M-file

- Un programma MATLAB è salvato in un M-file (estensione .m)
- Per creare un programma è perciò sufficiente creare un nuovo M-file.
- Gli M-file possono essere creati attraverso l'editor predefinito MATLAB o un qualunque editor di file di testo.
- Un M-file può essere poi eseguito in MATLAB
- Esistono due tipi di M-file: script e funzioni.

Script

- Sono utili per automatizzare una sequenza di operazioni che debbano essere eseguite molte volte
- Non accettano argomenti in input.
- Non ritornano argomenti in output.
- Memorizzano le variabili nel workspace (base workspace) condiviso con altri script e con la Finestra dei Comandi

```
v=ones(1,5);
A=diag(v);
any(A)
```

- Salvare le istruzioni precedenti in un file prova_script.m e digitare prova_script dalla shell MATLAB

Funzioni

- Sono utili per estendere il linguaggio MATLAB per applicazioni specifiche
- Possono accettare argomenti in input
- Possono restituire argomenti in output
- Memorizzano le variabili in un workspace separato (function workspace) interno alla funzione
- Sono implementate in M-file il cui nome è in genere lo stesso nome della funzione.

Funzioni

- Linea di definizione della funzione:
  - definisce il nome della funzione, il numero e l'ordine dei suoi argomenti in input e output
  - **function** [rv1 rev2] = MiaFunzione(iv1, iv2)

    Dove rv1 e rv2 sono valori di ritorno e iv1 e iv2 sono valori di input. A differenza di molti linguaggi di programmazione non è necessario specificare il tipo dei dati di input o ritorno.
- Gli argomenti in input sono racchiusi tra parentesi tonde () e devono essere separati da virgole.
- Le variabili associate agli argomenti in input sono locali alla funzione.
- NOTA: il nome degli argomenti in input non ha alcuna relazione con il nome della variabile che concretamente verrà passata alla funzione.
- Gli argomenti in output sono racchiusi tra parentesi quadre [] e devono essere separati da virgole.

Funzioni

- Vediamo una semplice funzione per sommare 2 valori:

```
function s = somma(a, b)
s = a+b
```

- E una per il fattoriale, da salvare nel file fattoriale.m

```
function s = fattoriale(n)
  s = 1; % inizializzazione di s
  for i = 1:n
    s = s*i;
end
```

```
>> fattoriale(5)
   ans =
            120
```

- Le righe precedute da % sono commenti

Funzioni

- La prima funzione che appare in un M-file è detta funzione primaria (primary function).
- La funzione primaria di un M-file può essere chiamata dalla Finestra dei Comandi o da un altro M-file.
- Nello stesso file possibile definire anche altre funzioni, dette **sottofunzioni** (subfunctions).
- Le sottofunzioni possono essere chiamate dalla funzione primaria e da altre sottofunzioni all'interno di un M-file, ma NON sono visibili all'esterno dell'M-file.
- Quindi le sottofunzioni NON possono essere chiamate dalla Finestra dei Comandi o da altri M-file.

Funzioni

- function [avg, med] = newstats(u)
  % Primary function. NEWSTATS Find mean and median
  n = length(u);
  avg = mean(u, n);
  med = median(u, n);

  function a = mean(v, n) % Subfunction, Calculate average
  a = sum(v) / n;

  function m = median(v, n) % Subfunction, calculate median.
  w = sort(v);
  if rem(n, 2) == 1
  m = w((n + 1) / 2);
  else
  m = (w(n / 2) + w(n / 2 + 1)) / 2;
  end % ultimo end necessario

Comandi utili

- **find** : permette di cercare degli elementi in un vettore/matrice che soddisfano una certa condizione
- Restituisce un vettore contenente gli indici degli elementi che soddisfano la condizione

```
>> v=[1 3 6 4 11 5 7];
>> find(v > 5)
ans =
     3     5     7
```

- **numel** : numero elementi in un vettore/matrice
- **length** : lunghezza di un vettore
- **size** : dimensioni di una matrice

  size(A, 1) numero di righe, size(A, 2) numero di colonne

Esercizi

- Scrivere una funzione MATLAB che conta il numero di occorrenze di un numero all'interno di un vettore
- Scrivere una funzione MATLAB che conta il numero di occorrenze il numero di elementi pari in un vettore
- Scrivere una che preso in input un vettore restituisce il prodotto delle occorrenze del numero 5 e delle occorrenze dei numeri dispari
- Scrivere una funzione che chiede all'utente una matrice restituisce un vettore contenente la somma degli elementi di ogni riga della matrice
- Utilizzando l'istruzione 'for', scrivere una funzione MATLAB che presa in input una matrice, restituisca un vettore che contiene per ogni possibile coppia di vettori, la differenza delle corrispondenti medie.

Input / Output a video

- Le stringhe e i valori numerici possono essere visualizzate tramite la funzione **disp**:

>> disp(‘Visualizzazione di questa stringa sullo standard output’)

>> disp(15);

- E' possibile leggere valori in input digitati da tastiera con il comando **input**: >> var = input('messaggio')

>>a = input('a= ');

- Sul video comparirà:

>> a=

- Dopo aver digitato per esempio [5,3](oppure solo 5) in a sarà memorizzato il vettore riga [5 3] (risp. l'intero 5).

Input / Output a video

>>val = input(’Inserire la dimensione dell’’array: ’);

>>A = zeros(1, val);

- input può anche essere usato per leggere delle stringhe:

>> val = input('Inserire la dimensione dell''array: ' ,'s');

>> val = str2num(val)

>> A = zeros(1, val);

Tempi di Esecuzione

- Per visualizzare il tempo di esecuzione necessario al sistema per svolgere una operazione è possibile usare i comandi **tic** e **toc:**

```
tic
fattoriale(5);
toc

tic
fattoriale(5000);
toc
```

Calcolo del minimo

- Dato v vettore

```
function min = minimo(v)
n = length(v);
min = v(1);
for i=2:n
  if(v(i) < min)
    min = v(i);
  end
end
```

- Quante operazioni vengono svolte?

Calcolo del massimo

- Dato v vettore

```
function max = massimo(v)
n = length(v);
max = v(1);
for i=2:n
  if(v(i) > max)
    max = v(i);
  end
end
```

- Quante operazioni vengono svolte?

Calcolo di minimo e massimo

- Dato v vettore
  function [min, max] = minimomassimo(v)
  min=minimo(v);
  max=massimo(v);
- Quanti confronti vengono svolti?
- Si può fare meglio?

Calcolo di minimo e massimo

```
function [rmin,rmax ]= minimomassimo2(v)

n = numel(v);
if v(1)< v(2)
    rmin=v(1); rmax=v(2);
else
    rmin=v(2); rmax=v(1);
end
for i = 3:2:(n-1)
   if v(i) < v(i+1)
        if v(i) < rmin
            rmin = v(i);
        end
        if v(i+1) > rmax
            rmax = v(i+1);
        end
```

```
   else
        if v(i+1) < rmin
            rmin=v(i+1);
        end
        if v(i) > rmax
            rmax = v(i);
        end
    end
end
if mod(n,2)~=0
       if v(n) > rmax
           rmax = v(n);
       else
           if rmin > v(n)
               rmin = v(n);
           end
       end
end
end
```

Calcolo di minimo e massimo

- minimomassimo compie 2n-2 confronti

- minimomassimo2 compie 1+(n/2 - 1)*3 confronti

  quando n è pari

- 2n-2 – [3(n/2)-2] = n/2
  - per n → infinito la differenza è considerevole

Minimo e Massimo

```
>> tic;minimomassimo(1:100000000);toc
Elapsed time is 6.458089 seconds.

>> tic;minimomassimo2(1:100000000);toc
Elapsed time is 4.652109 seconds.

>> tic;min(1:100000000);max(1:100000000);toc
Elapsed time is 9.543298 seconds.
```

Numeri Casuali

1. A volte in un algoritmo può essere necessario ottenere un numero casuale. La più semplice funzione per generare numeri casuali è **rand**

   - rand() restituisce un numero uniformemente estratto in [0, 1]

   >>Mrd = rand(3,4); % matrice 3x4 di numeri casuali

2. Molto più generica è la funzione random() che restituisce valori presi dalle più comuni distribuzioni di probabilità:

   >>val1 = random('Normal',0,1,2,4)

   >>val2 = random('poiss',1,2,4)

3. Usare il comando help per descrizione dei comandi

Numeri Casuali

1. MATLAB implementa parecchie funzioni statistiche. Di particolare interesse sono la media **mean**() e la deviazione standard **std**()

```
M1 = random('Normal',1,2,10,6)
Media = mean(M1,2);
DevStd = std(M1, 0, 2);
```

Esercizi

1. Scrivere uno script MATLAB che crea due vettori, uno contenente i primi 100 interi, l'altro contenente i primi 100 numeri pari, e ne restituisce il prodotto scalare visualizzando a video il tempo di esecuzione impiegato per tutte le istruzioni
2. Scrivere una funzione MATLAB che chiede all'utente un intero n < 10 e stampa a video il fattoriale dei numeri interi da 1 fino ad n
3. Scrivere uno script MATLAB che cicla all'infinito finché l'utente non digita la stringa 'esci'. Ad ogni iterazione, se il valore inserito è diverso da 'esci', ne stampa a video il fattoriale.

Esercizi

4. Confrontare i tempi di esecuzione della funzione scritta al punto precedente con la funzione MATLAB **max** su vettori di taglia 10000, 100000, 1000000.
6. Scrivere una funzione MATLAB che legge dall'input un intero n, crea una matrice quadrata A di numeri casuali di dimensione n, e restituisce una matrice nx2, contenente, per ogni riga A, i corrispondenti elementi minimo e massimo.
8. Scrivere una funzione che presa in input una matrice, ne calcola il minimo e il massimo.

Complessità degli Algoritmi

- La complessità di un algoritmo è misurata in base al tempo di calcolo impiegato (complessità temporale) e alle risorse utilizzate, e.g. celle di memoria (complessità spaziale)
- Considereremo prevalentemente la complessità temporale
  - Complessità temporale misurata come numero di istruzioni eseguite
  - Consideriamo un modello in cui ogni operazione ha costo unitario

Dimensione dell'input

- Misureremo le risorse di calcolo usate da un algoritmo in funzione della dimensione dell'istanza in input

  - Generalmente numero di elementi che compongono gli oggetti in input all'algoritmo

  - Esempio: numero di elementi di una vettore da ordinare

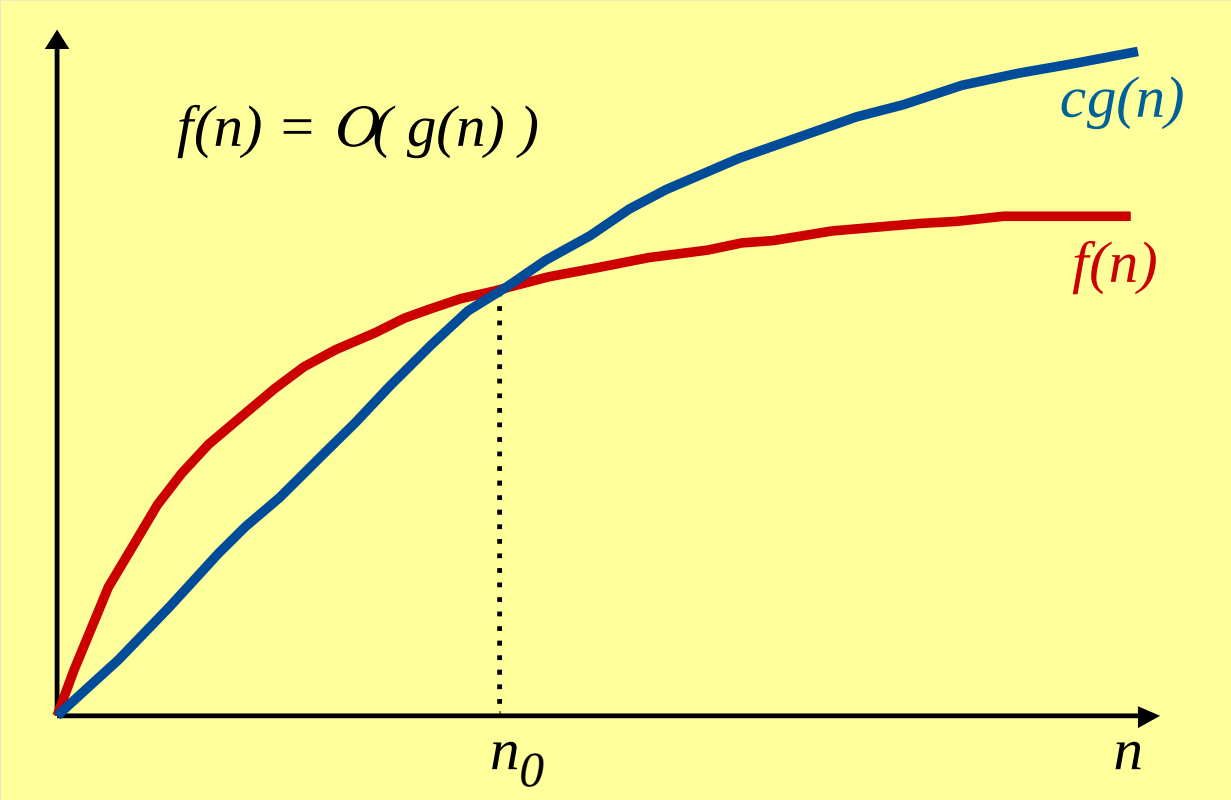
- Qual è la taglia in input del problema del calcolo del massimo?
- E il numero di operazioni?
- Dato un algoritmo che ha un input di taglia n, indicheremo con $T(n)$ il numero di operazioni che l'algoritmo esegue sull'input

Notazione Asintotica

- Complessità temporale e spaziale saranno espresse in notazione asintotica rispetto alla dimensione dell'input
- La notazione asintotica è un'astrazione utile per descrivere l'ordine di grandezza di una funzione ignorando i dettagli non influenti, come costanti moltiplicative e termini di ordine inferiore
- Ai fini dell'analisi asintotica, sarà sufficiente considerare le cosiddette operazioni dominanti, ovvero quelle che nel caso peggiore vengono eseguite più spesso

Notazione O

- O(g(n)) = {f(n) : esistono le costanti c>0 ed $n_0 \geq 0$ tali che $0 \leq f(n) \leq c*g(n)$ per ogni $n \geq n_0$ }



Notazione O

- Dato che O(g(n)) descrive un insieme di funzioni, è corretto scrivere

$$T(n) \in O(g(n))$$

  Per dire che il tempo di esecuzione T(n) di un algoritmo appartiene alla classe O(g(n)). Talvolta si abusa scrivendo T(n) = O(g(n)).

- Sia $T(n) = 2n^2 + 3n$, qual è la funzione g(n) per cui T(n) = O(g(n))?
- Se T(n) è il tempo di esecuzione del programma che calcola il minimo, T(n) = O(?) ?

Legame con il concetto di limite

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \quad \Rightarrow \quad f(n) = O(g(n))$$

$$f(n) = O(g(n)) \quad \not\Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) = O(g(n)) \quad \Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} \text{ (se esiste) } < \infty$$

Esempi

Sia $T(n) = 2n^2 + 3n$

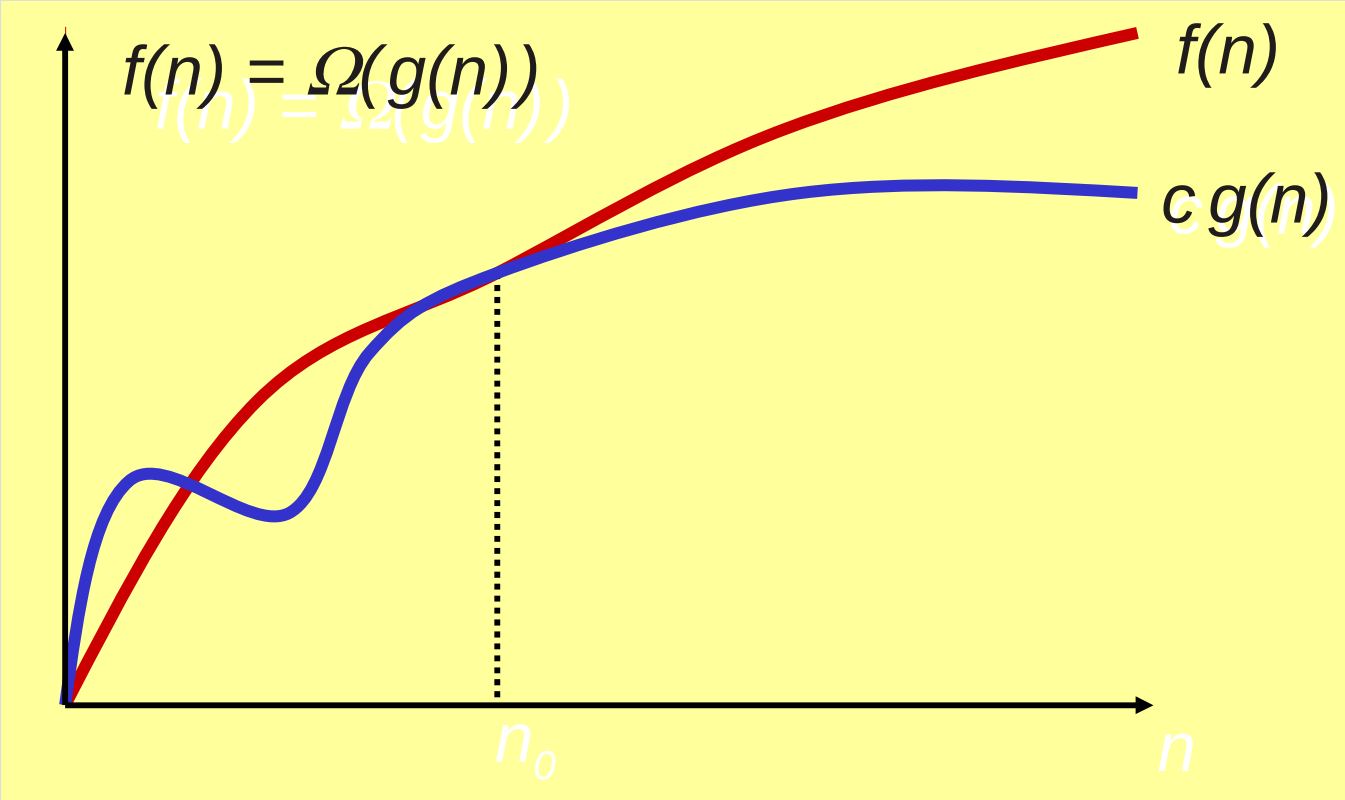
$f(n) = O(n^3)$ $(c=1, n_0=3)$

$f(n) = O(2^n)$ $(c=4, n_0=3)$

Invece, $f(n) \neq O(n)$

Notazione Ω

$f(n) = \Omega(g(n))$ se $\exists$ due costanti $c>0$ e $n_0 \geq 0$ tali che $f(n) \geq c\, g(n)$ per ogni $n \geq n_0$



Notazione Ω

Sia $f(n) = 2n^2 - 3n$; $f(n) = \Omega(n^2)$.

per $n \geq 3$ (quindi $n_0=3$)

scegliendo $c = 1$

avremo che $2n^2 - 3n \geq 1n^2$ per $n \geq n_0=3$.

$f(n) = \Omega(n)$ $(c=1, n_0=2)$

$f(n) = \Omega(\log n)$ $(c=1, n_0=2)$

Invece, $f(n) \neq \Omega(n^3)$

Legame con il concetto di limite

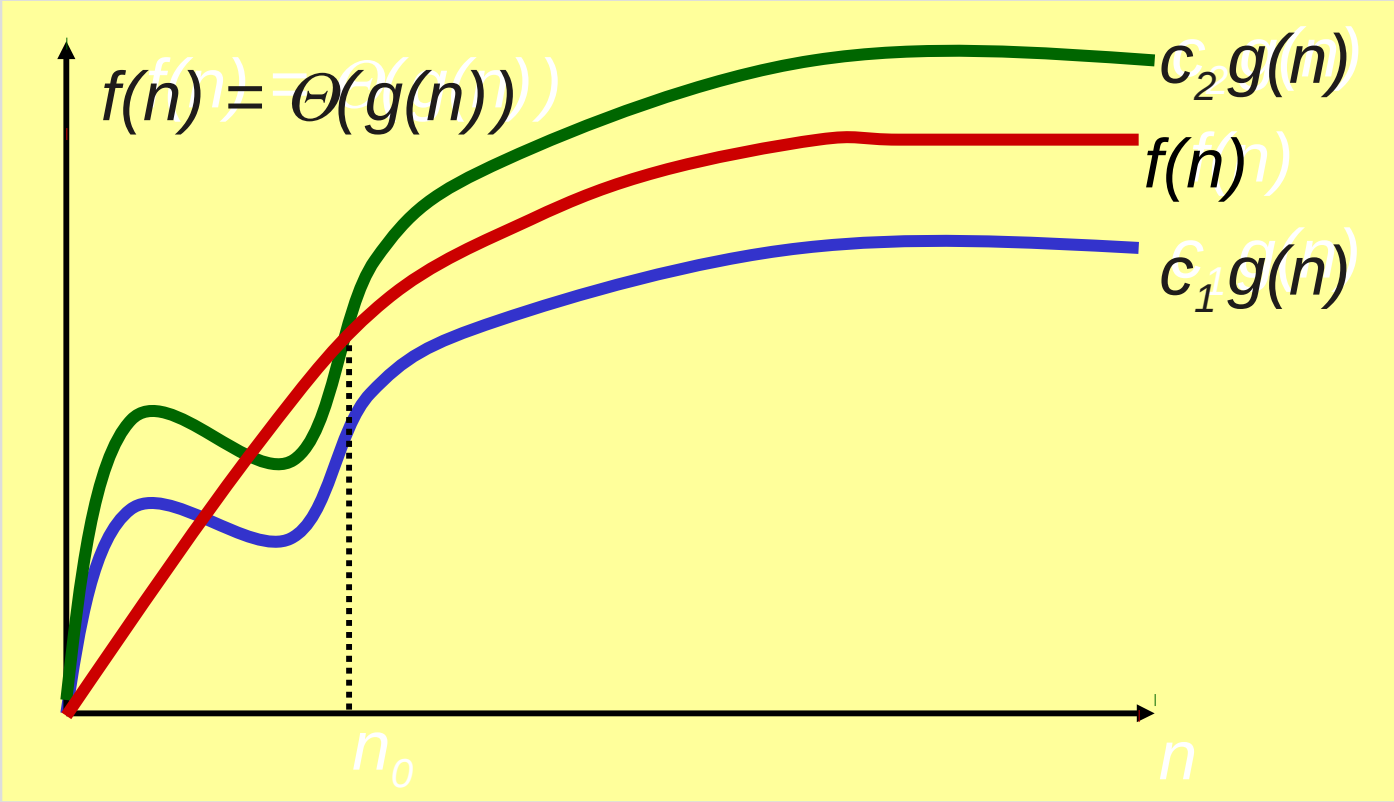
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \quad \Rightarrow \quad f(n) = \Omega(g(n))$$

$$f(n) = \Omega(g(n)) \quad \not\Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

$$f(n) = \Omega(g(n)) \quad \Rightarrow \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} \ (\text{se esiste}) \ > 0$$

Notazione Θ

$f(n) = \Theta(g(n))$ se $\exists$ tre costanti $c_1, c_2 > 0$ e $n_0 \geq 0$ tali che $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ per ogni $n \geq n_0$



Relazioni tra O, Ω e Θ

$$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$$

$$f(n) = O(g(n)) \not\Rightarrow f(n) = \Theta(g(n))$$

$$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$$

$$f(n) = \Omega(g(n)) \not\Rightarrow f(n) = \Theta(g(n))$$

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = \Omega(g(n)) \text{ e } f(n) = O(g(n))$$

Esercizio

C1. A quale classe di complessità temporale appartengono gli algoritmi minimomassimo e minimomassimo2?

C2. Scrivere un algoritmo che ordina in maniera crescente gli elementi di un vettore e studiarne la complessità temporale

Esercizi

C3. Scrivere un script che generare una matrice quadrata di dimensione 100 con valori tra 1 e 10000 generati con distribuzione uniforme, e stampa a video il valore massimo ed il valore minimo della matrice. Calcolare la complessita' temporale.

C4. Scrivere una funzione che calcola la media di ogni riga di una matrice generata secondo una distribuzione normale con media 1 e deviazione standard 2.