《北校门外的未来》命题报告

西北工业大学附属中学 刘承奥

1 概述

本文讨论2018年集训队互测中作者所命制的一道试题。

本文首先给出题目信息，然后介绍解题算法，之后会分析命制思路并讨论本题模型的一些应用和展望。

2 题目

2.1 题意

对于一棵树$T = (V, E)$，V 中每个点有一个互不相同的正整数标号。我们用点i 表示编号为i 的点。

定义这棵树的**谷图**为$G(T) = (V, E')$。$G(T)$ 是无向简单图。存在边$(u, v) \in E'$ 当且仅当在T 中，不存在一个异于u, v 的点x 满足x 在从u 到v 的简单路径上且其编号大于$\min(u, v)$。

有一棵树T，初始时只有一个点，编号为1，接下来有q 次操作，操作有以下两种：

1. `1 u v` 表示加入一个编号为v 的节点并与当前编号为u 的节点相连（保证任何时刻不会有两个编号相同的节点）；
2. `2 u v` 表示查询$G(T)$ 中点u 到v 的最短路（每条边长度均为1）。

请你回答所有查询。

2.2 数据范围

子任务编号	分值	n	q	性质
1	6	≤ 100	≤ 500	
2	10	≤ 2000	≤ 5000	
3	12	≤ 2000	$\leq 5\times 10^5$	
4	15	$\leq 10^5$	$\leq 5\times 10^5$	一、二
5	12	$\leq 10^5$	$\leq 5\times 10^5$	二
6	10	$\leq 10^5$	$\leq 2\times 10^5$	一、三
7	10	$\leq 10^5$	$\leq 2\times 10^5$	一
8	10	$\leq 10^5$	$\leq 2\times 10^5$	
9	15	$\leq 10^5$	$\leq 5\times 10^5$	

性质一：所有1 操作（修改）在所有2 操作（询问）之前。

性质二：任何时刻保证树是一条链。

性质三：最终形成的树在所有n 个点的有标号无根树中均匀随机，随机数生成器使用梅森旋转算法。

2.3 评测配置

时间限制：2s

空间限制：512MiB

语言：g++，编译选项为-O2 [-std=c++11]

评测环境：

3 试题考点

笛卡尔树及其扩展，复杂度均摊分析，树链剖分，树上倍增，Link-Cut-Tree。

4 算法介绍

为了方便，我们约定：对于任意有根树T，设$f_T(u)$ 表示u 在T 中的父亲（如

果存在）。

4.1 子任务1

按题意模拟，每次重建T 和$G(T)$ 并用Floyd 算法预处理出最短路，或在每次询问时BFS 求出最短路。

时间复杂度：$O(n^4 + q)$ 或$O(n^3 + q)$ 或$O(qn^2)$。

4.2 子任务2

对于序列最值问题我们有笛卡尔树这个工具，可以把大小关系变成树上的祖先后代关系。对于树我们也可以使用这个方法：

我们称树T 关于点权函数$w(v)$ 的**笛卡尔树**（该定义仅在本文中有效）为一棵有根树$C(T)$，满足：

1. 堆性质：$C(T)$ 中任意祖先节点的权值不小于子孙节点的权值；
2. 搜索树性质：$C(T)$ 中任意子树对应的点在T 中连通。

在这个问题中，我们令点权为点的编号。

那么，由点权两两不同，可知$C(T)$ 是唯一的，且堆性质是严格的（祖先严格大于子孙）。

我们先不考虑修改。对于T 构造出$C(T)$，可以发现：

性质1 $G(T)$ 中的边在$C(T)$ 中必然是返祖边。

因为$C(T)$ 的横叉边对应的两个点在$C(T)$ 中的最近公共祖先必然异于两点，权值严格大于两点，且在T 中两点之间的路径上，故由$G(T)$ 的定义这样的边不会在$G(T)$ 中存在。

性质2 $G(T)$ 中若存在边(u, v)，且在$C(T)$ 中u 是v 的祖先，则要么$f_{C(T)}(v) = u$，要么边$(f_{C(T)}(v), u)$ 在$G(T)$ 中存在。

显然成立：在T 中$f_{C(T)}(v)$ 到v 的路径和v 到u 的路径上都不存在大于$\min(f_{C(T)}(v), u)$ 的点，那么$f_{C(T)}(v)$ 到u 的路径作为上述两条路径并的子集也不会含有这样的点。

性质3 $G(T)$ 中如果同时存在边$(u, v), (u, x)$ 使得在$C(T)$ 中u 是v, x 的祖先且v, x 不互为祖先，那么u 在$C(T)$ 中一定是v, x 的最近公共祖先。

假设存在这样的两条边，那么T 中v 到u 再到x 这条路径必然包含v 到x 的路径，也就包含$C(T)$ 中v,x 的最近公共祖先，由于该点权值大于v 和w 且两条边存在，由$G(T)$ 的定义该点只能为u。

那么，我们可以得到，将$G(T)$ 的边连在$C(T)$ 中，必然是每个点向自己的每棵子树中的一条自顶向下的链上的所有点连边。而某个点u 对应的各个子树中链的底端实际上就是在T 中所有与u 相邻的权值小于u 的点。

于是，我们可以每次重建$C(T)$ 并处理出上述每个点相邻的点组成的链。我们可以发现原问题（不修改）等价于这样一个问题：

给定一棵有根树$C(T)$，除根外以每个点为链顶有一条向子树中延伸的链，链上的所有点向链顶的父亲连边，每次查询两点之间的最短路。

我们有以下性质：

性质4 $G(T)$ 中任意两点之间最短路上的点编号先递增后递减，即最短路中不存在一个点使得其编号小于最短路中与之相邻的两点。

否则，直接删掉该点即可得到一条更短的路径。

于是，我们设询问最短路的点为u,v，最短路中最大的点为w，则w 必然是$C(T)$ 中u,v 的公共祖先，最短路也就等于u,v 分别跳到w 的最小步数之和。

那么，给定点u 和其祖先w，如何求u 到w 所需的最小步数呢？

我们有贪心方法：**每次从u 跳到其相邻的点中深度最小且在w 子树中的点。**

这个方法是正确的，因为考虑最优决策中第一步不满足该方法的点x，x 必然是该方法求出的到达点y 的子孙，那么根据性质2，可以把方案中x 及其后所有是y 的子孙的点替换成一个y，方案不会变劣。

于是我们就可以按照贪心策略$O(n)$ 处理出询问点所有祖先的路径长度并枚举祖先求答案了。

时间复杂度：$O(qn)$。

4.3 子任务3

在子任务2 的基础上，我们要支持快速询问。直接每次预处理的时间复杂度为$O(n^3)$，难以接受。

对于询问u,v，我们设u,v 的最近公共祖先为l，设u,v 各自跳到某个l 的祖先的最小步数分别为a,b，根据前述结论，答案显然不小于$a+b$。我们还可以得到答案不大于$a+b+1$：设两个点各贪心地向上跳a,b 步后，分别到达x,y，不妨

设x是y的祖先，那么考虑跳到x的上一步z，l必然是z的祖先，而y又是l的祖先，由性质2可得$G(T)$中存在边(x,y)，这样就有了一条长$a+b+1$的路径，故答案不超过$a+b+1$。

于是，只要判断答案是否等于$a+b$即可。答案等于$a+b$意味着u,v分别向上跳a,b步后可能到达同一个点。根据贪心方法，我们先让u,v各自向上贪心地跳$a-1,b-1$步到达x',y'，则x',y'分别在l的不同儿子的子树中，故根据性质3，如果存在长$a+b$的路径，则它们下一步必然都跳到l。于是只要判定x',y'各自能否到达l即可。

建立一棵树$F(T)$，设每个点在$C(T)$中贪心向上跳的点为其在$F(T)$中的父亲。

在无修改的情况下，预处理$F(T)$可以按$C(T)$中的拓扑序直接枚举计算（从每个点对应的底端向上标记所有未标记点），每个点贪心地跳若干步后的位置和第一次跳到某个点祖先的步数可以用树上倍增、树链剖分或Link-Cut Tree等多种算法/数据结构计算。

我们在每次修改时预处理$T,C(T)$及$F(T)$，并用树链剖分等方法支持快速查询$C(T)$中点的最近公共祖先，$F(T)$中某个点的k级祖先以及祖先中第一个不小于l的点，查询时二分判定即可。

时间复杂度：$O(n^2+q\log n)$或$O(n^2+q\log^2 n)$。

4.4 子任务4

子任务4是一道集训队作业自选题：最短路练习题[1]。

原题做法为倍增，总时间复杂度为$O(q\log n)$，不详叙。我们介绍一种线性算法。

我们仍然建立笛卡尔树，注意到序列比一般树多了一些特殊性质，包括**每个点仅与至多两个比它大的点连边**。这其中必有一条边是$C(T)$的树边，故非树边至多只有一条。进一步观察可得，某个点u的非树边指向其祖先中最近的“改变方向”（即，在u的祖先序列中，该点的相邻两点在其同侧）的节点的父亲。利用这个性质，我们可以预处理父边方向改变次数的到根前缀和，并$O(1)$计算出跳到一个祖先的最小次数。

于是，我们只要使用$O(n)-O(1)$的最近公共祖先算法就可以将复杂度优化

[1] 见https://saac.cs.tsinghua.edu.cn/#!/contest/23/problem/137。

到$O(n+q)$。

当然，直接使用上一题的算法——只需要一次预处理，也可以做到$O((n+q)\log n)$ 的时间复杂度。

时间复杂度：$O(n+q)$ 或$O((n+q)\log n)$。

4.5 子任务5

在子任务4 的基础上进行维护，使用离线的重链剖分或Link-Cut Tree 等算法即可。

时间复杂度：$O((n+q)\log n)$ 或$O((n+q)\log^2 n)$。

4.6 子任务6

由于随机树树高较小（$O(\sqrt{n})$），可以直接使用暴力方法维护和计算。

时间复杂度：$O((n+q)\sqrt{n})$。

4.7 子任务7

使用子任务3 的算法——只进行一次预处理即可解决此子任务。

时间复杂度：$O((n+q)\log n)$ 或$O((n+q)\log^2 n)$。

4.8 子任务8,9

考虑插入一个叶子这种修改。我们考察在v 处插入u 对$C(T)$ 的影响：$C(T)$ 中新增了点u，其被插入到了v 的祖先中编号大小合适（满足堆性质）的位置。原来$C(T)$ 中的链的两端点都不受影响，新增了一条从u 到v 的链（如果$u>v$ 的话）。查询一个点是否在某条链上可以使用简单的平衡树或Link-Cut Tree 维护。

考虑如何维护$F(T)$。我们使用Link-Cut Tree 来维护：每个点如果和其$C(T)$ 上父亲在$F(T)$ 上的父亲（称为“后继”）相同，则点权为0 并向$C(T)$ 中父亲连边，否则点权为1 并向后继连边。这样，在查询时就可以通过提取出一个点到根路径的平衡树进行二分查找了。修改时，在$C(T)$ 上从u 向上暴力枚举，对于每一条覆盖到的后继比v 小的链，在Link-Cut Tree 上修改相应的链即可。

Link-Cut Tree 的修改（Link,Cut,修改点权）次数等于$C(T)$ 中覆盖到的较小链的条数。这相当于一棵一般的Link-Cut Tree 上的Access 操作，故总次数是$O(n\log n)$ 的。

总时间复杂度为$O(n\log^2 n + q\log n)$。

$C(T)$ 可以离线维护来降低常数。

如果常数过大，或者使用了$O(n\log^2 n + q\log^2 n)$ 等较差的算法，可能只能通过子任务8。

时间复杂度：$O(n\log^2 n + q\log n)$。

5 数据生成方式

基本逻辑是，先生成所有修改，然后生成操作。如无特别说明或性质的限制，生成操作时，若当前树的大小为s，则以$\frac{2n^{1.5}}{1.5qs^{0.5}}$ 概率生成修改，否则生成询问，询问点在当前T 的点中独立均匀随机两次。

生成修改的方式如下：

5.1 直接生成原树

直接生成原树的修改生成方式有以下几种：

1. 先利用Prüfer 序列随机生成最终的T，然后每次修改在所有与已知节点相邻的点中等概率随机选一个。

2. 先用$\frac{2n}{\log_2 n} \pm O(1)$ 个点建成一棵完全二叉树（如果是链则用所有点建立完全二叉树），然后修改保证链的切换次数最大，此时标准算法的预处理时间达到最大。

3. 先用$n - \frac{q}{n} \pm O(1)$ 个点生成一棵星型树（即除1 以外的点全部向1 连边），然后将剩下的点（称为额外节点）随机连在星型树当前的某些叶子上，询问时随机查询一个星型树部分的点和一个额外节点，此时$G(T)$ 接近完全图，$O(n^2q)$ 的BFS 算法的运行时间达到最大。

5.2 生成笛卡尔树

其余情况下，先生成了笛卡尔树，然后在此基础上随机生成一棵符合该笛卡尔树的原树，再随机生成修改，每次修改在所有与已知节点相邻的点中等概率随机选一个。

生成笛卡尔树的方法有以下几种：

1. 点i 的父亲为$i+\min(n-i,5,\text{rndshort})$。其中，rndshort 函数以$2^{-x}$ 概率返回x。这样生成的树高较大。

2. 点i 的父亲有$\frac{1}{2}$ 概率为n，否则在$[i+1,n]$ 中随机选择。这样生成的树最大点度较大。

3. 点i 的父亲在$[i+1,n]$ 中随机选择。这样生成的树树高较小。

在此基础上，生成原树的方法有以下几种：
生成原树时，需要确定每个点i 在自己的每个笛卡尔树子树S 中与谁连边。

1. 在S 中随机 $\sqrt{n}$ 个点，取最大的连边。

2. 从S 的根开始走，每次以0.6 的概率停止，否则随机走向一棵子树。最终停止的点和i 连边。这样生成的树询问答案较大。

3. 当要求性质二时，随机打乱子树顺序，然后得到笛卡尔树的一个深度优先遍历序，并按此顺序连成一条链。

6 题目难度及分数分布估计

子任务一是朴素算法，知识、思维、代码难度都很低。预计全部参赛选手都能够通过。

子任务二一般需要使用笛卡尔树或类似模型，并发现贪心策略。这一步的思维难度较大，同时也是优化算法的基础和本题最困难的部分之一。预计约有50% 的参赛选手能够解决。

子任务三需要更多的性质和一些数据结构，实现有一定难度，不过在子任务二基础上思维难度不高。预计约有32 – 40% 的参赛选手能够解决。

子任务四是集训队作业题。尽管难度不低，但参赛选手对这个问题比较了解，预计有60 – 80% 的选手能够解决。

子任务五是子任务四的扩展，多了数据结构，实现难度也较大，预计有25 – 40% 的选手能够解决。

子任务六数据随机，许多算法的期望效率比较优秀。不需要利用笛卡尔树模型，实现难度也不大，预计有50 – 70% 的选手能够解决。

子任务七与子任务三的解法类似，多了数据结构的一些细节。预计有25 – 40% 的选手能够解决。

子任务八已经和正解类似，只是数据范围较小，允许一些效率稍差的解法。预计有8 – 15% 的选手能够解决。

完整问题的思维难度和实现难度都很大，标程长度约8KB，能解决的选手预计不超过8%。

总体来说，这是一道需要选手发现和推导较多性质的数据结构题，同时具有较大的实现难度，是一道难度近于近年CTSC 的难题。

7 命题思路及过程

这道题目最早来源于最短路练习题。我在讨论该问题的过程中，发现其平面嵌入的对偶图含有原序列的笛卡尔树，且是三角剖分图，利用笛卡尔树和三角剖分图的性质比较系统地解决了该问题。

在之后的研究中，我通过挖掘三角剖分图的性质命制了MIN&MAX I和MIN&MAX II[2] 两道题目。后来我试图把序列推广到树上，但这样破坏了平面图性质，平面图方面的扩展失败了。

于是我开始研究图的其它性质。对于继承原题的最短路问题，我发现利用笛卡尔树仍可将其转化为链和贪心策略的模型。经过进一步研究，又发现了答案上下界的性质。

之后，我又研究了带修改的问题。最初的想法是连边和删边操作，我考虑了按大小启发式合并等方法，但这会导致笛卡尔树合并变得非常复杂，且原本依赖均摊分析的算法很难同时快速支持插入和删除。而只连边或只删边又会使题目变得很不优美。后来，根据维护连边的启发式方法，我发现加入一个叶子

[2] 见https://loj.ac/contest/22/problem/3 和https://loj.ac/contest/22/problem/5 。

的修改操作复杂度藉由类似Link-Cut Tree 的均摊分析有保证，故最终选择了这样的修改方式并命制了本题。

8 总结及展望

本题主要是应用笛卡尔树模型及其在树上的扩展并利用Link-Cut Tree 维护。笛卡尔树模型在解决区间最值问题上是一个重要的模型，能够把区间最值转化为树上公共祖先。本题利用笛卡尔树模型比较系统且简单地解决了关于树上路径最值生成的图的最短路问题，对原来最短路练习题的孤立模型做了一些扩展。笛卡尔树模型的应用还有很大的延伸空间。而Lick-Cut Tree 的灵活应用也是解决本题的关键。

在本题的模型上，关于无修改的问题是否能支持$O(1)$ 查询，以及连边删边等操作能否维护，最大和最小同时连边的图有何性质等问题，我没能得到较好的结论，欢迎讨论。

在竞赛方面，IOI经常出现这类既需要利用经典模型和理论，又要发现推导一些问题个例的性质才能解决的问题；尤以日本信息学竞赛中这类问题十分常见。这类综合问题出现在竞赛中对选手的锻炼和价值是很大的。