LOJ 1144 - Ray Gun

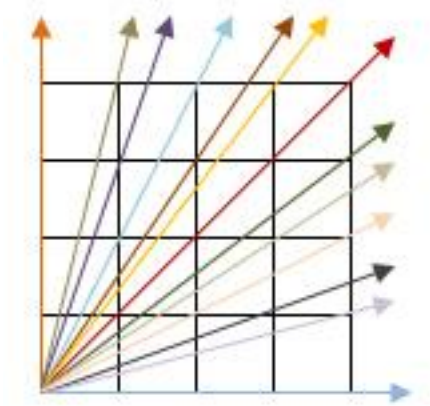
You are in an m x n grid. You are standing in position (0, 0) and in each of the other lattice points (points with integer co-ordinates) an enemy is waiting. Now you have a ray gun that can fire up to infinity and no obstacle can stop it. Your target is to kill all the enemies. You have to find the minimum number of times you have to fire to kill all of them. For a 4 x 4 grid you have to fire 13 times. See the picture below:

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case contains two integers m, n (**$0 \le m, n \le 10^9$**) and **at least one of them will be less than or equal to 10^6.**

For each case, print the case number and the minimum number of times you have to fire to kill all the enemies.

Sample Input:	**Sample Output:**
5	
4 4	**13**
0 0	**0**
0 4	**1**
4 0	**1**
1 4	**6**
1000000000 1000000	**607927103575025**



****

IDEA

- ➔ ANSWER = Number of co-prime pair in (NxM) pair + 2(for x axis and y axis). That is how many integer/lattice co-ordinate (x_i, y_i) are there in (NxM) grid such that GCD(x_i,y_i)=1; how many ways we can select one pair/point from (N*M) points such that their GCD is 1.
- ➔ We know a straight line between coordinate (x1,y1) and (x2,y2) has the number lattice point = GCD(abs(x1, x2), abs(y1, y2)) + 1;
- ➔ Actually we have to count all that points that if we fire up ray gun from origin(0,0) which lattice point it first attack in this grid.
- ➔ If a point let say (x,y) is attacked by the i'th gun shot then first point that is attacked by the i'th gun shot is ((x/gcd(x,y)), (y/gcd(x,y)))
- ➔ We can use inclusion exclusion principle and Mobius function to solve the problems.
- ➔ First we add all the points whose GCD is atleast 1, then subtract whose GCD is atleast 2, then subtract whose GCD is atleast 3, do nothing for whose GCD is atleast 4, … , add whose GCD is atleast 6,…, do nothing for whose GCD is atleast 12, …, subtract whose GCD is atleast 30… add whose GCD is atleast 210 etc. Thai is subtract(GCD-2,3,5,30 etc, number of odd prime factors) or add(GCD-6,10,14,15, etc number of odd prime factors) all the triples from ${}^{N}C_3$ ways. and do not add or subtract for(GCD-4,12,18 etc divisibe by any perfect square numbers i.e. have a prime factor p^k where k>1).

Sample Code:

```
#define ll long long
ll mob[1000005],vis[1000005];
void mobius()
{
    for(int i=1; i<=1000000; i++)mob[i]=1;
    mob[1]=1;
    for(ll i=2; i<=1000000; i++){
        if(vis[i]==0){
            mob[i]=-1;
            for(ll j=i+i; j<=1000000; j+=i){
                if(j%(i*i)==0)mob[j]=0;
                mob[j] *= (-1);
                vis[j]=1;
            }
        }
    }
}
ll solve(ll n,ll m)
{
    ll ans=n*m;
    for(ll g=2; g<=n; g++){
        ans += mob[g]*((n/g) * (m/g));
    }
    ans += 2;
    return ans;
}
int main()
{
    mobius();
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++)
    {
        ll n,m; scanf("%lld%lld",&n,&m);
        if(n>m)swap(n,m);

        ll ans;
        if(n==0 && m==0)ans=0;
        else if(n==0 || m==0)ans=1;
        else ans = solve(n,m);

        printf("Case %d: %lld\n",ks,ans);
    }
    return 0;
}
```