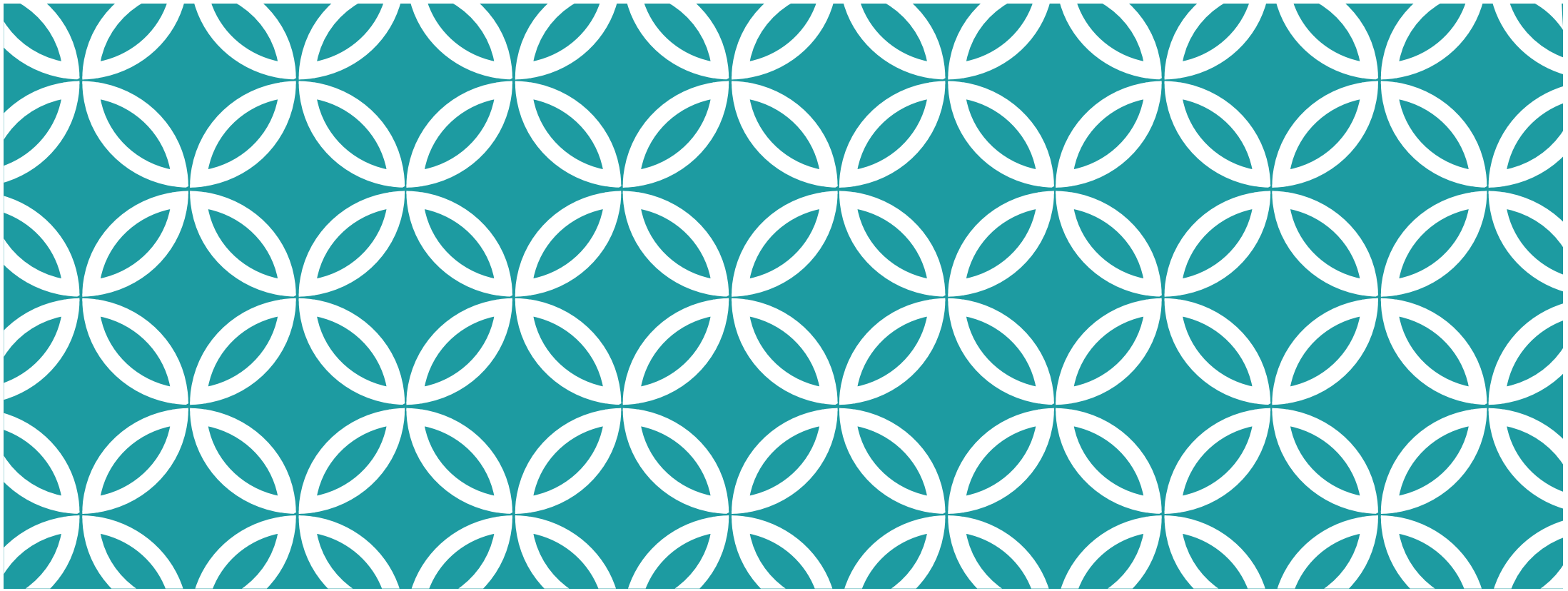
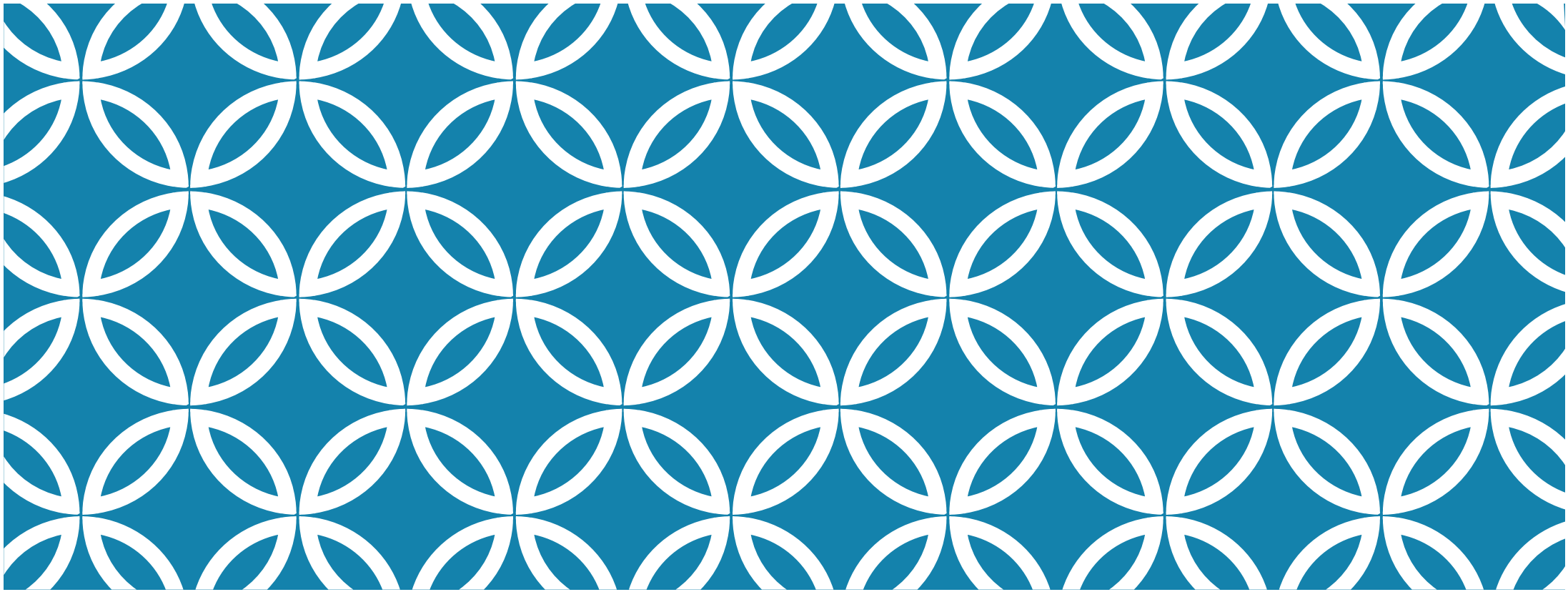


点分治与点分树

CZC

2023-5-20



分治上树——点分治

一个问题

统计完全二叉树上距离为d的点对数量。

具体来说：

统计对象：树上路径(不带修)

仅统计树上满足某条件的路径数量

树上点对问题

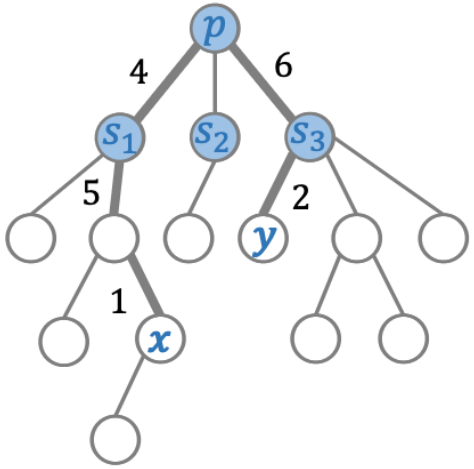
类比线性空间上的序列问题。

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。

例 TREE POJ1741

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

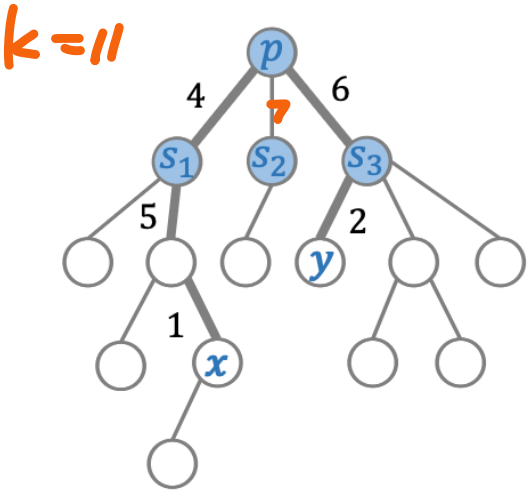
Case2 包含于p的某一颗子树中（不经过根节点）

可以把case2 转化为 case1 问题

对于case1来说，可以看作 p->x 和 p->y

例 TREE POJ1741

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。



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]<=k ，且b[x]!=b[y]所有点对.
- 例如k=11情况下，满足条件的点对只有(s1,s3)

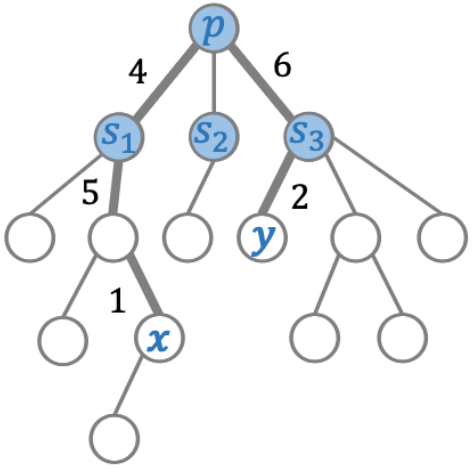
问题转化

只需求出case1

经过p点的满足条件的点对数量

如何求

双指针法： 求解CASE1



【已经有】求经过点p的路径长度<=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<=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]] ]
```

可视化

CASE1 解决

排序后用双指针法
求出了经过点p的路径长度<=k的点对数量

接下来分治求解其余子树

方法小结

树上分治

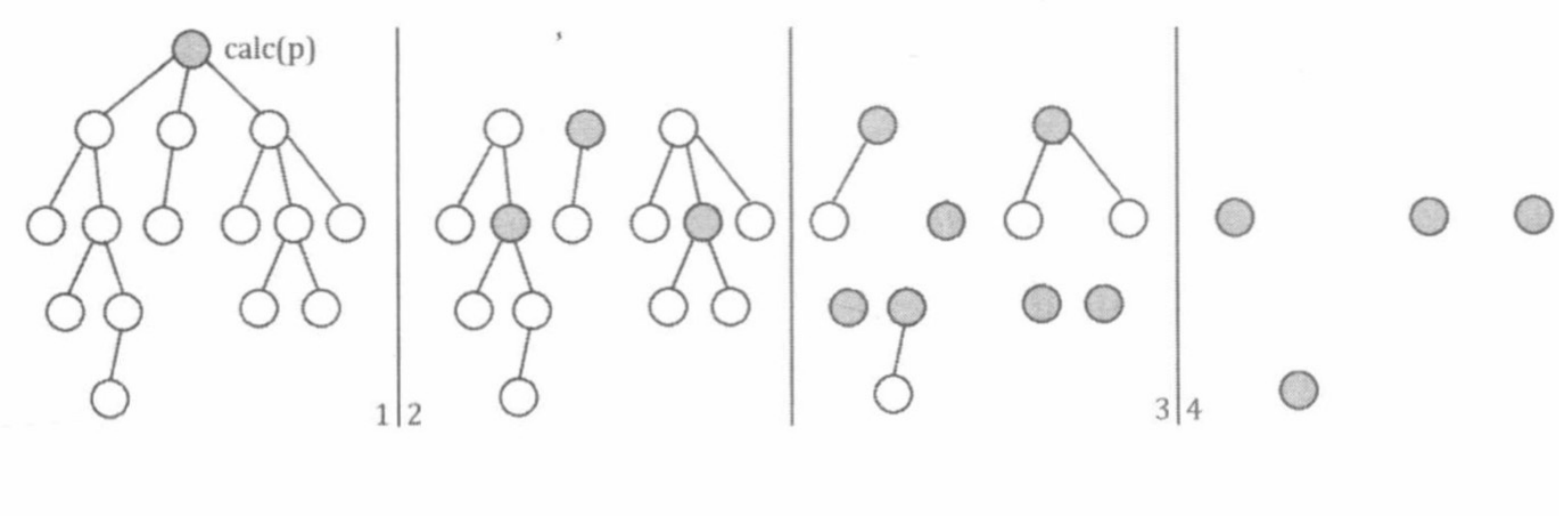
+

用一些特殊的结构/算法维护/求出需要的信息 （难点）

例如：例题的方法叫 点分治+双指针法

点分治

只需要删除p点，分别对子树递归求解即可。



点分治：具体过程

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?

重心?

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(NlogN)
实际复杂度达到了$O(N^2logN)$

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) -> O(logN)
整体时间复杂度 = $O(Nlog^2N)$

流程回顾，点分治时间复杂度?

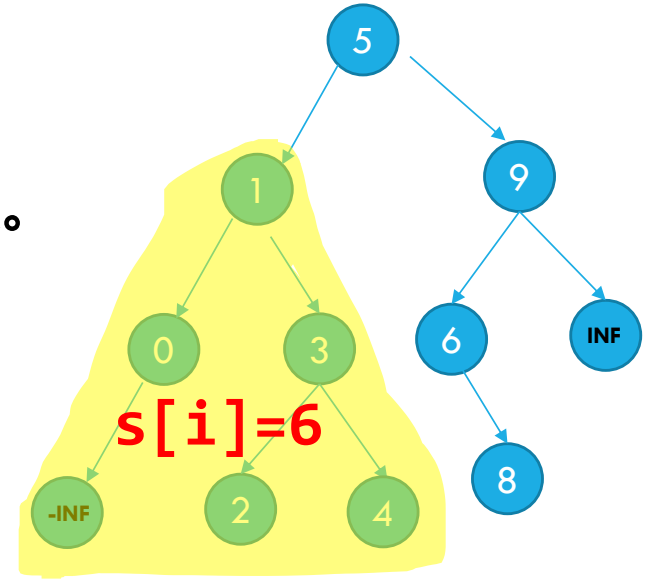
过程：
1. 选一个重心p
2. 从p出发DFS，求出d 和b
3. 在2基础上LR双指针扫经过p的满足条件的路径数
4. 删p，对其子树重复1-3过程

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 * logN * logN)
感觉，其应该正确，具体如何算的还在研究（应该列个公式求和就可以得到这个公式）

复习：找重心?

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。

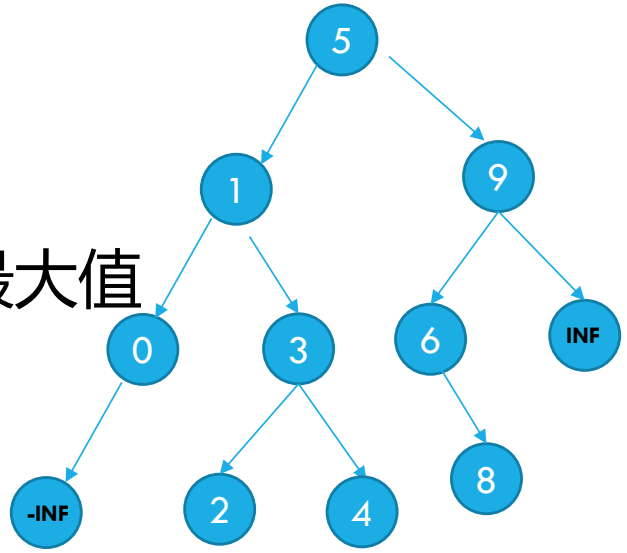


复习：找重心?

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

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



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)

```
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

```cpp
#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;
}
```

```cpp
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;
}
```

书上提到的解法2
使用双指针L与R进行答案统计
PS 感谢ZJY提供的代码实现

```cpp
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;
}
```

例题参考代码

```cpp
#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 ;
}
```

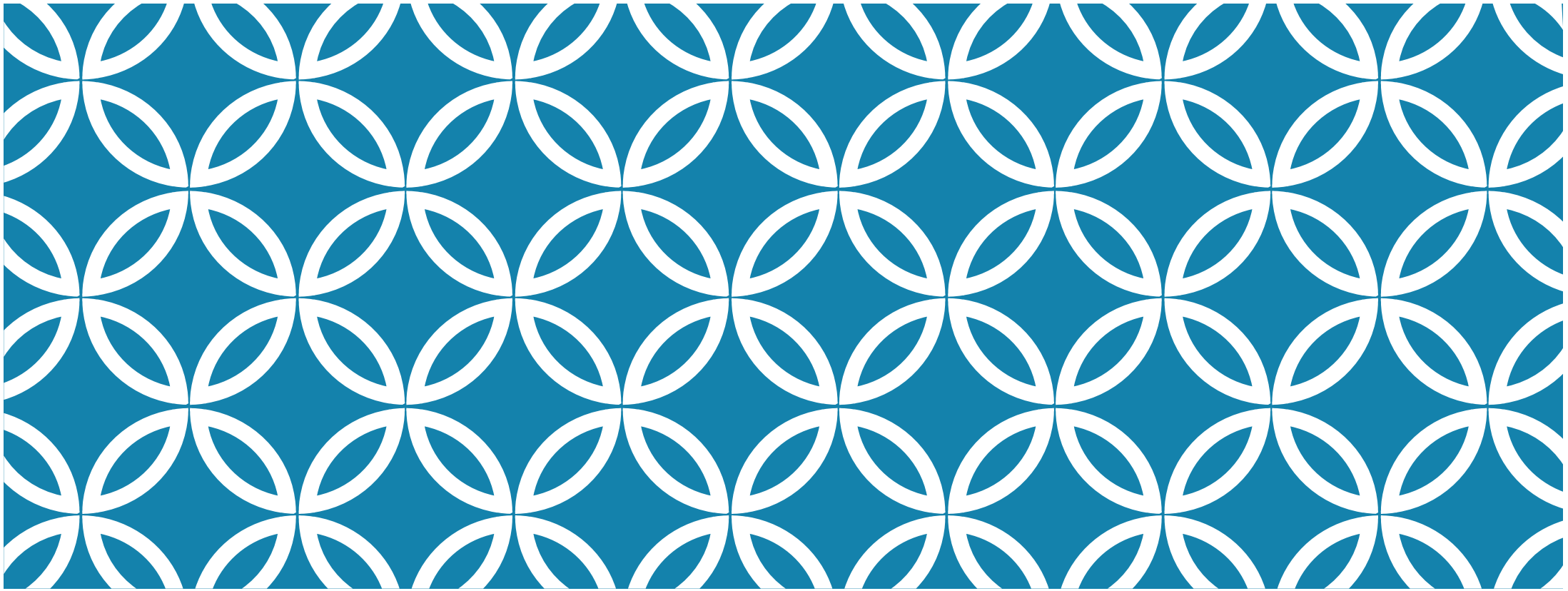
```cpp
inline void search_son( int u ){
    for(int i = fst[u] ; i ; i = nex[i] ){
        if(vis[v[i]]) continue;

        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);
    }
}
```

```cpp
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;
}
```

书上提到的解法1
使用数据结构进行答案统计

参考

蓝书



分治上树——点分树

动态点分治

点分树

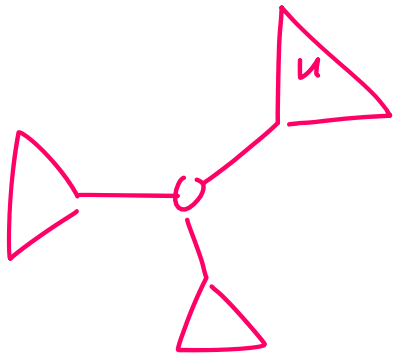
例题：开店

有一颗树，这颗树的每一个节点有一个妖怪，每个妖怪有一个年龄。树边有权

多组询问，给定区间[L,R] 节点u 问年龄在[L,R]内的妖怪离节点u的距离和是多少。

树上点度数最多为3

询问强制在线。节点<=150000, 询问<=200000



点分治做法

假设有路径(u,x),其中ux不在同一子树中，要统计的信息为：

1 u到重心的距离
2 其他子树中满足条件的节点个数
3 其他子树中满足条件的节点到重心的距离和
答案 = 距离*个数 + 距离和

假设有路径(u,x),其中ux在同一子树中，递归做。

但是有多个询问，如果依次处理会超时。

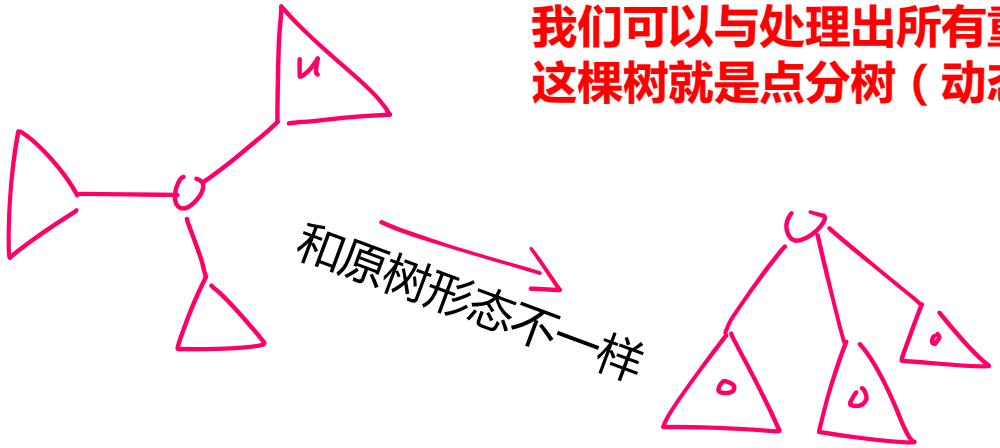
发现，每次询问的时候，不会影响树的形态，则重心不变。

动态点分治——点分树

通过分析题目发现，操作不影响重心，即不影响树的形态
所以可以以重心为结点重构成一颗树

点分树

发现，每次修改的时候，不会影响树的形态，则重心不变。
我们可以与处理出所有重心，并以重心为root构建出一颗树
这棵树就是点分树（动态点分治）

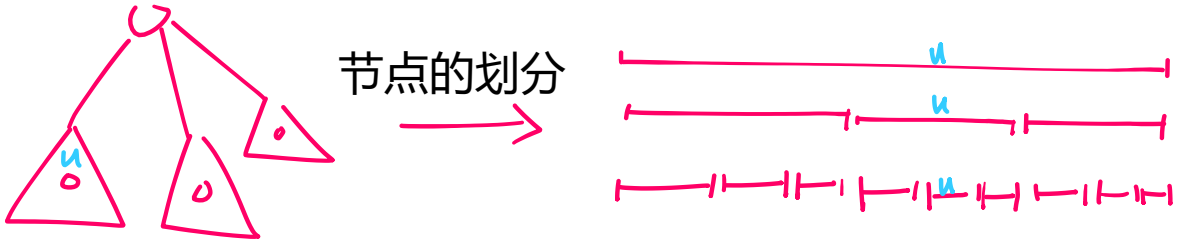


点分树：支持快速求答案。

注意每个子树里重心为root递归建树即可

如何快速点分治？
先看看点分治的过程有没有变化

点分树支持的点分治

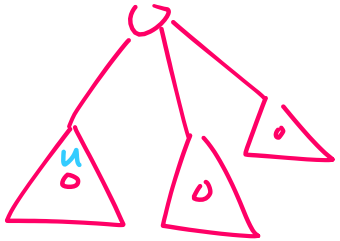


u最多在logn层中被计算。接下来仍然是点分治的部分。

考虑第一层中的路径（u，x）分为2种情况
ux不在同一个子树中：即，x和u不在同一颗树中的满足要求的点的所有距离和。
ux在同一个子树中：在第二层归并去做即可。

继续考虑第二层（u,x）的情况，不在同一子树中（直接求），在同一子树中（下一层归并做）
And so on
直到：u所在子树只有u，或u为一个重心 时停止。
最多递归logn层 点分树的如何支持点分治？（有一些问题还需要进一步考虑）

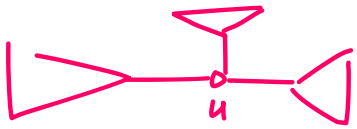
点分树支持的点分治



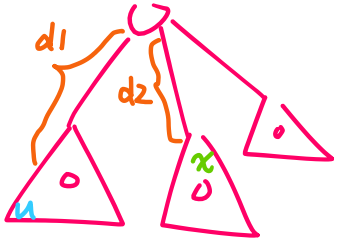
有一些问题还需要进一步考虑：

思想：通过与处理，将树的形态进行预处理，构建出一颗点分树。

考虑问题1：　如何处理其他树节点与u的信息进行计算？
即：如何处理u和他的兄弟结点的关系？



考虑问题2：　如果u是重心，怎么处理u和他儿子结点的关系？



问题1：假设有路径(u,x),其中ux不在同一子树中，要统计的信息为：

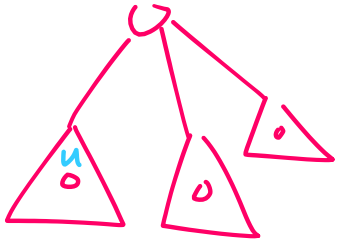
1 u到重心的**距离d1**
2 其他子树中满足条件的节点**个数**
3 其他子树中满足条件的节点到重心的**距离和**
答案 = **距离d1*个数** + **距离和**

即从u出发，计算到达其他子树中满足要求的所有结点的路径长度和

一共有多少条路径呢？（d1要被统计多少次？）

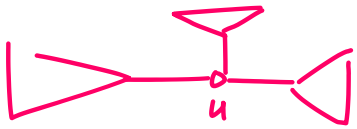
= X所在子树中年龄在[L,R]内的结点个数。
= 直接将子树所有节点**年龄存入重心vector**，**排序后二分**即可。

点分树支持的点分治



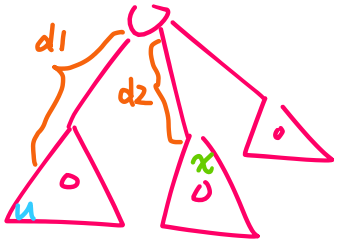
有一些问题还需要进一步考虑：

思想：通过与处理，将树的形态进行预处理，构建出一颗点分树。



考虑问题1：　如何处理其他树节点与u的信息进行计算？
即：如何处理u和他的兄弟结点的关系？

考虑问题2：　如果u是重心，怎么处理u和他儿子结点的关系？



问题1：假设有路径(u,x),其中ux不在同一子树中，要统计的信息为：

1 u到重心的**距离d1**
2 其他子树中满足条件的节点**个数**
3 其他子树中满足条件的节点到重心的**距离和**
答案 = **距离d1*个数 + 距离和**

即从u出发，计算到达其他子树中满足要求的所有结点的路径长度和

x到root的路径和？ （d2的和一共有多少？）

= 在root的存年龄的时候同时存距离

= 满足要求[L,R]区间求和，前缀和预处理。

一些细节维护

对于每个点需要求一下他的重心是谁。（方便快速找到u）

代码实现

https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/开店.cpp
点击下载，有详细注释

复杂度

空间复杂度：

- 每个rt会存入整个树的信息，一层树是n。Logn层树就是nlogn

时间复杂度：

- 整体涉及到排序、递归建树有O(nlogn)复杂度
- 每次询问复杂度为$O(\log^2 n)$ (logn层，logn二分)
- n次询问复杂度$O(n\log^2 n$ ~~+ nlogn~~)
- 整体复杂度为$O(n\log^2 n)$

- 实际上，由于归并操作变化万千，
- 可以假定查询一次的复杂度为T
- 则查Q次的复杂度应为 O(Q*T*logn)
- e.g., 线段树+点分树的时间复杂度为$O(n\log^2 n)$

点分树： 震波

代码（复制粘贴到浏览器中，获得下载）
https://ipic-1254235966.cos.ap-chongqing.myqcloud.com/upic/震波.cpp

前言

由于这篇题解思路并没有什么区别，所以这篇题解的意义在于稍稍更细致地讲下思路和卡常方法。~~估计也只有我常数这么大了~~

思路

第一感

由于题目要查询到一个点距离为k以内的所有点的权值和，一个显然的想法就是对每个点开一个线段树维护权值和，下标维护距离，然后暴力查询。显然这是$MLE+TLE$的。考虑优化这个过程。

爆空间?

首先MLE的解决很简单，动态开点就行了，于是问题在于TLE。

超时间?

考虑之前TLE是因为每次修改点权的时候我们都要对每个点去修改。这表明我们这样子的维护方式是不行的。所以考虑把所有点给答案的贡献分为两个部分。一个常用的方法是分为在子树内和在子树外两个部分。但是由于树高不确定，这导致我们的复杂度可能退化。所以考虑点分树来保证复杂度。

点分树

考虑一组(x,k)，对于x在点分树上的子树的贡献我们可以开一棵动态开点的线段树按之前的方法维护。设这个值为$sm1(x,p)$。然后我们考虑子树外的贡献，也就是我们跳到点分树上的父亲结点时需要计算的贡献。如果我们直接计算$sm1(fa[x],k-dis(x,fa[x]))$很显然我们会把在$x$子树内的点算重复。所以我们还要开一棵线段树记下$x$的子树内到$fa[x]$的权值和，记为$sm2(x,p)$,那么每次向上跳的时候获得的答案就是$sm1(i,k-dis(x,i))-sm2(lst,k-dis(x,i))$其中$i$是当前跳到的祖先，$lst$是上一个跳到的祖先。那么这样的时间复杂度就是$nlog_2^2n$的。

卡常&实现细节

- 动态开点的线段树第一棵在x这个节点的最大深度开到$siz[x]$就行了，第二棵要开到$siz[fa[x]]$。
- 线段树查询的时候查到空节点直接返回0，不要像我这个蒟蒻一样还去开节点啊....
- 最好使用RMQ求LCA。~~然鹅我并没有用~~
- 使用$namespace$让你毫无顾虑的使用重复数组名。
- 把初始的值看做修改可以让你省很多事。

点分树的更多题解

1见：已经毕业现在在川大的yxy学姐：手把手教你点分树

https://blog.csdn.net/RA100FDM/article/details/107460101

2见打印纸质稿

点分治AND 点分树题目

UOJ题号	题目名	short-mark
2343	「模板」点分治「POJ2114」Boatherds	不带修点分治 =k
1114	「模板」「BZOJ1468」Tree	<k
2428	「IOI2011」Race	给一棵树，每条边有权。求一条简单路径，权值和等于 k，且边的数量最小。
2342	聪聪可可	相对模版有点变化
2958	「BZOJ3730」震波	带修
7177	atm的树	二分答案后又是一只震波
7175	[BJOI2017]树的难题	
6792	「HNOI2015」开店	
6802	[2010国家集训队]Crash的旅游计划	有出题人题解pdf，在资源包中（国家集训队作业08）
2957	「ZJOI2015」幻想乡战略游戏	
2959	「BZOJ4372」烁烁的游戏	和模板题震波类似，用一个数据结构进行维护
5833	小清新数据结构题	
5830	[Ynoi2011] 成都七中	
6803	「CF936E」 Iqea	挑战题
7176	模式字符串	挑战题

HINT 关于复杂问题思考过程

1. 先看题目，思考10～20分钟，直至你认为再也想不到更多内容。
2. 看题解（找一篇写的好的）看懂，分析自己思路哪里薄弱了（记录到总结中）
3. 关掉题解，自己再实现一遍代码。
4. 发现代码有问题（那一定是自己理解不到位）继续打开题解重新理解（必要时再换一篇题解）
5. 看懂了，关掉，继续做。
6. 重复操作4～5，直到这个题自己一口气能写对为止（看悟性和运气0.5-10h不等）。
7. 记录这个题的过程（在日常总结中）
8. 如果一个题，超过40小时还没有头绪。记录并放弃。等自己实力up后再来解决。