树链剖分

树链剖分的思想及能解决的问题

树链剖分用于将树分割成若干条链的形式，以维护树上路径的信息。

具体来说，将整棵树剖分为若干条链，使它组合成线性结构，然后用其他的数据结构维护信息。

树链剖分（树剖/链剖）有多种形式，如**重链剖分**、**长链剖分**和用于 Link/cut Tree 的剖分（有时被称作“实链剖分”），大多数情况下（没有特别说明时），“树链剖分”都指“重链剖分”。

重链剖分可以将树上的任意一条路径划分成不超过 $O(\log n)$ 条连续的链，每条链上的点深度互不相同（即是自底向上的一条链，链上所有点的 LCA 为链的一个端点）。

重链剖分还能保证划分出的每条链上的结点 DFS 序连续，因此可以方便地用一些维护序列的数据结构（如线段树）来维护树上路径的信息。

如：

1. 修改**树上两点之间的路径上**所有点的值。
2. 查询**树上两点之间的路径上**结点权值的**和/极值/其它（在序列上可以用数据结构维护，便于合并的信息）**。

除了配合数据结构来维护树上路径信息，树剖还可以用来 $O(\log n)$ （且常数较小）地求 LCA。在某些题目中，还可以利用其性质来灵活地运用树剖。

重链剖分

我们给出一些定义：

定义**重子结点**表示其子结点中子树最大的子结点。如果有多个子树最大的子结点，取其一。如果没有子结点，就无重子结点。

定义**轻子结点**表示剩余的所有子结点。

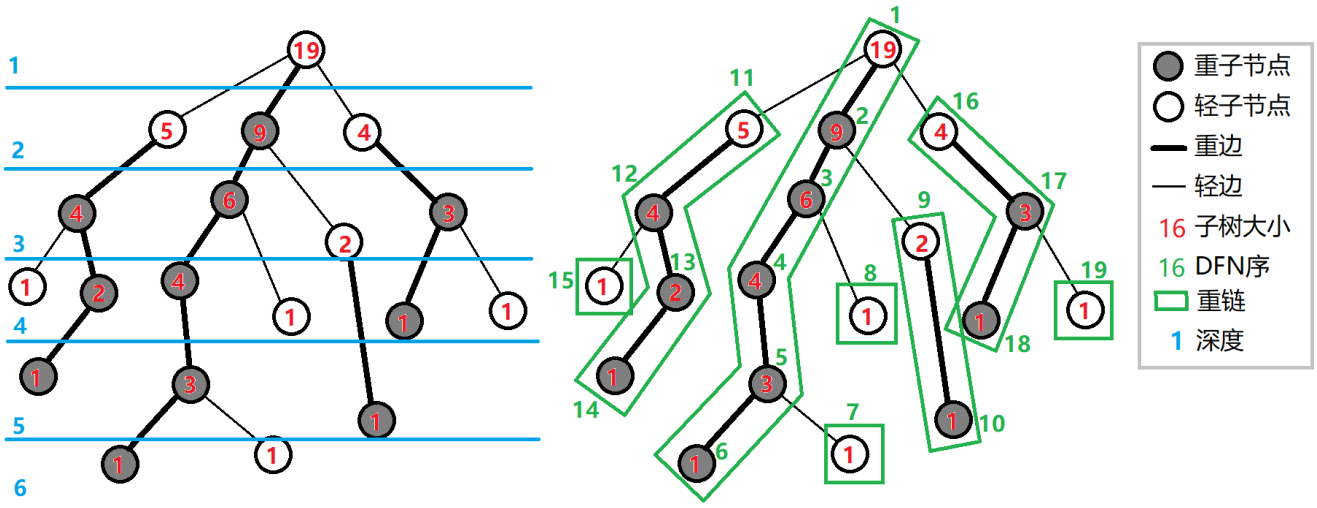
从这个结点到重子结点的边为**重边**。

到其他轻子结点的边为**轻边**。

若干条首尾衔接的重边构成**重链**。

把落单的结点也当作重链，那么整棵树就被剖分成若干条重链。

如图：



实现

树剖的实现分两个 DFS 的过程。伪代码如下：

第一个 DFS 记录每个结点的父节点（father）、深度（deep）、子树大小（size）、重子结点（hson） $$ \begin{array}{l}\text{TREE-BUILD}(u,dep)\\ \begin{array}{ll} 1 & u.hson\gets 0\\ 2&u.hson.size\gets 0\\ 3 & u.deep\gets dep\\ 4&u.size\gets 1\\ 5 & \textbf{for}\ \text{each}\ u\text{'s son}\ v\\ 6 & \qquad u.size\gets u.size+\text{TREE-BUILD}(v,dep+1)\\ 7 & \qquad v.father\gets u\\ 8 & \qquad \textbf{if}\ v.size>u.hson.size\\ 9 & \qquad\qquad u.hson\gets v\\ 10 & \textbf{return}\ u.size \end{array}\end{array} $$ 第二个 DFS 记录所在链的链顶（top）（应初始化为结点本身）、重边优先遍历时的 DFS 序（dfn）（DFS 序对应的结点编号）（rank） $$ \begin{array}{l}\text{TREE-DECOMPOSITION }(u,top)\\ \begin{array}{ll} 1 & u.top\gets top\\ 2 & tot\gets tot+1\\ 3 & u.dfn\gets tot\\ 4 & rank(tot)\gets u\\ 5 & \textbf{if}\ u.hson\ \text{is not}\ 0\\ 6 & \qquad\text{TREE-DECOMPOSITION}(u.hson,top)\\ 7 & \qquad\textbf{for }\text{each }u\text{'s son }v\\ 8& \qquad\qquad\textbf{if }v\text{ is not }u.hson\\ 9 & \qquad\qquad\text{TREE-DECOMPOSITION }(v,v) \end{array}\end{array} $$ 以下为代码实现。

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E6%A0%91%E9%93%BE%E5%89%96%E5%88%86_lgwza&rev=1594106140

Last update: **2020/07/07 15:15**