《树据结构》题解

unzcjouhi

2022 年 1 月 17 日

题意简介

有一个 n 个顶点的树，边上有边权，且边的边权是一个排列。你可以给定两个边权 x, y，交换对应边权的边，也可以询问根到点 u 的路径上所有边的边权最大值。你需要通过这两个操作来还原出树的结构和初始时每条边的权值。

得分情况

集训队情况：
4 位同学 64 分（所有 $O(n \log n)$ 部分）
3 位同学 43 分（通过 $n \leq 1000$，特殊性质 A、B 的 $O(n \log n)$ 部分）
5 位同学 29 分（$n \leq 1000$ 和特殊性质 A 的 $O(n \log n)$ 部分）
1 位同学 20 分（$n \leq 50$ 的特殊性质 A 和 $n \leq 20000$ 的特殊性质 B ???）
6 位同学 15 分（$n \leq 1000$）
2 位同学 6 分（$n \leq 50$ 的特殊性质 A）
据说非集训队目前最高分 76 分。

吐槽环节

算法零

记 $Q(u)$ 表示询问根到 u 的路径上的的边权最大值得到的结果。
我们介绍一个简单的树上 $O(n^2)$ 的算法。考虑对每个 i，把边权为 i 的边交换成 $n-1$，然后询问 $Q(2), Q(3), \cdots, Q(n)$，再交换回来。这样我们可以知道：边权为 i 的边下面的子树有哪些节点。记这个集合为 S_i。
根据 S_i 可以还原出整棵树的结构。一种还原方法是：对每个点 u，找所有 $u \in S_i$ 中 $|S_i|$ 最小的那个和次小的那个（如果存在的话）。根据最小的那个，我们可以知道 u 到它父亲的边的边权。根据次小的那个，我们可以知道它父亲到它父亲的父亲的边的边权（或者它的父亲就是根），从而还原出它的父亲的编号。

算法一

首先注意到只需记录下每次交换的过程，并知道操作之后的边权分布，就可以知道初始时的边权分布了。以下的算法都需要使用这一步。此外，我们可以假设点权和边权都是随机的，否则可以随机打乱点的编号，再用 n 次交换操作随机排列边权。

我们先讲树是一条链情况下的各种做法。链的各种 $O(n^2)$ 做法应该很多，这里我们再讲一种。

考虑任意地找一个点 u，并将边权 1 与 $Q(u)$ 交换，再将 2 与 $Q(u)$ 交换，...，将边权 i 与 $Q(u)$ 交换，直到 $i = Q(u)$ 了为止。这样，我们可以确保 1 到 u 的路径上所有的边是 $1, 2, \cdots, Q(u)$ 的一个排列。然后，我们再一次询问 $Q(2), ..., Q(n)$，就可以知道每个点是否在根到 u 的路径上。

对每个 $u = 2, 3, \cdots, n$ 做一遍操作，我们就可以知道链的排列结构，也可以把链从根到底端的边权交换成 $1, 2, \cdots, n-1$，因此也可以获知初始边权。

算法二

算法一可以优化成 $O(n \log n)$。
考虑随机选一个点，通过算法一的操作可以知道哪些点在它的左边，哪些点在它的右边。这样，我们可以把问题转化为两个子问题。
实现时，记录 $solve(root, l, r, S)$ 表示处理深度区间为 $[l, r]$ 中的点，且其中根为 $root$，点编号的集合为 S 的函数。每次随机 S 中的一个点做算法一的操作（边权区间是 [l, r - 1]，等价地看成 [1, r - l]）。

算法三

我们再介绍一种特殊性质 A 的 $O(n\log n)$ 做法。这种做法更容易优化到 $O(n)$。

假设我们知道了 $1,2,\cdots,i$ 在链上的顺序，并且满足 $1,2,\cdots,i$ **中相邻两个点之间的边权是连续的，每一段的权值是递增的**。我们还需要维护每个点 $u(2\le u\le i)$ 的 $Q(u)$ 的 set。

那么我们要考虑怎样把 $i+1$ 号点“插入”进前 i 号点，并继续维护上述信息。询问一次 $Q(i+1)$，再询问 $Q(2),\cdots,Q(i)$ 中最大的 $\le Q(i+1)$ 的数（记为 x）。随后，我们用类似算法一、二的方法，交换 $x+1,Q(i+1)$，交换 $x+2,Q(i+1)$，...，直到 $i+1$ 到左边第一个编号 $\le i$ 的点之间所有边的边权是连续的一段。

注意到期望上相邻两个点的边的个数是 $O(n/i)$，因此第 i 次插入的期望代价是 $O(n/i)$ 的，总复杂度为 $O(n\log n)$。

举个例子：$n=10$，$1,2,3,4$ 号点在链上的排列为 $1,4,2,3$，且 1 到 4 的边的边权是 1 到 4 的排列，4 到 2 之间的边权是 5 到 7 的排列，2 到 3 之间的边权是 8 到 9 的排列。假如我们询问出 $Q(5)=3$，就可以直到 5 的位置在 1，4 之间。

算法四

算法三的思想如何优化？
假设我们忽略边权 $< n - 2^k$ 的边，并且知道 $1, 2, \cdots, 2^k$ 在链上的“顺序”。但是我们只需要满足 $1, 2, \cdots, 2^k$ 的相邻的两个点的所有边权 $\geq n - 2^k$ 的边的边权构成连续的一段。注意这里相邻的两个点之间可能没有边，因此我们不能确定 $1, 2, \cdots, 2^k$ 在链上的顺序，但可以将它分成若干段，段与段之间的顺序已知。
用算法三的方法将 $2^k + 1, \cdots, 2^{k+1}$ 依次“插入”。接着我们需要考虑边权 $\geq n - 2^{k+1}$ 的边了。我们也像算法三一样依次插入，维护顺序关系，但是我们需要让相邻两个点的所有边权 $\geq n - 2^{k+1}$ 的边的边权构成连续的一段。此外，我们需要 **从最浅的段到最深的段的顺序**插入，段内的顺序任意。
通过倍增，我们最终可以还原出所有点之间的顺序。注意到我们考虑的点和边的大小总是同级别，因此相邻两个点之间的边的个数是 $O(1)$，并且点被分成的段的大小期望也是 $O(1)$ 的。容易发现总代价正比于 $1 + 2 + 4 + \cdots$，是 $O(n)$ 的。

算法五

对于特殊性质 B，我们有许多常数或小或大的 $O(n \log n)$ 到 $O(n \log^2 n)$ 做法，这里我们简叙几种。我们随机选一个点。用类似算法一的操作，随机一个点 u，可以用 $O(n)$ 次操作问出 1 到 u 上所有点有哪些，那么也就很容易还原出 1 到 u 整条链的顺序，自然我们知道 1 的某个儿子。把 1 到儿子的边赋值成 n，再询问一遍 $Q(2), \cdots, Q(n)$，就可以知道这个儿子的子树的所有点。这样我们把树分成子树内和子树外两块。通过分治（需要把这个原先 $n-1$ 的边重新变成 1），我们可以还原出整棵树的信息。

赛后发现，其实分治过程可以更简单。我们可以直接询问 $Q(2), \cdots, Q(n)$，找到其最小者 x，再把它交换成 $n-1$，询问一遍 $Q(2), \cdots, Q(n)$。再把 $n-1$ 交换成 1，询问一遍 $Q(2), \cdots, Q(n)$。在随机树的情况下，这个算法按实现细节的差异，复杂度可以做到 $O(n \log n)$ 或 $O(n \log^2 n)$ 不等。

算法六

事实上，一般树的情况可以化为链的情况。想象一个维护根到 2 号点，3 号点，$\cdots$，i 号点的链的并的过程。记录 S_i 表示 1 号点到 $2, 3, \cdots, i$ 的链的并含有的点的集合（不含根）。我们需要找到 $S_2, \cdots, S_n$，并且要求：S_i 上面的每个点到它父亲的边的权值构成 $1, 2, \cdots, |S_i|$ 的排列。
假设我们得到了 $|S_i|$ 的值。若 $Q(i+1) \leq S_i$，则通过找 $Q(i+1)$ 与 $|S_1|, ..., |S_i|$ 的 lower_bound，我们可以找到某个 j 使得 $i+1 \in S_j - S_{j-1}$，且 $S_{i+1} = S_i$。否则，我们交换 $|S_i|+1$ 和 $Q(i+1)$, ...，直到我们把 S_{i+1} S_i 上所有边变成权值连续的一段（此时通过 $Q(i+1) = |S_{i+1}|$）在 n 次加入之后，我们其实可以得到 $S_{i+1} - S_i$ 这个集合，也就是说 $i+1$ 号点加入后新增的一条链。
调用链的做法之后，我们可以知道 $S_i - S_{i-1}$ 这条链的顺序，以及它顶端的点 x 到父亲的那条边的边权。只需考虑怎样将所有的链“接起来”。按 i 递增的顺序，我们把 x 到父亲的边的边权交换成 1，再询问 $Q(x)$，最后把 $S_i - S_{i-1}$ 的所有点到它父亲的边边权交换成 $|S_{i-1}|+2, \cdots, |S_i|+1$（按由浅到深的顺序）。容易发现这样我们每次都可以知道 $S_i - S_{i-1}$ 的顶端 x 的父亲编号。
如果你使用 $O(n)$ 的链上的做法，总复杂度仍然为 $O(n)$。

谢谢大家！