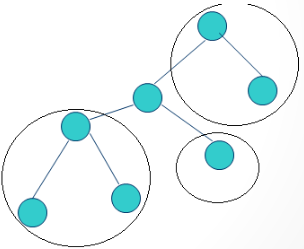
[树分治（点分治模板）poj-1741 Tree](http://blog.csdn.net/u010660276/article/details/44920725)

题意：求树上距离小于等于m的点对有多少个。

对于一棵有根树， 树中满足要求的一个数对所对应的一条路径，必然是以下两种情况之一：  
1、经过根节点  
2、不经过根节点，也就是说在根节点的一棵子树中  
对于情况2，可以递归求解，下面主要来考虑情况1。  
  
设点i的深度为Depth[i]，父亲为Parent[i]。  
若i为根，则Belong[i]=-1，若Parent[i]为根，则Belong[i]=Parent[i]，否则Belong[i]=Belong[Parent[i]]。  
这三个量都可以通过一次BFS求得。  
我们的目标是要统计：有多少对(i,j)满足i<j，Depth[i]+Depth[j]<=K且Belong[i]<>Belong[j]  
  
如果这样考虑问题会变得比较麻烦，我们可以考虑换一种角度：  
设X为满足i<j且Depth[i]+Depth[j]<=K的数对(i,j)的个数  
设Y为满足i<j，Depth[i]+Depth[j]<=K且Belong[i]=Belong[j]数对(i,j)的个数  
那么我们要统计的量便等于X-Y  
  
求X、Y的过程均可以转化为以下问题：  
已知A[1],A[2],...A[m]，求满足i<j且A[i]+A[j]<=K的数对(i,j)的个数  
  
对于这个问题，我们先将A从小到大排序。  
设B[i]表示满足A[i]+A[p]<=K的最大的p(若不存在则为0)。我们的任务便转化为求出A所对应的B数组。那么，若B[i]>i，那么i对答案的贡献为B[i]-i。  
显然，随着i的增大，B[i]的值是不会增大的。利用这个性质，我们可以在线性的时间内求出B数组，从而得到答案。  
  
综上，设递归最大层数为L，因为每一层的时间复杂度均为“瓶颈”——排序的时间复杂度O(NlogN)，所以总的时间复杂度为O(L\*NlogN)  
  
然而，如果遇到极端情况——这棵树是一根链，那么随意分割势必会导致层数达到O(N)级别，对于N=10000的数据是无法承受的。因此，我们在每一棵子树中选择“最优”的点分割。所谓“最优”，是指删除这个点后最大的子树尽量小。这个点可以通过树形DP在O(N)时间内求出，不会增加时间复杂度。这样一来，即使是遇到一根链的情况时，L的值也仅仅是O(logN)的。  
  
简单来说：点分治就是每次找到重心，然后把重心去掉，对分成的每两棵树之间分别统计路径信息（以重心的每个相邻点为根，遍历整棵子树即可得到这个根到每个结点的统计信息），就可以知道包含这个重心的所有路径的信息，然后对于剩下的路径就是在子树里面进行同样的操作了，直到只剩一个点为止（注意这一个点所构成的路径有时也要处理一下）。边分治就是每次找到一条边，使得删掉这条边后分成的两棵子树大小尽可能平均，然后以删掉的边的两端点为根，分别统计根到两棵树中的每个结点的路径信息，最后合并算路径，即可得到包含这条边的所有路径的信息，剩下的路径在两棵树中递归处理。  
  
对于树分治可以参考qzc大神的论文：[《分治算法在树的路径问题中的应用》](http://wenku.baidu.com/view/8861df38376baf1ffc4fada8.html?re=view" \t "_blank)

**Tree**

|  |  |  |
| --- | --- | --- |
| **Time Limit:** 1000MS |  | **Memory Limit:** 30000K |
| **Total Submissions:** 12417 |  | **Accepted:** 3939 |

Description

Give a tree with n vertices,each edge has a length(positive integer less than 1001).  
Define dist(u,v)=The min distance between node u and v.  
Give an integer k,for every pair (u,v) of vertices is called valid if and only if dist(u,v) not exceed k.  
Write a program that will count how many pairs which are valid for a given tree.

Input

The input contains several test cases. The first line of each test case contains two integers n, k. (n<=10000) The following n-1 lines each contains three integers u,v,l, which means there is an edge between node u and v of length l.  
The last test case is followed by two zeros.

Output

For each test case output the answer on a single line.

Sample Input

5 4

1 2 3

1 3 1

1 4 2

3 5 1

0 0

Sample Output

8

Source

[LouTiancheng@POJ](http://poj.org/searchproblem?field=source&key=LouTiancheng%40POJ" \t "_blank)

题意：找出树中有多少点对，满足dis（u，v）<K

思路：点分治模板题

#include<iostream>

#include<cstdio>

#include<string>

#include<cstring>

#include<vector>

#include<cmath>

#include<queue>

#include<stack>

#include<map>

#include<set>

#include<algorithm>

using namespace std;

const int maxn=10010;

int N,K;

int ans,root,Max;

struct node

{

int v,next,w;

}edge[maxn\*2];

int head[maxn],tot;

int size[maxn];//树的大小

int maxv[maxn];//最大孩子节点的size

int vis[maxn];

int dis[maxn];

int num;

void init()

{

tot=0;

ans=0;

memset(head,-1,sizeof(head));

memset(vis,0,sizeof(vis));

}

void add\_edge(int u,int v,int w)

{

edge[tot].v=v;

edge[tot].w=w;

edge[tot].next=head[u];

head[u]=tot++;

}

//处理子树的大小

void dfssize(int u,int f)

{

size[u]=1;

maxv[u]=0;

for(int i=head[u];i!=-1;i=edge[i].next)

{

int v=edge[i].v;

if(v==f||vis[v])continue;

dfssize(v,u);

size[u]+=size[v];

if(size[v]>maxv[u])maxv[u]=size[v];

}

}

//找重心

void dfsroot(int r,int u,int f)

{

if(size[r]-size[u]>maxv[u])//size[r]-size[u]是u上面部分的树的尺寸，跟u的最大孩子比，找到最大孩子的最小差值节点

maxv[u]=size[r]-size[u];

if(maxv[u]<Max)Max=maxv[u],root=u;

for(int i=head[u];i!=-1;i=edge[i].next)

{

int v=edge[i].v;

if(v==f||vis[v])continue;

dfsroot(r,v,u);

}

}

//求每个点离重心的距离

void dfsdis(int u,int d,int f)

{

dis[num++]=d;

for(int i=head[u];i!=-1;i=edge[i].next)

{

int v=edge[i].v;

if(v!=f&&!vis[v])

dfsdis(v,d+edge[i].w,u);

}

}

//计算以u为根的子树中有多少点对的距离小于等于K

int calc(int u,int d)

{

int ret=0;

num=0;

dfsdis(u,d,0);

sort(dis,dis+num);

int i=0,j=num-1;

while(i<j)

{

while(dis[i]+dis[j]>K&&i<j)j--;

ret+=j-i;

i++;

}

return ret;

}

void dfs(int u)

{

Max=N;

dfssize(u,0);//处理子树的大小

dfsroot(u,u,0);//找重心

ans+=calc(root,0);//计算以root为根的子树中有多少点对的距离小于等于K

vis[root]=1;

for(int i=head[root];i!=-1;i=edge[i].next)

{

int v=edge[i].v;

if(!vis[v])

{

ans-=calc(v,edge[i].w);

dfs(v);

}

}

}

int main()

{

while(scanf("%d%d",&N,&K)!=EOF)

{

if(!N&&!K)break;

int u,v,w;

init();

for(int i=1;i<N;i++)

{

scanf("%d%d%d",&u,&v,&w);

add\_edge(u,v,w);

add\_edge(v,u,w);

}

dfs(1);

printf("%d\n",ans);

}

return 0;

}