# 树链剖分

## 数据预处理1 确定重儿子、子树大小以及节点深度

```
vector<int> siz(n + 10), fath(n + 10);
vector<int> dep(n + 10), son(n + 10);
auto dfs1 = [&](auto self, int fa, int x) -> void {
    fath[x] = fa; siz[x] = 1; dep[x] = dep[fath[x]] + 1;
    int maxson = 0;
    for(auto v : g[x]){
        if(v == fa) continue;
        self(self, x, v);
        siz[x] += siz[v];
        if(siz[v] > maxson) maxson = siz[v], son[x] = v;
    }
    return ;
};
dfs1(dfs1, 0, r);
```

根节点第一次$dfs$更新重儿子。

## 数据预处理2 确定链顶点、$dfs$序左右边界以及节点值

```
vector<int> lef(n + 10), rig(n + 10); // 左右边界
vector<int> top(n + 10), node(n + 10);
int cnt = 0;
auto dfs2 = [&](auto self, int fa, int x, int tp) -> void {
    lef[x] = ++cnt; top[x] = tp; node[cnt] = nums[x];
    if(siz[x] == 1){
        rig[x] = cnt;
        return ;
    }
    self(self, x, son[x], tp);
    for(auto v : g[x]){
        if(v == fa || v == son[x]) continue;
        self(self, x, v, v);
    }
    rig[x] = cnt;
    return ;
};
dfs2(dfs2, 0, r, r);
```

根节点第二次$dfs$更新节点值

## 更新操作

```
auto add = [&](int x, int y, int z) -> void {
    while(top[x] != top[y]){
        if(dep[top[x]] < dep[top[y]]) swap(x, y);
        update(update, 1, n, 1, lef[top[x]], lef[x], z);
        x = fath[top[x]];
    }
    if(dep[x] < dep[y]) swap(x, y);
    update(update, 1, n, 1, lef[y], lef[x], z);
    return ;
```

```
};
add(x, y, z);
```

当在不同链的情况下循环迭代直到两个节点在同一个链上同时进行操作。

## 询问操作

```
auto sum = [&](int x, int y) -> i64 {
    i64 ans = 0;
    while(top[x] != top[y]){
        if(dep[top[x]] < dep[top[y]]) swap(x, y);
        ans = MOD(ans + query(query, 1, n, 1, lef[top[x]], lef[x]), mod);
        x = fath[top[x]];
    }
    if(dep[x] < dep[y]) swap(x, y);
    ans = MOD(ans + query(query, 1, n, 1, lef[y], lef[x]), mod);
    return ans;
};
```

不同链的情况下循环迭代并且求和。