### 点分治

#include <cstdio>   
#include <algorithm>   
#include <vector>   
#include <cstring>   
using namespace std;   
const int N=1e5+5;   
struct node {   
 int v, l;   
};   
vector<node> g[N];   
int n, k, Size, s[N], f[N], root, d[N], K, ans;   
vector<int> dep;   
bool done[N];   
void getroot(int now, int fa) {   
 int u;   
 s[now] = 1; f[now] = 0;   
 for (int i=0; i<g[now].size(); i++)   
 if ((u = g[now][i].v) != fa && !done[u]) {   
 getroot(u, now);   
 s[now] += s[u];   
 f[now] = max(f[now], s[u]);   
 }   
 f[now] = max(f[now], Size-s[now]);   
 if (f[now] < f[root]) root = now;   
}   
void getdep(int now, int fa) {   
 int u;   
 dep.push\_back(d[now]);   
 s[now] = 1;   
 for (int i=0; i<g[now].size(); i++)   
 if ((u = g[now][i].v) != fa && !done[u]) {   
 d[u] = d[now] + g[now][i].l;   
 getdep(u, now);   
 s[now] += s[u];   
 }   
}   
int calc(int now, int init) {   
 dep.clear(); d[now] = init;   
 getdep(now, 0);   
 sort(dep.begin(), dep.end());   
 int ret = 0;   
 for (int l=0, r=dep.size()-1; l<r; )   
 if (dep[l] + dep[r] <= K) ret += r-l++;   
 else r--;   
 return ret;   
}   
void work(int now) {   
 int u;   
 ans += calc(now, 0);   
 done[now] = true;   
 for (int i=0; i<g[now].size(); i++)   
 if (!done[u = g[now][i].v]) {   
 ans -= calc(u, g[now][i].l);   
 f[0] = Size = s[u];   
 getroot(u, root=0);   
 work(root);   
 }   
}   
signed main() {   
   
 while (scanf("%d%d", &n, &K)) {   
 if (n == 0 && K == 0) break;   
 for (int i=0; i<=n; i++) g[i].clear();   
 memset(done, false, sizeof(done));   
 int u, v, l;   
 for (int i=1; i<n; i++) {   
 scanf("%d%d%d", &u, &v, &l);   
 g[u].push\_back(node(v, l));   
 g[v].push\_back(node(u, l));   
 }   
 f[0] = Size = n;   
 getroot(1, root=0);   
 ans = 0;   
 work(root);   
 printf("%d\n", ans);   
 }   
 return 0;   
}

### simpson 自适应积分

#define LD long double   
   
   
LD simpson(LD l, LD r) {   
LD c = (l + r) / 2;   
return (f(l) + 4 \* f(c) + f(r)) \* (r - l) / 6;   
}   
LD asr(LD l, LD r, LD eps, LD S) {   
 LD m = (l + r) / 2;   
 LD L = simpson(l, m), R = simpson(m, r);   
 if (fabs(L + R - S) < 15 \* eps) return L + R + (L + R - S) / 15;   
 return asr(l, m, eps / 2, L) + asr(m, r, eps / 2, R);   
}   
   
LD asr(LD l, LD r, LD eps) { return asr(l, r, eps, simpson(l, r)); }