Informatica B
2017-2018

Esercitazione IV

Funzioni

Alessandro A. Nacci
alessandro.nacci@polimi.it - www.alessandronacci.it

Esercizio 1

Nel file temperature.mat sono contenute le misurazioni di temperatura rilevate da diverse stazioni di misurazione nell'arco di una giornata. Le misurazioni sono raccolte in una matrice in cui ogni riga rappresenta una stazione e ogni colonna una misurazione (le misurazioni vengono raccolte con una frequenza di due misurazioni ogni ora).

Scrivere 5 funzioni che presi in ingresso i valori di misurazione (quindi una matrice) rispondano alle seguenti domande:

1. Quale é la stazione più calda in ogni momento della giornata?
2. Quale é la massima temperatura rilevata da ognuna delle stazioni? E quella minore?
3. Quale é stata la stazione che ha misurato la maggior escursione termica?
4. Quali sono le massime temperature per ogni stazione del giorno (6:00-18:59)? E della notte (19:00-05:59)?
5. Quali stazioni hanno avuto temperature per almeno metà della giornata maggiori di una soglia data dall'utente?

Esercizio 1

Nel file temperature.mat sono contenute le misurazioni di temperatura rilevate da diverse stazioni di misurazione nell'arco di una giornata. Le misurazioni sono raccolte in una matrice in cui ogni riga rappresenta una stazione e ogni colonna una misurazione (le misurazioni vengono raccolte con una frequenza di due misurazioni ogni ora).

	T(h0.5)	T(h1)	T(h1.5)	T(h2)	T(h2.5)	T(h3)	T(h3.5)
S1							
S2							
S3							
S4							

temperature.mat

Esercizio 1 - Funzione 1

- Quale é la stazione più calda in ogni momento della giornata?

```
function [ maxTempStation ] = es1Fun1( temperature )

    [values, indexes] = max(temperature);
    maxTempStation = indexes;

end
```

Documentazione

```
octave:6> help max
'max' is a built-in function from the file libinterp/corefcn/max.cc

 -- Built-in Function: max (X)
 -- Built-in Function: max (X, [], DIM)
 -- Built-in Function: [W, IW] = max (X)
 -- Built-in Function: max (X, Y)
     Find maximum values in the array X.

     For a vector argument, return the maximum value.  For a matrix
     argument, return a row vector with the maximum value of each
     column.

If called with one input and two output arguments, 'max' also
     returns the first index of the maximum value(s).  Thus,

          [x, ix] = max ([1, 3, 5, 2, 5])
              =>  x = 5
                  ix = 3
```


Esempio

```
octave:11> a = [3 4 5; 8 2 3; 9 12 13]
a =

    3    4    5
    8    2    3
    9   12   13

octave:13> [v,i] = max(a)
v =

    9   12   13

i =

   3   3   3
```

Esercizio 1 - Funzione 2

- Quale é la massima temperatura rilevata da ognuna delle stazioni? E quella minore?

```
function [ minTemps, maxTemps ] = es1Fun2( temperature )

    minTemps = min(temperature, [], 2);
    maxTemps = max(temperature, [], 2);

end
```

Documentazione

```
M = max(A)example
M = max(A,[],dim)example
[M,I] = max(___)
-------------------------------------------------------------
M = max(A) returns the largest elements of A.
• If A is a vector, then max(A) returns the largest element of A.
• If A is a matrix, then max(A) is a row vector containing the maximum
  value of each column.
• If A is a multidimensional array, then max(A) operates along the first
  array dimension whose size does not equal 1, treating the elements as
  vectors. The size of this dimension becomes 1 while the sizes of all
  other dimensions remain the same. If A is an empty array with first
  dimension 0, then max(A) returns an empty array with the same size as A.
-------------------------------------------------------------
M = max(A,[],dim) returns the largest elements along dimension dim. For
example, if A is a matrix, then max(A,[],2) is a column vector containing
the maximum value of each row.
```

Esempio

```
octave:23> a
a =

    3    4    5
    8    2    3
    9   12   13

octave:24> max(a, [], 2)
ans =

    5
    8
   13
```

Esercizio 1 - Funzione 3

- Quale é stata la stazione che ha misurato la maggior escursione termica?

```
function [ maxDiff ] = es1Fun3( temperature )
    [minTemps maxTemps] = es4Fun2(temperature);
    diffs = maxTemps - minTemps;
    maxDiff = find(diffs == max(diffs));

end
```

Esercizio 1 - Funzione 4

- Quali sono le massime temperature per ogni stazione del giorno (6:00-18:59)? E della notte (19:00-05:59)?

```
function [ maxDay, maxNight ] = es1Fun4( temperature )

    dayTemp = temperature(:, 12:35);
    nightTemp = temperature(:, [1:11, 36:48]);

    maxDay = max(dayTemp, [], 2);
    maxNight = max(nightTemp, [], 2);

end
```

Esercizio 1 - Funzione 5

- Quali stazioni hanno avuto temperature per almeno metà della giornata maggiori di una soglia data dall'utente?

Esercizio 1 - Funzione 5

- Quali stazioni hanno avuto temperature per almeno metà della giornata maggiori di una soglia data dall'utente?

```
function [ result ] = es1Fun5( temperature, soglia )

    count = sum(temperature>soglia, 2);
    result = find(count >= size(temperature,2)/2);

end
```

Documentazione

```
S = sum(A)example
S = sum(A,dim)example
S = sum(___,outtype)example
S = sum(___,nanflag)example

Description
-----------------------------------------
S = sum(A) returns the sum of the elements of A along the first array
dimension whose size does not equal 1.
• If A is a vector, then sum(A) returns the sum of the elements.
• If A is a matrix, then sum(A) returns a row vector containing the sum of
  each column.
• If A is a multidimensional array, then sum(A) operates along the first
  array dimension whose size does not equal 1, treating the elements as
  vectors. This dimension becomes 1 while the sizes of all other
  dimensions remain the same.
-----------------------------------------
S = sum(A,dim) returns the sum along dimension dim. For example, if A is a
matrix, then sum(A,2) is a column vector containing the sum of each row.
```

Esempio

```
octave:25> a
a =

    3    4    5
    8    2    3
    9   12   13

octave:26> sum(a,2)
ans =

   12
   13
   34
```

```
octave:27> a > 3
ans =

   0   1   1
   1   0   0
   1   1   1

octave:28>
sum(a > 3,2)

ans =

   2
   1
   3
```

Esercizio 1 - Script

```
load('temperature.mat');

q1 = es1Fun1(temperature);

[q2Min, q2Max] = es1Fun2(temperature);

q3 = es1Fun3(temperature);

[q4Day, q4Night] = es1Fun4(temperature);

soglia = input('Inserisci una soglia : ');
q5 = es1Fun5(temperature, soglia);
```

Esercizio 2

Scrivere un programma che letti da tastiera due vettori numerici scambi gli elementi di indice pari del primo vettore con quelli di indice dispari del secondo. Si continui a chiedere all'utente di inserire dei vettori fino a quando questi non soddisfano le condizioni necessarie per risolvere l'esercizio.

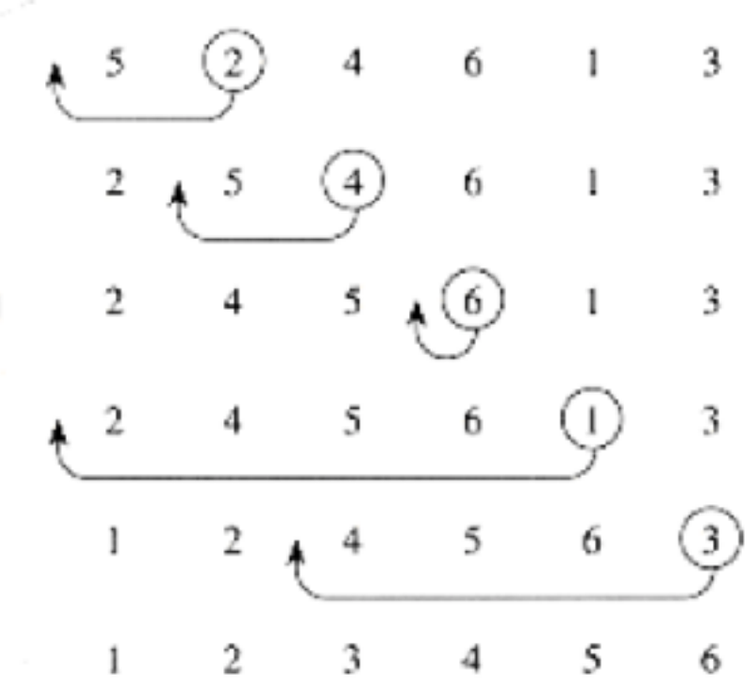
```
pari = false;
vettore = false;
numerico = false

while(pari == false || vettore == false || numerico == false)
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il second vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
    numerico = isnumeric(v1) & isnumeric(v2);
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;

v1
v2
```

Esercizio 3

In Matlab esiste è possibile ordinare un vettore utilizzando la funzione *sort*.
Implementare una funzione *mysort* che riceve in ingresso un vettore e lo restituisce ordinato. Verificare quindi la correttezza della proria implementazione confrontando il risultato con la sort messa a disposizione da Matlab.

Suggerimento 1
Utilizzare l'algoritmo *Insertion Sort*:

Suggerimento 2
Può essere utile implementare una funzione di supporto inserisci che inserisce in un array già ordinato un nuovo elemento (mantenendo corretto l'ordinamento del vettore).



Esercizio 3

In Matlab esiste è possibile ordinare un vettore utilizzando la funzione *sort*. Implementare una funzione *mysort* che riceve in ingresso un vettore e lo restituisce ordinato. Verificare quindi la correttezza della proria implementazione confrontando il risultato con la sort messa a disposizione da Matlab.

```
function sv = mysort(v)

sv = v(1);
for i=2:length(v)
    sv = inserisci(sv,v(i));
end
```

```
function v2 = inserisci(v,e)
v2 = [v(v<=e) e v(v>e)];
```

```
v = rand(1,5000);

sorted1 = sort(v);
sorted2 = mysort(v);

if (all(sorted1==sorted2))
    disp('OK');
else
    disp('NO');
end
```

Esercizio 4

Scrivere un programma che chiede all'utente un numero intero positivo (se non è intero positivo, continua a chiederne un altro). Dato il numero, determina se il numero è perfetto. Un numero è perfetto se è uguale alla somma dei suoi divisori propri (i divisori propri sono gli interi tali per cui il numero diviso il divisore da resto zero, escluso il numero stesso)

Se il numero non è perfetto, viene richiesto un secondo numero intero positivo e il programma determina se i numeri sono amici (o amicabili). Due numeri sono amici se ciascuno è uguale alla somma dei divisori propri dell'altro).

Scrivere le seguenti funzioni:

- leggiNumeroInteroPositivo (legge intero positivo)
- divisoriPropri (ritorna array divisori propri)
- amici (determina se due numeri sono amici)

Esercizio 4 - leggiNumeroInteroPositivo

```
function n = leggiNumeroInteroPositivo()
    n = input('Comunicami un numero intero positivo: ');
    while (n<=0 || round(n)~=n)
        disp('Il numero è negativo o non intero')
        n = input('Comunicami un numero intero positivo: ');
    end
end
```

Esercizio 4 - divisoriPropri

```
function div = divisoriPropri(n)
candidati = 1:n/2;
div = candidati(mod(n,candidati)==0);
end
```

Esercizio 4 - amici

```
function r = amici(a,b)

    if (a==sum(divisoriPropri(b)) && b==sum(divisoriPropri(a)))
        r = true;
    else
        r = false;
    end

end
```

Esercizio 4 - Script

```
a = leggiNumeroInteroPositivo();
if a==sum(divisoriPropri(a));
    disp([num2str(a) ' è perfetto'])
else
  disp([num2str(a) ' non è perfetto'])
  b = leggiNumeroInteroPositivo();
  if amici(a,b)==1
      disp([num2str(a) ' e ' num2str(b) ' sono amici '])
  else
      disp([num2str(a) ' e ' num2str(b) ' non sono amici '])
  end
end
```

Esercizio 5 - Esercizio per casa

Scrivere uno script Matlab per costruire una tabella delle probabilità di riuscire a conquistare un territorio in un gioco del Risiko, sulla base del numero di carri armati dell'attaccante e del difensore (nella tabella considerare i casi in cui i carri armati sono compresi fra 1 e 10).

Per costruire la tabella, si suggerisce di usare il metodo Montecarlo: simulare N tentativi di attacco e calcolare empiricamente la probabilità di successo sulla base di questi N tentativi.

Si suggerisce di implementare le seguenti funzioni:

- *simula*, che simula un attacco: riceve in ingresso il numero di carri armati dell'attacante e del difensore e restituisce true se l'attacco ha successo, false altrimenti
- *lancio*, che simula un singolo lancio di dati durante un attacco: riceve in ingresso il numero di dadi usati dall'attaccante, il numero di dadi usati dal difensore e restituisce il numero di carri armati persi dall'attaccante e dal difensore nel lancio.

Tutte il materiale sarà disponibile sul mio sito internet!

www.alessandronacci.it