浅谈一类利用数据结构解决的树上问题

Cai

目 录

1 DFS 序的运用

我们对树进行一次 DFS 遍历，得到每个节点被遍历的顺序，称这个顺序为 DFS 序，这个 DFS 序有一个性质：树上的任何一颗子树的 DFS 序一定是连续的一段，我们可以利用这个性质套用数据结构进行维护

例题 1. 谈笑风生[1]

给出一颗含 N 个节点，以 1 号结点为根的有根树，及 M 个询问 (p,k)，询问有多少个二元组 (a,b) 满足以下三个条件：

1. $a \neq p, b \neq p, a \neq b$
2. p, a 都是 b 的祖先
3. p, a 在树上的距离不超过 k

$N, M \leq 300000$

可能用到的符号：D_x：节点 x 的深度，$D_1 = 1$；S_x：以 x 为根的子树的节点数量

分 a 是 p 的祖先和 p 是 a 的祖先两种情况考虑

如果 a 是 p 的祖先，则 a 可能的方案为 $\min(D_p - 1, k)$，c 可能的方案为 $S_p - 1$

如果 p 是 a 的祖先，则 a 只可能在以 p 为根的子树且深度在 $[D_p + 1, D_p + k]$ 的范围内，这里可以需要用到树的 DFS 序，因为一棵树的子树在 DFS 序上一定是连续的一段，而我们需要求的值为 $\sum(S_b - 1)$，节点 b 满足 a 是 b 的祖先且 $D_b \in [D_p + 1, D_p + k]$，显然这个是可以差分的，于是我们就可以用可持久化线段树来维护：以深度为下标，$S_x - 1$ 为值加入到可持久化线段树上，询问就是两颗线段树区间和的差。

最后答案就是两种情况的方案和

时间复杂度:$O((N + M) \log N)$

1.1 利用 DFS 序用线段树替代树链剖分

对于一般的子树修改子树查询问题都可以利用 DFS 序用线段树解决，而对于路径上的查询问题也可以用线段树解决，设 $F(u,v)$ 表示路径 (u,v) 的信息，如果这个信息满足查分性质 (设 $L(u,v)$ 表示节点 u 与节点 v 在树上的最近公共祖先，r 表示树的跟，$P(u)$ 表示节点 u 的父节点)

对于权值在边上的问题有

$$F(u,v) = F(r,u) + F(r,v) - 2 \times F(r, L(u,v))$$

对于权值在点上的问题有

$$F(u,v) = F(r,u) + F(r,v) - F(r, L(u,v)) - F(r, P(L(u,v)))$$

这样的话我们就可以在线段树上维护每一个节点到其根节点上的信息，化路径查询为单点查询，这样做可以比树链剖分少一个 log 的时间复杂度

[1]BZOJ 3653

例题 2. 树链剖分[2]

给出有一颗以 1 号结点为根节点的有根树，每条边上有一个边权，使其支持以下四个操作

1(1 i w): 将第 i 条边的权值增加 w

2(2 u w): 将以 u 为根节点的子树内所有边的边权增加 w

3(3 u w): 将以 u 为根节点的子树内所有边的权值设为 w

4(4 u v): 询问 u 到 v 的简单路径的权值和

$N \leq 10^6, M \leq 1666666$，$M$ 为操作数量，3s/512MB

维护每一个节点到根节点的简单路径的权值和，求两个节点的简单路径的权值和就可以运用上面的公式。

对于第一个操作: 修改一条边的边权会影响到以其为根的子树内所有节点的到根节点权值和，打上区间加标记即可；

对于第二个操作: 修改一颗子树的边权会使以其为根的所有子树到根节点的权值和，而且是关于其深度的一次函数，也可以打上标记；

对于第三个操作: 也是类似的，先将字数内所有节点的权值改成修改节点的父亲的节点权值，这一步就相当于将子树内所有节点的权值设为 0，然后再加上即可，

时间复杂度:$O(N \log N)$

例题 3. fibonacci[3]

给出一颗以 1 号结点为根节点的有根树，每一个节点有一个初始值为 0 的权值 W，令 D_x 表示点 x 的深度，令 $D_1 = 1$，定义数列 F:$F(1) = F(2) = 1$，$\forall i > 2 : F(i) = F(i-1) + F(i-2)$，维护以下两种操作:

1(U x k): 对于在子树 x 内的所有节点 y(包括 x 本身):$W_y \leftarrow W_y + F(k + D_y - D_x)$

2(Q x y): 询问节点 x 到节点 y 上简单路径的权值和对 1000000007 取模的结果

$N, M \leq 100000, 1 \leq k \leq 10^{15}$，1s/512MB

这个题目权值再点上，满足只有子树修改路径求和的性质，所以可以使用上面的方法在 $O(N \log N)$ 的时间内出解，令 A_y 表示节点 y 到根节点路径上的权值之和，那么对于修改操作 x k 则有 (假定 y 在子树 x 内):

$$A_y \leftarrow A_y + \sum_{i=D_x}^{D_y} F(k + i - D_x)$$

将这个式子中的 F 函数写成转移矩阵的形式则有

$$A_y \leftarrow A_y + \sum_{i=D_x}^{D_y} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{k+i-D_x}$$

运用公式

$$\sum_{i=1}^{r} F(i) = F(i+2) - 1$$

去掉上式的求和符号得将上式转移矩阵中的常数项分离则有

$$A_y \leftarrow A_y + \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{k+D_y-D_x+2} - \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{k+1}$$

[2]原创

[3]雅礼中学 WC2017 集训

这样修改操作就变成了两个区间复制操作，一个与节点的深度有关，而另一个与节点的深度无关，套用线段树即可

时间复杂度:$O(N \log N)$，涉及到矩阵操作，常数略大

1.2 Splay 维护括号序列

括号序列也是一种 DFS 序，对于我们在 DFS 遍历时：在进入节点和离开节点时都记录时间戳，我们用一个左括号表示进入节点，用一个右括号表示离开节点，这样一棵树就可以对应一个括号序列

括号序列的性质：树上的任何一颗子树一定对应括号序列中一段连续的括号

这样如果我们想在树中删除一颗子树的话我们可以直接删除那一段括号，如果我们想加入一颗子树的化我们可以插入一段括号，这样我们就可以用 Splay 来维护括号序列来做到一些伪的动态树问题

对于一类与树的结构不是静态，没有链上操作的问题，但有子树修改，查询的问题中，我们可以使用 Splay 来动态维护树的括号序列来解决，但是如果你会 ETT 的话就没必要了

例题 4. Gty 的游戏[4]

给出一颗含 N 个节点的且以 1 号结点为根有根树，每个节点有一些石子，每次可以在某个节点将不超过 L 的石子移动到其父节点，不能移动者输，M 个操作，每个操作为以下三种：

1($1\ x$): 在以 x 为根的子树内玩游戏，询问先手是否有必胜策略，如果有则输出 MeiZ，否则输出 GTY

2($2\ x\ y$): 将第 x 个节点上的石子数改成 y

3($3\ u\ v\ x$): 为节点 u 添加一个子节点 v，其石子数为 x

强制在线，每个操作除了类型名 (输入的第一个数字) 外，所有其他数字都需要异或上之前输出 MeiZ 的个数

$N \leq 50000$，节点上的石子数小于 10^9，节点数量不超过 10^5

首先解决游戏部分，对于以 x 为根的子树，令 x 的深度为 0，则我们只需要考虑深度为奇数的节点，因为对于深度为偶数的节点，如果移动偶数层的石子，那么另一个人一定可以继续将其移动到偶数层，若干次之后这些石子就不能移动了 (到达节点 x)，所以我们只需要管奇数层的石子，其次这是一个 NIM 游戏，假设技术层的棋子有 $A_1, A_2..A_k$ 个，那么如果 $\text{xor}_{i=1}^{k}(A_i \mod (L+1)) \neq 0$ 的话先手有必胜策略，此时输出 MeiZ，否则输出 GTY

其次这是一个关于子树的问题，如果你会 ETT 的话就可以不用往下看了。子树可以想到 DFS 序，这样的话子树内的节点就是一个连续的数列了，但是有插边操作怎么办，我们可以想到括号序列，在节点 x 下方插入一个子节点 y 就相当于在 x 的左括号的后面插入一个表示节点 y 的左右括号，我们认为左括号的权值为节点的权值，右括号的权值为 0，计算的时候按照深度分类就可以了，那么这样询问的问题就解决了：询问 x 的时候提取 x 左括号和 x 右括号的一段子括号序列，直接判断即可

但是如何定位一个括号的位置，我们可以直接把 DFS 序当成 Splay 内节点的标号

时间复杂度:$O(N \log N)$

[4]BZOJ 3729

2 树链剖分

2.1 主要思想

将一颗有根树剖分成若干条链，这些链我们称为重链，重链上的边称为重边，一条重链的两个断点 (u, v)，如果在树上 u 是 v 的父节点，那么我们称 v 是 u 的重儿子，树上的任何一个节点最多只有一个重儿子，v 是 u 的重儿子需要满足的条件是 v 所在的子树是 u 的儿子的子树中节点数量最多的，u 的其他儿子被称为轻儿子。树上不是重边的边称为轻边。

性质 (*)：树上任何一条路径上轻边的数量不超过 $\log N$，重边的数量不超过 $\log N$

我们对整棵树进行一次遍历，遍历的规则是：如果这个节点有重儿子，那么优先遍历重儿子，得到 DFS 序，这样树上任何一颗子树的 DFS 序一定是连续的一段，任何一条重链的 DFS 序都是连续的一段，这样我们就可以利用这个 DFS 序用数据结构来维护这棵树上的信息了，由于性质 (*)，我们可以知道这样的单次操作的时间复杂度是 $O(\log N)$ 的

我们一般采用线段树维护，那么时间复杂度就是 $O(\log^2 N)$

2.2 一些模板题

例题 5. Query on a tree[5]

给出一棵有 N 个节点的树，树上每条边都有一个边权，你需要维护一下两种操作

1(QUERY u v): 询问节点 u 到节点 v 的简单路径的边权最大值

2(CHANGE i x): 将第 i 条边的边权修改为 x

$N \leq 10000$，0.851s/1536MB

模板题，将边权下放为点权，树链剖分 + 线段树

例题 6. 树的统计[6]

给出一颗 N 个节点的树，树上结点编号为 1-N，每个节点都有一个点权 x，M 个操作，每个操作为一下三种

1(CHANGE u x): 将节点 u 的权值修改为 x

2(QMAX u v): 询问节点 u 到节点 v 的简单路径的点权的最大值

3(QSUM u v): 询问节点 u 到节点 v 的简单路径的点权和

$1 \leq N \leq 30000, 0 \leq M \leq 200000, |x| \leq 30000$，1s/256MB

例题 7. 月下“毛景树”[7]

给出一颗 N 个节点的树，每条边都有一个边权，使其支持以下四个操作:

1(Change k w): 将第 k 条边的权值改为 x

2(Cover u v w): 将节点 u 到节点 v 的简单路径上的权值修改为 w

3(Add u v w): 将节点 u 到节点 v 的简单路径上的权值增加 w

4(Max u v): 询问节点 u 到节点 v 的简单路径上的权值最大值

$N \leq 100000$，2s/64MB

[5]SPOJ QTREE

[6]ZJOI 2008

[7]BZOJ 1984

模板，注意线段树上的标记下传

例题 8. 软件包管理器[8]

给出一个含有 N 个节点的有根数，每个节点都有一个初值为 0 的点权，M 个操作，每个操作为以下两种：

1(install x): 讲节点 x 到根路径上的所有点设为 1，并输出有多少个节点的点权发生了变化

2(uninstall x): 将以节点 x 为根的子树的所以点权设为 0，并输出有多少个节点的点权发生了变化

$N \leq 100000, M \leq 100000$

例题 9. 染色[9]

给出一颗有 N 个节点的无根树，每个节点都有一个颜色，M 个操作，每个操作为以下两种：

1(C u v x): 询问节点 u 到节点 v 的简单路径上的所有点的点权变成 x

2(Q u v): 询问节点 u 到节点 v 的简单路径上有多少段颜色

$1 \leq N \leq 100000, 1 \leq M \leq 100000$，1s/128MB

模板，不过操作有点麻烦，注意回答询问的时候路径上两端的颜色

2.3 树链剖分维护轻边信息

例题 10. sub[10]

给出一颗 N 个节点的无根树，每个节点 i 都有一个权值 A_i，M 次操作，每次操作为以下两种：

1(1 x y): 将 A_x 修改为 y

2(2): 输出一个点权和最大的联通块的点权和

$N, M \leq 10^5$，任何时候，每个节点的权值的绝对值不超过 1000，1s/256MB

如果只有一个询问的话这显然就是一个 dp 问题了，但是这个题目还有修改操作，所以我们不能直接 dp

这是一个树链剖分维护轻边信息的题目，对于一个节点 i，我们另一个节点有另一个权值 B，B_i 表示 A_i 加上 i 的子树中所有与节点 i 相连的轻边的 dp 值之和 (也就是节点 i 所在的重链中关于 B 的包含节点 i 的最大子段和)，然后答案就只可能是某一条重链上的一段关于 B 的最大子段和，最大字段和我们可以用 set 来维护

修改的时候我们维护 B 数组和 set 即可

时间复杂度:$O(N \log^2 N)$

[8]NOI 2015

[9]SDOI 2011

[10]Vijos Coaching Test

3 LCT

3.1 主要思想

LCT 全称为 Link-Cut-Tree，主要思想就是像树链剖分一样维护轻重边，但是在 LCT 中，我们将重边称为实边，将轻边称为虚边，实边用 Splay 来维护，虚边直接记录父亲，这是因为这一点，造成了 LCT 中孩子认父亲但是父亲不认儿子的现象。

3.2 主要操作

LCT 支持关于链上的操作，但是由于孩子认父亲但是父亲不认儿子，所以一般来说无法维护子树信息。

3.2.1 Splay 树上的 Splay 操作 Splay(x)

在 LCT 中 Splay 操作的用处与普通的 Splay 树的 Splay 操作有少许的不同：普通的 Splay 树的 Splay 操作有两个参数 (x, y)：表示将节点 x 旋转至节点 y 的下方，而 LCT 中的 Splay 操作则只有一个参数 (x): 将节点 x 旋转至根节点

LCT 的 Splay 操作中，如果有标记的话要先递归下传标记

3.2.2 Splay 树上的 Splay 操作 Rotate($x, kind$)

LCT 中根节点的交换是在 Rotate 操作中完成的，所以在执行 LCT 的 Rotate 函数时要判断 Splay 根节点

代码：

```
void Rotate(int x, int kind) {
    int y = pre[x];
    pre[ch[y][!kind]=ch[x][kind]] = y;
    if(!rt[y]) ch[pre[y]][ch[pre[y]][1]==y] = x;
    else rt[y] = 0, rt[x] = 1; // 注意在这里维护Splay根节点的关系
    pre[x] = pre[y];
    pre[ch[x][kind]=y] = x;
    Update(y), Update(x);
}
```

3.2.3 Access 操作 Access(x)

Access 操作是打通节点 x 到根的路径，也就是说将根到 x 的边设为实边

具体操作：将节点 x 旋转至 Splay 的根，这个时候 Splay 中右子树部分的节点的深度都比节点 x 大，我们需要把节点 x 与其下方的边设为虚边，然后切断节点 x 与右儿子的联系，然后继续打通节点 x 的父节点与根的关系，再旋转，切断右子树，此时应该将 x 设为其的右儿子，那么这样子的话，我们就打通了一段，继续执行这个操作直到到达根节点

Access 操作有返回值，返回节点 x 到根节点的 Splay 树的根

代码：

```
int Access(int x) {
    int y = 0; // y表示用来记录当x的上一次的值
    while(x) { // 当x=0时表示节点已经打通了与根节点的关系
        Splay(x);
        rt[ch[x][1]] = 1; ch[x][1] = 0;
            // rt[x]表示节点x在Splay中是否为根，因为我们维护的是Splay森林
            // 因为断开x与其右子树的关系所以ch[x][1]分裂成一颗新的Splay
        ch[x][1] = y; rt[y] = 0;
            // 因为我们在上一次循环中只是断开了y与其右子树的关系
            // 但是并没有与其父节点连实边，连边之后y所在的Splay与x合并成一棵Splay
        Update(x); // 因为x在Splay的孩子关系改变，所以要更新节点信息
        y = x, x = pre[x]; // 记录x，因为x尚未与其父节点连出实边，更改变量x的值进行下一次循环
    }
    return y;
}
```

3.2.4 **换根操作** ChangeRoot(x)

换根操作可以讲节点 x 变成所在树的根，主要操作就是先执行 Access 函数，这样节点 x 和原来的根节点就是用实边连接的，然后将节点 x 在 Splay 中旋转至根，打上翻转标记就可以了，因为对于一颗有根树，我们要换根的话只需要讲那么节点和根节点的简单路径的所有边的方向反向即可

3.2.5 **连接操作** Link(x, y)

连接操作可以在节点 x 和节点 y 之间连一条边，做法是让节点 y 变成树根，然后直接把 y 的父亲设为 x 即可

3.2.6 **删边操作** Cut(x, y)

删边操作可以在树中删除边 (x, y)，做法是让 x 变成树的根，然后执行 Access(y), Splay(y)，这样在 Splay 树中节点 y 的做儿子一定只有一个并且这个节点就是 x，直接打断他们的关系即可

3.3 LCT **的一些模板题**

例题 11. 弹飞绵羊[11]

N 个装置，编号为 0-$N-1$，每个装置都有一个系数 k，如果一只绵羊在第 i 个装置上，这个装置的系数是 k_i，那么绵羊会被弹到第 $i+k_i$ 个装置上，如果不存在第 $i+k_i$ 个装置，那么绵羊会被弹飞，M 个操作，每个操作为一下两种

1(1 x): 将一只绵羊放到第 i 个装置上，询问这只从被放置到被弹飞经过了多少个装置

2(2 i x): 将第 i 个装置的系数设为 x

$N \leq 100000, M \leq 200000$，1s/512MB

[11]HNOI 2010

对于一个装置 i，我们连边 $(i, i+k_i)$，如果 $i+k_i \geq N$，那么我们连边 (i, T)，那么这个题目的询问就变成了询问节点 x 到节点 T 的路径上有多少个节点，输出节点数-1，可以使用 LCT 维护

LCT 维护时注意到这个题的加边删边操作是一起进行的 (先删边再加边)，所以我们不需要换根操作就可以完成了，而且加边的时候只需要令 y 的父亲是 x 就好了，根本不需要什么换根操作

例题 12. 洞穴勘测[12]

N 个节点的图，初始时图上没有边，M 个操作，每个操作为以下三种

1(Query u v): 询问图上节点 u 和节点 v 是否联通

2(Connect u v): 在节点 u 与节点 v 之间连一条边，保证节点 u 与节点 v 不连通

3(Destory u v): 在图中删除边 (u, v)，保证存在这条边

$N \leq 10000, M \leq 200000$

(u, v) 在图上联通相当于 $pre[pre[pre[pre[...u]]]] = pre[pre[pre[pre[pre[...v]]]]]$

例题 13. tree[13]

给出一颗 N 个节点的树，每个节点都有一个初值为 1 的点权，M 个操作，每个操作为以下三种:

1(+ u v x): 将节点 u 到节点 v 的简单路径上的所有节点的点权增加 x

2(- u v a b): 在树中删除边 (u, v)，添加边 (a, b)，保证数据合法

3(* u v x): 将节点 u 到节点 v 的简单路径上的所有节点的点权增加 x

4(/ u v): 询问节点 u 到节点 v 的简单路径的权值之和对 51061 取模的值

$1 \leq N \leq 100000, 0 \leq x \leq 10000$

注意标记的下传

例题 14. Query on The Trees[14]

给出一颗 N 个节点的树，每个节点都有一个点权，你需要是这棵树支持以下四种操作:

1(1 u v): 在节点 u 与节点 v 之间连一条边，如果 u, v 在一个联通块则视为非法

2(2 u v): 当节点 u 为根节点时，删除节点 v 与其父节点的边，如果 $u = v$ 或者 u, v 不在一个联通块则视为非法

3(3 x u v): 在节点 u 到节点 v 的路径上的所有节点的点权增加 x，如果 u, v 不在一个联通块则视为非法

4(4 u v): 询问节点 u 到节点 v 路径上的点权最大值

如果操作非法则应该忽略并输出-1

$1 \leq N \leq 300000, 1 \leq M \leq 300000$，5s/65768KB

注意细节

[12]SDOI 2008

[13]BZOJ 2631

[14]HDU 4010

例题 15. 城市旅行[15]

给出一颗 N 个节点的树，每个节点都有一个权值 A_i，M 个操作，每个操作为以下四种之一:

1(1 u v): 如果存在边 (u,v)，那么删除这条边

2(2 u v): 如果节点 u 与节点 v 不连通，则加入边 (u,v)

3(3 u v x): 如果节点 u 与节点 v 联通，则将 u 到 v 的简单路径上的节点的权值增加 x

4(4 u v): 如果节点 u 与节点 v 不连通则输出-1，否则询问在节点 u 到节点 v 的简单路径上任取两点 (x,y)，x 可以等于 y，求节点 x 到节点 y 的点权和的期望

$1 \le N \le 50000, 1 \le M \le 50000, 1 \le A_i \le 10^6, 1 \le x \le 100, 1 \le u, v \le N$

询问相当于是问简单路径上的所有简单路径的权值之和比上简单路径上的所有简单路径的数量

设一条长度为 c 链上的值分别为 $b_1, b_2, b_3, .., b_c$

用 LCT 维护这几个量：链的长度 c，链上权值和 s，

$$f = \sum_{i=1}^{c} i \times b_i$$

$$g = \sum_{i=1}^{c} (n - i + 1) \times b_i$$

$$h = \sum_{i=1}^{c} \sum_{j=i}^{c} \sum_{k=i}^{j} b_k$$

h 也就是简单路径上的所有简单路径的权值和，然后推一推就可以知道他们的关系了，不复杂，这里给出当链加上 x 时，h 的值会增加

$$\frac{t(t+1)(t+2)}{6} \times x$$

时间复杂度:$O(N \log N)$

例题 16. 水题[16]

给出一颗 N 个节点的树，以 1 为根，点 i 有权值为 A_i ，进行 M 次操作，有 4 种操作:

1(1 u v): 将节点 u 的父亲改成 v

2(2 u v x): 将节点 u 到节点 v 的简单路径上的点的权值改成 x

3(3 u): 询问 $F(A_u)$

4(4 u v): 设节点 u 到节点 v 的简单路径上的点的权值分别为 $B_1, B_2, .., B_k$，询问

$$\sum_{i=1}^{k} \sum_{j=i}^{k} F(\sum_{p=i}^{j} B_p)$$

其中，$F(0) = 0, F(1) = 1, \forall n \ge 2, F(n) = F(n-1) + F(n-2)$

$1 \le N, M \le 10^5, A_i, x \in [1, 10^9]$

2s/128MB/O2

[15]BZOJ 3091

[16]WC simulation BY Owaski

类似于城市旅行，不过这次是要求斐波那契数列的和，我们可以维护转移矩阵，设 C 表示斐波那契数列的转移矩阵

设一条长度为 c 链上的值分别为 $b_1, b_2, b_3, .., b_c$

用 LCT 维护这几个量：链的长度 c，链上权值转移矩阵积 s，

$$f = \sum_{i=1}^{c} \prod_{j=1}^{i} C^{b_j}$$

$$g = \sum_{i=1}^{c} \prod_{j=c-i+1}^{c} C^{b_j}$$

$$h = \sum_{i=1}^{c} \sum_{j=i}^{c} \prod_{k=i}^{j} C^{b_k}$$

h 也就是简单路径上的所有简单路径的权值和，然后推一推就可以知道他们的关系了：设 l 表示 x 的左子树，r 表示右子树，v 表示节点 x 的转移矩阵，B 表示单位矩阵，那么有

$$f = f(r) + (f(l) + B) \times s(r) \times v$$

$$g = g(l) + (g(r) + B) \times s(l) \times v$$

$$h = h(l) + h(r) + (f(l) + B) \times (g(r) + B) \times v$$

当链上的值变成 x 时：

$$f = g = \sum_{i=1}^{c} A^{i \times x}$$

$$h = \sum_{i=1}^{c} (c - i + 1) \times A^{i \times x}$$

这个时候 f, h 都可以分治求

时间复杂度:$O(N \log^2 N)$

3.4 一类权值在边上的 LCT 问题

对于权值在边上的 LCT 问题，我们可以把在边的中间加一个顶点，我们可以吧边权放到新加的顶点上，但是这样做边数和顶点树都会加倍

例题 17. 魔法森林[17]

N 个点 M 条边的图，每条边都有两个权值 (a, b)，你需要找到一条从节点 1 到节点 N 的道路，最小化 $\max\{a_i\} + \max\{b_i\}$，$i$ 属于你选择的道路的集合

$2 \le N \le 50000, 0 \le M \le 100000, 1 \le a_i, b_i \le 50000$

1s/256MB

[17]NOI 2014

我们可以将边按照 a 排序，加入一条新边时，如果原来的两个顶点不连通，那么我们就直接添加，否则就找到树上两个节点之间的边权最大值，如果这个最大值比新加的边的值大的话，就把那一条边删掉，再加入这条边，否则不加入这条边

时间复杂度:$O(M \log N)$

例题 18. Codechef MARCH14 GERALD07 加强版[18]

N 个点 M 条边的无向图，Q 次询问，每次询问保留图中编号在 $[l,r]$ 内的边的时候图中的联通块个数，强制在线

$N, M, Q \leq 200000$

对于一个 k 个点的联通块，我们只需要保留其中的 $k-1$ 条边使其保持树的形态就可以了，那么对于这个题目，我们一样可以对一个联通块只保留一颗树，那么我们一次考虑一条边，如果会连出环的话，我们就把最早加入 LCT 的边删掉，用可持久化线段树维护每一次加边时我们联通块内保留的树的编号，就可以做到在线了

时间复杂度:$O(N \log N)$

例题 19. 二分图[19]

N 个点，M 条边，T 个时刻，每条边会在一个时刻产生，也会在一个时刻消失，对于 $[0,T)$ 内的每个时刻，询问当前边形成的图是否为二分图

$N \leq 100000, M \leq 200000, T \leq 100000$

如果图中存在奇环那么这张图就不是二分图

对于图上的一个环，如果形成一个偶环，那么我们可以删除环上最早消失的边，如果形成一个奇环，我们也可以删除环上最早消失的边，但是我们要打上一个计划，表示直到时刻 x(x 为被删除的边消失的时刻)，这个图不可能是二分图

输出的时候检查一下就好了

时间复杂度:$O(M \log N)$

3.5 LCT 维护虚子树信息

之前说过，LCT 有一个特点就是孩子认父亲但父亲不认孩子，这就导致了 LCT 解决一些子树信息的题目就不那么显然了，但是我们可以维护 LCT 虚子树上的信息

会影响到虚边信息的操作有两个,一个是加边操作,因为加上去的边一般都是用虚边表示,另一个是 Access 操作，因为它会导致一些边从虚边变成实边或者一些实边变成虚边

对于加边操作 (x,y)，我们在不需要维护子树信息的时候是直接 x 换成根，在将其 Splay 到 Splay 树的根节点，然后在 y 和 x 之间连上虚边，但是需要维护子树信息的时候，这条虚边会影响到节点 y 到根的所有虚边，解决的方法是打通节点 y 到根的实边再将 y 旋转至 Splay 的根节点，这样虚边 (x,y) 就只会影响到节点 x 的信息

对于 Access 操作，每次循环都会有一条边从实边变成虚边，一条边从虚边变成实边，我们只需要把原来那条虚边的信息删除，在将原来实边的信息加上去就可以了，复杂度不变

[18]BZOJ 3514

[19]BZOJ 4025

例题 20. 大融合[20]

给出 N 个孤点，M 个操作，每个操作为以下两种:

1(A u v): 在节点 u 与节点 v 之间连一条边，保证 u,v 在连边之前不连通

2(Q u v): 询问当前图上有多少条简单路径经过边 (u,v)，保证边 (u,v) 存在

$N,M \leq 100000$

1s/256MB

LCT 维护虚子树信息的模板题

例题 21. 首都[21]

N 个点的图，一开始图上没有边，定义一个联通块为一个国家，一个国家的首都为这个联通块 (树) 的重心，如果重心有两个，则取编号较小的那个，M 个操作，每个操作为以下三种之一

1(Xor): 询问当前所有是首都的节点的编号异或和

2(Ask x): 询问节点 x 所在的国家的首都

3(A u v): 在节点 u 与节点 v 之间连一条边，保证节点 u 与节点 v 不连通

$1 \leq N \leq 100000, 1 \leq M \leq 200000$

LCT 维护树的重心，支持加边操作

错误想法：运用结论：两棵树通过连一条边的构成的新树中，重心一定在两棵树原来重心的连线的路径上，但是具体在链上的那一个位置不确定，所以只能暴力求重心，这样子做在连成链的情况下退化成 $O(N^2 \log N)$

正确想法：运用结论：一棵树添加一个叶子节点是重心最多朝添加的叶子所在的链上移动一个节点，我们在合并的时候暴力把小的树拆开，在一个一个的合并到大的树里面，由于这个结论的存在我们每次只需要花费小树的大小的时间就可以完成此题，在添加叶子的时候，我们把可能变成重心的点转的 Splay 树的根，如果这个点的子树大小的两倍大于整棵树的节点树或者等于整棵树的节点树并且编号较小的话那么这个点就会变成新的重心，所以需要用 LCT 维护虚子树信息

时间复杂度:$O(N \log^2 N)$

例题 22. 共价大爷游长沙[22]

给出一个含 N 个节点的树，使其支持一下四种操作:

1(1 x y u v): 删除树边 (x,y)，添加树边 (u,v)，保证操作完之后依旧是一棵树

2(2 x y): 在可重集合 S 中添加点对 (x,y)

3(3 x): 在集合 S 中删除第 x 次添加的点对

4(4 x y): 询问是否对于所有的 $(u,v) \in S$，都有树上 u 到 v 的简单路径经过 (x,y) 这条边

$N \leq 100000, M \leq 300000$

2s/512MB/O2

对于询问操作，考虑询问的这条边 (x,y)，如果断开边 (x,y)，那么树就会被分成两个联通块，如果 S 中的所有点对 (u,v) 都满足节点 u 和节点 v 不在同一个联通块内就可以了

那么对于在 S 中加入一个点对 (x,y)，我们可以认为是在节点 x 和节点 y 上赋一个值，询问的话就可以看子树内的信息的并，这里可以使用异或，也就是维护子树异或和

[20]BJOI 2014

[21]BZOJ 3510

[22]UOJ 207

时间复杂度:$O(N \log N)$

例题 23. 远行[23]

N 个点，使其支持一下两个操作:

1(1 u v): 在节点 u 与节点 v 之间加一条边，保证加边之前 u, v 不连通

2(2 x): 询问树上与节点 x 距离最远的点的距离

$N \leq 300000, M \leq 500000$

2s/256MB/O2

对于一条链，我们考虑其深度最大的点和深度最小的点，由于在 Splay 中的一颗子树表示的是树上的一条链，我们令 $LS(i)$ 表示从链上深度最小的点出发的最远距离，令 $RS(i)$ 表示从链上深度最大的点出发的最远距离。

那么这样询问的话打通节点 x 与根节点的实链，此时 x 一定是其中深度最大的节点，此时 $RS()$ 就是答案

考虑加边的影响，我们使用一个可重集合来维护一个从一个节点出发的所有虚链的长度，并记录从节点出发走虚边的最大长度 (也就是可重集合中的最大值)，加边 (x, y) 的话就是修改虚边父节点 x 的虚链信息，用从 y 出发可以到达的最远长度 (也就是 $LS(y)$，因为此时 y 是子树的根并且在 Splay 的根节点，那么 y 的深度最小)

Access 操作相当于从可重集合中删除一个数在加入一个数，也是可以维护的

时间复杂度:$O(N \log^2 N)$

例题 24. 维护直径[24]

N 个点，使其支持以下三种操作:

1(link u v): 在节点 u 与节点 v 之间连一条边

2(cut u v): 删除节点 a 与节点 b 之间的边

3(query x): 询问节点 x 所在的树的直径

$N \leq 100000, M \leq 250000$

1.5s/512MB/O2

上一题的加强版，首先删除肯定是关于实链的操作，所以可以不用管它

对于直径我们还需要维护另一个可重集合，表示节点的虚子树的直径长度，以下量可以更新直径 (假定 Splay 中左右子树存在，为 l, r，上一题的可重集合的最大值和次大值为 a, b，直径用 M 表示)：$a + b + 1, M(l), M(r), RS(l) + a, LS(r) + a, RS(l) + LS(r) + 1$ 以及两个可重集合的最大值

时间复杂度:$O(N \log^2 N)$

[23]LOJ 6038

[24]Author：pyh

4 SPOJ 上的 QTREE 问题

推荐做题顺序:1236754

例题 25. Query on a tree[25]

给出一颗 N 个节点的树，每条边都有一个边权，M 个操作，每个操作为以下两种之一:

1(CHANGE i v): 将第 i 条边的权值修改为 v

2(QUERY u v): 询问简单路径 (u,v) 上的边权最大值

$N \leq 10000$

851ms/1572864MB

树链剖分模板题

例题 26. Query on a tree II[26]

给出一颗 N 个节点的树，每条边都有一个边权，M 个操作，每个操作为以下两种之一:

1(DIST u v): 询问简单路径 (u,v) 上的边权和

2(KTH u v k): 询问简单路径 (u,v) 上的第 k 个点的编号

$N \leq 10000$

433ms/1572864MB

倍增

例题 27. Query on a tree again![27]

给出一个 N 个节点的树，每个节点都有一个颜色，初始时为白色，M 个操作，每个操作为以下两种:

1(0 u): 改变节点 u 的颜色，如果是黑色则变成白色，如果是白色则变成黑色

2(1 u): 询问简单路径 $(1,u)$ 上的第一个黑色节点的编号

$N \leq 100000, M \leq 400000$

2s/1572864MB

树链剖分模板

[25]SPOJ QTREE

[26]SPOJ QTREE2

[27]SPOJ QTREE3

例题 28. Query on a tree IV[28]

给出一颗 N 个节点的树，每个节点都有一个颜色，每条边有一个边权，M 个操作，每个操作为以下两种:

1(0 u): 改变节点 u 的颜色，如果是黑色则变成白色，如果是白色则变成黑色

2(1 u): 询问树上两个白点的最大边权和

$1 \leq N \leq M \leq 100000$

100ms/1572864KB

树链剖分维护轻边信息，对于一个节点，我们用一个可重集合距离下节点 x 下面的轻子树到节点 x 的距离，用线段树维护 $[l,r]$ 一个端点为 l，经过区间 $[l,r]$ 的点 (可以只经过 l) 然后走轻边 (可以不走) 到一个白点的最大边权和，一个端点为 r，经过区间 $[l,r]$ 的点 (可以只经过 r) 然后走轻边 (可以不走) 到一个白点的最大边权和，从一个白点出发通过走轻边经过 $[l,r]$ 的点后在轻边到白点的最大边权和 X(白点两个端点可以相同，可以都在 $[l,r]$ 内) 就可以了，最后用一个可重集合维护每条重链的 X 的最大值就可以了

颜色修改的时候顺着轻边往上走就好了

时间复杂度:$O(N \log^2 N)$

例题 29. Query on a tree V[29]

给出一颗 N 个节点的树，每个节点都有一个颜色，M 个操作，每个操作为以下两种:

1(0 u): 改变节点 u 的颜色，如果是黑色则变成白色，如果是白色则变成黑色

2(1 u): 询问距离节点 u 最近的白点的距离

$1 \leq N \leq M \leq 100000$

714ms/1572864KB

树链剖分维护轻边信息，对于一个节点，我们用一个可重集合距离下节点 x 下面的轻子树到节点 x 的距离，用线段树维护 $[l,r]$ 一个端点为 l，经过区间 $[l,r]$ 的点 (可以只经过 l) 然后走轻边 (可以不走) 到一个白点的最大边权和，一个端点为 r，经过区间 $[l,r]$ 的点 (可以只经过 r) 然后走轻边 (可以不走) 到一个白点的最大边权和，维护和询问的时候往上跳就可以了

时间复杂度:$O(N \log^2 N)$

例题 30. Query on a tree VI[30]

给出一颗 N 个节点的树，每个节点都有一个颜色，初始时为黑色，M 个操作，每个操作为以下两种:

1(0 u): 询问有多少个 v 满足简单路径 (u,v) 上的颜色一样

2(1 u): 改变节点 u 的颜色，如果是黑色则变成白色，如果是白色则变成黑色

$1 \leq N, M \leq 100000$

1s/1572864KB

树链剖分，开两颗线段树，一颗维护白点，另一颗维护黑点，在白树上所有黑点的权值为负无穷，白点为 1；黑树上白点的权值为负无穷，黑点为 1，那么这个题目就是在某棵树上查联通块的最大权值和了

我们可以树链剖分维护用轻边连接的子树的最大联通块的和，可以与处理出来，然后查询和修改的话就是顺着轻边往上走

时间复杂度:$O(N \log^2 N)$

[28]SPOJ QTREE4

[29]SPOJ QTREE5

[30]SPOJ QTREE6

例题 31. Query on a tree VII[31]

给出一颗 N 个节点的树，每个节点都有一个点权和颜色，M 个操作，每个操作为以下两种:

1(0 u): 询问路径 (u,v) 上的点权最大值，其中路径 (u,v) 满足路径 (u,v) 上所有点的颜色相同

2(1 u): 改变节点 u 的颜色，如果是黑色则变成白色，如果是白色则变成黑色

3(2 u v): 将节点 u 的点权改成 v

$1 \leq N, M \leq 100000$

1s/1572864KB

类似于 Query on a tree VI，我们维护对于两种颜色各维护一颗线段树，转移的时候注意是点权最大值而不是点权和。轻链信息用一个可重集合维护

时间复杂度:$O(N \log^2 N)$

[31]SPOJ QTREE7