $[l,r]$，表示当前知道重心落在 $l\sim r$ 中，同时维护集合 S 表示当前这个区间内部点的集合。每一次先随机选择一个点，检查所有 S 中在链上的点，找到随机选择的点所对应的点在链上的位置，之后检查这个点的情况，然后将 S 划分为三个集合，向其中一个方向移动。

若当前的点不在链上部分的子树大小 $>n/2$，可以对这个点单独做一次摩尔投票，判断其是否为重心。最终的询问次数为 $O(n)$，可以通过。

3.3.7 常数分析

令 C 表示用经典的分治算法实现 nth_element 的常数。

在整个算法的第一步，我们随机了两个点，我们将此时的结果按照在两点之间是否存在一个点，其除去随机的链以外子树大小 $>n/2$ 分为两类，无论怎样我们先使用了 n 次操作得到了链上的所有点。

若存在，则期望随机 2 次就可以得到这个点，每一次随机会检查当前存在的所有点，每一次检查会进行两次询问，得到这个点之后需要用一次摩尔投票以及检查摩尔投票的结果。所以这部分的理论期望是不会超过 $7n$ 的。

若不存在，则可以看做一个带权的 nth_element，带权的 nth_element 的常数是一定不大于不带权的，所以这部分的常数一定不超过 $2Cn+n$。

每一次如果链上不存在点使得子树外大小 $> n/2$，则重心一定在链上，也就是一定会结束，如果存在有可能不会结束，假设视为一定不会结束，由于这个概率不超过 $1/2$，所以期望会有一次这样的情况。最后的总期望是 $2Cn+8n$。经过实际测验 C 的值在 $n=49999$ 时大致为 3.5，不过有一定的波动，但在多次 $(>10^6)$ 试验（单次试验为：进行 100 次模拟用分值算法求长度为 49999 序列的中位数，取平均值）中均未超过 3.9，若近似看作 4 则整道题的次数为 $16n$，实际上可以发现在上述讨论中很多部分完全不会达到上界，并且在各种测试中也表现较为优秀，又因为更高复杂度的算法也带有一定的常数，故将 5×10^7 作为满分标准。实际上本题还有一些很有用的常数优化，不过由于暂时无法证明其对算法带来的实际变化，没有对最终的分数进行调整。在目前的数据中中平均情况大致在 $12n$ 左右，多次测验也均可轻易通过题目。

如果您在 Subtask5 直接使用了系统的 nth_element，可以发现其常数会更小，因为其实现方法并不仅是普通的分治算法。

3.3.8 其它优化

过程中有很多步，我们需要判断一个点是属于左部还是右部或是不属于任何部分，我们原本用了两次在确定，但我们可以随机询问的顺序，便有概率一次问出该点所属的部分。

在询问出整条链时，我们其实也得出了挂在链两端的集合，于是可以特殊处理，实际情况中，将原算法卡到常数比较大的数据都有一条比较长的链，而此时删掉链两端的集合优化效果非常明显。

加上以上两个优化之后平均次数大致在 $9n$ 左右。

4 分数设计

由于本题的询问次数与期望有关，故采用单组数据多次测试来减小误差。

即使在 $t = 100$ 的情况下，在实验中仍然存在较大的误差，于是并没有采取交互题常用的根据交互次数给分而选择了子任务得分，由于不同的算法之间差距非常大（$O(n), O(n \log n), O(n^2)$），且针对不同的算法构造了不同的数据，这样的做法仍然可以区分掉不同复杂度的做法。每个 Subtask 的限制相对该 Subtask 的算法也是非常松的。未经任何剪枝的 std 在大量测试中均能轻松通过所有测试点。

5 致谢

感谢集训队员常瑞年同学，罗思远同学，精英培训冯政玮同学为本题验题。

6 参考资料

无