图论三

树上差分、子树和与倍增、最近公共祖先

郑欣

2024 年 7 月 27 日

树上前缀和

问题 1

给一个序列 a，进行 q 次询问，每次选择一个区间 $[\ell, r]$，求 $a_\ell + a_{\ell+1} + \cdots + a_r$。

前缀和：令 $S_i = \sum_{j=1}^{i} a_i\ (0 \le i \le n)$，则 $\sum_{i=\ell}^{r} a_i = S_r - S_{\ell-1}$。

问题 2

给一棵树，每个点有点权 a_i。进行 q 次询问，每次选两个点 ℓ, r，求 ℓ 到 r 路径上的点权之和。

用 S_i 表示 1（假设根节点为 1）到 i 的点权和。

则 ℓ 到 r 的点权和为 $S_\ell + S_r - S_u - S_{p_u}$，其中 p_u 表示 u 的父亲节点，u 是链 $(1, \ell)$ 与 $(1, r)$ 的最深交点。

我们称 u 为 ℓ 和 r 的最近公共祖先。

最近公共祖先

（u 和 v 的）最近公共祖先（Lowest Common Ancestor, LCA）：同时是 u 和 v 的祖先且深度最大的点。记作 $\operatorname{lca}(u, v)$。

性质：

- $\operatorname{lca}(u, v) = u$ 当且仅当 u 是 v 的祖先。
- 若 $\operatorname{lca}(u, v) \notin \{u, v\}$，则 u 与 v 在 $\operatorname{lca}(u, v)$ 的两个不同子树中。
- $\operatorname{lca}(u, v)$ 在 u 到 v 的最短路上。

暴力

记 u 的深度为 dep_u，不妨设 $\text{dep}_u \geq \text{dep}_v$。

- 当 u 的深度大于 v，不断将 u 变为 u 的父亲 p_u，直到 $\text{dep}_u = \text{dep}_v$。
- 同时令 $u \leftarrow p_u$，$v \leftarrow p_v$，直到 $u = v$。

每组询问时间复杂度 $O(n)$。

倍增

Step 1: 当 $\mathrm{dep}_u > \mathrm{dep}_v$，令 $u \leftarrow p_u$，直到 $\mathrm{dep}_u = \mathrm{dep}_v$。

用 $\mathrm{anc}_i(u)$ 表示 u 向上走 i 步得到的祖先。

直接令 $u \leftarrow \mathrm{anc}_{\mathrm{dep}_u - \mathrm{dep}_v}(u)$，则 $\mathrm{dep}_u = \mathrm{dep}_v$。如何快速求出 $\mathrm{anc}_{\mathrm{dep}_u - \mathrm{dep}_v}(u)$？

预处理 $\mathrm{anc}_{2^i}(u)$：$\mathrm{anc}_{2^0}(u) = p_u$，$\mathrm{anc}_{2^{i+1}}(u) = \mathrm{anc}_{2^i} \circ \mathrm{anc}_{2^i}(u)$。

对于 $\mathrm{anc}_k(u)$，将 k 二进制分解为 $k = 2^{k_0} + 2^{k_1} + \dots$，则 $\mathrm{anc}_k(u) = \cdots \circ \mathrm{anc}_{2^{k_1}} \circ \mathrm{anc}_{2^{k_0}}(u)$。

预处理时间复杂度 $O(n \log n)$，单组询问复杂度 $O(\log n)$。空间复杂度 $O(n \log n)$。

倍增

Step 2: 令 $u \leftarrow p_u$，$v \leftarrow p_v$，直到 $u = v$。

考虑倍增：

- 从 $\log n$ 到 0 枚举 i。
- 若 $\mathrm{anc}_{2^i}(u) = \mathrm{anc}_{2^i}(v)$，则令 $u \leftarrow \mathrm{anc}_{2^i}(u)$, $v \leftarrow \mathrm{anc}_{2^i}(v)$，否则 u, v 不变。
- 最后 p_u（或 p_v）即为 LCA。

时间复杂度 $O(\log n)$。

Code

```
// 完整代码: https://www.luogu.com.cn/record/165936526

void init() {
    // (DFS 预处理 par 和 dep 数组)
    for (int u = 1; u <= n; ++u) anc[0][u] = par[u];
    for (int i = 0; i < logn; ++i)
        for (int u = 1; u <= n; ++u)
            anc[i + 1][u] = anc[i][anc[i][u]];
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);  // 保证 dep_u ≥ dep_v

    // Step 1: u 往上倍增直到 dep_u = dep_v
    for (int i = logn; i >= 0; --i)
        if (dep[u] - (1 << i) >= dep[v])
            u = anc[i][u];
    if (u == v) return u;

    // Step 2: u,v 同时往上倍增直到 u = v
    for (int i = logn; i >= 0; --i)
        if (anc[i][u] != anc[i][v])
            u = anc[i][u], v = anc[i][v];
    return par[u];  // 此时 u,v 为深度最低的 u ≠ v 的点, 即 lca(u,v) 的子节点
}
```

欧拉序：把每条树边看作正反两条有向边，从根节点出发一笔画最后回到根节点经过节点的顺序。记这个序列为 $(a_1, a_2, \ldots, a_{2n-1})$ $(a_1 = a_{2n-1} = 1)$。

记点 u 第一次出现的时间为 t_u。

Code

```
vector<int> a(1);        // 欧拉序
vector<int> t(n + 1);    // t[u]: u 在 a 中第一次出现的下标
void dfs(int u) {
    t[u] = a.size();     // a.size() 为此时 u 在 a 中的下标
    a.push_back(u);
    for (auto v: G[u]) {
        if (!t[v]) {
            dfs(v);
            a.push_back(u);
        }
    }
}
```

性质：欧拉序中 u 和 v 之间第一次最早出现的点是 $\text{lca}(u,v)$。即 $t_{\text{lca}(u,v)} = \min_{t_u \le i \le t_v} t_{a_i}$。

- u 的子树在欧拉序中的位置连续，且形如 $u\ T_1\ u\ T_2\ u \dots u$。
  - 若 u 和 v 在 r 的同一个子树中，则 u, v 之间不会出现 r。
  - 若 u 和 v 在 r 的不同子树中，则 u, v 之间一定出现 r。
- 若 v 在 u 的子树中，则 $t_u \le t_v$。

求出欧拉序 a 和每个点第一次出现的时间 t 后，用 ST 表维护区间中 t 的最小值即可。

预处理时间复杂度 $O(n \log n)$，单组询问复杂度 $O(1)$。空间复杂度 $O(n \log n)$。

Code

```
// 完整代码: https://www.luogu.com.cn/record/165950526

vector<vector<int>> st(logn, vector<int>(2 * n));
unordered_map<int, int> mp;

void init() {
    // (DFS 求 a,t 数组)
    for (int i = 1; i <= n; ++i) mp[t[i]] = i;  // 从 t 映射回顶点 id

    // st 表, 存放 [u,u+2^i) 中 t 的最小值
    const int logn = 21;
    for (int u = 1; u < 2 * n; ++u)
        st[0][u] = t[a[u]];
    for (int i = 0; i < logn; ++i)
        for (int u = 1; u + (1 << i) < 2 * n; ++u)
            st[i + 1][u] = min(st[i][u], st[i][u + (1 << i)]);
}

int lca(int u, int v) {
    int l = t[u], r = t[v];
    if (l > r) swap(l, r);
    int i = __lg(r - l + 1);
    int mint = min(st[i][l], st[i][r + 1 - (1 << i)]);
    return mp[mint];
}
```

应用：

- 维护直径，支持修改边权：线段树维护 $\max_{u\le w\le v} \mathrm{dep}_u + \mathrm{dep}_v - 2\mathrm{dep}_w$。
- 维护动态树：平衡树维护欧拉序。

离线算法。用并查集维护已经访问过的点的 LCA。

- 初始时每个点自己构成一个连通块。
- DFS 遍历所有点。当访问到点 u 时，检查所有与 u 有关的询问 (u, v)。如果此时 v 被访问过，则 $\operatorname{lca}(u, v)$ 为 v 所在的连通块的根节点。
- 当 u 出栈时，将 u 所在连通块合并至父亲节点 p_u。

时间复杂度 $O(n + Q\alpha(n))$。

Code

```
// 完整代码: https://www.luogu.com.cn/record/165956083

vector<int> ans(q + 1);
vector<int> vis(n + 1);
DSU S(n + 1);   // 并查集
void dfs(int u) {
    vis[u] = 1;
    for (auto v: G[u]) {
        if (!vis[v]) {
            dfs(v);         // 递归搜索子树 v
            S.par[v] = u;   // 退出 v 时把 v 合并到 u 上
        }
    }
    // 遍历所有与 u 有关的询问 (u,v)
    for (auto [v, i]: Q[u]) {
        // 如果 v 已经被访问过，则答案为 v 所在连通块的根节点
        if (vis[v]) ans[i] = S.root(v);
    }
}
```

树链剖分

顾名思义就是把树切成若干链拼起来。

重链剖分：把每个节点和儿子中子树大小最大的点连起来。

性质：从任意叶子节点到根节点经过不超过 $\log n$ 条链。

求 LCA：每次将较深的点跳到所在链的根节点。

比起求 LCA 更多用于维护树上的一些动态信息。

例题 1

Luogu P3398　仓鼠找 sugar

有一棵树，Q 组询问，每组询问给 4 个点 a, b, c, d，问 a 到 b 的路径是否与 c 到 d 的路径有公共点。

范围：$n, Q \le 10^5$

两条路径有交点当且仅当 $\mathrm{lca}(a, b)$ 在 c 到 d 的路径上，或 $\mathrm{lca}(c, d)$ 在 a 到 b 的路径上。

- a 到 b 的路径上的点一定在 $\mathrm{lca}(a, b)$ 的子树中。
- 如果两条路径有交点，那么一定有一条路径的 lca 是另一条路径的祖先。
- 若 u 是 v 的祖先，那么从 u 到 v 的某个后代的路径一定会经过 v。

例题 2

AHOI2008　聚会

给一棵树，有 Q 组询问，每组询问给出三个点 x, y, z，求 $\min_{v\in V} \operatorname{dis}(x, v) + \operatorname{dis}(y, v) + \operatorname{dis}(z, v)$。

范围：$n, Q \le 5 \cdot 10^5$

考虑最小的包含 $\{x, y, z\}$ 的子树，v 只能选在这棵子树上。答案显然不小于这棵子树的边数。

事实上答案可以取到这个值，且此时 v 是 $\operatorname{lca}(x, y), \operatorname{lca}(y, z), \operatorname{lca}(z, x)$ 中深度最大的点：

- 若 $\operatorname{lca}(x, y) = \operatorname{lca}(y, z) = \operatorname{lca}(z, x)$，则 x, y, z 在 $\operatorname{lca}(x, y, z)$ 的三个不同子树中，结论成立。
- 否则令 $u = \operatorname{lca}(x, y, z)$，不妨设 x, y 在 u 的同一个子树中，z 在另一个子树。
  我们有 $\operatorname{lca}(x, z) = \operatorname{lca}(y, z) = u$。取 $v = \operatorname{lca}(x, y)$，可以验证 $(x, v), (y, v), (z, v)$ 三条路径不经过重复的边。

例题 3

Luogu P6374　树上询问

给定一棵 n 个点的无根树，有 Q 次询问。每次询问给一个参数三元组 (a,b,c)，求有多少个 i 满足这棵树在以 i 为根的情况下 $\mathrm{lca}(a,b)=c$。

范围：$n \le 5\cdot 10^5, Q \le 2\cdot 10^5$

c 必须在 a 到 b 的路径上，否则答案为 0。

当 $c \notin \{a,b\}$ 时，答案为 $n-a$ 所在的子树大小 $-\, b$ 所在的子树大小。

考虑怎么维护这个式子。以 1 为根节点，用 siz_u 表示 u 为根的子树大小。

- 若 $c=\mathrm{lca}(a,b)$，将 a,b 往上倍增到 c 的儿子层得到 a',b'，则答案为 $n-\mathrm{siz}_{a'}-\mathrm{siz}_{b'}$。
- 若 c 在 $\mathrm{lca}(a,b)$ 到 a 的链上，则答案为 $\mathrm{siz}_c-\mathrm{siz}_{a'}$。
- 若 c 在 $\mathrm{lca}(a,b)$ 到 b 的链上，则答案为 $\mathrm{siz}_c-\mathrm{siz}_{b'}$。

当 $c=a$ 或 b 时类似。

例题 4

NOIP2013　货车运输

给一个无向带权图，有 Q 组询问，每组询问给两个点 u, v，求一条 u 到 v 的路径，使得路径上最小的边尽可能大（即最大瓶颈路）。

范围：$n \leq 10^4, m \leq 5 \cdot 10^4, Q \leq 3 \cdot 10^4$

考虑从大到小加入边。如果加入某条边时 u, v 恰好连通，则 (u, v) 的答案为这条边的边权。

求最大生成树，输出 u 到 v 在生成树上的路径的最小边权即可。

LOJ 10137　跳跳棋

棋盘上有 3 颗棋子，坐标分别为 a, b, c。要求通过最少的跳动把位置移动到 a', b', c'（棋子之间没有区别，可能无解）。每次跳动任意选一颗棋子，对一颗中轴棋子跳动。跳动前后两颗棋子距离不变，一次只允许跳过一颗棋子。

范围：所有坐标绝对值 $\leq 10^9$

棋子至多只有 3 种跳动方式：假设棋子坐标为 (a, b, c)，则跳动后可以变为

- b 跨过 a 向左跳：$(2a - b, a, c)$；
- b 跨过 c 向右跳：$(a, c, 2c - b)$；
- a 或 c 跨过 b 往中间跳：
  - 若 $b - a < c - b$，则选 a 跳，变为 $(b, 2b - a, c)$；
  - 若 $b - a > c - b$，则选 c 跳，变为 $(a, 2b - c, b)$。

考虑以坐标状态为节点建图：若 (a, b, c) 可以跳到 (x, y, z) 则连一条边。则这个图是以 $\{(a, b, c) : a + c = 2b\}$ 为根节点的二叉树森林。

例题 5

当 (a, b, c) 和 (a', b', c') 在不同二叉树时无解，否则答案为 (a, b, c) 到 (a', b', c') 的路径长度。

不需要显式地把树构造出来，只需实现两个操作：

- 求 (a, b, c) 的深度：辗转相除。
- 求 (a, b, c) 和 (a', b', c') 的 LCA：用倍增法，需要求出 (a, b, c) 向上跳 k 步的结果。

复杂度 $O(\log^2 |W|)$.

1. 最近公共祖先
2. 树上差分

树上差分

问题 1

给一个序列 a，进行 q 次操作，每次操作选择一个区间 $[\ell, r]$，将 $a_\ell, a_{\ell+1}, \ldots, a_r$ 加 k。求最终序列。

差分：令 $d_i = a_i - a_{i-1}$ $(1 \le i \le n)$，其中 $a_0 = 0$。

对于操作 $[\ell, r]$，令 $d_\ell \leftarrow d_\ell + k$，$d_{r+1} \leftarrow d_{r+1} - k$。最后 $a_i = \sum_{j=1}^{i} d_j$。

d_j 的含义：对区间 $[j, n]$ 进行了多少次 $+1$ 的操作。

- 只有当 $j \le i$ 时，对区间 $[j, n]$ 的操作才会对 a_i 产生贡献，因此 $a_i = \sum_{j \le i} d_j$。
- 操作 $[\ell, r] + k$ 可以拆成 $[\ell, n] + k$ 和 $[r+1, n] - k$。

问题 2

给一棵树，每个点有点权 a_i。进行 q 次操作，每次操作选两个点 ℓ, r，将 ℓ 到 r 路径上的点权加 k。求最终所有点的权值。

树上差分：用 d_j 记录进行了多少次对 1（根节点）到 j 的路径 $+1$ 的操作。

- 只有当 j 在 i 的子树中时，对路径 $[1, j]$ 的操作才会对 a_i 产生贡献，因此 $a_i = \sum_{j \in T_i} d_j$，其中 T_i 表示 i 为根的子树。
- 操作 $[\ell, r] + k$ 可以拆成 $[1, \ell] + k$，$[1, r] + k$，$[1, \mathrm{lca}(\ell, r)] - k$，$[1, p_{\mathrm{lca}(\ell, r)}] - k$。

d_i 的初始值：$d_i = a_i - \sum_{j \in \mathrm{son}_i} a_j$。

例题 1

例题 1

给一棵树，每个点有点权。Q 组询问，要求实现链加、单点求值两种操作。

树上差分，转化为单点加、子树求和。

把树按 DFS 序展开成序列，则每棵子树对应区间 $[\mathrm{dfn}_u, \mathrm{dfn}_u + \mathrm{siz}_u)$，用树状数组实现单点修改、区间求和即可。

复杂度 $Q \log n$。

例题 2

例题 2

给一棵树，每个点有点权。Q 组询问，要求实现链加、子树求和两种操作。

用 d_i 表示对链 $[1, i]$ 的权值修改。

一次操作对根节点为 u 的贡献为 $(\text{dep}_i - \text{dep}_u + 1) \cdot d_i$，且必须 i 在 u 的子树中才会产生贡献。

因此 $a_u = \sum_{i \in T_u} (\text{dep}_i - \text{dep}_u + 1) \cdot d_i = \sum_{i \in T_u} \text{dep}_i \cdot d_i + (-\text{dep}_u + 1) \sum_{i \in T_u} d_i$。

与上一题类似把树按 DFS 序展开，用两棵树状数组分别维护区间 d_i 和、区间 $\text{dep}_i \cdot d_i$ 和即可。

复杂度 $Q \log n$。

例题 3

NOIP2015　运输计划

有一棵树，每条边有边权。给定 m 组 (u_i, v_i)，你需要选择一条边将边权变成 0，使得最小化 $\max_{1 \le i \le m} \operatorname{dis}(u_i, v_i)$。

范围：$n, m \le 3 \cdot 10^5$

考虑二分答案。

假设答案为 x，则选出来的边必须在所有长度 $> x$ 的路径上，否则距离最大值不变。

将所有长度 $> x$ 的路径求交，选择交路径中最长的边变成 0，检查减去这条边的长度后，最长路是否 $\le x$。

路径求交：将每条路径上的边权 $+1$，最后查询每条边的边权，如果 $= m$ 说明这条边是所有路径的交。

例题 4

NOIP2016　天天爱跑步

有一棵树，每条边长度为 1。树上有 m 个人在第 0 时刻同时开始跑步，其中第 i 个人从 s_i 跑到 t_i。对每个 i，求 i 号点在时刻 w_i 的人数。

范围：$n, m \leq 3 \cdot 10^5$

考虑在每个点维护一个向量 a_u，$a_u[i]$ 表示节点 u 在时刻 i 的人数。

我们只需实现以下操作：

- 给定 u, v，令 $[u, v]$ 上的每个点 i，令 $a_i[\mathrm{dis}(u, i)] \leftarrow +1$。

上述操作可以拆分成下面两种操作：

- 操作 1：给定 u, t, k，对 $[1, u]$ 上的每个点 v，令 $a_v[t + (\mathrm{dep}_u - \mathrm{dep}_v)] \leftarrow +k$。
- 操作 2：给定 u, t, k，对 $[1, u]$ 上的每个点 v，令 $a_v[t - (\mathrm{dep}_u - \mathrm{dep}_v)] \leftarrow +k$。

例题 4

我们可以分别考虑操作 1 和操作 2 的贡献，最后加起来。

令 $b_v[i] = a_v[i - \mathrm{dep}_v]$，则操作 1 变为 $b_v[t + \mathrm{dep}_u] \leftarrow +k$。

这是经典的链修改、单点求值问题，可以对 b 差分。现在问题转化为

- 给定 u, k，令 $d_u[t + \mathrm{dep}_u] \leftarrow +k$。
- 最后对所有 u 查询 $\left(\sum_{v \in T_u} d_v\right)[t + \mathrm{dep}_v]$。

可以用线段树合并或 map 启发式合并做，复杂度 $O((m + n)\log n)$。

Thanks