格式

1. 英文/公式左右两边跟汉字需要空格
2. 结点 u, v 等也需要使用公式
3. log 使用 $\log$
4. 句子结束请使用句号
5. 建议写 merge，mergE 很怪

内容：个人感觉讲得不够清楚，需要反复看几遍，并结合代码才能大致看懂。建议用形式化的语言写明合并过程，在简介部分举例（有图最好）等方法改进。另外这个复杂度怎么是个玄学。。没有严谨一点的证明吗？建议再检索一下资料。

线段树合并

简介

有些树型的题目会需要在一个结点维护一棵线段树，例如每次把 u到v 的路径上的每个点内加一个元素 k，然后求每个节点中出现次数最多的数，这时可以在每个点开一棵权值线段树，树上差分之后在 dfs 中不断合并父子节点代表的两棵线段树。

其中的关键步骤是**动态开点**和**线段树的合并操作**

因为不可能真的在每个点建一棵完整的线段树（而且一般情况下每个点所代表的权值都只是完整线段树的小部分），所以要动态开点。

```
void update(int &id,int l,int r,int pos,int k)
{
    if(!id) id=++tot;  //动态开点
    if(l==r){
        //更新操作（例如把这个位置的值加1
        return ;
    }
    int mid = (l+r)>>1;
    if(pos<=mid) update(tr[id].lc,l,mid,pos,k);
    else update(tr[id].rc,mid+1,r,pos,k);
    push_up(id);
}
```

然后合并的操作主要基于两棵线段树具有相同的值域所以有相同的结构，那么递归的处理左右子树即可。

例如在维护区间内数字个数时：都从各自的根开始递归左右子树，如果有一个棵树的子树为空则返回另一棵树的子树（的编号），否则一直合并到叶子。

注：结点中的数字表示他的子树中有多少个数字，不是编号

```
int merge(int p,int q,int l,int r)
{
    if(!p)return q;if(!q)return p;
```

```
    if(l==r){
        // merge_leaf 操作
        return p;
    }
    int mid = (l+r)>>1;
    tr[p].lc = merge(tr[p].lc,tr[q].lc,l,mid);
    tr[p].rc = merge(tr[p].rc,tr[q].rc,mid+1,r);
    push_up(p);
    return p;
}
```

复杂度

空间复杂度比较要注意（因为涉及数组开多大QAQ□□每次 update 最多开 $logn$ 个点，所以一般开 $mlogn$ 的数组□m 表示 update 次数□n 表示权值线段树的值域上限。

同空间复杂度，每次 update 最多开 $logn$ 个点，所以这里的时间复杂度是 $mlogn$□而 $merge$ 的复杂度正比于叶子节点的个数，又因为 m 次操作最多产生 m 个叶子，所以最坏的情况就是合并了 m 个叶子节点及其路径，每次合并都是 $logn$ 级别的，所以这部分的复杂度也是 $mlogn$□

例题

P4556雨天的尾巴

给出一棵树□m 次操作，每次操作让 path<u,v> 上的点都发放第 k 种粮食。所有 m 次操作完后，求每个点存放最多的粮食的种类是什么。

显然是树上点差分，但是值域很大，可以在每个点建一棵权值线段树，然后合并即可，注意差分在 update 上有常数 4，所以开数组要多乘以 4。

```
#include<bits/stdc++.h>
#define p_ pair<int,int>
#define mp_ make_pair
#define ll long long
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
int
n,m,fa[maxn][35],dep[maxn],lg[maxn],X[maxn],Y[maxn],Z[maxn],mxz,ans[maxn],c
```

```
nt,rt[maxn],idz[maxn];
vector<int> g[maxn];
struct node
{
    int lc,rc,mx,id;
}tr[maxn*80];
inline void dfs(int u,int f)
{
    dep[u]=dep[f]+1,fa[u][0]=f;
    rep(i,1,lg[dep[u]]) fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int v:g[u])if(v!=f){
        dfs(v,u);
    }
}
inline int lca(int x,int y)
{
    if(dep[x]<dep[y])swap(x,y);
    while(dep[x]>dep[y]) x=fa[x][lg[dep[x]-dep[y]]-1];
    if(x==y) return x;
    for(int i=lg[dep[x]-1];i>=0;i--){
        if(fa[x][i]!=fa[y][i])x=fa[x][i],y=fa[y][i];
    }
    return fa[x][0];
}
inline void push_up(int id)
{
    int lc=tr[id].lc,rc=tr[id].rc;
    if(tr[lc].mx>=tr[rc].mx) tr[id].mx=tr[lc].mx,tr[id].id=tr[lc].id;
    else tr[id].mx=tr[rc].mx,tr[id].id=tr[rc].id;
}
void update(int &id,int l,int r,int pos,int v)
{
    if(!id) id=++cnt;
    if(l==r){
        tr[id].mx+=v;tr[id].id=l;return ;
    }
    int mid = (l+r)>>1;
    if(pos<=mid) update(tr[id].lc,l,mid,pos,v);
    else update(tr[id].rc,mid+1,r,pos,v);
    push_up(id);
}
int mergE(int p,int q,int l,int r)
{
    if (!p) return q; if (!q) return p;
    if(l==r){
        tr[p].mx+=tr[q].mx;return p;
    }
    int mid = (l+r)>>1;
    tr[p].lc = mergE(tr[p].lc,tr[q].lc,l,mid);
    tr[p].rc = mergE(tr[p].rc,tr[q].rc,mid+1,r);
    push_up(p);
```

```
    return p;
}
void dfs2(int u,int f)
{
    for(int v:g[u])if(v!=f){
        dfs2(v,u);
        rt[u] = mergE(rt[u],rt[v],1,mxz);
    }
    if(tr[rt[u]].mx) ans[u] = tr[rt[u]].id;
}
int main()
{
    fastio();
    cin>>n>>m;
    rep(i,1,n) lg[i]=lg[i-1]+(1<<lg[i-1]==i);
    rep(i,1,n-1) {int u,v;cin>>u>>v;g[u].pb(v);g[v].pb(u);}
    dfs(1,0);
    rep(i,1,m){
        cin>>X[i]>>Y[i]>>Z[i];
        idz[i] = Z[i];
    }
    sort(idz+1,idz+m+1);
    mxz = unique(idz+1,idz+m+1)-idz-1;
    rep(i,1,m){
        int z = lower_bound(idz+1,idz+mxz+1,Z[i])-idz;
        update(rt[X[i]],1,mxz,z,1);
        update(rt[Y[i]],1,mxz,z,1);
        int Lca = lca(X[i],Y[i]);
        update(rt[Lca],1,mxz,z,-1);
        update(rt[fa[Lca][0]],1,mxz,z,-1);
    }
    dfs2(1,0);
    rep(i,1,n) cout<<idz[ans[i]]<<endl;
    return 0;
}
```

P3605 [USACO17JAN]Promotion Counting P

一棵带点权的树，求每个点 u 的子树中有多少个点的点权比 u 自身大。

也是每个点建一个权值线段树，权值线段树维护每个值出现的次数，然后dfs过程中合并完后query 一下。

P3224 HNOI2012]永无乡

n 个点，两个操作，一是连接两个点，二是求某个点和所有和他相连的点的第k小。

第 k 小也是权值线段树干的事儿，连接的操作用并查集维护。

CF600E. Lomsat gelral

树上每个点染色 c_i （好像树上染色可以用 dsu （原来我不会啊，那没事了），求每个点子树出现次数最多的颜色的和（可能多个颜色同时出险次数最多）。

也是线段树瞎搞搞，最后 dfs 合并一下。