这道题让找能够被至少一个集合中的数整除的，那么我们转换一下思路，考虑集合中的数任意搭配（注意这里用二进制就可以很巧的实现）**能够让那些数被整除**，那自然是这些数的最小公倍数的倍数，但是一个数可能被多个集合所计算到，也就是可能是多个集合的最小公倍数的倍数，所以用容斥计算一下

求lcm是必须的，如果只是这个集合里每个数相乘的话，会漏掉一些比它们的乘积小的数

注意0的情况

还要注意找的是1-n-1范围内的，所以除的时候要是n-1除

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<cmath>

#include<stack>

#include<vector>

#include<queue>

#include<algorithm>

#define ll long long

using namespace std;

ll n,ans;

int m;

vector<ll> v;

inline ll solve()

{

ll ans=0;

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

{

ll sum=1;

int cnt=0;

for(int j=0;j<m;j++)

{

if(i&(1<<j))

{

cnt++;

sum=sum/\_\_gcd(sum,v[j])\*v[j];

}

}

if(cnt&1)

ans+=(n-1)/sum;

else ans-=(n-1)/sum;

}

return ans;

}

int main()

{

//freopen("input.txt","r",stdin);

while(cin>>n>>m)

{

v.clear();

ll x;

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

{

cin>>x;

if(x>0)

v.push\_back(x);

}

m=v.size();

cout<<solve()<<endl;

}

return 0;

}