[**树链剖分详解及模板**](http://blog.csdn.net/y990041769/article/details/40348013)

树链剖分是解决在树上进行插点问线，插线问点等一系列树上的问题。假如现在给你一棵树，然后没两条边之间有一条权值，有一些操作，1：x---y之间的最大权值是多少，2：改变x---y之间的权值。当前这样的操作有很多，如果直接用暴力的方法的话肯定不行，那么就要想一个好的方法，我们可以想一下能不能借助线段树解决，能不能想一种方法对树上的边进行编号，然后就变成区间了。那么我们就可以在线段树上进行操作了，树链剖分就是这样的一个算法。

当然编号不是简单的随便编号，如果我们进行随便的编号，然后建立一个线段树，如果要更新一个边的权值，是log2（n）的复杂度，而查找的话，我们要枚举x--y的之间的所有的边，假如我们随便以一个点为根节点进行编号，最大的长度是树的直径，这个值本身是比较大的，而在线段树上查找任意一个区间的复杂度也是log2（n），这样查找一次的时间复杂度比直接暴力还要高，所以很明显是不行的。

那么就要想想办法了，我们能不能把x--y之间的一些边一块儿查找，这就是关于树链剖分的重边和轻边，

重边：某个节点x到孩子节点形成的子树中节点数最多的点child之间的边，由定义发现除了叶子节点其他节点只有一条重边

重边是可以放在一块儿更新的，而有

性质：从根到某一点的路径上轻边、重边的个数都不大于logn。

所以这样查找的时间复杂度相当于log2（n）

其实树链剖分就是把边哈希到线段树上的数据结构。

实现的话很简单，用两个dfs处理数数的信息，重边以及轻边，然后就是一些线段树的操作了。

模板“：以spoj 375 为例

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/y990041769/article/details/40348013)[在CODE上查看代码片](https://code.csdn.net/snippets/491444)

#include <cstdio>

#include <cstring>

#include <vector>

#include <algorithm>

**using** **namespace** std;

#define Del(a,b) memset(a,b,sizeof(a))

**const** **int** N = 10005;

**int** dep[N],siz[N],fa[N],id[N],son[N],val[N],top[N]; //top 最近的重链父节点

**int** num;

vector<**int**> v[N];

**struct** tree

{

**int** x,y,val;

**void** read(){

        scanf("%d%d%d",&x,&y,&val);

    }

};

tree e[N];

**void** dfs1(**int** u, **int** f, **int** d) {

    dep[u] = d;

    siz[u] = 1;

    son[u] = 0;

    fa[u] = f;

**for** (**int** i = 0; i < v[u].size(); i++) {

**int** ff = v[u][i];

**if** (ff == f) **continue**;

        dfs1(ff, u, d + 1);

        siz[u] += siz[ff];

**if** (siz[son[u]] < siz[ff])

            son[u] = ff;

    }

}

**void** dfs2(**int** u, **int** tp) {

    top[u] = tp;

    id[u] = ++num;

**if** (son[u]) dfs2(son[u], tp);

**for** (**int** i = 0; i < v[u].size(); i++) {

**int** ff = v[u][i];

**if** (ff == fa[u] || ff == son[u]) **continue**;

        dfs2(ff, ff);

    }

}

#define lson(x) ((x<<1))

#define rson(x) ((x<<1)+1)

**struct** Tree

{

**int** l,r,val;

};

Tree tree[4\*N];

**void** pushup(**int** x) {

    tree[x].val = max(tree[lson(x)].val, tree[rson(x)].val);

}

**void** build(**int** l,**int** r,**int** v)

{

    tree[v].l=l;

    tree[v].r=r;

**if**(l==r)

    {

        tree[v].val = val[l];

**return** ;

    }

**int** mid=(l+r)>>1;

    build(l,mid,v\*2);

    build(mid+1,r,v\*2+1);

    pushup(v);

}

**void** update(**int** o,**int** v,**int** val)  //log(n)

{

**if**(tree[o].l==tree[o].r)

    {

        tree[o].val = val;

**return** ;

    }

**int** mid = (tree[o].l+tree[o].r)/2;

**if**(v<=mid)

        update(o\*2,v,val);

**else**

        update(o\*2+1,v,val);

    pushup(o);

}

**int** query(**int** x,**int** l, **int** r)

{

**if** (tree[x].l >= l && tree[x].r <= r) {

**return** tree[x].val;

    }

**int** mid = (tree[x].l + tree[x].r) / 2;

**int** ans = 0;

**if** (l <= mid) ans = max(ans, query(lson(x),l,r));

**if** (r > mid) ans = max(ans, query(rson(x),l,r));

**return** ans;

}

**int** Yougth(**int** u, **int** v) {

**int** tp1 = top[u], tp2 = top[v];

**int** ans = 0;

**while** (tp1 != tp2) {

        //printf("YES\n");

**if** (dep[tp1] < dep[tp2]) {

            swap(tp1, tp2);

            swap(u, v);

        }

        ans = max(query(1,id[tp1], id[u]), ans);

        u = fa[tp1];

        tp1 = top[u];

    }

**if** (u == v) **return** ans;

**if** (dep[u] > dep[v]) swap(u, v);

    ans = max(query(1,id[son[u]], id[v]), ans);

**return** ans;

}

**void** Clear(**int** n)

{

**for**(**int** i=1;i<=n;i++)

        v[i].clear();

}

**int** main()

{

    //freopen("Input.txt","r",stdin);

**int** T;

    scanf("%d",&T);

**while**(T--)

    {

**int** n;

        scanf("%d",&n);

**for**(**int** i=1;i<n;i++)

        {

            e[i].read();

            v[e[i].x].push\_back(e[i].y);

            v[e[i].y].push\_back(e[i].x);

        }

        num = 0;

        dfs1(1,0,1);

        dfs2(1,1);

**for** (**int** i = 1; i < n; i++) {

**if** (dep[e[i].x] < dep[e[i].y]) swap(e[i].x, e[i].y);

            val[id[e[i].x]] = e[i].val;

        }

        build(1,num,1);

**char** s[200];

**while**(~scanf("%s",&s) && s[0]!='D')

        {

**int** x,y;

            scanf("%d%d",&x,&y);

**if**(s[0]=='Q')

                printf("%d\n",Yougth(x,y));

**if** (s[0] == 'C')

                update(1,id[e[x].x],y);

        }

        Clear(n);

    }

**return** 0;

}