树链剖分

树链剖分

```
#include<iostream>
#include<algorithm>
#include<vector>
#define int long long
using namespace std;
int n,m,r,pr,mod;
int
siz[100000+50],top[100000+50],dfsn[100000+50],fa[100000+50],dep[100000+5
0],hson[100000+50];
int wt[100000+50],cnt=0,w[100000+50],rdfsn[100000+50];
struct node
{
    int x,tag;
}tree[400000+50];
int ls(int p){return p<<1;}
int rs(int p){return p<<1|1;}
void build(int p,int l,int r){
    if(l==r){
        tree[p].x=wt[l];
        tree[p].x%=mod;
        return;
    }
    int mid=(l+r)>>1;
    build(ls(p),l,mid);
    build(rs(p),mid+1,r);
    tree[p].x=(tree[ls(p)].x+tree[rs(p)].x)%mod;
}
void pushdown(int p,int l,int r){
    int mid=(l+r)>>1;
    (tree[ls(p)].x+=tree[p].tag*(mid-l+1))%mod;
    (tree[ls(p)].tag+=tree[p].tag)%mod;
    (tree[rs(p)].x+=tree[p].tag*(r-(mid+1)+1))%mod;
    (tree[rs(p)].tag+=tree[p].tag)%mod;
    tree[p].tag=0;
}
void updata(int p,int l,int r,int ql,int qr,int x){
    if(l>qr||r<ql)return;
    if(ql<=l&&qr>=r){
        tree[p].x+=x*(r-l+1);
        tree[p].x%=mod;
        tree[p].tag+=x;
        tree[p].tag%=mod;
        return;
    }
    pushdown(p,l,r);
    int mid=(l+r)>>1;
    updata(ls(p),l,mid,ql,qr,x);
    updata(rs(p),mid+1,r,ql,qr,x);
    tree[p].x=(tree[ls(p)].x+tree[rs(p)].x)%mod;
}
int query(int p,int l,int r,int ql,int qr){
    if(l>qr||r<ql)return 0;
    if(ql<=l&&qr>=r){
        return tree[p].x%mod;
    }
    pushdown(p,l,r);
    int mid=(l+r)>>1;
    int res=0;
    res+=query(ls(p),l,mid,ql,qr);
    res+=query(rs(p),mid+1,r,ql,qr);
    tree[p].x=(tree[ls(p)].x+tree[rs(p)].x)%mod;
    return res%mod;
}
vector<int>p[100000+50];
void dfs1(int rt,int fas,int d){
    int size=1;
    int mx=0;
    dep[rt]=d;
    for(auto j:p[rt]){
        if(j==fas)continue;
        dfs1(j,rt,d+1);
        size+=siz[j];
        if(siz[j]>mx){
            mx=siz[j];
            hson[rt]=j;
        }
        fa[j]=rt;
    }
    siz[rt]=size;
}
```

```
void dfs2(int x,int fas){
    dfsn[x]=++cnt;
    wt[cnt]=w[x];
    if(hson[x]!=0){
        top[hson[x]]=top[x];
        dfs2(hson[x],x);
    }
    for(auto j:p[x]){
        if(j==fas||dfsn[j])continue;
        if(!top[j]){
            top[j]=j;
            dfs2(j,x);
        }
    }
    rdfsn[x]=cnt;
}
int qrange(int x,int y){
    int ans=0;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        ans+=query(1,1,n,dfsn[top[x]],dfsn[x]);
        ans%=mod;
        x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    ans+=query(1,1,n,dfsn[x],dfsn[y]);
    return ans%mod;
}
int qson(int x){
    return query(1,1,n,dfsn[x],rdfsn[x])%mod;
}
void qupl(int x,int y,int k){
    k%=mod;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        updata(1,1,n,dfsn[top[x]],dfsn[x],k);
        x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    updata(1,1,n,dfsn[x],dfsn[y],k);
}
void qups(int x,int k){
     updata(1,1,n,dfsn[x],rdfsn[x],k);
}
void solve(){
    cin>>n>>m>>r>>pr;
    mod=pr;
    for(int i=1;i<=n;++i)cin>>w[i];
    for(int i=1;i<=n-1;++i){
        int u,v;
        cin>>u>>v;
        p[u].push_back(v);
        p[v].push_back(u);
    }
    dfs1(r,0,1);
    dfs2(r,r);
    build(1,1,n);
    while(m--){
        int k,x,y,z;
        cin>>k;
        if(k==1){
            cin>>x>>y>>z;
            qupl(x,y,z);
        }
        else if(k==2){
            cin>>x>>y;
            cout<<qrange(x,y)<<"\n";
        }
        else if(k==3){
            cin>>x>>y;
            qups(x,y);
        }
        else{
            cin>>x;
            cout<<qson(x)<<"\n";
        }
    }
}
signed main(){
    std::ios::sync_with_stdio(false);
    solve();
}
```

splay

```
Splay 完成区间查询
#include<iostream>
#include<algorithm>
#include<vector>
#include<set>
#include<map>
using namespace std;
#define int long long
int ch[100000+50][2];
int rt,siz[100000+50];//子树大小
```

```
int val[100000+50],cnt[100000+50];//val[x]在x上出现多少次
int fa[100000+50],tot;
int tag[100000+50];
int n,m;
void push_up(int x){
    siz[x]=siz[ch[x][0]]+siz[ch[x][1]]+cnt[x];
}
int bulid(int l,int r,int fat){//一般是bulid(0,n+1,0);
    if(l>r)return 0;
    int x=++tot;
    int mid=(l+r)>>1;
    fa[x]=fat;
    cnt[x]=1;
    val[x]=mid;
    ch[x][0]=bulid(l,mid-1,x);
    ch[x][1]=bulid(mid+1,r,x);
    push_up(x);
    return x;
}
void push_down(int x){
    if(x&&tag[x]){
        if(ch[x][0])tag[ch[x][0]]^=1;
        if(ch[x][1])tag[ch[x][1]]^=1;
        swap(ch[x][0],ch[x][1]);
        tag[x]=0;
    }
}
bool get(int x){
    return x==ch[fa[x]][1];
}
void rotate(int x){
    int y=fa[x],z=fa[y],flag=get(x);
    push_down(x);
    push_down(y);
    ch[y][flag]=ch[x][flag^1];
    if(ch[x][flag^1]){
        fa[ch[x][flag^1]]=y;
    }
    ch[x][flag^1]=y;
    fa[y]=x;
    fa[x]=z;
    if(z)ch[z][y==ch[z][1]]=x;
    push_up(y);
}

void clear(int x){
    ch[x][0]=ch[x][1]=fa[x]=val[x]=siz[x]=cnt[x]=0;
}
```

```
void splay(int x,int goal=0){
    for(int f=fa[x];(f=fa[x])!=goal;rotate(x)){
        if(fa[f]!=goal)rotate(get(x)!=get(f)?f:x);
    }
    if(goal==0)rt=x;
}
void ins(int k){
    if(!rt){
        val[++tot]=k;
        cnt[tot]++;
        rt=tot;
        push_up(rt);
        return ;
    }
    int cur=rt,f=0;
    while(1){
        if(val[cur]==k){
            cnt[cur]++;
            push_up(cur);
            push_up(f);
            splay(cur);
            break;
        }
        f=cur;
        cur=ch[cur][val[cur]<k];
        if(!cur){
            val[++tot]=k;
            cnt[tot]++;
            fa[tot]=f;
            ch[f][val[f]<k]=tot;
            push_up(tot);
            push_up(f);
            splay(tot);
            break;
        }
    }
}
int rk(int k) {
    int res = 0, cur = rt;
    while (1) {
        if (k < val[cur]) {
            cur = ch[cur][0];
        }
        else {
            res += siz[ch[cur][0]];
            if (k == val[cur]) {
                splay(cur);
                return res + 1;
            }
```

```
            res += cnt[cur];
            cur = ch[cur][1];
        }
    }
}
int kth(int k) {
    int cur = rt;
    while (1) {
        push_down(cur);
        if (ch[cur][0] && k <= siz[ch[cur][0]]) {
            cur = ch[cur][0];
        }
        else {
            k -= cnt[cur] + siz[ch[cur][0]];
            if (k <= 0) {
                splay(cur);
                return cur;
            }
            cur = ch[cur][1];
        }
    }
}
int fin(int x){
    return kth(x+1);
}
void rev(int l ,int r){
    int fl=fin(l-1);
    int fr=fin(r+1);
    splay(fl,0);
    splay(fr,fl);
    int pos=ch[rt][1];
    pos=ch[pos][0];
    tag[pos]^=1;
}
void dfs(int x){
    push_down(x);
    if(ch[x][0])dfs(ch[x][0]);
    if(val[x]!=0&&val[x]!=n+1)cout<<val[x]<<" ";
    if(ch[x][1])dfs(ch[x][1]);
}
void solve(){
    cin>>n>>m;
    bulid(0,n+1,0);
    rt=1;
    while(m--){
        int l,r;
        cin>>l>>r;
        rev(l,r);
    }
    dfs(rt);
}
signed main(){
    std::ios::sync_with_stdio(false);
    solve();
}
```

线段树分治

```
#线段树分治
#include<iostream>
#include<vector>
#include<algorithm>
#define int long long
using namespace std;
int left_son(int x){return x<<1;}
int righ_son(int x){return x<<1|1;}
int fa[500000+50],si[500000+50];
vector<pair<int &,int>>hisfa;
vector<pair<int &,int>>hissi;
void init(int n){
    for(int i=1;i<=n;++i){
        fa[i]=i;
        si[i]=1;
    }
}
int find(int x){
    while(x!=fa[x])x=fa[x];
    return x;
}
int size(int x){
    return si[find(x)];
}
void merge(int u,int v){
    int x=find(u),y=find(v);
    if(x==y)return ;
    if(si[x]<si[y])swap(x,y);
    hisfa.push_back({fa[y],fa[y]});
    fa[y]=x;
    hissi.push_back({si[x],si[x]});
    si[x]=si[x]+si[y];
}
bool cmp(int u,int v){
    return find(u)==find(v);
}
int history(){
    return hisfa.size();
}
```

```
void roll(int h){
    while(hisfa.size()>h){
        hisfa.back().first=hisfa.back().second;
        hisfa.pop_back();
        hissi.back().first=hissi.back().second;
        hissi.pop_back();
    }
}
vector<pair<int,int>>col[500000+80];
int n;
vector<int>seg[500000*4+5];
void add_seg(int p,int l,int r,int al,int ar,int e){
    if(al>r||ar<l)return ;
    if(ar>=r&&al<=l){
        seg[p].push_back(e);
        return ;
    }
    int mid=(l+r)>>1;
    add_seg(left_son(p),l,mid,al,ar,e);
    add_seg(righ_son(p),mid+1,r,al,ar,e);
}
void add_seg(int l,int r,int e){
    add_seg(1,1,n,l,r,e);
}
int ans=0;
void query(int p,int l,int r){
    if(l==r){
        int h=history();
        for(auto c:seg[p]){
            for(auto j:col[c]){
                merge(j.first,j.second);
            }
        }
        for(auto p:col[l]){
            ans+=size(p.first)*size(p.second);
        }
        cout<<ans<<"\n";
        roll(h);
    }
    else{
        int h=history();
        for(auto c:seg[p]){
            for(auto j:col[c]){
                merge(j.first,j.second);
            }
        }
        int mid=(l+r)>>1;
        query(left_son(p),l,mid);
        query(righ_son(p),mid+1,r);
        roll(h);
    }
}
void solve(){
    cin>>n;
    for(int i=1;i<=n-1;++i){
        int u,v,w;
        cin>>u>>v>>w;
        col[w].push_back({u,v});
    }
    init(n);
    for(int i=1;i<=n;++i){
        if(i==1){
            add_seg(2,n,i);
        }
        else if(i==n){
            add_seg(1,n-1,i);
        }
        else{
            add_seg(1,i-1,i);
            add_seg(i+1,n,i);
        }
    }
    query(1,1,n);
    //cout<<ans<<"\n";
}
signed main(){
    std::ios::sync_with_stdio(false);
    solve();
}
```

lct2

```
Lct2
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<vector>
#include<cstring>
using namespace std;
//#define int long long
int ch[200050][2],fa[200050],tag[200050],siz[200050],val[200050];
void clear(int x){ch[x][0]=ch[x][1]=fa[x]=tag[x]=val[x]=siz[x]=0;}
int getch(int x){return x==ch[fa[x]][1];}
```

```
int isroot(int x){return ch[fa[x]][0]!=x&&ch[fa[x]][1]!=x;}
void pushup(int x){siz[x]=val[x]^siz[ch[x][0]]^siz[ch[x][1]];}
void pushdown(int x){
    if(tag[x]){
        if(ch[x][0])swap(ch[ch[x][0]][0],ch[ch[x][0]][1]),tag[ch[x]
[0]]^=1;
        if(ch[x][1])swap(ch[ch[x][1]][0],ch[ch[x][1]][1]),tag[ch[x]
[1]]^=1;
        tag[x]=0;
    }
}

//黑盒  旋转到当前块的根
//也就是说，我们对于同一块内的splay进行选择，他之后还是中序遍历的结果还是对应着原树中
的一条实链
void updata(int x){
    if(!isroot(x))updata(fa[x]);
    pushdown(x);
}
void rorate(int x){
    int y=fa[x],z=fa[y],chx=getch(x),chy=getch(y);
    fa[x]=z;
    if(!isroot(y))ch[z][chy]=x;
    ch[y][chx]=ch[x][chx^1];
    fa[ch[x][chx^1]]=y;
    ch[x][chx^1]=y;
    fa[y]=x;
    pushup(y),pushup(x);
}
void splay(int x){
    updata(x);
    for(int f=fa[x];f=fa[x],!isroot(x);rorate(x)){
        if(!isroot(f))rorate(getch(x)==getch(f)?f:x);
    }
}

//access 在原树中把从根到x的所有点放在一条实链里，使根到x成为一条实路径，并且在同一棵
Splay 里。并且 下面没有实边。
void access(int x){
    int y=0;
    while(x){
        splay(x);
        ch[x][1]=y;
        pushup(x);
        y=x;
        x=fa[x];
    }
}
// makeroot(x)在原树中，把x当成根
```

```
void makeroot(int x){
    access(x);// 此操作后 x的splay块的 根 变成了 原树的根
    splay(x);
    swap(ch[x][0],ch[x][1]);
    tag[x]^=1;
}
int find(int x){
    access(x);
    splay(x);
    while (ch[x][0])x=ch[x][0];
    splay(x);
    return x;
}
void link(int x,int y){
    if(find(x)!=find(y))makeroot(x),fa[x]=y;
}
void split(int x,int y){
    makeroot(x);
    access(y);
    splay(y);
    //在辅助树上，y的子树就包含了 路径上的信息
}
void cut(int x,int y){
    split(x,y);
    if(ch[y][0]==x&&!ch[x][1])ch[y][0]=fa[x]=0;
}
int que(int x,int y){
    split(x,y);
    return siz[y];
}
void sets(int x,int y){
    splay(x);
    val[x]=y;
    pushup(x);
}
void solve(){
    int n,m;
    cin>>n>>m;
    int x,y,z;
    for(int i=1;i<=n;++i)cin>>val[i];
    while (m--){
        cin>>x>>y>>z;
        if(x==0){
            cout<<que(y,z)<<"\n";
        }
        else if(x==1){
            link(y,z);
        }
        else if(x==2){
```

```
            cut(y,z);
        }
        else sets(y,z);
    }

}
signed main(){
    std::ios::sync_with_stdio(false);
    solve();
}
```

lct1

```
Lct1
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<vector>
#include<cstring>
using namespace std;
//#define int long long
int ch[200050][2],fa[200050],tag[200050];
void clear(int x){ch[x][0]=ch[x][1]=fa[x]=tag[x]=0;}
int getch(int x){return x==ch[fa[x]][1];}
int isroot(int x){return ch[fa[x]][0]!=x&&ch[fa[x]][1]!=x;}
void pushup(int x){}
void pushdown(int x){
    if(tag[x]){
        if(ch[x][0])swap(ch[ch[x][0]][0],ch[ch[x][0]][1]),tag[ch[x]
[0]]^=1;
        if(ch[x][1])swap(ch[ch[x][1]][0],ch[ch[x][1]][1]),tag[ch[x]
[1]]^=1;
        tag[x]=0;
    }
}
//黑盒  旋转到当前块的根
void updata(int x){
    if(!isroot(x))updata(fa[x]);
    pushdown(x);
}
void rorate(int x){
    int y=fa[x],z=fa[y],chx=getch(x),chy=getch(y);
    fa[x]=z;
    if(!isroot(y))ch[z][chy]=x;
    ch[y][chx]=ch[x][chx^1];
    fa[ch[x][chx^1]]=y;
    ch[x][chx^1]=y;
    fa[y]=x;
    pushup(y),pushup(x);
}
void splay(int x){
    updata(x);
    for(int f=fa[x];f=fa[x],!isroot(x);rorate(x)){
        if(!isroot(f))rorate(getch(x)==getch(f)?f:x);
    }
}

//access 在原树中把从根到x的所有点放在一条实链里，使根到x成为一条实路径，并且在同一棵
Splay 里。并且 下面没有实边。
void access(int x){
    int y=0;
    while(x){
        splay(x);
        ch[x][1]=y;
        pushup(x);
        y=x;
        x=fa[x];
    }
}
// makeroot(x)在原树中，把x当成根
void makeroot(int x){
    access(x);// 此操作后 x的splay块的 根 变成了 原树的根
    splay(x);
    swap(ch[x][0],ch[x][1]);
    tag[x]^=1;
}
int find(int x){
    access(x);
    splay(x);
    while (ch[x][0])x=ch[x][0];
    splay(x);
    return x;
}
void link(int x,int y){
    if(find(x)!=find(y))makeroot(x),fa[x]=y;
}
void split(int x,int y){
    makeroot(x);
    access(y);
    splay(y);
    //在辅助树上，y的子树就包含了 路径上的信息
}
void cut(int x,int y){
```

```
    split(x,y);
    if(ch[y][0]==x&&!ch[x][1])ch[y][0]=fa[x]=0;
}
void solve(){
    int n,m;
    cin>>n>>m;
    string op;
    int x,y;
    while (m--){
        cin>>op>>x>>y;
        if(op[0]=='Q'){
            if(find(x)==find(y))cout<<"Yes\n";
            else cout<<"No\n";
        }
        else if(op[0]=='C'){
            link(x,y);
        }
        else cut(x,y);
    }

}
signed main(){
    std::ios::sync_with_stdio(false);
    solve();
}
```

dsu

```
Dsu
 #include<iostream>
    #include<algorithm>
    #include<vector>
    #include<map>
    using namespace std;
    typedef long long ll;
    /*
    vector<int>sub[500005];
    vector<int>e[500005];
    int id[500005];
    int tot=1;
    void dfs(int rt,int fa){
        int mx=-1,k=0;
        id[rt]=++tot;
        for(auto j:e[rt]){
            if(rt==fa)continue;
            dfs(j,rt);
            if(sub[j].size()>mx){
                mx=sub[j].size();
                k=j;
            }
        }
        if(mx!=-1)id[rt]=id[k];
        for(auto j:e[rt]){
            if(j==k)continue;
            if(j==fa)continue;
            for(auto x:sub[id[j]]){
                sub[id[rt]].push_back(x);
            }
        }
        sub[id[rt]].push_back(rt);
    }
    */
    int c[100000+5];
    struct node
    {
        int mx_cnt=0;
        long long mx_sum=0;
        map<int,int>cnt;
        vector<int>list;
        void add(int u){
            cnt[c[u]]++;
            if(cnt[c[u]]>mx_cnt)mx_cnt=cnt[c[u]],mx_sum=c[u];
            else if(cnt[c[u]]==mx_cnt)mx_sum+=c[u];
            list.push_back(u);
        }
        int size(){return list.size();}
    }sub[100000+5];
    const int N=1e5+5;
    vector<int>e[100005];
    int id[100005];
    long long ans[N];
    int tot=0;
    void dfs(int rt,int fa){
        int mx=0,k=-1;
        id[rt]=++tot;
        for(auto j:e[rt]){
            if(j==fa)continue;
            dfs(j,rt);
            if(sub[id[j]].size()>mx){
                mx=sub[id[j]].size();
                k=j;
            }
        }
        if(k!=-1)id[rt]=id[k];
```

```
        for(auto j:e[rt]){
            if(j==k)continue;
            if(j==fa)continue;
            for(auto x:sub[id[j]].list){
                sub[id[rt]].add(x);
            }
        }
        sub[id[rt]].add(rt);
        ans[rt]=sub[id[rt]].mx_sum;
    }
    void solve(){
        int n;
        cin>>n;
        for(int i=1;i<=n;++i)cin>>c[i];
        for(int i=1;i<=n-1;++i){
            int u,v;
            cin>>u>>v;
            e[u].push_back(v);
            e[v].push_back(u);
        }
        dfs(1,0);
        for(int i=1;i<=n;++i)cout<<ans[i]<<" ";
        cout<<"\n";
    }
    int main(){
        std::ios::sync_with_stdio(false);
        solve();
    }
```

动态开点线段树

动态开点线段树

```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
#include<set>
#define ls(p) tree[p].l
#define rs(p) tree[p].r
#define sum(p) tree[p].sum
#define mark(p) tree[p].mark
#include<cstdio>
#include<cstring>
using namespace std;
const int maxn=2e5+5;
int ans[maxn];
int tot=1;
int one=1;
struct node
{
    int sum=0,l=0,r=0,mark=0;
}tree[maxn<<6];
struct line
{
    int l,r,id;
};
vector<line>p[maxn];
void pushdown(int p,int len){
    if(len<=1)return;
    if(!ls(p))ls(p)=++tot;
    if(!rs(p))rs(p)=++tot;
    sum(ls(p)) += mark(p) * (len / 2);
    mark(ls(p)) += mark(p);
    sum(rs(p)) += mark(p) * (len - len / 2);
    mark(rs(p)) += mark(p);
    mark(p) = 0;
}
long long query(int p,int l,int r,int ql,int qr){
    if(ql<=l&&qr>=r)return sum(p);
    pushdown(p,r-l+1);
    long long mid=(l+r-1)/2;
    long long res=0;
    if(mid>=ql)res+=query(ls(p),l,mid,ql,qr);
    if(mid<qr)res+=query(rs(p),mid+1,r,ql,qr);
    return res;
}
void updata(int p,int l,int r,int ql,int qr,int k){
    if(ql<=l&&qr>=r){
        sum(p)+=k*(r-l+1),mark(p)+=k;
        return;
    }
    int mid=(l+r-1)/2;
    pushdown(p,r-l+1);
    if(mid>=ql)updata(ls(p),l,mid,ql,qr,k);
    if(mid<qr) updata(rs(p),mid+1,r,ql,qr,k);
    sum(p)=sum(ls(p))+sum(rs(p));
}
void solve(){
    tot=1;
    int n;
    cin>>n;
    for(int i=1;i<=n;++i)p[i].clear();
    multiset<int>L,R;
    for(int i=1;i<=n;++i){
```

```
        int l,r,c;
        cin>>l>>r>>c;
        p[c].push_back({l,r,i});
        updata(1,1,1e9,l,r,one);
        L.insert(l);
        R.insert(r);
    }
    for(int i=1;i<=n;++i){
        for(auto j:p[i]){
            updata(1,1,1e9,j.l,j.r,-1*one);
            L.erase(L.find(j.l));
            R.erase(R.find(j.r));
        }
        for(auto j:p[i]){
            if(query(1,1,1e9,j.l,j.r))ans[j.id]=0;
            else{
                int res=1e9;
                auto it=R.lower_bound(j.l);
                if(it!=R.begin()){
                    it--;
                    res=j.l-*it;
                }
                 it=L.upper_bound(j.r);
                if(it!=L.end()){
                    res=min(res,*it-j.r);
                }
                ans[j.id]=res;
            }
        }
        for(auto j:p[i]){
            updata(1,1,1e9,j.l,j.r,one);
            L.insert(j.l);
            R.insert(j.r);
        }
    }
    for(int i=1;i<=n;++i)cout<<ans[i]<<" ";
    cout<<"\n";
    memset(tree,0,sizeof(tree[0])*(tot+2));
}
signed main(){
    std::ios::sync_with_stdio(false);
    int t;
    cin>>t;
    while(t--)
    solve();
}
```

主席树

动态开点线段树

```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
#include<set>
#define ls(p) tree[p].l
#define rs(p) tree[p].r
#define sum(p) tree[p].sum
#define mark(p) tree[p].mark
#include<cstdio>
#include<cstring>
using namespace std;
const int maxn=2e5+5;
int ans[maxn];
int tot=1;
int one=1;
struct node
{
    int sum=0,l=0,r=0,mark=0;
}tree[maxn<<6];
struct line
{
    int l,r,id;
};
vector<line>p[maxn];
void pushdown(int p,int len){
    if(len<=1)return;
    if(!ls(p))ls(p)=++tot;
    if(!rs(p))rs(p)=++tot;
    sum(ls(p)) += mark(p) * (len / 2);
    mark(ls(p)) += mark(p);
    sum(rs(p)) += mark(p) * (len - len / 2);
    mark(rs(p)) += mark(p);
    mark(p) = 0;
}
long long query(int p,int l,int r,int ql,int qr){
    if(ql<=l&&qr>=r)return sum(p);
    pushdown(p,r-l+1);
    long long mid=(l+r-1)/2;
    long long res=0;
    if(mid>=ql)res+=query(ls(p),l,mid,ql,qr);
    if(mid<qr)res+=query(rs(p),mid+1,r,ql,qr);
    return res;
}
```

```
void updata(int p,int l,int r,int ql,int qr,int k){
    if(ql<=l&&qr>=r){
        sum(p)+=k*(r-l+1),mark(p)+=k;
        return;
    }
    int mid=(l+r-1)/2;
    pushdown(p,r-l+1);
    if(mid>=ql)updata(ls(p),l,mid,ql,qr,k);
    if(mid<qr) updata(rs(p),mid+1,r,ql,qr,k);
    sum(p)=sum(ls(p))+sum(rs(p));
}
void solve(){
    tot=1;
    int n;
    cin>>n;
    for(int i=1;i<=n;++i)p[i].clear();
    multiset<int>L,R;
    for(int i=1;i<=n;++i){
        int l,r,c;
        cin>>l>>r>>c;
        p[c].push_back({l,r,i});
        updata(1,1,1e9,l,r,one);
        L.insert(l);
        R.insert(r);
    }
    for(int i=1;i<=n;++i){
        for(auto j:p[i]){
            updata(1,1,1e9,j.l,j.r,-1*one);
            L.erase(L.find(j.l));
            R.erase(R.find(j.r));
        }
        for(auto j:p[i]){
            if(query(1,1,1e9,j.l,j.r))ans[j.id]=0;
            else{
                int res=1e9;
                auto it=R.lower_bound(j.l);
                if(it!=R.begin()){
                    it--;
                    res=j.l-*it;
                }
                 it=L.upper_bound(j.r);
                if(it!=L.end()){
                    res=min(res,*it-j.r);
                }
                ans[j.id]=res;
            }
        }
        for(auto j:p[i]){
            updata(1,1,1e9,j.l,j.r,one);
            L.insert(j.l);
            R.insert(j.r);
        }
    }
    for(int i=1;i<=n;++i)cout<<ans[i]<<" ";
    cout<<"\n";
    memset(tree,0,sizeof(tree[0])*(tot+2));
}
signed main(){
    std::ios::sync_with_stdio(false);
    int t;
    cin>>t;
    while(t--)
    solve();
}
```

dijkstra

```
#include<iostream>
#include<algorithm>
#include<queue>
#define int long long
using namespace std;
struct node
{
    int from,to,w;
}e[500050];
struct po
{
    int id,w;
    bool operator<(const po &o)const{
        return w>o.w;
    }
};
int nex[500050];
int cnt=0;
void add(int u,int v,int w){
    ++cnt;
    e[cnt].from=nex[u];
    e[cnt].to=v;
    nex[u]=cnt;
    e[cnt].w=w;
}
```

```
priority_queue<po>qu;
int dis[500005],book[500005];
int n,m,s;
void solve(int s){
    book[s]=1;
    for(int i=nex[s];i;i=e[i].from){
        dis[e[i].to]=min(dis[e[i].to],e[i].w);
    }
    for(int i=1;i<=n;++i){
        if(i==s||dis[i]==1234567890)continue;
        po tmp;
        tmp.w=dis[i];
        tmp.id=i;
        qu.push(tmp);
    }
    while(!qu.empty()){
        po tmp=qu.top();
        qu.pop();
        if(book[tmp.id])continue;
        book[tmp.id]=1;
        for(int i=nex[tmp.id];i;i=e[i].from){
            if(dis[e[i].to]>dis[tmp.id]+e[i].w){
                dis[e[i].to]=dis[tmp.id]+e[i].w;
                 po t;
                 t.id=e[i].to;
                 t.w=dis[e[i].to];
                 qu.push(t);
            }
        }
    }
    for(int i=1;i<=n;++i){
        if(dis[i]!=1234567890)
        cout<<dis[i]<<" ";
        else cout<<((1ll*1<<31)-1)<<" ";
    }
    cout<<"\n";
}
signed main(){
    std::ios::sync_with_stdio(false);cin>>n>>m>>s;
    for(int i=1;i<=n;++i)dis[i]=1234567890;
    dis[s]=0;
    for(int i=1;i<=m;++i){
        int u,v,w;
        cin>>u>>v>>w;
        add(u,v,w);
    }
    solve(s);
}
```