## Matematica Modulare

L’aritmetica modulare viene anche spesso detta **aritmetica dell’orologio**. Questo perché fa uso del concetto di modulo n (12 o 24 nel caso dell’orologio). In termini matematici le ore 17 corrispondono (sono congruenti) alle 5 ≡ mod 12 cosi come anche le ore 29 5 ≡ mod 12.

Inoltre la relazione di congruenza è una relazione di equivalenza, ossia gode della proprietà riflessiva, simmetrica e transitiva.

* Riflessiva: a congruente ad a mod m
* Simmetrica: se a congruente a b, allora b congruente ad a
* Transitiva: se a congruente a b e b congruente a c, allora a congruente a c

Facile da dimostrare che se **a** ≡ b mod m allora a – b è multiplo di m!

Esempio: m= 17 23 ≡ 40 mod 17, 40 – 23 = 17

Anche la moltiplicazione si comporta nella stessa maniera, esempio m=17, a=37, b=21

|  |  |
| --- | --- |
| Notazione semplificata (mod 17) | Notazione aritmetica |
| a\*b = 777 | a\*b = 777 |
| 12 = [777] se m=17 | 12 ≡ 777 mod 17 |
|  |  |
| 3 ≡ [37] | 3 ≡ 37 mod 17 |
| 4 ≡ [21] | 4 ≡ 21 mod 17 da cui |
| 12 ≡ [3\*4] ≡ [37\*21] ≡ [37]\*[21] | 12 ≡ 3\*4 mod 17 |
| -5 ≡ [12] |  |

## Inverso moltiplicativo e Bezout

Caso particolare è l’**inverso moltiplicativo** (IM). Per inciso se in aritmetica modulare ho:

1 ≡ a \* x mod m

Allora x è l’inverso moltiplicativo (IM) di a, in pratica x = a-1 mod m per cui a\*x ≡ a/a ≡ 1

Dato per assunto che a e b siano **co-primi** (non hanno fattori in comune) allora si possono trovare grazie all’algoritmo esteso di Eulero (AEE, vedi sotto) coefficienti di Bezout (x, y) tali per cui:

a\*x + b\*y = 1

esempio: se a=29 e b=31 possiamo verificare che:

x=15 e y=-14

infatti 29\*15 – 31\*14 = 1 da questo si deduce il seguente:

x è l’IM di a mod b

y è l’IM di b mod a

|  |  |
| --- | --- |
|  |  |
| a \* x mod b = 1 | 29 \* 15 mod 31 = 1 |
| b \* y mod a = 1 | 31 \* (-14) mod 29 =  31 \* (-14 + 29) mod 29 = 31 \* 15 mod 29 = 1 |

Esempio: se a=47 e b=31 allora  
 x=2 e y=-3

47 \* 2 mod 31 = 1

normalizzando il numero negativo ‘y’ per la proprietà della matematica modulare che indica che:  
 a ≡ (a + km) mod m per cui -3 ≡ -3 + 47 = 44

allora  
 31 \* 44 mod 47 = 1

## Massimo comun divisore (MCD)

Per il calcolo del MCD esiste l’algoritmo di Eulero.

L’AE permette di trovare il massimo comun divisore (mcd) fra 2 numeri interi.   
 Ad esempio:

MCD (12,18) = 6

MCD (4,14) = 2   
 MCD (4,14) = 2

MCD (5,0) = 5   
 MCD (5,0) = 5

In linguaggio Java potrebbe essere:

int mcd(int a, int b) {

int r;

while (b != 0) {

r = a % b;

a = b;

b = r;

}

return Math.abs(a);

}

oppure in forma ricorsiva

public int mcd(int a, int b) {

if (b == 0)

return a;

return mcd(b, a % b);

}

## Algoritmo Euclideo Esteso (AEE)

In aritmetica e nella programmazione l'algoritmo esteso di Euclide è un'estensione dell'algoritmo di Euclide che calcola non solo il massimo comun divisore (indicato con MCD nel seguito) tra due interi a e b, ma anche i coefficienti dell'identità di **Bezout** x e y tali che:

ax + by = MCD(a,b)

L'algoritmo esteso di Euclide è particolarmente utile quando a e b sono **interi coprimi.**

Nella teoria dei numeri, due interi a e b sono primi tra loro se il numero intero positivo che è divisore di entrambi è 1 in pratica MCD (a, b) = 1, esempio: (4,9), (6,35).   
In questo caso x è l'**inverso moltiplicativo** di a modulo b e y è l'inverso moltiplicativo di b modulo a.

In linguaggio Java (versione>=16) utilizzando i Java [BigInteger](https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html) potrebbe essere la seguente:

public record GcdRec(BigInteger resto, BigInteger x, BigInteger y) { }  
  
public GcdRec gcd(BigInteger a, BigInteger b) {

if (b == null || b.signum() == 0)

return new GcdRec(a, BigInteger.ONE, BigInteger.ZERO);

GcdRec res = gcd(b, a.mod(b));

BigInteger d = res.resto();

BigInteger x = res.y();

BigInteger y = res.x() //

.subtract( //

a.divide(b) //

.multiply(x) //

);

return new GcdRec(d, x, y);

}

## Funzione di Eulero

Con la funzione di Eulero (fi greco di n) φ(n) si calcola la **quantità** di interi a dove 0<a<n

che soddisfa   
 MCD (a, n) = 1

In pratica la **quantità** di numeri che sono **primi** con n, in altre parole la quantità di fattori primi dato per assunto che:

n = p1α1 \* p2α2 \* p3α3 \* … \* pkαk

dove pk sono numeri primi e αk sono esponenti, avremo

esempio: se n=1600 allora avremo 1600=26\*52

da qui:

Ma se n stesso è primo !!! allora abbiamo che φ(n) = n – 1

Ma se n = p \* q, dove sia p che q sono primi allora avremo φ(pq) = (p – 1)(q - 1)

## Piccolo teorema di Fermat

Se abbiamo un intero a>0 e un numero primo p allora avremo la relazione:

a p-1 ≡ 1 mod p

esempi:

|  |  |  |  |
| --- | --- | --- | --- |
| a | p | ap-1 | ap-1 mod p |
| 8 | 3 | 64 | 1 |
| 19 | 3 | 361 | 1 |
| 17 | 7 | 24.137.569 | 1 |
| 11 | 5 | 14.641 | 1 |

Facendo come considerazione che:

1. se n è primo
2. MCD(a,n) = 1
3. Allora aφ(n) ≡ 1 mod n

## Algoritmo RSA

1. Scelti due primi (molto alti) p e q
2. E il modulo n = p \* q
3. Il messaggio in forma numerica è M
4. Si sceglie un numero e tale che MCD(e, φ(n)) = 1   
   quindi e è co-primo con φ(n)
5. Si sceglie un numero d tale sia l’inverso moltiplicativo di e in φ(n)  
   si sa che esiste perché MCD(e, φ(n)) = 1 !  
   a tal punto avremo che:  
   de ≡ 1 mod φ(n)
6. A questo punto possiamo **crittare** il messaggio M con:  
   C = Me mod n
7. Per riavere il messaggio M da C allora  
   M = Cd mod n  
   in altri termini d essendo l’inverso moltiplicativo di e, abbiamo  
   M = ( M e )d mod n

Se si dimostra tale uguaglianza si dimostra la validità dell’algoritmo RSA.

## Proviamo dimostrare RSA

Se MCD (m, n) = 1 e per il teorema di Euler :  
 mφ(n) ≡ 1 mod n

Sappiamo che d ed e sono inversi moltiplicativi per cui:

d e – 1 ≡ 0 mod φ(n)

ovvero esiste un valore k intero tale che:

d e – 1 ≡ k φ(n)

d e ≡ k φ(n) + 1

quindi con un paio di passaggi

(me)d = med = m k φ(n) + 1 = (m φ(n))km ≡ 1k m mod n = m mod n

Otteniamo l’equivalenza desiderata.