

```

1  #include <algorithm>
2  #include <bitset>
3  #include <cstdio>
4  #include <cstring>
5  #include <cstdlib>
6  #include <cmath>
7  #include <deque>
8  #include <iostream>
9  #include <map>
10 #include <queue>
11 #include <set>
12 #include <stack>
13 #include <string>
14 #include <utility>
15 #include <vector>
16 #include <list>
17 #define For(i,a,b) for(int i=(a); i<=(b) ; i++)
18 #define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
19 #define Memset(a,b); memset((a),(b),sizeof((a)));
20 using namespace std;
21 const int INF = 0x3f3f3f3f;
22 typedef long long LL;typedef unsigned long long ULL;typedef long double
LDB;
23 inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0' || ch>'9'){if(ch=='-'
)f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-
'0';ch=getchar();}return x*f;}
24 inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-'
)f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
25 const int N = 1e4+5;
26 int n,k,ans;
27 struct Edge{
28     int to,val;
29 };
30 vector<Edge>edge[N];
31 int subtree_size[N];
32 int book[N];
33 int ds[N],n_ds;
34 int tds[N],n_tds;
35 void init()
36 {
37     int x,y,v;Edge tmp;
38     For(i,1,n-1)
39     {
40         x = Cin();y = Cin();v = Cin();

```

```

41         tmp.to = y;tmp.val = v;
42         edge[x].push_back(tmp);
43         tmp.to = x;
44         edge[y].push_back(tmp);
45     }
46 }
47 void csh()
48 {
49     Memset(book,0);
50     Memset(subtree_size,0);
51     For(i,1,n) edge[i].clear();
52     ans = 0;
53 }
54 int count_subtree_size(int now,int pre)
55 {
56     subtree_size[now] = 1;
57     For(i,0,edge[now].size()-1)
58     {
59         int _to = edge[now][i].to;
60         if(_to == pre || book[_to]) continue;
61         subtree_size[now] += count_subtree_size(_to,now);
62     }
63     return subtree_size[now];
64 }
65 int found_center(int now,int pre,int half)
66 {
67     For(i,0,edge[now].size()-1)
68     {
69         int _to = edge[now][i].to;
70         if(_to == pre || book[_to]) continue;
71         if(subtree_size[_to] > half) return found_center(_to,now,half);
72     }
73     return now;
74 }
75 void count_subtree_path(int now,int pre,int path)
76 {
77     tds[++n_tds] = path;
78     For(i,0,edge[now].size()-1)
79     {
80         int _to = edge[now][i].to;
81         if(_to == pre || book[_to]) continue;
82         count_subtree_path(_to,now,path + edge[now][i].val);
83     }
84 }
85 int count_pairs(int d[],int siz)
86 {

```

```

87     sort(d+1,d+siz+1);
88     int res = 0;
89     For(i,1,siz)
90     {
91         int c = (k - d[i]);
92         int j = upper_bound(d+1,d+siz+1,c) - d-1;
93         if(j > siz) j = siz;
94         if(j > i)
95             res+=(j - i);
96     }
97
98     return res;
99 }
100 void solve_subtree(int now)
101 {
102     count_subtree_size(now,-1);
103     now = found_center(now,-1,subtree_size[now]/2);
104     book[now] = 1;
105     For(i,0,edge[now].size()-1)
106     {
107         int _to = edge[now][i].to;
108         if(book[_to]) continue;
109         solve_subtree(_to);
110     }
111     n_ds = 0;
112     ds[++n_ds] = 0;
113     For(i,0,edge[now].size()-1)
114     {
115         int _to = edge[now][i].to;
116         if(book[_to]) continue;
117         n_tds = 0;
118         count_subtree_path(_to,now,edge[now][i].val);
119         int t = count_pairs (tds,n_tds);
120         ans -= t;
121         For(i,1,n_tds)
122             ds[++n_ds] = tds[i];
123     }
124     int t = count_pairs(ds,n_ds);
125     ans+=t;
126     book[now] = 0;
127 }
128 int main()
129 {
130     while(scanf("%d%d",&n,&k)!=EOF && n && k)
131     {
132         init();

```

```
133     solve_subtree(1);
134     printf("%d\n",ans);
135     csh();
136 }
137 }
```