**Number Theory 1**

**Diophantine Equations**

These are the polynomial equations for which integral solution exists.

Example: 3x + 7y = 1 or x2 - y2 = z3.

For competitive programming, we only need to study linear diophantine equations of the form

*ax + by = c*

Note: a,b,c ∊ I (set of integers).

Solutions to these equations exist only if gcd(a,b) divides c.

**Extended Euclid Algorithm**

It is the extended form of euclid’s algorithm. GCD(a,b) has the property that it can be written in the form of an equation like

*ax + by = gcd(a,b)*

We will find values of x and y

*ax + by = gcd(a, b)*

*gcd(a, b) = gcd(b, a%b)*

*gcd(b, a%b) = bx1 + (a%b)y1*

*a%b = a − (a/b) ∗ b*

*From the above equations we get,*

*ax + by = bx1 + (a%b)y1*

*ax + by = bx1 + (a − (a/b) ∗ b)y1*

*ax + by = ay1 + b(x1 − (a/b) ∗ y1)*

*Comparing the coefficients of a and b, we get*

*x = y1*

*y = x1 − (a/b) ∗ y1*

**Code**

struct Triplet

{

int x,y,gcd;

};

Triplet extendedEuclid(int a, int b)

{

if(b == 0){

Triplet ans;

ans.gcd = a;

ans.x = 1;

ans.y = 0;

return ans;

}

Triplet smallAns = extendedEuclid(b, a%b);

Triplet ans;

ans.gcd = smallAns.gcd;

ans.x = smallAns.y;

ans.y = smallAns.x - (a/b)\*smallAns.y;

return ans;

}

**Multiplicative Modulo Inverse**

Consider the equation

*(A\*B)%m = 1*

We are given A and m. Our task is to find the value of B such that R.H.S becomes 1.

Memory tip: To remember MMI, just remember this line

**For what value of B eqn** (A ∗ B)%m = 1 **holds true**.

Finding MMI,

Consider the equation

*A ∗ B ≡ 1 (mod m)*

*⇒ (A ∗ B − 1) ≡ 0 (mod m)*

*⇒ A ∗ B − 1 = mq*

*⇒ A ∗ B + mQ = 1*

This is our normal diophantine equation. For its solution to exist gcd(A,m)

should divide 1, which means gcd(A,m) = 1

Our MMI will simply be the values of x from our Extended Euclid Algorithm.

Code

int mmInverse(int a, int m)

{

Triplet ans = extendedEuclid(a, m);

return ans.x;

}

void solve()

{

int a=19, m=17;

int ans = mmInverse(a,m);

cout <<"MMI is "<< ans << endl;

}