# Shufenzhi

1741

#include<cstring>

#include<iostream>

#include<algorithm>

#include<vector>

#include<cstdio>

using namespace std;

const int M= 12345;

struct Edge{

int to,val;

Edge(int to=0,int val=0) : to(to),val(val){}

};

vector<Edge> G[M];

int vis[M];

int s[M];

int d[M];

int dfs(int num,int dist,int fa){

s[num]=1;

d[++d[0]]=dist;

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(vis[to]||to==fa) continue;

s[num]+=dfs(to,dist+G[num][i].val,num);

}

return s[num];

}

int K,sum;

int n,m;

int find\_point(int num,int root,int fa){

int mx=s[root]-s[num];

if(mx>=sum) return 0;

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(vis[to]||to==fa) continue;

mx=max(mx,s[to]);

}

if(mx<sum){

K=num;

sum=mx;

}

for(int i=0;i<G[num].size();i++){

int to=G[num][i].to;

if(vis[to]||to==fa) continue;

find\_point(to,root,num);

}

return 0;

}

long long ans[M];//ans[i]保存的是lca(x,y)=i的答案，即两点连线一定过i点

int work(int num,int sz,int length){

if(sz<=1){ans[num]=0;return 0;}

d[0]=0;

dfs(num,0,0);

//第一次dfs 找出每个点对于上次选取的分治的点的距离

int cc=0;

int t2=sz;

sort(1+d,1+d+sz);

for(int i=1;i<=sz;i++){

while(t2&&d[t2]+d[i]>m-length){

//如果在当前集合的x,y 有d[x]+d[y]<=m-(f[num]到num的权值)\*2,

//那么说明在统计f[num]的时候我们也会计算（x,y），所以要减去

t2--;

if(!t2) break;

}

cc+=t2;

if(d[i]\*2<=m-length) cc--;

}

cc/=2;

sum=sz;K=0;

find\_point(num,num,0);

//找到树的重心进行切割 这样保证严格 log 层

vis[K]=1;

num=K;

d[0]=0;

dfs(num,0,0);

//第二次dfs 找出每个点对于这次选取的分治的点的距离

int t=sz;

sort(1+d,1+d+sz);

ans[num]=0;

for(int i=1;i<=sz;i++){

if(d[i]>m) break;

while(t&&d[t]+d[i]>m){

t--;

if(!t) break;

}

ans[num]+=t;

if(d[i]\*2<=m) ans[num]--;//自己和自己不能构成一对

}

ans[num]/=2;//有序对

for(int j=0;j<G[num].size();j++){

int to=G[num][j].to;

if(vis[to]) continue;

ans[num]=ans[num]-work(to,s[to],G[num][j].val\*2);;

}

vis[num]=0;

return cc;//cc统计的是有可能对该点父亲结点造成重复的答案

}

int main(){

freopen("A.in","r",stdin);

while(1){

cin>>n>>m;

if(n+m==0) break;

for(int i=1;i<=n;i++){

G[i].clear();

vis[i]=0;

}

for(int i=1;i<n;i++){

int x,y,z;

cin>>x>>y>>z;

G[x].push\_back(Edge(y,z));

G[y].push\_back(Edge(x,z));

}

work(1,n,0);

long long Ans=0;

for(int i=1;i<=n;i++) Ans+=ans[i];

printf("%lld\n",Ans);

}

return 0;

}

5977

/\*\*此代码借鉴了某大神的，我看懂了后又分析的\*/

#include<bits/stdc++.h>

#include <vector>

using namespace std;

typedef long long ll;

const int MAXN = 5e5 + 10;

int n, k, Max, root;

ll ans;

vector <int> tree[MAXN];

vector <int> sta;

int sz[MAXN], maxv[MAXN], a[MAXN];

ll Hash[1200];

bool vis[MAXN];

void init(){

memset(vis, false, sizeof(vis));

for (int i = 1; i <= n; i++) tree[i].clear();

}

void dfs\_size(int u, int pre){

sz[u] = 1;

maxv[u] = 0;

int cnt = tree[u].size();

for (int i = 0; i < cnt; i++){

int v = tree[u][i];

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

dfs\_size(v, u);

sz[u] += sz[v];

maxv[u] = max(maxv[u], sz[v]);

}

}

void dfs\_root(int r, int u, int pre){

maxv[u] = max(maxv[u], sz[r] - sz[u]);

if (Max > maxv[u]){

Max = maxv[u];

root = u;

}

int cnt = tree[u].size();

for (int i = 0; i < cnt; i++){

int v = tree[u][i];

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

dfs\_root(r, v, u);

}

}

void dfs\_sta(int u, int pre, int s)

{

sta.push\_back(s);

int cnt = tree[u].size();

for (int i = 0; i < cnt; i++)

{

int v = tree[u][i];

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

dfs\_sta(v, u, s | (1 << a[v]));

}

}

ll cal(int u, int s)

{

ll res = 0;

sta.clear();

dfs\_sta(u, -1, s);

memset(Hash, 0, sizeof(Hash));

int cnt = sta.size();

/\*\*暴力去跑

for(int i=0;i<cnt;i++)

{

for(int j=i+1;j<cnt;j++)

{

if(sta[i]|sta[j]==((1<<k)-1))

res+=2;

}

}\*/

for (int i = 0; i < cnt; i++) Hash[sta[i]]++;

for (int i = 0; i < cnt; i++)

{

Hash[sta[i]]--;///减去自身值

res += Hash[(1 << k) - 1];///i点到状态为((1<<k)-1)肯定符合，所以直接加上

for (int s0 = sta[i]; s0; s0 = (s0 - 1) & sta[i])///枚举i节点状态子集

{

res += Hash[((1 << k) - 1) ^ s0];

}

Hash[sta[i]]++;

}

return res;

}

void dfs(int u)

{

Max = n;

dfs\_size(u, -1);

dfs\_root(u, u, -1);

ans += cal(root, (1 << a[root]));

vis[root] = true;

int cnt = tree[root].size(), rt = root;

for (int i = 0; i < cnt; i++)

{

int v = tree[rt][i];

if (vis[v]) continue;

ans -= cal(v, (1 << a[rt]) | (1 << a[v]));

dfs(v);

}

}

int main(){

while (scanf("%d%d", &n, &k) == 2){

init();

for (int i = 1; i <= n; i++){

scanf("%d", &a[i]);

--a[i];

}

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

{

int u, v;

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

tree[u].push\_back(v);

tree[v].push\_back(u);

}

if (k == 1)

{

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

continue;

}

ans = 0;

dfs(1);

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

}

return 0;

}