



信息学
点分治



树上点对问题



西南大学附属中学
High School Affiliated to Southwest University

- 类比线性空间上的序列问题。
- 例如对 2 3 4 5 6 的指定范围求和。
- 树上的序列问题，其实就是点对问题。



| 西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University

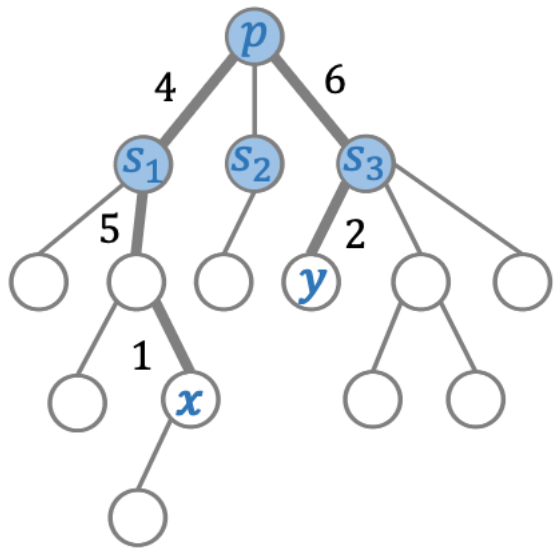


例 Tree POJ1741



西南大学附属中学
High School Affiliated to Southwest University

- 给定一颗有 N 个点的无根树，每条边都有一个权值。树上两个节点 x, y 之间的路径长度就是路径上各条边的权值之和。求长度不超过 K 的路径有多少条。



考虑根节点 p

对于 p 来说，树上**路径分为两类**

Case1 经过根节点 p

Case2 包含于 p 的某一颗子树中（不经过根节点）

可以把 case2 转化为 case1 问题

对于 case1 来说，可以看作 $p \rightarrow x$ 和 $p \rightarrow y$



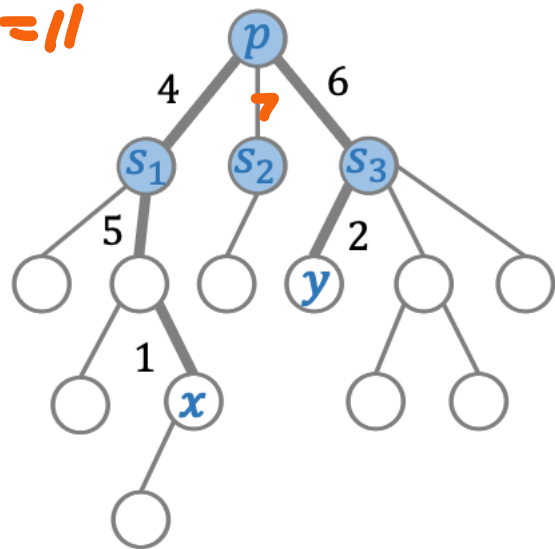
例 Tree POJ1741



西南大学附属中学
High School Affiliated to Southwest University

- 给定一颗有 N 个点的无根树，每条边都有一个权值。树上两个节点 x, y 之间的路径长度就是路径上各条边的权值之和。求长度不超过 K 的路径有多少条。

$k=11$



可以通过DFS预处理出 $d[i], b[i]$

- $d[i]$ 表示 i 到根节点的距离
- $b[i]$ 表示 i 属于根节点的哪一个子树 (定义 $b[p]=p$)

如果是case1 的情况，即经过 p 点。

- 则需要统计 $d[x]+d[y] \leq k$ ，且 $b[x] \neq b[y]$ 所有点对。
- 例如 $k=11$ 情况下，满足条件的点对只有 $(s1, s3)$



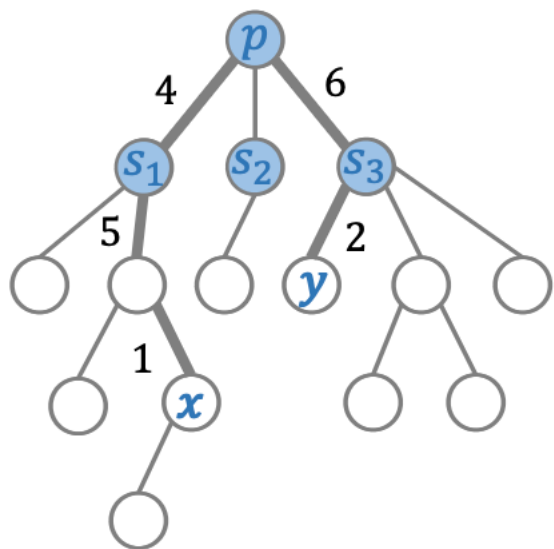
只需求出case1

经过 p 点的满足条件的点对数量

如何求

附 | 中 | 信 | 息 | 学 | 竞 | 赛 |

High School Affiliated to Southwest University



【已经有】求经过点 p 的路径长度 $\leq k$ 的点对数量

【已经有】从 p 开始dfs，得到深度 d 与子树关系 b

(例如 $b[x] == s1, b[y] == s3$)

- 将深度 d 进行排序，放入 a 数组中
- 两个变量 L 和 R 从数组两端向中间扫

L 从左向右扫的时候，如果要满足 $\leq k$ 的条件， R 恰好是从右向左扫

在 L 和 R 区间移动的时候，利用 $cnt[s]$ 维护子树关系（都是子树 s 的结点的结点数）

- 这样从 $a[L]$ 出发的，满足条件的结点有：

$R - L - cnt[b[a[L]]]$



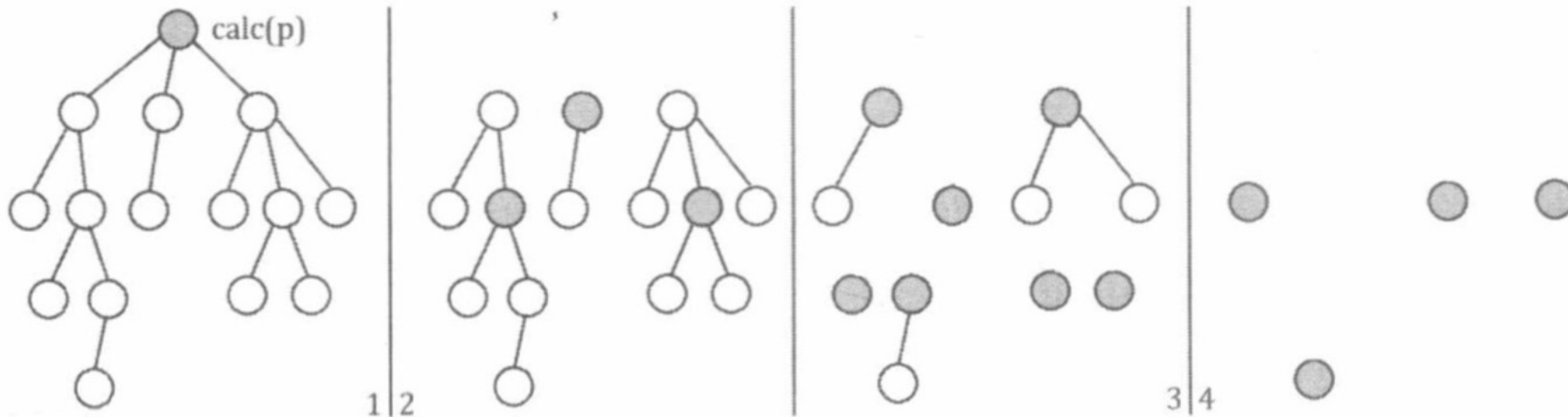
排序后用双指针法
求出了经过点p的路径长度 $\leq k$ 的点对数量

| 西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University

接下来分治求解其余子树



只需要删除p点，分别对子树递归求解即可。





总而言之，整个点分治算法的过程就是：

1. 任选一个根节点 p （后面我们将说明， p 应该取树的重心）。
2. 从 p 出发进行一次 DFS，求出 d 数组和 b 数组。
3. 执行 $Calc(p)$ 。
4. 删除根节点 p ，对 p 的每棵子树（看作无根树）递归执行 1~4 步。

why?

为什么选择重心？

如果选择最边缘的结点。
那么拆出来的树很可能只有一个子树。
那么递归需要 $O(N)$ 复杂度
再配合求解过 p 点点对数的复杂度 $O(N \log N)$
实际复杂度达到了 $O(N^2 \log N)$

所以我们尽可能希望，拆出的子树多。

选择重心可以满足这个希望，并且可以知道，子树的最大规模至多是原来的一半，那么递归复杂度 $O(N) \rightarrow O(\log N)$
整体时间复杂度 = $O(N \log^2 N)$



流程回顾，点分治时间复杂度？



西南大学附属中学
High School Affiliated to Southwest University

过程：

1. 选一个重心p
2. 从p出发DFS，求出d 和b
3. 在2基础上LR双指针扫经过p的满足条件的路径数
4. 删p，对其子树重复1-3过程

每次在处理的问题规模在不断减少。

这里整体复杂度为 $O(N * \log N * \log N)$



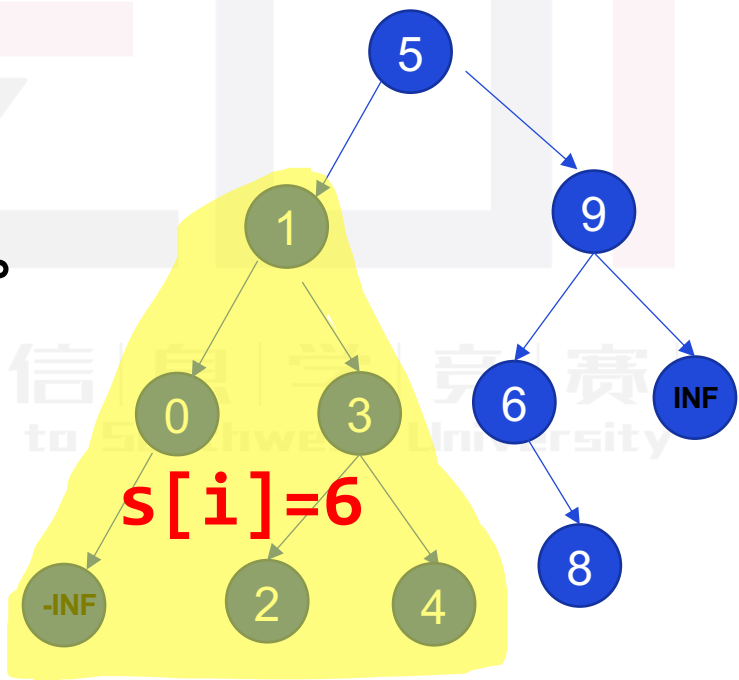
复习：找重心？



西南大学附属中学
High School Affiliated to Southwest University

什么是重心：删除树中一点，使其余子树中子树节点数最大值最小，这个点就是重心。

方法：树上DP求重心
找任意一点为根，DFS搜下去，
 $s[i]$ 记录当 i 为根节点时，子树上的节点数。





复习：找重心？



西南大学附属中学
High School Affiliated to Southwest University


什么是重心：删除树中一点，使其余子树中子树节点数最大值最小，这个点就是重心。

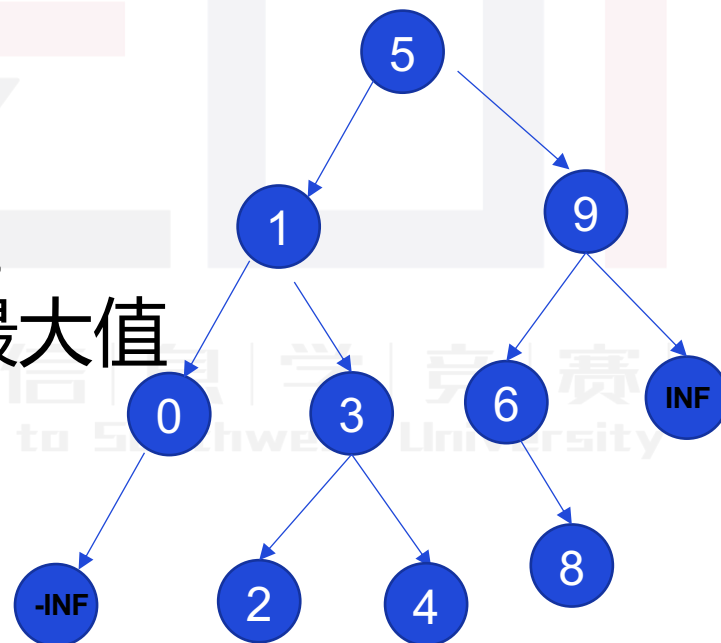
方法：树上DP求重心

找任意一点为根，DFS搜下去，

$s[i]$ 记录当 i 为根节点时，子树上的节点数。

$dp[i]$ 记录以 i 为根节点时，其子树节点数最大值


$$dp[i] = \max(n - s[i], \max_{j \in i's \text{ son}} (s[j]))$$



因为在dfs的时候我们不知道 $s[j]$ 的值，所以，回溯的时候处理。



复习：找重心？

Dfs $O(N)$ 完成后
再对dp数组
 $O(N)$ 扫一遍，找min即可
总共时间复杂度 $O(N)$



西南大学附属中学

```
11 int n;  
12 int dp[maxn+10],s[maxn+10];  
13 vector<int> v[maxn+10];  
14  
15 void dfs(int u,int pre)  
16 {  
17     int k=v[u].size();  
18  
19     s[u]=1;  
20     for(int i=0;i<k;++i){  
21         int j=v[u][i];  
22         if(j==pre)  
23             continue;  
24         dfs(j,u);  
25         dp[u]=max(dp[u],s[j]);  
26         s[u]+=s[j];  
27     }  
28     dp[u]=max(dp[u],n-s[u]);  
29 }
```



例题参考代码

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=4e4+10;
int dp[maxn],s[maxn],head[maxn];
int rt,cnt;
struct node{
    int to,nxt,w;
    int b,d;
}e[maxn*4],a[maxn];
void add(int u,int v,int w){
    e[++cnt].to=v;
    e[cnt].nxt=head[u];
    head[u]=cnt;
    e[cnt].w=w;
}
int K,vis[maxn],dis[maxn],tot,ans,s1;
//求重心
void dfs(int x,int fa,int sum/*树的大小*/){
    s[x]=1; dp[x]=0;//最大子树的大小
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].to;
        if(v!=fa&&!vis[v]){
            dfs(v,x,sum);
            s[x]+=s[v];
            dp[x]=max(dp[x],s[v]);
        }
    }
    dp[x]=max(dp[x],sum-s[x]);
    if(dp[x]<dp[rt])rt=x;
}
```

```
int point[maxn],dep[maxn];
void getdis(int x,int fa){
    a[++tot].d=dis[x];
    point[x]=tot;
    if(dep[x]>1)a[tot].b=a[point[fa]].b;
    else a[tot].b=x;
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].to;
        if(v!=fa&&!vis[v]){
            dis[v]=dis[x]+e[i].w;
            dep[v]=dep[x]+1;
            getdis(v,x);
        }
    }
}
int c[maxn];
int cmp(node x,node y){return x.d<y.d;}
int getans(int x){
    dis[x]=tot=s1=dep[x]=0;
    getdis(x,0);
    sort(a+1,a+1+tot,cmp);
    int l=1,r=tot,s=0;
    memset(c,0,sizeof(c));
    for(int i=2;i<=tot;i++)c[a[i].b]++;
    while(l<r){
        if(a[l].d+a[r].d<=K){
            s+=r-l-c[a[l].b];
            l++;
            c[a[l].b]--;
        }else{
            c[a[r].b]--;
            r--;
        }
    }
    return s;
}
```



```
void Solve(int x){
    vis[x]=1;
    a[x].b=x;
    ans+=getans(x);
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].to;
        if(!vis[v]){
            rt=0;
            dfs(v,x,s[v]);
            Solve(rt);
        }
    }
}
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<n;i++){
        int x,y,w;
        scanf("%d%d%d",&x,&y,&w);
        add(x,y,w);
        add(y,x,w);
    }
    dp[0]=maxn;
    scanf("%d",&K);
    dfs(1,0,n);
    Solve(rt);
    printf("%d",ans);
    return 0;
}
```

```

#include <bits/stdc++.h>
#define lowbit(x) x & (-x)
using namespace std;
int n , k , siz[44000] , maxp[44000] ;
int a , b , c , t[22000] , rt , cl[44000];
int fst[44000] , nex[88000] , v[88000];
int val[88000] , vis[44000] , tot , ans;
inline void add( int a , int b , int c ){
    nex[++tot] = fst[a]; fst[a] = tot;
    v[tot] = b; val[tot] = c;
    return ;
}
void upd( int x , int a ){
    while(x <= k + 1){t[x] += a;x += lowbit(x);}
}
int ask( int x ){
    int ans = 0;
    while(x){ans += t[x];x -= lowbit(x);}
    return ans;
}
void getrt( int u , int fa , int num ){
    siz[u] = 1; maxp[u] = 0;
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(v[i] == fa || vis[v[i]]) continue;
        getrt(v[i] , u , num);
        siz[u] += siz[v[i]];
        maxp[u] = max(maxp[u] , siz[v[i]]);
    }
    maxp[u] = max(maxp[u] , num - maxp[u]);
    if(maxp[u] < maxp[rt]) rt = u;
    return ;
}
void dfs( int u , int fa , int dis ){
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(fa == v[i] || vis[v[i]]) continue;
        dfs(v[i] , u , dis + val[i]);
    }
    cl[++tot] = dis;
    return ;
}

```

```

inline void search_son( int u ){
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(vis[v[i]]) continue;

        int las = tot;
        dfs(v[i] , u , val[i]);

        for(int j = las + 1 ; j <= tot ; j++ )
            if(k - cl[j] >= 0)
                ans += ask(k - cl[j] + 1);

        for(int j = las + 1 ; j <= tot ; j++ )
            upd(cl[j] + 1 , 1);
    }
    for(int j = 1 ; j <= tot ; j++ )
        upd(cl[j] + 1 , -1);
}
void calc( int u , int fa ){
    siz[u] = 1;
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(v[i] == fa || vis[v[i]]) continue;
        calc(v[i] , u);
        siz[u] += siz[v[i]];
    }
    return ;
}
void solve( int u ){
    vis[u] = 1;
    tot = 0;
    search_son(u);
    calc(u , 0);
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(vis[v[i]]) continue;
        rt = 0;
        getrt(v[i] , u , siz[v[i]]);
        solve(rt);
    }
}

```

```

int main(){
    scanf("%d" , &n);
    for(int i = 1 ; i < n ; i++ ){
        scanf("%d%d%d" , &a , &b , &c);
        add(a , b , c); add(b , a , c);
    }
    scanf("%d" , &k);
    upd(1 , 1);
    rt = 0; maxp[0] = 1e9;
    getrt(1 , 1 , n);
    solve(rt);
    printf("%d" , ans);
    return 0;
}

```

息 | 学 | 竞 | 赛 |
outhwest University

书上提到的解法1
使用数据结构进行答案统计