Università di Salerno
Dipartimento di Ingegneria Industriale

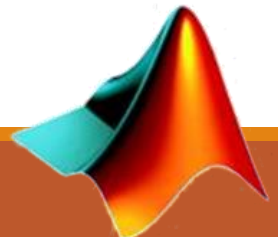
UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Programmazione in MATLAB | Gestione File | Grafici
Esercitazione Per Casa – 2 | Possibili Soluzioni

Prof. Raffaele Pizzolante

A.A. 2016/17



MATLAB

M

valori

<<mazzo_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

valori

<<punteggio_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

NOTA: Negli esercizi possono essere utilizzate funzioni viste a lezione (negli esempi), funzioni realizzate negli esercizi precedenti e/o funzioni built-in di MATLAB

- In questa esercitazione verranno utilizzate una matrice **M** ed un array riga **P**
  - La matrice **M** rappresenta **un mazzo (non necessariamente completo) di carte da gioco**
  - L'elemento **`M(indice_seme, indice_valore)`** varrà `1` se la carta con valore specificato da `indice_valore`, ed appartenente al seme specificato da `indice_seme`, risulterà presente nel mazzo, varrà 0 altrimenti
    - ***Esempio***: `M(1, 7)` → ha valore `0` e indica che la carta di valore 7 e seme 2 *non è presente* nel mazzo
    - ***Esempio***: `M(4, 2)` → ha valore `1` e indica che la carta di valore 2 e seme 4 *è presente* nel mazzo
  - L'elemento **`P(indice)`** rappresenta il punteggio della carta (a prescindere dal seme) avente valore specificato da `indice`
    - ***Esempio***: `P(2)` → indica che una carta con valore 2 (colonna 2), ottiene un punteggio pari a `3` punti
- Sia l'array riga che la matrice contengono esclusivamente dati numerici (evidenziati in arancio nell'esempio)
- Link Script Matrici Esempio
  - http://www.di.unisa.it/dottorandi/pizzolante/FI_201617/materiale/matlab/aggiuntivo/Lezione_25_EsercitazioneCasa_ScriptMatrici.txt

M *valori*

<<mazzo_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P *valori*

<<punteggio_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 1

Scrivere una funzione chiamata `informazioni_mazzo`, che prenda come argomenti di input la matrice `M` (*mazzo_carte*) e l'array riga `P` (*punteggio_carte*), e restituisca due argomenti di output: il numero di carte contenute nel mazzo ed il punteggio totale relativo a tutte le carte contenute nel mazzo

- ***Esempio:*** `[nc, pc] = informazioni_mazzo(M, P)` → restituisce `nc = 20` e `pc = 84`
  - **OSSERVAZIONI**
    - Il valore `20` si riferisce al numero di carte all'interno del mazzo
    - Il valore `84` si riferisce al punteggio totale relativo a tutte le carte contenute nel mazzo

Possibile Soluzione 1/2

```
function [ num_carte_mazzo, punteggio_carte ] = informazioni_mazzo(M, P)
    num_carte_mazzo = sum(sum(M));

    punteggio_carte = sum(P * M');
end
```

M

<<mazzo_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

<<punteggio_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 1

Scrivere una funzione chiamata `informazioni_mazzo`, che prenda come argomenti di input la matrice `M` (*mazzo_carte*) e l'array riga `P` (*punteggio_carte*), e restituisca due argomenti di output: il numero di carte contenute nel mazzo ed il punteggio totale relativo a tutte le carte contenute nel mazzo

- ***Esempio:*** `[nc, pc] = informazioni_mazzo(M, P)` → restituisce `nc = 20` e `pc = 84`
  - **OSSERVAZIONI**
    - Il valore `20` si riferisce al numero di carte all'interno del mazzo
    - Il valore `84` si riferisce al punteggio totale relativo a tutte le carte contenute nel mazzo

Possibile Soluzione 2/2

```
function [ num_carte_mazzo, punteggio_carte ] = informazioni_mazzo(M, P)
    num_carte_mazzo = sum(sum(M));
    [num_semi, num_valori] = size(M);
    somma = 0;

    for indice_seme = 1:num_semi
        somma = somma + sum(M(indice_seme, :) .* P);
    end

    punteggio_carte = somma;
end
```

M

<<mazzo_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

<<punteggio_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 2

Scrivere una funzione chiamata `seme_massimo`, che prenda come argomenti di input: la matrice `M` (*mazzo_carte*) e l'array riga `P` (*punteggio_carte*), e restituisca due argomenti di output: l'indice del seme di cui sono presenti più carte all'interno del mazzo, e l'indice del seme le cui carte totalizzano il punteggio massimo all'interno del mazzo

- ***Esempio***: `[i1, i2] = seme_massimo(M, P)` → restituisce `i1 = 2` e `i2 = 2`
  - **OSSERVAZIONI**
    - Il valore `2` (di `i1`) si riferisce al seme `2`, che ha il maggior numero di carte nel mazzo di carte
    - Il valore `2` (di `i2`) si riferisce al seme `2`, le cui carte all'interno del mazzo totalizzano il punteggio massimo
    - In questo caso, `i1` ed `i2` sono uguali, ma potrebbe non essere sempre così

Possibile Soluzione

```
function [i1, i2] = seme_massimo(M, P)
    [valore1, i1] = max(sum(M, 2));

    [num_semi, num_valori] = size(M);

    for indice_seme = 1:num_semi
        punteggio_seme(indice_seme) = sum(M(indice_seme, :) .* P);
    end

    [valore2, i2] = max(punteggio_seme);
end
```

M

valori

<<mazzo_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

valori

<<punteggio_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 3

Scrivere una funzione chiamata `carta_non_presente`, che prenda come argomenti di input: la matrice `M` (*mazzo_carte*), un intero `indice_seme` ed un intero `indice_valore`, e restituisca come argomento di output

- `1`, se la carta non è presente nel mazzo
- `0`, altrimenti

- ***Esempio 1:*** `carta_non_presente(M, 2, 3)` → restituisce `0`
- ***Esempio 2:*** `carta_non_presente(M, 3, 9)` → restituisce `1`

Possibile Soluzione 1/3

```
function [ non_presente ] = carta_non_presente(M, indice_seme, indice_valore)
    non_presente = ~M(indice_seme, indice_valore);
end
```

Possibile Soluzione 2/3

```
function [ non_presente ] = carta_non_presente(M, indice_seme, indice_valore)
    non_presente = 1 - M(indice_seme, indice_valore);
end
```

M

<<mazzo_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

<<punteggio_carte.txt>>	valori 1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 3

Scrivere una funzione chiamata `carta_non_presente`, che prenda come argomenti di input: la matrice `M` (*mazzo_carte*), un intero `indice_seme` ed un intero `indice_valore`, e restituisca come argomento di output

- `1`, se la carta non è presente nel mazzo
- `0`, altrimenti

- ***Esempio 1:*** `carta_non_presente(M, 2, 3)` → restituisce `0`
- ***Esempio 2:*** `carta_non_presente(M, 3, 9)` → restituisce `1`

Possibile Soluzione 3/3

```
function [ non_presente ] = carta_non_presente(M, indice_seme, indice_valore)
    if M(indice_seme, indice_valore) == 1
        non_presente = 0;
    else
        non_presente = 1;
    end
end
```

M *valori*

<<mazzo_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P *valori*

<<punteggio_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 4

Scrivere una funzione chiamata `carta_punteggio_massimo`, che prenda come argomenti di input: l'array riga `P` (*punteggio_carte*), e restituisca come argomento di output l'indice del valore della carta che ottiene il punteggio massimo

- ***Esempio:*** `carta_punteggio_massimo(P)` → restituisce `10`

Possibile Soluzione

```
function [ indice_carta ] = carta_punteggio_massimo(P)
    [valore, indice_carta] = max(P);
end
```

M

<<mazzo_carte.txt>>	valori									
	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P

<<punteggio_carte.txt>>	valori									
	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 5

Scrivere un M-File Script chiamato `carte_script.m` che effettui le seguenti operazioni

1. Importi la matrice `M` dal file `mazzo_carte.txt`
2. Importi la matrice `P` dal file `punteggio_carte.txt`
3. Invochi la funzione dell'Esercizio 1 (chiamata `informazioni_mazzo`) con gli argomenti di input: `M` e `P`, e mostri a video i due output prodotti dalla funzione stessa
4. Generi un grafico a barre con le seguenti caratteristiche
    1. Asse *X* → Rappresenta gli indici dei semi
    2. Asse *Y* → Ad ogni punto di *Y*, rappresenti il numero di carte, all'interno del mazzo, in base al seme specificato sull'asse *X*

NOTA: I file `mazzo_carte.txt` e `punteggio_carte.txt` (mostrati in seguito) contengono solo dati numerici. È utilizzato il separatore virgola (**,**) per separare le colonne (***suggerimento:*** utilizzare la funzione `importdata`). Si assuma che i file siano memorizzati all'interno della **Current Directory**

M *valori*

<<mazzo_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Seme 1 (indice 1)	1	0	0	1	1	0	0	1	1	0
Seme 2 (indice 2)	1	1	1	0	0	0	1	0	1	1
Seme 3 (indice 3)	0	0	0	1	1	0	0	1	0	1
Seme 4 (indice 4)	0	1	0	1	0	1	0	0	1	1

P *valori*

<<punteggio_carte.txt>>	1	2	3	4	5	6	7	8	9	10
Punteggio	5	3	2	4	7	3	3	2	1	9

Esercizio 5

Possibile Soluzione (contenuto di `carte_script.m`)

```
M = importdata('mazzo_carte.txt');
P = importdata('punteggio_carte.txt');
[nc, np] = informazioni_mazzo(M, P)

[num_semi, num_valori] = size(M);
x = 1:num_semi;
y = sum(M, 2);

bar(x, y);
```

Esempio Esercizio 5