# 虚树

## 算法简介

一种用于加速树上 $\text{dp}$ 算法，时间复杂度 $O(k\log k)$，其中 $k$ 为树上的关键节点数。

## 算法思想

树上关键点数量较少时很多节点不需要进行 $\text{dp}$，考虑重建一棵树，只保留必要的节点。

发现所有关键节点的 $\text{LCA}$ 必须保留，但如果暴力枚举每个关键节点的 $\text{LCA}$ 并考虑是否保留时间复杂度至少为 $O(k^2)$。

事实上只需要将关键点序列$\text{dfs}$ 序排序，然后枚举序列中相邻节点的 $\text{LCA}$。

可以证明该方法枚举得到的 $\text{LCA}$ 不会有遗漏。假设 $p=\text{LCA}(u,v)$，且 $u,v$ 不是序列中相邻的节点。

于是必有 $u,v$ 属于 $p$ 的不同子树，考虑取序列中与 $v$ 相邻且 $\text{dfs}$ 序小于 $v$ 的节点，记为 $v_2$。

首先易知 $v_2$ 也位于 $p$ 的子树。于是如果 $v_2$ 与 $v$ 不在同一棵子树，那么 $\text{LCA}(v_2,v)=p$。

否则把 $v$ 换成 $v_2$，继续重复上述操作，最后总有 $v_{k-1},v_k$ 不在同一棵子树(最差的情况是 $v_k=u$)，证毕。

然后为了保证最终选取的点一定会构成树而不是森林，考虑强制将根节点加入虚树或者构造一个虚根。

接下来是建边过程，考虑用单调栈维护从根节点出发的一条树链。

每新加入一个节点时，先计算新节点与栈顶节点(事实上栈顶节点一定是上一次加入的节点，即与新元素相邻的节点)的 $\text{LCA}$，记为 $p$。

如果 $p$ 恰好为栈顶节点，则直接将新节点入栈。

否则，将栈顶节点弹出直到栈顶栈顶节点的下一个节点的 $\text{dfs}$ 序不大于 $p$ 的$\text{dfs}$ 序。

注意到每当栈顶节点被弹出时他在虚树中的位置已经确定，可以直接将他与下一个栈顶节点连一条边。

然后如果 $p$ 恰为栈顶栈顶节点的下一个节点，则将栈顶节点弹出并将他与下一个栈顶节点连一条边。

否则将栈顶节点弹出并将他与 $p$ 连一条边，再将 $p$ 加入栈。这一系列操作结束后再将新节点入栈。

所有关键节点都访问结束后将栈的剩余元素出栈并连边。

## 算法模板

洛谷p2495

## 题意

给定一棵以 $1$ 为根的边权树，设切断一条边的费用为这条边的边权。接下来 $q$ 次询问。

每次询问给定 $k_i$ 个节点(保证不含根节点)，问使得这 $k_i$ 个节点与根节点不连通的最小费用和为多少。

## 题解

首先考虑 $\text{dp}$ 过程，有状态转移方程

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

Permanent link:
**https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E8%99%9A%E6%A0%91&rev=1595765106**

Last update: **2020/07/26 20:05**