目录

最小生成树 Kruskal 2
树分治入门 3
部分树上 dp 5
Dfs 序 7
树链剖分 11

最小生成树 Kruskal

```
struct node{
	int u,v,len;
	bool operator<(const node &A)const{
		if (len!=A.len) return len<A.len;
		if (u!=A.u) return u<A.u;
		return v<A.v;
	}
}Edge[maxn];
priority_queue<node> Q;
int fa[maxn];
inline void getfather(int x){
	if (x==fa[x]) return x;
	return fa[x]=getfather(fa[x]);
}
int n,m;
int main()
{
	scanf("%d%d",&n,&m);
	REP(i,m) scanf("%d%d%d",Edge[i].u,Edge[i].v,Edge[i].len);
	sort(Edge,Edge+m);
	while(Q.size()){
		edge=Edge[]();
		if (getfather(edge.u)==getfather(edge.v)) continue;
		fa[getfather[u]]=v;
		edge[u].push_back(v);
	}
}
```

树分治入门

```
int n,k;
vector<pair<int,int> > edge[maxn];
int size[maxn],root,minw;
bool mark[maxn];
void dfs1(int u,int from,int n){//root
	int i,v,weight=0;
	size[u]=1;
	REP(i,edge[u].size()){
		v=edge[u][i].first;
		if (v==from||mark[v]) continue;
		dfs1(v,u,n);
		size[u]+=size[v];
		weight=max(weight,size[v]);
	}
	weight=max(weight,n-size[u]);
	if (weight<minw) {root=u;minw=weight;}
}
vector<int> leng;
void dfs2(int u,int from,int depth){//len
	int i,v;
	size[u]=1;
	leng.push_back(depth);
	REP(i,edge[u].size()){
		v=edge[u][i].first;
		if (v==from||mark[v]) continue;
		dfs2(v,u,depth+edge[u][i].second);
		size[u]+=size[v];
	}
}
int calc(int root,int len){
	leng.clear();
	dfs2(root,0,len);
	sort(leng.begin(),leng.end());
	int l,r,ret=0;
	for (l=0,r=leng.size()-1;l<r;){
		if (leng[l]+leng[r]<=k) ret+=r-l,l++;
		else r--;
	}
	return ret;
}
int ans;
void dfs3(int u){
```

```
    int i,v,l,r;
    ans+=calc(u,0);
    mark[u]=1;
    REP(i,edge[u].size()){
        v=edge[u][i].first;
        if (mark[v]) continue;
        ans-=calc(v,edge[u][i].second);
        minw=size[v];//注意
        dfs1(v,0,size[v]);
        dfs3(root);
    }
}
int i,u,v,len;
int main(){
    while (~scanf("%d%d",&n,&k)&&(n||k)){
        FOR(i,1,n) edge[i].clear(),mark[i]=0;
        REP(i,n-1){
            scanf("%d%d%d",&u,&v,&len);
            edge[u].push_back(make_pair(v,len));
            edge[v].push_back(make_pair(u,len));
        }
        ans=0;
        minw=n;
        dfs1(1,0,n);
        size[root]=n;
        dfs3(root);
        printf("%d\n",ans);
    }
}
```

部分树上 dp

到叶结点最大距离

```
void dfs1(int u,int from){
    int v,w,i;
    REP(i,edge[u].size()){
        v=edge[u][i].first;
        if (v==from) continue;
        w=edge[u][i].second;
        dfs1(v,u);
        if (l1[u]<l1[v]+w) l2[u]=l1[u],l1[u]=l1[v]+w,son[u]=v;
        else if (l2[u]<l1[v]+w) l2[u]=l1[v]+w;
    }
}
void dfs2(int u,int from,LL d){//从叶子开始
    int v,w,i;
    len[u]=max(d,l1[u]);
    REP(i,edge[u].size()){
        v=edge[u][i].first;
        if (v==from) continue;
        w=edge[u][i].second;
        if (son[u]==v) dfs2(v,u,max(d,l2[u])+w);
        else dfs2(v,u,max(d,l1[u])+w);
    }
}
```

另一种方法

```
void dfs1(int u,int x,int length){//需要好多次(findmaxlen)
    int i;
    if (length>len[u]) len[u]=length;
    if (length>mxlen) mx=u,mxlen=length;
    REP(i,edge[u].size())
        if (edge[u][i]!=x) dfs1(edge[u][i],u,length+1);
}
void dfs2(int x,int father){
    int i;
    root[x]=father;
    value[father].push_back(len[x]);
    num[father]++;
    REP(i,edge[x].size())
        if (!root[edge[x][i]]) dfs2(edge[x][i],father);
}
```

从求含某条边的最小生成树截下来的代码(当然前面 sort 了)合并(要记得 merge 咋写)

```
inline int Union(int u,int v,int len){
    int ret=0;
```

```
    while (u!=v&&(fa[u]!=u||fa[v]!=v)){
        if (fa[u]==u||fa[v]!=v&&sz[u]>sz[v]) {ret=max(ret,val[v]);v=fa[v];}
        else {ret=max(ret,val[u]);u=fa[u];}
    }
    if (u==v) return ret;
    if (sz[u]>sz[v]) swap(u,v);
    fa[u]=v;val[u]=len;
    sz[v]+=sz[u];ans=ans+len;
    return len;
}
```

树上距离除 k 向上取整

```
LL count[maxn][6];
vector<int> edge[maxn];
LL num[maxn],cnt[maxn];//端点,满足条件的次数
int k;
LL ans;
void dfs(int u,int from){
    int i,j,c1,c2;
    count[u][0]=1;
    cnt[u]=1;
    REP(i,edge[u].size()){
        int v=edge[u][i];
        if (from==v) continue;
        dfs(v,u);
        REP(c1,k)
            REP(c2,k){
                ans+=count[u][c1]*count[v][c2];
                if (c1+c2+1>k) ans+=count[u][c1]*count[v][c2];
            }
        ans+=cnt[u]*num[v]+num[u]*cnt[v];
        num[u]+=num[v]+count[v][k-1];
        cnt[u]+=cnt[v];
        REP(c1,k) count[u][c1]+=count[v][(c1-1+k)%k];
    }
}
```

Dfs 序

时间戳

```
struct Segtree{
    struct node{
        int left,right;
    }tree[maxn*4];
    int mx[maxn*4],lazy[maxn*4];
    void pushdown(int x){
        if (lazy[x]){
            mx[x<<1]=lazy[x<<1]=lazy[x];
            mx[x<<1|1]=lazy[x<<1|1]=lazy[x];
            lazy[x]=0;
        }
    }
    void pushup(int x){
        mx[x]=max(mx[x<<1],mx[x<<1|1]);
    }
    void build(int x,int l,int r){
        tree[x].left=l;tree[x].right=r;lazy[x]=0;
        if (l==r) return;
        int mid=(l+r)/2;
        build(x<<1,l,mid);
        build(x<<1|1,mid+1,r);
    }
    void update(int x,int l,int r,int val){
        int L=tree[x].left,R=tree[x].right;
        if (l<=L&&R<=r){
            lazy[x]=mx[x]=val;
            return;
        }
        pushdown(x);
        int mid=(L+R)/2;
        if (mid>=l) update(x<<1,l,r,val);
        if (r>mid) update(x<<1|1,l,r,val);
        pushup(x);
    }
    int query(int x,int l,int r){
        int L=tree[x].left,R=tree[x].right;
        if (l<=L&&R<=r) return mx[x];
        pushdown(x);
        int mid=(L+R)/2,t=0;
        if (mid>=l) t=max(t,query(x<<1,l,r));
        if (r>mid) t=max(t,query(x<<1|1,l,r));
```

```
//          pushup(x);
            return t;
        }
}T1,T2;
int n,q;
int i,j,k;
int u,v;
vector<int> edge[maxn];
int in[maxn],out[maxn];
int tot;
void dfs(int u,int from){
    int v,i;
    in[u]=++tot;
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (v==from) continue;
        dfs(v,u);
    }
    out[u]=tot;
}
int main(){
    scanf("%d",&n);
    REP(i,n-1){
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    dfs(1,0);
    T1.build(1,1,tot);
    T2.build(1,1,tot);
    scanf("%d",&q);
    FOR(i,1,q){
        scanf("%d%d",&j,&k);
        if (j==1){
            T1.update(1,in[k],out[k],i);//时间戳
        }
        if (j==2){
            T2.update(1,in[k],in[k],i);
        }
        if (j==3){
            printf("%d\n",T1.query(1,in[k],in[k])>T2.query(1,in[k],out[k]));
        }
    }
}
```

复杂线段树

```
struct node{
	int left,right;
}tree[maxn*4];
int a[maxn],lazy[maxn*4],mark[maxn];//lazy 保存的是为 1 的 pushdown
void pushdown(int x){
	if (lazy[x]){
		if (tree[x].left==tree[x].right){
			a[tree[x].left]+=mark[tree[x].left]*lazy[x];
			lazy[x]=0;
		}
		else {
			lazy[x<<1]+=lazy[x];
			lazy[x<<1|1]+=lazy[x];
			lazy[x]=0;
		}
	}
}
void build(int x,int l,int r){
	tree[x].left=l;tree[x].right=r;lazy[x]=0;
	if (l==r) return;
	int mid=(l+r)/2;
	build(x<<1,l,mid);
	build(x<<1|1,mid+1,r);
}
void update(int x,int l,int r,LL val){
	int L=tree[x].left,R=tree[x].right;
	if (l<=L&&R<=r) {
		lazy[x]+=val;
		return;
	}
	int mid=(L+R)/2;
	if (mid>=l) update(x<<1,l,r,val);
	if (r>mid) update(x<<1|1,l,r,val);
}
int query(int x,int pos){
	int L=tree[x].left,R=tree[x].right;
	pushdown(x);
	if (L==R) return a[L];
	int mid=(L+R)/2;
	if (mid>=pos) return query(x<<1,pos);
	else return query(x<<1|1,pos);
}
int n,m;
```

```
int i,j,k,val;
int u,v;
vector<int> edge[maxn];
int in[maxn],out[maxn],tot;
void dfs(int u,int from,int color){
	in[u]=++tot;
	if (color) mark[tot]=1;//这里已经映射
	else mark[tot]=-1;
	int i,v;
	REP(i,edge[u].size()){
		v=edge[u][i];
		if (v==from) continue;
		dfs(v,u,color^1);
	}
	out[u]=tot;
}
int ori[maxn];
int main(){
	scanf("%d%d",&n,&m);
	FOR(i,1,n) scanf("%d",&ori[i]);
	REP(i,n-1) {
		scanf("%d%d",&u,&v);
		edge[u].push_back(v);
		edge[v].push_back(u);
	}
	dfs(1,0,1);
	FOR(i,1,n) a[in[i]]=ori[i];//映射
	build(1,1,tot);
	REP(i,m){
		scanf("%d%d",&j,&k);
		if (j==1){
			scanf("%d",&val);
			update(1,in[k],out[k],val*mark[in[k]]);
		}
		else printf("%d\n",query(1,in[k]));
	}
```

树链剖分

水题(按边的)

```
int tot;
inline int lowbit(int x){return x&-x;}
int c[maxn];
int getsum(int x){
	int ret=0;
	while (x){
		ret+=c[x];
		x-=lowbit(x);
	}
	return ret;
}
int query(int l,int r){
	return getsum(r)-getsum(l-1);
}
void add(int x,int d){
	while (x<=tot){
		c[x]+=d;
		x+=lowbit(x);
	}
}
void build(){
	int i;
	FOR(i,1,tot) c[i]=0;
}
int n,i,q,k;
int u,v;
int U[maxn],V[maxn];
vector<int> edge[maxn];
int fa[maxn],son[maxn],sz[maxn],top[maxn],id[maxn],dep[maxn];
void dfs1(int u,int from,int depth){
	int v,i,mx=-1;
	sz[u]=1;fa[u]=from;dep[u]=depth;son[u]=0;
	REP(i,edge[u].size()){
		v=edge[u][i];
		if (v==from) continue;
		dfs1(v,u,depth+1);
		sz[u]+=sz[v];
		if (sz[v]>mx) son[u]=v;
	}
}
void dfs2(int u,int x){
```

```
    int v,i;
    top[u]=x;id[u]=++tot;
    if (son[u]) dfs2(son[u],x);
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (v==fa[u]||v==son[u]) continue;
        dfs2(v,v);
    }
}
inline int Query(int x,int y){
    int ret=0;
    while (top[x]!=top[y]){
        if (dep[top[x]]<dep[top[y]]) swap(x,y);
        if (query(id[top[x]],id[x])) return -1;
        ret+=id[x]-id[top[x]]+1;
        x=fa[top[x]];
    }
    if (dep[x]>dep[y]) swap(x,y);
    if (son[x]){//不按边就直接加
        if (query(id[son[x]],id[y])) return -1;
        ret+=id[y]-id[son[x]]+1;
    }
    return ret;
}
int main(){
    scanf("%d",&n);
    FOR(i,1,n-1){
        scanf("%d%d",&U[i],&V[i]);
        edge[U[i]].push_back(V[i]);
        edge[V[i]].push_back(U[i]);
    }
    tot=0;
    dfs1(1,0,1);
    dfs2(1,1);
    FOR(i,1,n-1) if (dep[U[i]]>dep[V[i]]) swap(U[i],V[i]);
    build();
    scanf("%d",&q);
    while (q--){
        scanf("%d",&k);
        if (k==1){
            scanf("%d",&i);
            add(id[V[i]],-1);
        }
        if (k==2){
```

```
            scanf("%d",&i);
            add(id[V[i]],1);
        }
        if (k==3){
            scanf("%d%d",&u,&v);
            printf("%d\n",Query(u,v));
        }
    }
}
```

难题(区间合并)

```
int tot;
struct node{
    int lval,rval,ldown,lup,rdown,rup,upmx,downmx;
    node():upmx(0),downmx(0){};
}tree[maxn<<2];
int a[maxn];
node merge(node L,node R){
    if (L.upmx==0) return R;
    if (R.upmx==0) return L;
    node ret;
    ret.upmx=max(L.upmx,R.upmx);
    ret.downmx=max(L.downmx,R.downmx);
    ret.lval=L.lval;
    ret.lup=L.lup;
    ret.ldown=L.ldown;
    ret.rval=R.rval;
    ret.rup=R.rup;
    ret.rdown=R.rdown;
    if (L.rval<R.lval){
        ret.upmx=max(ret.upmx,L.rup+R.lup);
        if (L.downmx==1) ret.lup=L.lup+R.lup;
        if (R.downmx==1) ret.rup=L.rup+R.rup;
    }
    if (L.rval>R.lval){
        ret.downmx=max(ret.downmx,L.rdown+R.ldown);
        if (L.upmx==1) ret.ldown=L.ldown+R.ldown;
        if (R.upmx==1) ret.rdown=L.rdown+R.rdown;
    }
    return ret;
}
void build(int x,int l,int r){
    if (l==r){
        tree[x].lval=tree[x].rval=a[l];
```

```
tree[x].lup=tree[x].ldown=tree[x].rup=tree[x].rdown=tree[x].upmx=tree[x].downmx=1;
            return;
        }
        int mid=(l+r)/2;
        build(x<<1,l,mid);
        build(x<<1|1,mid+1,r);
        tree[x]=merge(tree[x<<1],tree[x<<1|1]);
}
node query(int x,int l,int r,int L,int R){
        node ret;
        if (l<=L&&R<=r) return tree[x];
        int mid=(L+R)/2;
        if (mid>=l&&r>mid) return merge(query(x<<1,l,r,L,mid),query(x<<1|1,l,r,mid+1,R));
        if (mid>=l) return query(x<<1,l,r,L,mid);
        return query(x<<1|1,l,r,mid+1,R);
}
int n,i,j,q;
int u,v;
vector<int> edge[maxn];
int fa[maxn],son[maxn],top[maxn],dep[maxn],id[maxn],sz[maxn];
int b[maxn];
void dfs1(int u,int depth){
        int v,i,mx=-1;
        son[u]=0;sz[u]=1;dep[u]=depth;
        REP(i,edge[u].size()){
            v=edge[u][i];
            dfs1(v,depth+1);
            sz[u]+=sz[v];
            if (sz[v]>mx) mx=sz[v],son[u]=v;
        }
}
void dfs2(int u,int x){
        int v,i;
        top[u]=x;id[u]=++tot;
        if (son[u]) dfs2(son[u],x);
        REP(i,edge[u].size()){
            v=edge[u][i];
            if (v==fa[u]||v==son[u]) continue;
            dfs2(v,v);
        }
}
int Query(int x,int y){//这里需要注意方向
        node up,down;
        int ret,mark1=0,mark2=0;
```

```
        while (top[x]!=top[y]){
            if (dep[top[x]]>dep[top[y]]){
                up=merge(query(1,id[top[x]],id[x],1,tot),up);
                x=fa[top[x]];
                mark1=1;
            }else {
                down=merge(query(1,id[top[y]],id[y],1,tot),down);
                y=fa[top[y]];
                mark2=1;
            }
        }
        if (dep[x]>dep[y]) up=merge(query(1,id[y],id[x],1,tot),up),mark1=1;
        else down=merge(query(1,id[x],id[y],1,tot),down),mark2=1;
        ret=max(up.downmx,down.upmx);
        if (mark1&&mark2&&up.lval<down.lval) ret=max(ret,up.ldown+down.lup);
        return ret;
}
int T,t;
int main(){
        scanf("%d",&T);
        FOR (t,1,T){
            scanf("%d",&n);
            FOR(i,1,n) edge[i].clear();tot=0;
            FOR(i,1,n) scanf("%d",&b[i]);
            FOR(i,2,n){scanf("%d",&fa[i]); edge[fa[i]].push_back(i);}
            dfs1(1,1);
            dfs2(1,1);
            FOR(i,1,n) a[id[i]]=b[i];
            build(1,1,tot);
            scanf("%d",&q);
            printf("Case #%d:\n",t);
            while (q--){
                scanf("%d%d",&u,&v);
                printf("%d\n",Query(u,v));
            }
            if (t!=T) puts("");
        }
}
```