Calcolo del M.C.D
Algoritmo di Euclide

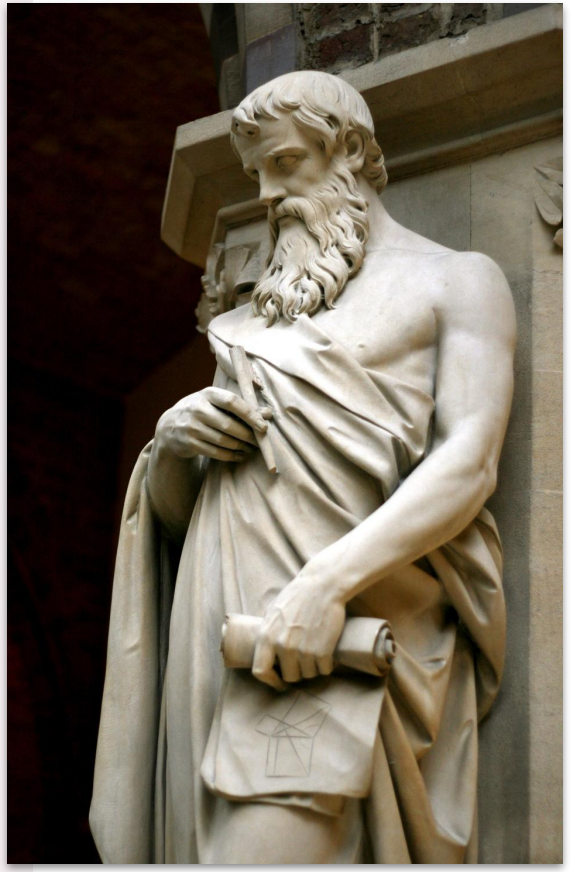
Chiara Luchini Nicolò Posta

Gli “Elementi” della storia di Euclide

Chi era Euclide?

Euclide è stato un **matematico** e **filosofo greco** antico vissuto tra il IV e il III secolo a.C., riconosciuto a livello mondiale per le sue scoperte in vari ambiti dall’astronomia alla musica fino alla matematica.

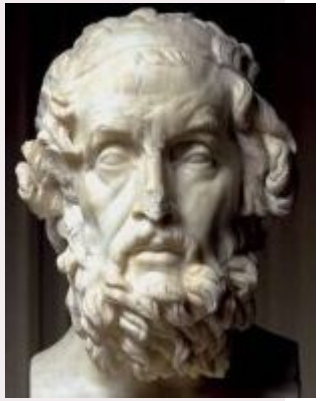
Fra queste ricordiamo l’**Algoritmo di Euclide** per il calcolo del **massimo comune divisore** trattato in una delle sue opere più importanti **“Elementi”**. Quest’ultima è composta da 13 libri diversi nei quali sono formulati differenti teoremi e regole della geometria e dell’aritmetica.



Sapevate che?

L'algoritmo di Euclide non è stato scoperto da Euclide stesso.

L'algoritmo era già conosciuto circa 75 anni prima della stesura dell'opera "**Elementi**" da parte di un altro matematico greco chiamato **Eudosso di Cnido** e anche pochi anni dopo da **Aristotele** che lo cita all'interno della sua opera intitolata "**I Topici**".



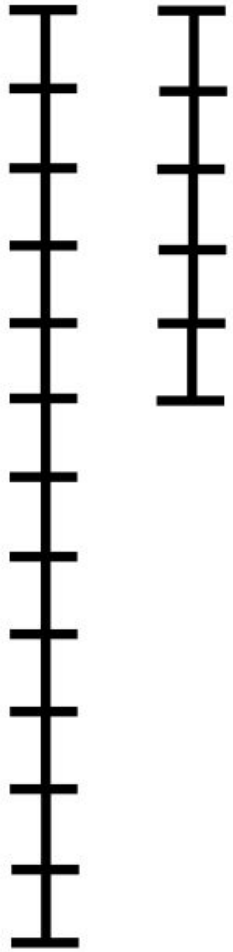
Eudosso di Cnido

Aristotele

Perché è stato creato?

Un problema geometrico...

Euclide originariamente formulò il problema geometricamente, per trovare una **"misura" comune** per la lunghezza di due segmenti, e il suo algoritmo procedeva sottraendo ripetutamente il più corto dal più lungo.



Un algoritmo *antico* ma *moderno.*

Anche se conosciuto ormai da oltre **2 millenni** l'algoritmo di Euclide è ancora utilizzato regolarmente in applicazioni matematiche ed informatiche come:

- Algoritmo **RSA** (Crittografia)
- Generare ritmi musicali (Musica)
- Equazioni Diottriche (Chimica)
- Teorema Cinese dei resti (Matematica)

Prima di iniziare...

Cosa è il M.C.D.?

Il **massimo comune divisore** di due o più numeri, indicato con il simbolo **MCD**, è il più grande divisore comune dei numeri considerati.

Come si calcola?

Esistono diversi per metodi per il calcolo del **MCD**:

- Scomposizione in fattori primi;
- Metodo di Euclide delle sottrazioni successive;
- Algoritmo euclideo delle divisioni successive.

Il metodo delle sottrazioni successive

Formulazione

*Prendiamo due numeri disuguali e si procede per sottrazioni successive togliendo ogni volta il minore dal maggiore, se il risultato di ogni sottrazione non **divide MAI** il sottraendo allora i due numeri sono coprimi tra loro.*

Traduzione "matematica"

Questo vuol dire che se abbiamo due numeri **a** e **b**:

1. Se **a>b** allora si fa **a−b = c**, sennò **b−a = c**;
2. Se **b>c** allora **b=a** e **c=b** e viceversa;
3. Se **c** non divide MAI **b** allora **a e b** sono coprimi.

Infatti l'ultima sottrazione avrà come risultato 1 se i numeri sono primi e 0 altrimenti.

Il metodo delle sottrazioni successive: esempi

Numeri primi tra loro

Con a = 31 e b = 13:

- 31−13= 18
- 18−13 = 5
- 13−5 =8
- 8−5 = 3
- 5−3 = 2
- 3−2=**1**

Numeri primi tra loro

Con a = 14 e b = 9:

- 14−9=5
- 9−5=4
- 5−4=**1**

Numeri NON primi tra loro

Con a = 24 e b =15:

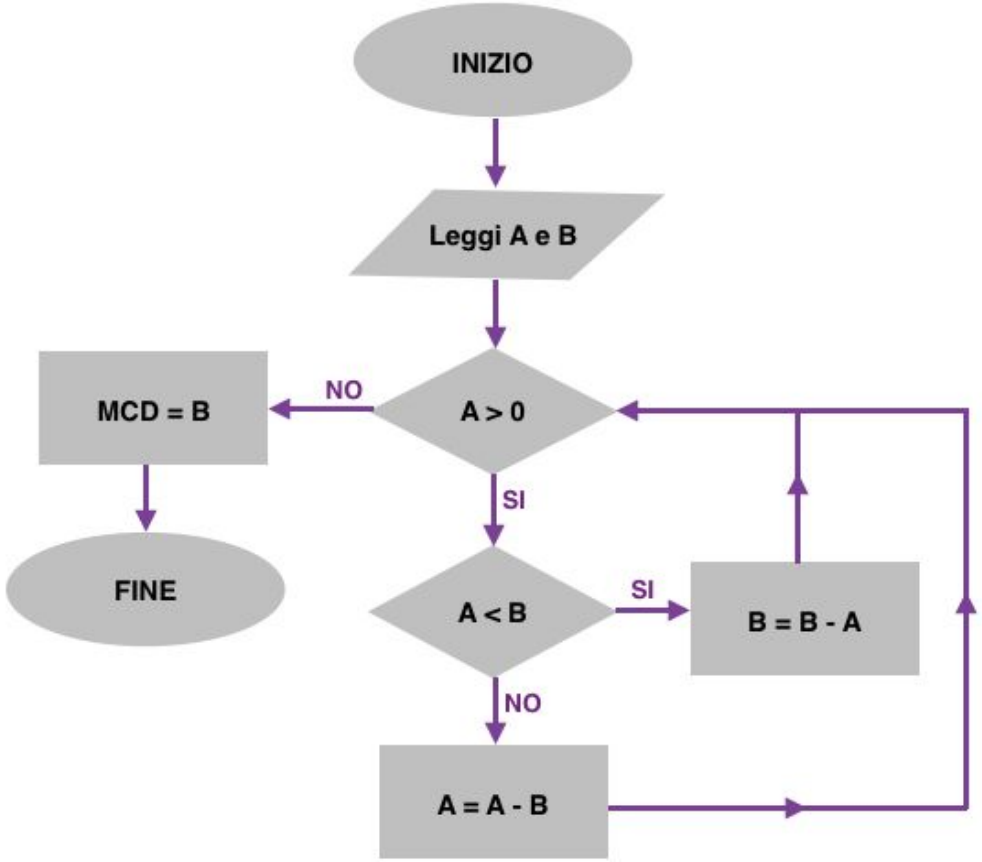
- 24−15=9
- 15−9=6
- 9−6=3
- 6−3=**3**
- 3−3=0

Il MCD(24,15) è 3.

Il metodo delle sottrazioni successive: l'algoritmo

Dati due numeri naturali a e b, diversi da 0:

1. se fosse **a<b** allora scambia **a** con **b**;
2. se **a = b** allora **b** è il M.C.D. tra **a** e **b** altrimenti calcola la differenza **a−b** ;
3. definiamo **a−b** come nuovo valore di **a** (mentre **b** resta invariato);
4. se fosse **a<b** allora scambia **a** con **b**;
5. ripeti il ciclo delle istruzioni fino a che si ottiene **a−b = 0**;
6. **b** è il M.C.D. tra **a** e **b**;



Il metodo delle sottrazioni successive: implementazione in Python 1/2

Alcune idee per costruirlo:

Versione iterativa

1. **Quali sono i casi di stop dell' algoritmo?**
   - In questo caso l'algoritmo si ferma quando **a = b**.
2. **Qual è il costrutto iterativo da utilizzare?**
   - Possiamo utilizzare il ciclo **while**.

Versione ricorsiva

1. **Qual è il caso base della ricorsione?**
   - Il caso base è quando **a = b**, lo verifichiamo tramite un **if..else...** .
2. **Quando richiamare la funzione?**
   - Quando abbiamo controllato chi è il numero maggiore nelle varie sottrazioni (es. **x>y**).

Il metodo delle sottrazioni successive: implementazione in Python 2/2

Versione iterativa

```
a = int(input("Immetti il primo numero:"))
b = int(input("Immetti il secondo numero:"))

def mcd_It(x,y):
    while x != y :
        if x > y:
            x = x-y
        else:
            y = y-x
    print(f"Il massimo comune divisore di {a} e {b} è {x}")

mcd_It(a,b)
```

Versione ricorsiva

```
a = int(input("Immetti il primo numero:"))
b = int(input("Immetti il secondo numero:"))

def mcd_Ric(x, y):
    if x != y:
        if x > y:
            x = x-y
        else:
            y = y-x
        mcd_Ric(x,y)
    else:
        print(f"Il massimo comune divisore di {a} e {b} è {x}")

mcd_Ric(a,b)
```

Una derivazione del metodo delle sottrazioni consecutive

Denominata dagli studiosi *algoritmo euclideo delle divisioni successive*

Dal metodo di Euclide ne deriva una variante basata sulle **divisioni successive** più efficiente rispetto a quella precedente.

In quanto una divisione può essere vista come una sottrazione successiva e perciò utilizzata in modo analogo per il calcolo del **M.C.D.** .

TH: Divisibilità del resto

*Se due numeri naturali **a** e **b**, con a > b, sono divisibili per uno stesso numero **c**, allora anche **r**, resto della divisione **a:b**, è divisibile per **c**.*

Nell algoritmo di Euclide, per diminuire il numero di operazioni da eseguire, possiamo allora procedere mediante divisioni successive, invece che sottrazioni, e utilizzare i resti ottenuti.

Esempio:

130 e **40** sono divisibili per **5**.

130 : 40 = 3 con resto di **10**

10 è divisibile per **5**

Il metodo delle divisioni successive

Formulazione

*Prendiamo due numeri disuguali e si procede per divisioni successive tra il divisore ed il relativo resto, se il resto di ogni divisione non **divide MAI** il relativo divisore, finchè rimarrà solo l'unità, allora i due numeri sono coprimi tra loro.*

Traduzione "matematica"

Questo vuol dire che se abbiamo due numeri **a** e **b**:

1. Se **a>b** allora si fa **a/b = c + r ,** sennò **b/a = c + r**;
2. Se **r != 1** allora **a=b** e **b=r**;
3. Se **r** non divide MAI **b** allora **a e b** sono coprimi.

Infatti l'ultima divisione avrà come resto 1 se i numeri sono primi e 0 altrimenti.

Il metodo delle divisioni successive: esempi

Numeri primi tra loro

Con a = 27 e b = 5:

- 27 / 5 = 4 + 7 di r.
- 7 / 5 = 1 + 2 di r.
- 5 / 2 = 2 + **1** di r.

Numeri primi tra loro

Con a = 3005 e b = 102:

- 3005 / 102 = 29 + 47 di r.
- 102 / 47 = 2 + 8 di r.
- 47 / 8 = 5 + 7 di r.
- 8 / 7 = 1 + **1** di r.

Numeri NON primi tra loro

Con a = 24 e b = 15:

- 24 / 15 = 1 + 9 di r.
- 15 / 9 = 1 + 6 di r.
- 9 / 6 = 1 + 3 di r.
- 6 / 3 = 2 + **0** di r.

Il MCD(24,15) è 3.

Il metodo delle divisioni successive: l'algoritmo 1/2

Per implementarlo abbiamo bisogno di una funzione speciale: Il Modulo

La funzione **modulo** funziona come la normale divisione, ma come risultato riporta il resto della divisione, non il suo quoziente.

La notazione matematica corretta è **mod**.

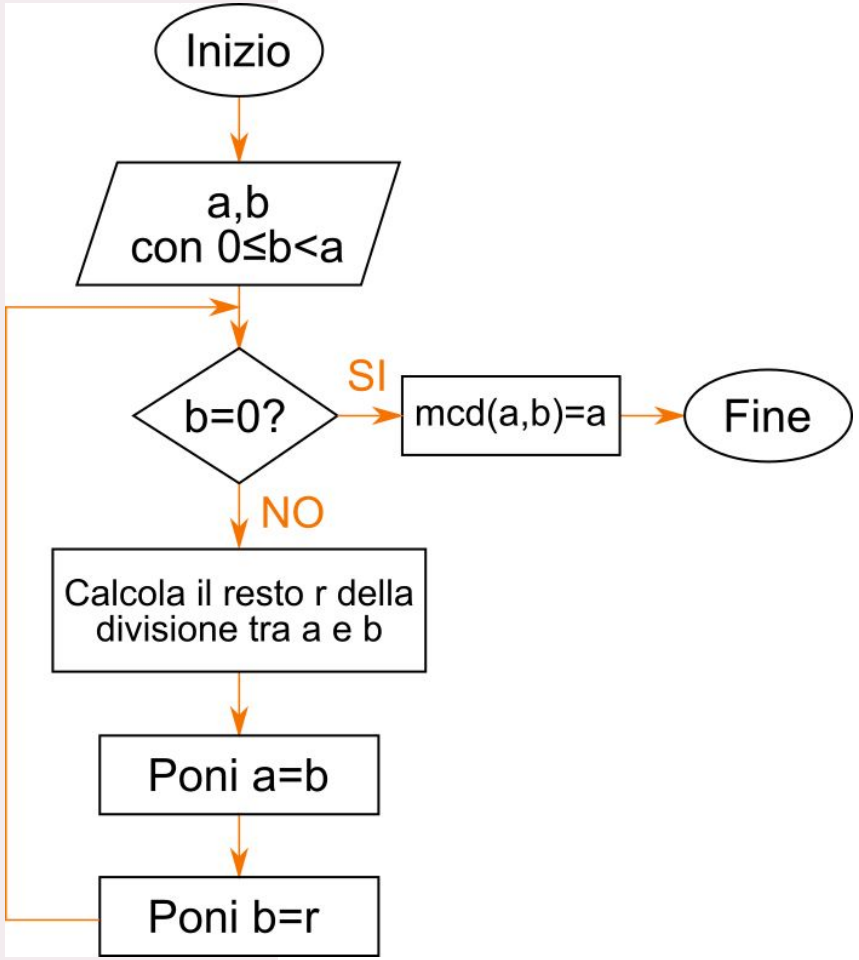
Esempio: 13 **mod** 5 = 3

Il risultato è 3 perché 13/5 è uguale a 2 con resto 3.

Il metodo delle divisioni successive: l'algoritmo 2/2

Dati due numeri naturali a e b, diversi da 0 con a ≥ b:

1. se **a = b** allora **a** è il M.C.D. tra **a** e **b** altrimenti calcola il resto **r** della divisione **a mod b**;
2. definiamo **b** come nuovo valore di **a** e **r** come nuovo valore di **b**;
3. ripeti il ciclo delle istruzioni fino a che si ottiene **r = 0**;
4. allora **a** è il M.C.D. tra **a** e **b**;



Il metodo delle divisioni successive: implementazione in Python 1/2

Alcune idee per costruirlo:

Versione iterativa

1. **Quali sono i casi di stop dell' algoritmo?**
   - In questo caso l'algoritmo si ferma quando **b = 0**.
2. **Qual è il costrutto iterativo da utilizzare?**
   - Possiamo utilizzare il ciclo **while**.

Versione ricorsiva

1. **Qual è il caso base della ricorsione?**
   - Il caso base è quando **b = 0**, lo verifichiamo tramite un **if..else...** .
2. **Quando richiamare la funzione?**
   - Dopo aver calcolato il resto **r**, richiamandola con i parametri **y** e **r**.

Il metodo delle divisioni successive: implementazione in Python 2/2

Versione iterativa

```
a = int(input("Immetti il primo numero:"))
b = int(input("Immetti il secondo numero:"))

def mcd_It_div(x,y):
    while y>0:
        r = x % y
        x, y = y, r
    print(f"Il massimo comune divisore di {a} e {b} è {x}")

mcd_It_div(a,b)
```

Versione ricorsiva

```
a = int(input("Immetti il primo numero:"))
b = int(input("Immetti il secondo numero:"))

def mcd_Ric_div(x,y):
    if y>0:
        r = x % y
        mcd_Ric_div(y,r)
    else:
        print(f"Il massimo comune divisore di {a} e {b} è {x}")

mcd_Ric_div(a,b)
```

Esercizio in Laboratorio

Esercizio di approfondimento

Calcolare m.c.m.

Data la relazione: $$mcm(a,b) = \frac{a \cdot b}{mcd(a,b)}$$

1. Disegnare il diagramma di flusso dell'algoritmo per il calcolo del m.c.m.
2. Implementare un algoritmo sia in versione ricorsiva che in versione iterativa che calcoli il minimo comune multiplo di due numeri **a** e **b**, utilizzando uno degli algoritmi proposti per il calcolo del M.C.D.

Grazie per l’attenzione