树分治

树分治是再树形结构上进行分治的一种操作，主要思想是除去树中的某些对象，使原树被分解成若干个互不相交的部分，由此可以解决许多以路径为询问对象的问题。

一颗树上的路径有经过根节点的或者不经过根节点的两种。

若我们把根节点删掉，则可以生成若干棵以原根节点的儿子为根节点的子树。

问题引入

给一颗带边权树，统计最短距离不超过k的顶点的对数。

解法

假设我们按重心把树分成若干子树，那么符合条件的顶点对有三种情况：

（1）顶点v,w属于同一子树的顶点对(v,w)

（2）顶点v,w属于不同子树的顶点对(v,w)

（3）顶点s与其他顶点v组成的顶点对(s,v)

对于(1)可以递归求解，其实就是在两点所在的子树上再进行划分的子问题。

对于(2)，路径上必经过s。记d_u表示结点u到s的路径长度，那么我们要统计的就是满足$d_u + d_v \leq k$且u, v来自不同子树的对数。（总对数 − 同一子树对数）

对于(3)，我们将s看做一个到s的路径长度为0的结点，那么就转化为了情况(2)

给定N个结点的带权树，其中结点分为两类，黑点和白点。要求找到一条路径，使得经过的黑点数不超过K个，且路径长度最大。

点分治

大概流程

1.随意选取一个点作为根

2.统计包含根的答案

3.将根拿掉，递归计算子树的答案

复杂度：递归层数*T(n)

希望递归层数尽量少，要先找到树的

定理：存在一个点使得分出的子树的结点个数均不大于N/2

在点分治时，每次把重心作为根，递归深度最坏是O(logn)的，总时间复杂度是O(nlogn)的。在树是一条链是达到上界。

边分治

1.重构树型：插入虚点使得新的树的度数不超过2，且总点数不超过2N

2.“分”——找到一条边使得可以均衡地将树分成两部分

3.“治”——统计包含这条边的答案

4.递归处理两棵子树

定理：如果一棵树中每个点的度均不大于D，那么存在一条边使得分出的两棵子树的结点个数在[N/(D+1),N*(D+1)]

步骤

①处理经过当前根节点的路径

②删掉根节点

③对生成的每棵子树的根节点重复①②

对于完全二叉树来说，时间复杂度是O(logn)的，但对于一条链O(n)

重心

对于一条链来说，取中点为根节点是更好的取法。

我们以O(n)的复杂度求出树的重心，就可以以O(logn)的复杂度分治点。

处理经过重心的路径

对于一条经过重心的路径来说,dis[u,v]=dis[u,rt]+dis[rt,v]

所以我们可以先处理出每个节点与重心结点的距离，并且记录一下。

建立一个judge数组，对于每一个dis,把judge[dis]置为true

对于每个询问k，遍历所有dis,若judge[k-dis]=true，则说明长度为k的路径存在。

树链剖分与树分治的联系

树分治的两种方式：树分治每次删除一个点或一条边，而树链剖分每次删除了一条链。所以树链剖分算法可以看作是基于链的分治，而且这种分治还有一个特点，每次被删除结点的儿子必将作为下一次删除的链的头节点。

例题

P3806 【模板】点分治1

给一颗带边权树，问树上距离为k的点对是否存在。

```
#include<bits/stdc++.h>
const int maxn=1e4+5;

struct E{
    int to,w,next;
}edge[maxn<<1];

int tot,head[maxn];

inline void read(int &data){
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){
        if(ch=='-') f=f*-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9'){
        x=x*10+ch-'0';
        ch=getchar();
    }
    data=x*f;
}

inline void addedge(int u,int v,int w){ //链式前向星建边
    edge[++tot].next=head[u];
    edge[tot].to=v;
    edge[tot].w=w;
```

```
    head[u]=tot;
}

int n,m,rt,sum,cnt;
int tmp[maxn],sz[maxn],dis[maxn],maxp[maxn],q[105];
bool judge[101011001],ans[105],vis[maxn];

void getrt(int u,int f){ //找重心
    sz[u]=1,maxp[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==f||vis[v]) continue; //被删掉了的不要算
        getrt(v,u); //下一层
        sz[u]+=sz[v]; //当前结点大小
        if(sz[v]>maxp[u]) maxp[u]=sz[v]; //找最大子树
    }
    if(maxp[u]<sum-sz[u]) maxp[u]=sum-sz[u];//maxp[u]=max(maxp[u],sum-sz[u]); //
两边取最大的子树大小
    if(maxp[u]<maxp[rt]) rt=u; //替换重心
}

void getdis(int u,int f){ //计算u的距离并且放到tmp中
    tmp[cnt++]=dis[u];  //
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==f||vis[v]) continue;
        dis[v]=dis[u]+edge[i].w;
        getdis(v,u);
    }
}

void solve(int u){ //处理u的子树
    std::queue<int>que;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]) continue;
        cnt=0; //计数处理器
        dis[v]=edge[i].w;
        getdis(v,u); //把距离都处理出来
        for(int j=0;j<cnt;j++){ //遍历所有的距离
            for(int k=0;k<m;k++){ //遍历所有的询问
                if(q[k]>=tmp[j]) //如果询问大于单条路径长度，那么就有可能存在
                    ans[k]|=judge[q[k]-tmp[j]]; //如果能用两条路径拼出来，那就存在。
            }
        }
        for(int j=0;j<cnt;j++){ //把存在的单条路径长度标true
            que.push(tmp[j]);
            judge[tmp[j]]=true;
        }
    }
    while(!que.empty()){ //清空judge数组，不要用memset
        judge[que.front()]=false;
        que.pop();
    }
}

void divide(int u){ //分治
    vis[u]=judge[0]=1; //删除根节点
```

```
    solve(u);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]) continue;
        maxp[rt=0]=sum=sz[v]; //把重心设为0
        getrt(v,0);
        getrt(rt,0); //与主函数相同，第二次更新sz大小
        divide(rt);
    }
}

signed main(){
    read(n);read(m);
    for(int i=1;i<n;i++){
        int u,v,w;
        read(u);read(v);read(w);
        addedge(u,v,w);
        addedge(v,u,w);
    }
    for(int i=0;i<m;i++) read(q[i]); //离线处理询问
    maxp[0]=sum=n;
    getrt(1,0); //找重心
    getrt(rt,0); //更新sz/fa等信息
    divide(rt); //开始分治
    for(int i=0;i<m;i++){
        if(ans[i]) puts("AYE");
        else puts("NAY");
    }
    return 0;
}
```

luoguP4178 Tree

给定一棵带边权树，求树上两点距离小于等于k的点对数量。

```
#include<bits/stdc++.h>
#define inf (1<<29)
#define lb(x) (x&-x)
using namespace std;
const int maxn=2e5+5;
int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

int n,m,ans;

struct E{
    int to,w,nxt;
}e[maxn<<1];
int tot,head[maxn];
inline void addedge(int u,int v,int w){ //链式前向星建边
    e[++tot].nxt=head[u];
    e[tot].to=v;
    e[tot].w=w;
    head[u]=tot;
```

```
}

int tr[maxn];
void update(int x,int val){
    for(;x<=m;x+=lb(x)) tr[x]+=val;
}
int query(int x){
    int res=0;
    for(;x;x-=lb(x)) res+=tr[x];
    return res;
}

int sz[maxn],vis[maxn],mn=inf,rt;
void getrt(int x,int f,int sum=n){ //寻找重心
    sz[x]=1;int mx=0;
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].to;
        if(vis[v]||v==f) continue;
        getrt(v,x);
        sz[x]+=sz[v];
        mx=max(mx,sz[v]);
    }
    mx=max(mx,sum-sz[x]);
    if(mx<mn) mn=mx,rt=x;
}

int s[maxn],sav[maxn];
void dfs(int x,int f,int dis){
    if(dis>m) return;
    s[++s[0]]=dis;sav[++sav[0]]=dis;  //第0位表示桶大小
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].to;
        if(vis[v]||v==f) continue;
        dfs(v,x,dis+e[i].w);
    }
}

void divide(int x){ //分治
    sav[0]=0;mn=n;
    getrt(x,0);
    getrt(rt,0,sz[x]);
    int u=rt;vis[u]=1;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].to;
        if(vis[v]) continue;
        s[0]=0;
        dfs(v,u,e[i].w); //查找子树中路径小于m的点，并记录路径
        for(int j=s[0];j>=1;j--){
            if(s[j]>m) continue;
            ans+=query(m-s[j]);
        }
        for(int j=s[0];j>=1;j--){  //树状数组维护
            if(s[j]>m) continue;
            update(s[j],1);
            ans++;
        }
    }
    for(int i=sav[0];i>=1;i--){ //清空树状数组
```

```
        if(sav[i]>m) continue;
        update(sav[i],-1);
    }
    for(int i=head[u];i;i=e[i].nxt){ //分治子树
        int v=e[i].to;
        if(vis[v]) continue;
        divide(v);
    }
}

void input(){  //输入
    n=read();
    for(int i=1,u,v,w;i<n;i++){
        u=read();v=read();w=read();
        addedge(u,v,w);
        addedge(v,u,w);
    }
    m=read();
}

signed main(){
    input();
    divide(1);
    cout<<ans<<"\n";
    return 0;
}
```

BZOJ2599 [IOI2011]Race

给一颗带边权树，问权值和等于k的路径数。$N \leq 2e5, K \leq 1e5$

点分治+排序优化

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
typedef long long ll;
const int N=2e5+7;

struct E{
    int v,w,nxt;
}e[N<<1];
int head[N],cnt=0;
inline void addedge(int u,int v,int w){
    e[++cnt]=(E){v,w,head[u]};head[u]=cnt;
}
struct Nod{
    int len,dep,bel;
    friend bool operator <(Nod A,Nod B){
        return A.len==B.len?A.dep>B.dep:A.len<B.len;
    }
}a[N];

int n,k,rt,sz[N],fa[N],sumsz,ans=inf,dep[N],dis[N],tot=0,tp,f[N];
bool vis[N];
```

```cpp
inline void dfs1(int x){
    sz[x]=1;
    a[++tot]={dis[x],dep[x],tp};
    for(int i=head[x];i;i=e[i].nxt){
        int to=e[i].v,w=e[i].w;
        if(to==fa[x]||vis[to]) continue;
        fa[to]=x;dep[to]=dep[x]+1;dis[to]=dis[x]+w;
        dfs1(to);
        sz[x]+=sz[to];
    }
}

inline void getroot(int x){
    f[x]=0;
    for(int i=head[x];i;i=e[i].nxt){
        int to=e[i].v,w=e[i].w;
        if(to==fa[x]||vis[to]) continue;
        getroot(to);
        f[x]=max(f[x],sz[to]);
    }
    f[x]=max(f[x],sumsz-sz[x]);
    if(f[x]<f[rt]) rt=x; //转移重心
}

inline void calc(int x){
    a[1].len=a[1].dep=a[1].bel=0;tot=1;
    sz[x]=1;dep[x]=dis[x]=0;
    for(int i=head[x];i;i=e[i].nxt){
        int to=e[i].v,w=e[i].w;
        if(vis[to]) continue;
        fa[to]=x;dep[to]=1;dis[to]=w;
        tp=to;dfs1(to);sz[x]+=sz[to];
    }
    stable_sort(a+1,a+1+tot);
    int l=1,r=tot;
    while(l<r){
        if(a[l].len+a[r].len>k) --r;
        else{
            int now=r;
            while(now>l&&a[l].bel==a[now].bel&&a[l].len+a[now].len==k) --now;
            if(now>l&&a[l].len+a[now].len==k) ans=min(ans,a[l].dep+a[now].dep);
            ++l;
        }
    }
}

inline void solve(int x){
    vis[x]=1;fa[x]=0;calc(x);
    for(int i=head[x];i;i=e[i].nxt){
        int to=e[i].v;
        if(vis[to]) continue;
        rt=0;sumsz=sz[to];
        getroot(to);solve(rt); //处理以to为根的子树
    }
}

signed main(){
    scanf("%d%d",&n,&k);  f[0]=inf;
```

```
    if(!k){puts("0");return 0;}
    for(int i=1,u,v,w;i<n;++i){
        scanf("%d%d%d",&u,&v,&w);
        ++u;++v;
        addedge(u,v,w);addedge(v,u,w);
    }
    dfs1(1);rt=0;
    sumsz=n;
    getroot(1);
    solve(rt);
    if(ans==inf) puts("-1");
    else printf("%d\n",ans);
    return 0;
}
```

参考资料

https://www.bilibili.com/video/BV1RT4y1L7Sb

https://www.bilibili.com/video/BV1PE41197md

《树分治》——黄哲威