数据结构3

mrsrz

前言

- 课件仅用于辅助讲课，里面可能会有胡言乱语
- 之前大家学习了一些维护序列信息的数据结构
- 本节课我们来研究一下维护树上信息的方法

基础的树上操作

- 链加、链求和、子树加、子树求和。

树上差分

- 考虑只有链加，要求求出最后树上所有点/边的信息。
- 在序列上，可以使用差分的方法。在树上，也有类似的方法。

树上差分

- 对于 u 到 v 的链上所有点加 x 的操作，在点 u 处加 x，在点 v 处加 x，在点 LCA(u,v) 处减 x，在点 fa[LCA(u,v)] 处减 x。最后对整棵树求子树和即可。
- 由于最后的求子树和操作，前面对点 t 的修改，相当于最终对 t 到树的根这条路径上所有点都进行了相同的修改。
- 对于边权的情况，我们一般将点 u 到 fa[u] 的边视为 u 对应的边。这样的对应方式是唯一的。
- 对于 u 到 v 的链上所有点加 x 的操作，在点 u 处加 x，在点 v 处加 x，在点 LCA(u,v) 处减 2*x，最后对整棵树求子树和即可。

P3258

- 给定一棵树以及一个序列 a。
- 一个人要从 a_1 走到 a_2 再走到 a_3 再走到 a_4 … 最后走到 a_n，问每个点被经过的次数。

P1600 天天爱跑步

- 给定一棵树，每个点有点权 w_i
- 有很多人，第 i 个人 0 时刻在 s_i，走到 t_i 消失，走一条边需要 1 的时间
- 对每个点 i，求出 w_i 时刻恰好在点 i 的人数

树的 dfs 序

- 选定一个结点作为树的根，然后对树进行深度优先遍历。
- 记录每个点被第一次访问到的时间，这就是树的 dfs 序。
- 可以发现，在一棵子树当中的所有点的 dfs 序都是连续的，且子树的根结点的 dfs 序是最小的。
- 容易用一个区间表示出一棵子树的所有 dfs 序。

树的 dfs 序

- 将树的结点编号改成 dfs 序，那么子树就对应一个区间。
- 对子树的修改和查询，就可以像维护序列一样进行维护了。

重链剖分

- 树链信息维护的常用方法。
- 对于每个非叶子结点，我们将它的儿子分成两类：重儿子和轻儿子
- 其中，重儿子有且仅有一个，是该结点子树大小最大的儿子结点（如果有多个最大的，则任选一个），其它的为轻儿子
- 将一个点连到重儿子的边称为重边，其它的边为轻边。

重链剖分

- 可以发现，所有的重边组成了若干条自上而下的链，每个点恰好在一条重链里。
- 这样的结构有一个重要的性质：一个结点到父亲结点的边若是轻边，则这个点的父亲结点的子树大小至少是这个点本身的 2 倍。
- 这条性质保证了一个点到根节点的路径上最多只有 log n 条轻边。

最近公共祖先

- 利用重链剖分，我们可以得到一个求 LCA 的方法。
- 我们记录每个结点所在重链中深度最小的结点（称为链顶，记为 top）。
- 若 u,v 两个点在同一条重链上（链顶相同），则 LCA 就是它们中深度较小的点。
- 否则，我们比较 top[u] 和 top[v] 的深度，将较浅的那个点变成 fa[top[*]]。
- 重复上面的步骤就可以求出 LCA。

最近公共祖先

- 这种方法相比倍增法，它的预处理只需要 O(n) 的时空复杂度，且算法的常数较小，因此一般情况推荐使用该方法代替倍增法。

维护链信息

- 考虑 u 到 v 这条路径，它中间会有 O(log n) 条轻边，这些轻边把整条路径划分为 O(log n) 个段，每段都是一条重链上的一个子链。
- 对于链上信息的维护，我们对树进行重链剖分以后，对每条重链分别使用数据结构维护。然后修改、查询的时候，对 O(log n) 个段分别在数据结构上查询，然后再进行合并。
- 这就是重链剖分维护信息的常规思路。

P3384 【模板】重链剖分/树链剖分

如题，已知一棵包含 N 个结点的树（连通且无环），每个节点上包含一个数值，需要支持以下操作：

- `1 x y z`，表示将树从 x 到 y 结点最短路径上所有节点的值都加上 z。
- `2 x y`，表示求树从 x 到 y 结点最短路径上所有节点的值之和。
- `3 x z`，表示将以 x 为根节点的子树内所有节点值都加上 z。
- `4 x` 表示求以 x 为根节点的子树内所有节点值之和

P3384 【模板】重链剖分/树链剖分

- 本题将基本的树上操作进行了结合。
- 考虑将 dfs 序和重链剖分相结合。
- 先对树进行重链剖分，然后再求 dfs 序。求 dfs 序的时候，我们先走重边，再走轻边。
- 可以发现，每条重链上的 dfs 序都是连续的，且自上而下递增。
- 这样，重链的子链，也能用 dfs 序上的一个区间进行表示。
- 使用线段树维护整棵树的信息即可。

P4092 [HEOI2016/TJOI2016] 树

- 给定一棵有根树，初始根结点有标记，有两种操作:
- 1. 给某个结点打上标记。
- 2. 询问一个结点的祖先中，离它最近的打了标记的结点。

轻重儿子分类维护

- 一种技巧。利用重链剖分的结构以及重儿子的唯一性，将轻重儿子的信息分开维护的方式。

P5305 [GXOI/GZOI2019] 旧词

给定一棵 n 个点的有根树，节点标号 $1 \sim n$，1 号节点为根。
给定常数 k。
给定 Q 个询问，每次询问给定 x, y。
求：

$$\sum_{i \le x} \mathrm{depth}(\mathrm{lca}(i, y))^k$$

$\mathrm{lca}(x, y)$ 表示节点 x 与节点 y 在有根树上的最近公共祖先。
$\mathrm{depth}(x)$ 表示节点 x 的深度，根节点的深度为 1。

P5305 [GXOI/GZOI2019] 旧词

- 将询问离线按照 x 排序，然后将结点按照编号从小到大插入到树上。
- 对每个点 u 维护 sz[u]*dep[u]^k，其中 sz[u] 表示 u 的轻儿子子树中当前已经插入的结点个数。
- 查询 y，对 y 的每个祖先 v，讨论 v 到 y 路径上第一条边的类型。

P4719 【模板】"动态 DP"&动态树分治

给定一棵 n 个点的树，点带点权。

有 m 次操作，每次操作给定 x, y，表示修改点 x 的权值为 y。

你需要在每次操作之后求出这棵树的最大权独立集的权值大小。

长链剖分

- 只需要将重链剖分中的“重儿子”改为深度最大的儿子结点，就是长链剖分。
- 性质一：一个点的 k 级祖先所在的长链长度至少为 k。
- 性质二：u 所在长链一定比 u 的轻儿子所在长链要长。因此有 u 到根的路径上的轻边数量是 sqrt(n) 级别的。

树上 K 级祖先（P5903）

- 长链剖分的一个经典应用。
- 使用倍增预处理出每个结点的 1,2,4,8,⋯,2^m 级祖先。
- 对于一条长度为 d 的长链的链顶 u，求出 u 的第 1,2,⋯,d 级祖先，和 u 所在长链上的第 1,2,⋯,d 个结点。
- 查询 k 级祖先时，我们先跳最大的 2^m 级祖先，然后跳到当前所在长链的链顶。此时 k 级祖先一定在链顶记录的这 2d 个点里。

长链剖分优化 dp

- 长链剖分可以优化一些与深度有关的 dp 问题。

CF1009F Dominant Indices

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。

CF1009F Dominant Indices

- 容易写出 O(n^2) 的 dp
- 令 f[i][j] 为 i 子树内距离 i 为 j 的点的个数。
- $f[u][i] = \sum_{s \in son_u} f[s][i-1]$ $f[u][0] = 1$

CF1009F Dominant Indices

- 由于 f[u][1..] 全是 0，因此对于第一个转移的子树，相当于把这个子树的 dp 值偏移一位，然后在 0 的位置放一个 1。因此可以直接将这个子树的 f 数组直接“继承”过来，复杂度 O(1)。
- 对树进行长链剖分
- 对于点 u，递归求出 u 的所有子树的信息
- 将 u 的重儿子的 dp 值继承到 u
- 将 u 的所有轻儿子的 dp 值合并上来
- 由于每条长链仅会在链顶被合并一次，因此总复杂度是 O(n)。

树上启发式合并

- 即 dsu on tree，是一种比较暴力的处理树上问题的方法。
- 利用树链剖分的结构，可以使复杂度变得很对。
- 可以解决部分无修改的查询子树信息的问题。

树上启发式合并

- dsu on tree 的一般步骤如下：
- 递归求轻儿子的答案
- 递归求重儿子的答案
- 将当前结点以及所有轻儿子子树内结点的贡献加入
- 求出当前结点的答案
- 如果当前是重链链顶，则清空所有子树的答案

树上启发式合并

- 考虑这个做法的复杂度。
- 每个点只会计算一次答案。
- 对于一条重链，它下面连的所有点都会在这条重链上被加入、清空一次。
- 由于一个点上面的重链最多 log n 条，因此它加入删除的次数也是 log n。
- 因此时间复杂度是 O(n log n) 的（假设加入、删除一个点的贡献是 O(1) 的）。

CF600E Lomsat gelral

- 给定一棵树，每个点有一个颜色。要求对每个点求出它子树内出现最多的颜色，并输出它子树内所有这种颜色的点的标号和。

点分治

- 处理树上路径信息的常用方法。
- 对一棵树，我们找到一个点 u，并处理所有经过 u 的路径的信息
- 然后，u 将整棵树分成了若干部分，对每部分递归执行上述操作
- 如果我们每次找的 u 满足 u 的每部分的结点个数不超过总数的一半，那么递归的层数就不超过 log n

树的重心

- 一个点作为根时，它的最大的儿子最小，称这个点为树的重心。求树的重心只需要一遍 dfs，维护子树大小即可
- 可以发现，点分治每次要找的那个点，就是树的重心

P3806 【模板】点分治 1

- 给定一棵有 n 个点的树，询问树上距离为 k 的点对是否存在。

P3806 【模板】点分治 1

- 点分治以后，只需要考虑如何统计经过点 u 的路径的信息
- 常用的做法有，将路径从 u 分成两段，统计 u 作为一个端点时的所有路径的信息，然后考虑合并两条不同子树中的路径信息。
- 对于本题，我们可以通过 dfs 求出 u 到所有点的距离，并使用桶记录每个距离的出现次数
- 然后对于点 v，将答案加上 k-dis(u,v) 的出现次数
- 这样会把两条来自 u 的同一棵子树的路径也统计进去。使用同样的方式去重即可
- O(n log n)

P4178 Tree

- 给定一棵 n 个节点的树，每条边有边权，求出树上两点距离小于等于 k 的点对数量。

P4178 Tree

- 和上一题类似，这次查询的是小于等于 k 的信息
- 其他过程都一样
- 对于点 v，将答案加上 0-k-dis(u,v) 的出现次数
- 使用树状数组或线段树来加速查询
- 时间复杂度 O(n log^2 n)

点分治

- 之前的例题，统计信息的时候都遇到统计两条来自同一子树的问题
- 有的题目中，无法通过减去同一子树信息的方式抵消这些贡献
- 可以按照子树大小建立哈夫曼树/带权分治，然后逐层计算贡献后合并
- 这是一种常见的代替边分治的方法

动态点分治/点分树

- 有时会需要修改一个点的信息，和查询一个点相关的路径信息
- 将点分治的结构保存下来，修改/查询的时候，在这个点所在的所有子树里进行修改/查询
- 这就是动态点分治/点分树

P6329 【模板】点分树 | 震波

- 给定一棵树，每个点有点权，有以下操作
- 0 x k 查询距离 x 不超过 k 的所有点的点权
- 1 x y 修改点 x 的点权为 y

虚树

- 尽管这是个 10 级算法，但是它本身不难，而且挺有用的

虚树

- 给定树上的 m 个关键点，要求用尽可能少的点来表示这 m 个点在树上的关系
- 虚树点集满足，任意点集内两个点的 LCA 都在点集里
- 因此一个 $O(m^2)$ 的建虚树的方法就是求出两两 LCA
- 事实上，根据 dfs 的性质，只需要求 dfs 序相邻的所有点对的 LCA

建虚树方法 1

- 将点集按照 dfs 序排序，求出所有相邻两个点的 LCA，去重后根据原树上的祖先后代关系建边。

建虚树方法 2

- 还是将点集按照 dfs 序排序
- 使用单调栈来建树，单调栈中维护一条自顶向下的链
- 每次加入一个点，若它是栈顶点的后代，则直接入栈
- 否则不断弹栈，弹出栈的点和栈中下一个点连边
- 直到栈顶下一个点是当前点的祖先，此时求出栈顶点和当前点的 LCA，弹栈，栈顶点和 LCA 连边，然后依次将 LCA 和当前点入栈

P2495 [SDOI2011] 消耗战

- 给定一棵树，有边权。
- 每次询问给出若干个关键点，要求断掉一些边，使得 1 和所有关键点都不连通。要求出最小的断掉的边的边权和

P2495 [SDOI2011] 消耗战

- 容易写出 O(n) 的 dp
- 令 f[u] 表示 u 的子树都无法到达 1 的最小代价，w[u] 表示 u 到父亲的边的边权
- 若 u 是关键点，则 f[u]=w[u]
- 若 u 不是关键点，则 f[u]=min(∑f[v], w[u])
- 对这些关键点建虚树，然后虚树上的边的边权为两点在原树上的路径的最小边权
- 然后再 dp 即可

WC2018 通道

- 给定三棵树，带边权。要求找到 u,v，使得三棵树上 u 到 v 的路径长度之和最大

WC2018 通道

- 一棵树的情况是树的直径，可以用经典的 dp 来解决。
- 考虑两棵树的情况
- 对第一棵树进行点分治，对这些点建虚树，在虚树上类似求直径
- 三棵树的情况可以点分治套点分治，$O(n \log^2 n)$
- 考虑 x 和 y 的贡献为 $d1(x)+d1(y)+dis2(x,y)+dis3(x,y)$
- 即 $d1(x)+d1(y)+d2(x)+d2(y)-2*d2(LCA(x,y))+dis3(x,y)$
- 利用直径的可合并性