数据结构

栈

单调栈

- 给一个序列，对于每个数左边第一个比它的的数。
- 求每段区间最小值的和。
- 笛卡尔树。

队列

单调队列

- 求所有长度为k的区间的最小值。

STL

- vector, stack, queue
- set, map, priority_queue
- unordered_map, unordered_set

例题

- 维护一个集合，支持插入一个数，删除一个数，求所有数从小到大排序之后的异或和。
- 有一段数字，支持区间覆盖，区间单点查询。

树上倍增

- LCA
- 求一个点往上第k个祖先
- 求两点之间路径的最大值
- 例题：有若干个操作，每次操作将一条路径上的边全部加上一个值，操作完之后求每条边的权值。

ST表

- 求RMQ, LCA
- 二维RMQ。

分治

- 求所有区间的最小值的和。
- 平面最近点对。
- 有一个n*n的矩阵，我们知道每一行最小值的位置单调不降，求每一行最小值的位置。

并查集

- 离线求lca。
- 动态加边，询问两点之间路径的xor和，强制在线
- 有一棵树，每次操作会将一条路径上打上标记。最后回答每条边上面打的标记的最小值。
- 动态加边，求图中的桥的个数，可以离线。

线段树

- 序列，支持区间加，区间乘，区间求和。
- 平面上有n个点，每次询问为一个矩形内有多少点。允许离线。
- 序列，支持区间加，询问区间第一个大于k的数字。
- 一个长度为n的全排列，进行m次操作，每次操作是把[l,r]这个区间排成升序或者降序，求m次操作后x这个位置的数。

DFS序

- 每个点有个f值，每个点的g值为子树的f值的和。支持修改f值，询问子树g值的和。
- 给你一棵树，边权可能是负数，要求查询u这个点到v这个子树里面的最远点。
-

启发式合并

- 有若干个集合，支持合并两个集合，求第k大。
- 求出每个子树的众数。

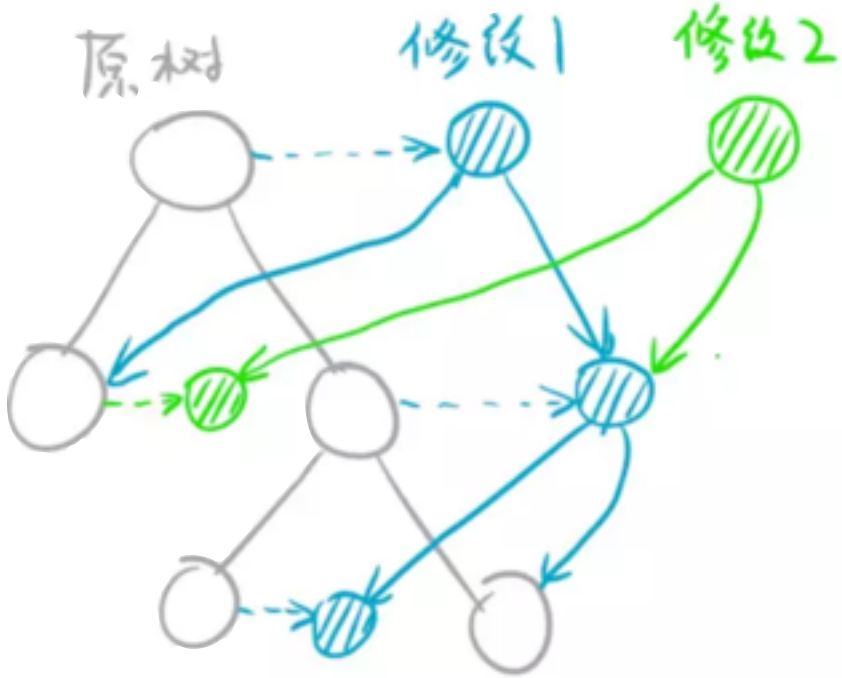
DSU on Tree

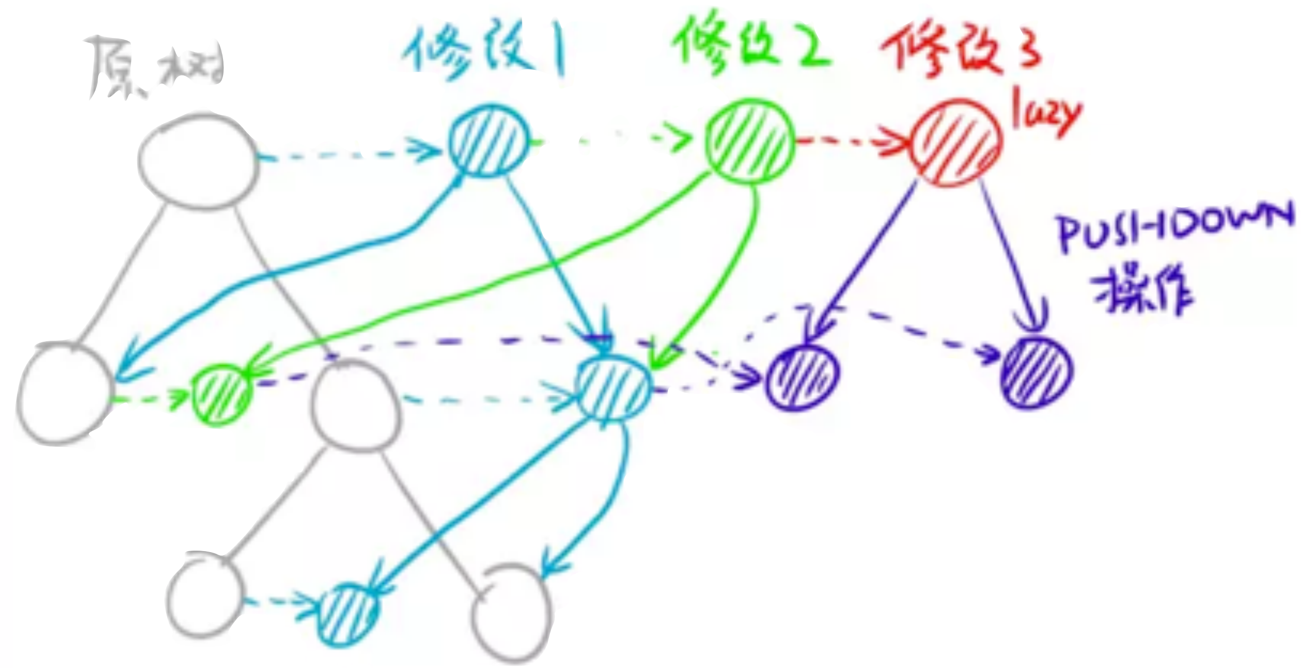
- Talk is cheap, show me code.
- 大致写法，先通过一遍dfs，求出size，重儿子，和dfs序。
- 然后第二遍dfs，先遍历轻儿子，并且不保留，然后遍历重儿子，保留，然后把所有轻儿子的信息加进去。
- 然后如果不保留就整体清空。

```
void dfs(int u,int f) {
    l[u]=++tot;
    id[tot]=u;
    sz[u]=1;
    hs[u]=-1;
    for (auto v:e[u]) if (v!=f) {
        dfs(v,u);
        sz[u]+=sz[v];
        if (hs[u]==-1||sz[v]>sz[hs[u]]) hs[u]=v;
    }
    r[u]=tot;
}

void dfs2(int u,int f,bool fg) {
    for (auto v:e[u]) if (v!=f&&v!=hs[u]) {
        dfs2(v,u,0);
    }
    if (hs[u]!=-1) {
        dfs2(hs[u],u,1);
    }
    for (auto v:e[u]) {
        if (v!=f&&v!=hs[u])
            rep(x,l[v],r[v]+1) modify(1,1,n,id[x],1);
    }
    modify(1,1,n,u,1);
    ans+=(ll)n*(n+1)/2-nd[1].s;
    if (!fg) {
        rep(i,l[u],r[u]+1) modify(1,1,n,id[i],0);
    }
}
```

主席树

- 



例题

- 二维数点，强制在线。
- 区间第k大查询。
- 求树上两点之间第k大。
- 给你一个序列a1, a2, …, an。给Q个询问，每次给出l, r, x求l,r之间与x异或之后最大的数。

CDQ分治

- 三维数点。
- 一个图，对于所有i,j,k求出从i到j不经过k的最短路长度。

线段树分治

- 动态图支持加边删边，询问连通性，可以离线。

Segment Beats

- 给一个序列，支持区间求min，然后询问区间max，区间和。

Solution

- 维护每个区间最大值，次大值，以及最大值出现了几次。
- 当打标记的时候，如果当前标记大于最大值，那么跳过。
- 如果在最大值与次大值之间，那么我们求改最大值。
- 否则打上标记，并且递归下去，求出新的最大值，次大值。

Solution

- 时间复杂度，容易发现如果递归下去，那么这个区间里面的最大值和次大值就会变得相同，也就是整个区间不同的数值个数会-1。
- 一开始所有区间不同的数字总和是O(n log n)的，所以总的时间复杂度是O((n+q) log n)。