树上问题

租酥雨

2021 年 8 月 19 日

outline

- 树的基础内容
- 链分治
  - 重链剖分
  - 长链剖分
- 树上启发式合并 (dsu on tree)
- 虚树
- 树分治
  - 点分治
  - 边分治

什么是树

树就是一张有 n 个节点和 $n-1$ 条边的连通图。

连通，无环，点数 $=$ 边数 $+1$。以上三个条件中任意两者都可以推出第三者。

对于有根树，可以自然地定义父亲、儿子、祖先、子孙等关系。分别称所有祖先和子孙构成的点集为**祖先链**和**子树**。同时，定义一个点的深度为到根路径的长度。

从根出发对有根树做深度优先搜索可以得到 DFS 序，**子树**中的所有点在 DFS 序上是一个连续的区间，但**祖先链**并不满足：经过重链剖分后，祖先链在 DFS 序上被分成不超过 $\log n$ 个区间。

最近公共祖先

树上两点 x, y 的最近公共祖先 (Lowest Common Ancestor, LCA) 为两点祖先链的交集中深度最大的点。

LCA 的（至少）四种求法?

最近公共祖先

树上两点 x, y 的最近公共祖先 (Lowest Common Ancestor, LCA) 为两点祖先链的交集中深度最大的点。

LCA 的（至少）四种求法?

倍增、重链剖分都是 $O(n \log n) \sim O(\log n)$，而欧拉序 + ST 表的做法可以 $O(n \log n) \sim O(1)$ 实现更优秀的复杂度，此外，存在一种被称作 Tarjan LCA 的算法，可以把求 LCA 问题的复杂度规约到并查集的复杂度，但缺点在于必须离线所有询问。

利用 LCA 可以简便地计算树上任意两点之间的距离：
$dis(x, y) = dep(x) + dep(y) - 2dep(lca(x, y))$，其中 $dep(x)$ 为 x 节点的深度。

树的直径

一棵树的直径被定义为相距最远的点对之间的距离。

所有直径都会交于同一点，有时候会称这个点为“中心”（$\neq$ 重心）。

求树的直径的两种方法：树 DP，或者两遍 DFS：对于树上的任意一个点 x 而言，任意一个离他最远的点 y 都会是某条直径的一个端点。

关于直径有一个比较有趣的结论：我们记 $diam(S)$ 表示树上点集 S 的直径（仍用原树中的距离来定义），那么 $diam(S \cup T)$ 的两端点必然是 $diam(S), diam(T)$ 各自端点这四个点中的两个。这为我们提供了快速合并直径的方法：只需要记录直径端点即可（但需要支持快速求距离）。

例题：子树直径

description

一棵 n 个点的树，q 次询问，每次询问给出 k 个点 $x_1, x_2, \cdots, x_k$（保证不存在祖孙关系），问去掉以这些点为根的子树后，剩下的部分（仍是一棵树）的直径是多少。

constraint

$n, q \leq 3 \times 10^5$.

例题：子树直径

tutorial

去掉若干棵子树后，剩下的部分仍然是 DFS 序上若干个连续的区间。

线段树维护直径即可。使用 ST 表来查询 LCA 可以实现 $O((n+q)\log n)$ 的复杂度。

树的重心

对于树上的一个点 s，我们定义 $mss(s)$ 为以 s 为根时，其儿子中最大的子树大小。对于一棵树，我们定义 s 是这棵树的重心，若 s 可以使得 $mss(s)$ 取到整棵树上的最小值。

我们可以依次地得到以下几个结论：

1. 重心等价于 $mss(s) \leq \lfloor \frac{n}{2} \rfloor$；
2. 若存在两个重心，则它们相邻，从而重心至多两个，且当重心有两个时，存在一条树边把整棵树分成大小相同的两部分；
3. 重心等价于最小化所有点到某一点的距离之和；
4. 用一条边连通两棵树时，新树的重心在原来两棵树重心的连线上。

证明可以参考这篇知乎专栏：https://zhuanlan.zhihu.com/p/357938161

LOJ6042 跳蚤王国的宰相

description

给一棵 n 个点的树，定义一次操作为，砍掉树的一条边，再用一条边把两棵树连成一棵。
对于每个点 x，你需要求出需要对原树操作至少多少次，才能使 x 成为重心。

constraint

$n \leq 10^6$.

LOJ6042 跳蚤王国的宰相

tutorial

要想让 x 成为重心，就需要把 x 的所有子树砍到大小不超过 $\lfloor \frac{n}{2} \rfloor$。当 x 不是原重心时，显然只有原重心所在子树是不满足条件的。

砍原重心的儿子一定是一种不会更劣的方案，而砍完可以直接连到 x 上，也就是说砍掉的部分无须再考虑，我们只需要砍掉原重心的一些儿子，使以 x 为根时的最大子树大小不超过 $\lfloor \frac{n}{2} \rfloor$。

贪心选最大的砍掉即可。同时这题还有一个结论是除了原重心以外，所有点的答案的差不超过 1。

树的链并

“树的链并”一般是指对于一个树上点集 S，维护 S 中所有点的祖先链的并这样一个结构。

做法就是把所有点按照 DFS 序排好序，加上每个点的祖先链，然后减去相邻点的 LCA 的祖先链。可以归纳证明其正确性。

当然也可以考虑使用树上差分，在 S 中的每个点上打上加法标记，在相邻 LCA 处打上减法标记，不难验证此时某个点的子树和为 1 等价于这个点在链并里。

往往会使用 `std::set` 来维护点集，每次插入一个新点时，需要查询其前驱后继以更新相邻 LCA 关系。

HDU5709 Claris Loves Painting

description

给一棵 n 点的树，每个节点上有一个颜色 c_i，q 次询问一个点的子树中与这个点距离不超过 d 的点的颜色有多少种。
(强制在线)

constraint

$n, q \leq 5 \times 10^5$.

HDU5709 Claris Loves Painting

tutorial

按深度离线，对每种颜色维护链并，需要实现一个数据结构支持单点加以及子树（区间）求和。

(强制在线的话只需要把这个数据结构可持久化就行了。)

重链剖分

每个点选取子树大小最大的儿子作为重儿子，把树剖分成若干条链。

称非重儿子的儿子为轻儿子，由重儿子的父向边连成的链为重链。

由于每从一个轻儿子跳到父亲时子树大小至少增大一倍，因而一个点到根的路径上经过的轻边不超过 $\log n$，也即路径上重链条数是 $O(\log n)$。

任意两点间的路径可以被拆分成不超过 $\log n$ 段**重链前缀**加上不超过一段**重链区间**。

LOJ6669 Nauuo and Binary Tree

description

这是一道交互题。
有一棵以 1 号点为根的二叉树。你可以询问交互库“树上某两点间的距离是多少”不超过 30000 次。
你需要还原出二叉树的结构。由于左右儿子是无法分辨的，因此你只需要回答每个节点的父节点是谁就可以了。

constraint

$n \leq 3000$.

LOJ6669 Nauuo and Binary Tree

tutorial

先通过 $n-1$ 次询问得到每个点的深度。按深度从小到大增量地还原整棵树。

每次尝试加入一个节点 x 时，对树进行树链剖分，假设根所在的重链的最后一个点是 y，那么询问 $dis(x,y)=dep(x)+dep(y)-2dep(lca(x,y))$，就可以唯一确定 $lca(x,y)$（因为 $lca(x,y)$ 一定在重链上，而重链上的点深度各异）。

于是 y 就一定在 $lca(x,y)$ 的另一个儿子的子树里（当然有可能就是 $lca(x,y)$ 的儿子）。递归处理，会发现得到了一个规模不超过原问题规模一半的子问题，因此需要的询问次数不超过 $\log n$。

长链剖分

每个点选取子树内最深深度最大的儿子作为长儿子。

长链剖分可以解决一些与深度有关的信息的维护问题。具体地，一般会维护 $f_{i,j}$ 表示 i 节点子树内和 i 距离 j 的某些信息，如果此时 i 是 fa_i 的长儿子，那么 $f_{fa_i,j}$ 可以直接由 $f_{i,j-1}$ 继承而来，通过一些技巧（比如指针）可以 $O(1)$ 实现这个继承过程。如果此时再暴力遍历所有短儿子的 f 数组计算答案并合并信息，那么总复杂度等同于遍历了树上的所有长链，因此是 $O(n)$ 的。

CodeForces1009F Dominant Indices

description

给定一棵以 1 为根，n 个节点的树。设 $d(u,x)$ 为 u 子树中到 u 距离为 x 的节点数。
对于每个点，求一个最小的 k，使得 $d(u,k)$ 最大。

constraint

$n \leq 10^6$.

CodeForces1009F Dominant Indices

tutorial

$f_{i,j}$ 表示节点 i 子树内和 i 距离为 j 的点数，再维护和变量记录 f_i 的最大值出现位置，转移比较简单。

例题

description

给出一棵 n 个点的树，每条边都有一个正整数权值，要求对于每个点 u，求出在 u 子树中，经过点 u 的，边数在 $[L_u, R_u]$ 之间的最长链。

constraint

$n \leq 5 \times 10^5, 1 \leq L_u \leq R_u \leq n.$

例题

tutorial

同一棵子树内，相同深度的节点中只需要保留到根权值和最大的那一个。

每次合并两棵子树时，把较短的合并到较长的上去，这样合并的总量是 $O(n)$ 的。(这是由于每合并一次就少一个点，否则总量会达到 $O(n \log n)$。)

对每条长链开一棵线段树维护，时间复杂度 $O(n \log n)$。

树上启发式合并（dsu on tree）

用于解决一些难以快速合并的子树信息询问。

算法流程是当处理到点 u 时，先递归处理 u 的所有轻儿子并在回溯时清空已统计信息，再递归处理 u 的重儿子，再遍历所有轻儿子统计子树信息，此时便可以根据已统计的点 u 的子树信息处理一些询问。

每条轻边会使其子树内的所有点被暴力遍历一次，而每个点的祖先轻边数量都不超过 $\log n$，因此算法的总时间复杂度为 $O(n \log n)$。

CodeForces741D Arpa's letter-marked tree and Mehrdad's Dokhtar-kosh paths

description

给一棵 n 个点的树，每个点上有一个字符，字符集大小为 σ。
定义一条路径是好的当且仅当这条路径上的所有字符经过重新排列后可以形成回文串。
求每个点子树内最长的好的路径的长度。

constraint

$n \leq 2 \times 10^5, \sigma \leq 22.$

CodeForces741D Arpa's letter-marked tree and Mehrdad's Dokhtar-kosh paths

solution

一条路径是好的当且仅当只有至多一种字符出现奇数次。

由于严格在子树内的路径可以由子树的答案取 $\max$ 继承上来，因此在计算 ans_u 时，只需要考虑经过 u 的好的路径即可。

状压，记 f_i 表示子树内到根（注意不是 u）路径上每种字符出现次数奇偶性状态为 i 的点的最大深度，加入一棵子树时，先利用桶中信息更新答案，再把整棵子树加入桶。每次更新答案时都有 $\sigma+1$ 种可能的情况，因此时间复杂度为 $O(n\sigma\log n)$。

虚树

对于给出的一棵树，有时我们只需要用到其中一个点集的信息。

我们希望能用尽量少的点来表示出这个点集并且维持原有树结构，只需要在这个点集之外额外加入两两点的 LCA——进一步地，只需要加入 DFS 序相邻两点的 LCA。

把额外的 LCA 加入点集，可以使用模拟栈的方法来线性构建出虚树结构：按 DFS 序遍历所有点，用栈模拟 DFS 的递归栈，每访问到一个点时，弹栈，直到栈顶元素是该点的祖先。

SDOI2011 消耗战

description

给一棵 n 个节点的树，边带权。
有一些询问，每次询问给出 k_i 个点，要求删除边权和最小的边集使得从 1 号点出发不能到达这 k_i 个点中的任意一个。

constriction

$n, \sum k_i \leq 10^6$.

SDOI2011 消耗战

tutorial

对给出的 k 个点建虚树，简单 DP 即可。

需要求出树上路径最小值。实际上这个路径最小值可以用到根路径最小值代替。

HNOI2018 毒瘤

description

给一张 n 个点 m 条边的连通图，求独立集个数，对 10^9+7 取模。

constriction

$n \leq 10^6, n-1 \leq m \leq n+15.$

HNOI2018 毒瘤

tutorial

朴素做法是 $O(2^{m-n+1}n)$ 枚举非树边的选取后做一遍树 DP。

可以发现只需要把这些非树边涉及到的点拿出来建虚树后做 DP 就行了，由于每次虚树的形态是一样的，所以可以预处理出虚树上每条边从 $dp_{v,0/1}$ 转移到 $dp_{u,0/1}$ 的转移系数，复杂度为 $O(n+2^{m-n+1}(m-n+1))$。

点分治

选取树的重心作为分治重心，处理经过重心的所有信息，再将重心删除，递归处理剩下的子树。

由重心的性质可以保证剩下的子树大小均不会超过原树的一半，因此每个点只会被处理不超过 $\log n$ 次。

根据点分治的过程可以建出一棵点分树，这棵树的树高是 $O(\log n)$。

Luogu2634 聪聪可可

description

一棵 n 个点的树，边有边权，求有多少点对 (i,j) 满足树上 i 到 j 的路径上所有边边权之和是 3 的倍数。

constraint

$n \leq 10^6$.

Luogu2634 聪聪可可

tutorial

点分治，每次统计过重心的所有路径。对当前分治层的每个点求出该点到重心路径上的边权之和除以 3 的余数，合并两条路径时，只需要满足两端点来自重心的不同子树，且边权和 3 的倍数即可。

动态点分治

习惯上，我们把“记录点分树结构的点分治算法”称为动态点分治。

动态点分治之于点分治，相当于线段树之于普通序列分治。

因此，类似线段树地，动态点分治能帮助我们动态地实现一些询问以及修改操作。

Luogu6329 震波

description

一棵树，点有点权，支持两种操作：

- 修改某个点的点权；
- 查询与某个点距离不超过 k 的所有点的点权和。

强制在线。

constraint

$n, q \leq 10^5$.

Luogu6329 震波

solution

建出点分树，对于询问点 x，考虑枚举与 x 距离不超过 k 的一点 y 跟 x 在点分树上的 LCA（一定是 x 的祖先，由深度可知不超过 $\log n$ 个）。

假设 x 距离其某个祖先 z 的距离是 d，那么就只需要计算点分树上 z 子树中与 z 距离不超过 $k-d$ 的点权和即可，可以在每层分治重心处，构建一个以距离为下标的数据结构（如树状数组），记录权值和。

但这样可能会额外计算一些 y，它们与 x 的 LCA 并不是 z，而是 z 的子孙，因此需要对每个点额外构建一个数据结构记录该点子树对其父亲的贡献，查询时减一下即可。

无修改链上询问

有一类问题是给出一棵树，（不带修改）q 次询问一条链 (x, y) 上的某个信息。

- 倍增算法 $O(n \log n) \sim O(\log n)$；
- 树剖 $+$ 线段树 $O(n) \sim O(\log^2 n)$，一个优化是预处理重链前缀的信息，这样一次询问只需要做 $\log n$ 次前缀信息查表和一次线段树区间查询，复杂度 $O(\log n)$。

而点分治可以做到 $O(n \log n) \sim O(1)$，在单次询问的复杂度上可以说是非常的优秀了。

无修改链上询问

对于链上询问 (x, y)，考虑这条链经过的最上层分治重心 z（也即 x, y 在点分树上的 LCA）。于是我们可以把询问拆成 (x, z) 和 (z, y)。

只需要事先维护好所有 $(*, z)$ 与 $(z, *)$ 的信息，那么每次询问就都可以在**一次**信息合并下求出。当然，需要实现 $O(1)$ 求 LCA 这个子问题。

值得注意的是，这个做法只需要在一次信息合并后求出答案（也就是说并不需要合并出完整的信息，只需要答案就可以了），这与前面提到过的倍增和树剖都是有所不同的。下面会展示一个存在明显区别的例题。

Wannafly 挑战赛 24E 旅行

description

给一棵 n 个节点的树和一个常数 k，树上的每个点都有一个整数权值 $val_i \in [0, k)$。
有 q 组询问，每组询问给出 x, y，询问从树上 x 到 y 路径上的点集中选出一个子集其权值和是 k 的倍数的方案数，输出答案对 998244353 取模后的结果。

constraint

$n \leq 10^5, k \leq 50, q \leq 10^6$.

Wannafly 挑战赛 24E 旅行

tutorial

找到 x, y 在点分树上的最近公共祖先 z，在 z 点处理这组询问。

点分时，从每层点分重心出发求出到当前分治层每个点路径上所有 val_i 在模 k 意义下的循环背包。

两个循环背包的合并本需要 $O(k^2)$ 的时间复杂度，但由于最终答案只需求一项，并且只需要一次合并，因而总时间复杂度为 $O((n \log n + q)k)$。

边分治

顾名思义地与点分治的区别在于选取的分治位置是一条边。在选取问题上，我们类似地取“两端子树大小较大值 $mss(e)$ 最小”的那条边 e 作为分治重心边。

边分治在树上每个点度数都为常数的情况下，复杂度与点分治同阶（只需要常数轮分治就可以达到不劣于点分治的划分结果）但常数较大；当点度数很大时，边分治的复杂度会退化，因此需要先对原树进行“三度化”操作（见板书）。

边分治相较于点分治的优势在于每层分治的结构比较简单（只有两侧的两棵子树，而点分治的分治重心可能很多子树），在部分问题上，其解法会比点分治更优美。

边分治例题

description

给出两棵均为 n 个点的树，边带任意整数权值（可能为负），求两个点，最大化这两个点在两棵树上的路径边权和。

constraint

$n \leq 10^5$.

边分治例题

tutorial

对第一棵树进行边分治，把分治重心边两侧子树中的点分别标记为红色和蓝色。

只考虑所有跨越当前分治重心边（即红蓝匹配）的点对。把红点和蓝点都放到第二棵树上，建出虚树，在虚树上简单 DP 即可。

“只有红蓝两种颜色”构成了这个 DP “简单”的充分条件。

谢谢大家！

祝大家学业有成！