13 Ricorsione con MATLAB

13.1 Esercizi

Esercizio 13.1

Scrivere una funzione che verifichi iterativamente se una stringa è palindroma. Scrivere poi una funzione che implementi la stessa funzionalità in modo ricorsivo.

Si stampi a schermo il passo iterativo e ricorsivo che viene seguito dalla soluzione proposta.

Esercizio 13.2

Implementare una funzione iterativa (e poi una sua versione ricorsiva) per tradurre i caratteri di una stringa da minuscoli a maiuscoli. Assumere che la funzione riceva in ingresso una stringa di caratteri minuscoli.

Suggerimento: la traduzione viene effettuata semplicemente sottraendo 32 al carattere da tradurre, e applicando `char()`. Ad esempio:

```
1 trad = char('a' - 32)
```

stampa a video il carattere `A`.

Esercizio 13.3

Si implementi in `MATLAB` una funzione che, preso un intero positivo `n` ne restituisca i coefficienti binomiali di ordine `n`.

Suggerimento: la costruzione del triangolo di tartaglia restituisce come risultato i coefficienti binomiali. Esso viene definito in maniera ricorsiva come:

- Il primo elemento è il numero uno;
- La riga successiva viene costruita mettendo degli 1 agli estremi e sommando a coppie i numeri vicini della riga precedente.

Esercizio 13.4

Si implementi in MATLAB uno script che, preso un array A, calcola il massimo comun divisore (MCD) fra tutti gli elementi di A. Ad esempio: sull'array `[9 18 6 27 30 42]` restituisce 3.

Si ricorda che, dati tre numeri a, b, c, l'MCD tra a, b e c è uguale all'MCD tra c e l'MCD tra a e b.

Esercizio 13.5

Si scriva una funzione `calcolaZeri(f, a, b, tolX, tolY)` che prende in ingresso:

- un function handle f
- un estremo inferiore di un intervallo a
- un estremo superiore di un intervallo b
- una tolleranza per la variabile x chiamata $tolX$
- una tolleranza per la variabile y chiamata $tolY$

che calcoli uno zero della funzione f, se esso esiste, in maniera ricorsiva tramite il metodo di bisezione. La funzione si deve fermare se l'intervallo è diventato più corto di $tolX$ oppure se la funzione ha valore assoluto in un punto minore di $tolY$.

Si chiami la funzione sull'intervallo $[-1; 3]$ per la funzione $y = x^2 - 4$ con tolleranze 0.0001 per entrambe le variabili.

Esercizio 13.6

Si implementi in MATLAB uno script che, preso un array v, lo ordini ricorsivamente utilizzando l'algoritmo Mergesor, definito come segue:

- Se la sequenza da ordinare ha lunghezza 1, è già ordinata
- Se la sequenze è lunga 2 o più, la sequenza viene divisa in due metà;
- Ognuna di queste sottosequenze viene ordinata, applicando ricorsivamente l'algoritmo;
- Le due sottosequenze ordinate vengono fuse. Per fare questo, si estrae ripetutamente il minimo delle due sottosequenze e lo si pone nella sequenza in uscita, che risulterà ordinata.

Soluzioni

Soluzione dell'esercizio 13.1

```
1
2
3  % parola palindroma
4  R = 'abbAbba';
5
6  % parola non palindroma
7  P = 'abbiibaia';
8
9  % invocazione funzione iterativa
10 fprintf('
      ----------------------------------------------------\n');
11 palindroma_iterativa(R);
12 fprintf('
      ----------------------------------------------------\n');
13 palindroma_iterativa(P);
14
15 % invocazione funzione ricorsiva
16 fprintf('
      ----------------------------------------------------\n')
      ;
17 palindroma_ricorsiva(R, 0);
18 fprintf('
      ----------------------------------------------------\n')
      ;
19 palindroma_ricorsiva(P, 0);
```

```
1  function [res] = palindroma_iterativa(parola)
2  res = 1;
3
4  for ii = 1:floor(length(parola)/2)
5      %Stampa a schermo del passo iterativo
6      fprintf('passo iterativo = %d: %c =?= %c\n', ii, parola(ii)
          , parola(end - ii + 1));
7
8      %Controllo di caratteri nella parola
9      if parola(ii) ~= parola(end - ii + 1)
10         res = 0;
11         return;
12     end
```

```
end
```

```
function [res] = palindroma_ricorsiva(parola, passo)

    %Stampa inizio passo ricorsivo
    for ii = 1:passo
        fprintf('\t');
    end
    fprintf('|--> inizio palindroma_ricorsiva(%s, %d)\n',
       parola, passo);

    if length(parola) < 2
        res = 1;
    else
        % controllo se gli estremi sono uguali
        %
        if parola(1) == parola(end)
            % Passo ricorsivo
            res = palindroma_ricorsiva(parola(2:end-1), passo
               +1);
            % da qui, l'esecuzione e` bloccata fino a che la
               funzione
            % precedente non ritorna
        else
            res = 0;
        end
    end

    %Stampa fine passo ricorsivo
    for ii = 1:passo
        fprintf('\t');
    end
    fprintf('|--> fine   palindroma_ricorsiva(%s, %d)\n',
       parola, passo);
```

Soluzione dell'esercizio 13.2

```
clc
clear
close all

s = 'ciaocomestai';

```

```
fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_iterativa(s);
disp(S)
toc

fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_ricorsiva1(s);
disp(S);
toc

fprintf('
   ------------------------------------------------------\n');
tic
S = maiuscola_ricorsiva2(s);
disp(S);
toc
```

```
function S = maiuscola_iterativa(s)

S = s;
for ii = 1:length(s)
    S(ii) = char(s(ii) - 32);
end
```

```
function S = maiuscola_ricorsiva1(s)

% Caso base: stringa di un carattere
if length(s) == 1
    S = [char(s(1) - 32)];
else
    S = [char(s(1) - 32) maiuscola_ricorsiva1(s(2:end))];
end
```

```
function S = maiuscola_ricorsiva2(s)

% Caso base: stringa di un carattere
if length(s) == 1
    S = char(s(1) - 32);
else
```

```
    m = round(length(s) / 2);
    S = [maiuscola_ricorsiva2(s(1:m)) maiuscola_ricorsiva2(s((m
       +1):end))];
end
```

Soluzione dell'esercizio 13.3

```
clear
clc
close all;

triangoloTartaglia(10)
```

```
function T = triangoloTartaglia(n)

% Caso base
if n == 0
    T = 1;
% Chiamata ricorsiva
else
    T = triangoloTartaglia(n-1);

    % Soluzione alla C :creo un vettore I con la somma delle
       coppie degli elementi di T al passo n-1
    %     I = [];
    %     for ii = 2 : numel(T)
    %         I = [I, T(ii- 1)+ T(ii)];
    %     end
    %     T=[1, I, 1];

    % Soluzione alla matlab, shifto T
    T = [1,T(1:end-1)+T(2:end),1];
end
disp(T)
```

Soluzione dell'esercizio 13.4

```
clear
clc
close all

res = mcd(15, 18, 0)
res = mcd_array([12 15 18])
```

```
% Calcolo dell'MCD con algoritmo di Euclide
%
%     * se m = n, MCD(m,n) = m              (caso base)
%     * se m > n, MCD(m,n) = MCD(m-n, n)    (ricorsione)
%     * se m < n, MCD(m,n) = MCD(m, n-m)    (ricorsione)

function [M] = mcd(m, n, passo)
% la variabile "passo" non e` parte della soluzione.

% stampo "passi volte" il carattere TAB per "visualizzare" a
   che punto
% della ricorsione mi trovo
for ii = 1:passo
    fprintf('\t');
end
% stampo l'inizio dell'invocazione corrente
fprintf('|--> inizio esecuzione mcd: m = %d, n = %d, passo = %d
   \n', m, n, passo);

if m == n
    M = m;
else
    if m > n
        M = mcd(m-n, n, passo+1);
    else
        M = mcd(m, n-m, passo+1);
    end
end

% stampo "passi volte" il carattere TAB per "visualizzare" a
   che punto
% della ricorsione mi trovo
for ii = 1:passo
    fprintf('\t');
end
% stampo la fine dell'invocazione corrente
fprintf('|--> fine esecuzione mcd: m = %d, n = %d, passo = %d\n
   ', m, n, passo);
```

```
function res = mcd_array(v)

len_v = length(v);

```

```
5  if len_v == 1
6      res = v;
7  else
8      res = mcd(v(1),v(2),0);
9      for ii = 3:len_v
10         res = mcd(res,v(ii),0);
11     end
12 end
```

Soluzione dell'esercizio 13.5

```
1  clear
2  clc
3  close all
4
5  f = @(x)(x.^2 - 4);
6  a = -1;
7  b = 3;
8  tolX = 0.0001;
9  tolY = 0.0001;
10
11 zero = calcolaZeri(f,a,b,tolX,tolY);
12
13 x = a:0.01:b;
14 y = f(x);
15
16 plot(x,y);
17 hold on;
18 plot(zero,f(zero),'ro');
```

```
1  function x = calcolaZeri(f, a, b, tolX, tolY)
2
3  if (f(a) * f(b) > 0)
4      x = NaN;
5      return;
6  end
7
8  if (abs(f(a)) < tolY)
9      x = a;
10     return;
11 elseif (abs(f(b)) < tolY)
12     x = b;
13     return;
```

```
14 end
15
16 if ((b-a) < tolX)
17     x = (b-a) / 2;
18     return;
19 end
20
21 c = (b - a) / 2;
22 if (f(a) * f(c) < 0)
23     x = calcolaZeri(f, a, c, tolX, tolY);
24 else
25     x = calcolaZeri(f, c, b, tolX, tolY);
26 end
```

Soluzione dell'esercizio 13.6

```
1 clear
2 clc
3 close all
4
5 v = rand(100,1);
6 plot(v);
7 hold on;
8 sorted_v = merge_sort(v);
9 plot(sorted_v,'r');
```

```
1  function v = merge_sort(v)
2
3  len_v = length(v);
4  if  len_v > 1
5      %Divide
6      middle_point = round(len_v / 2);
7      v1 = merge_sort(v(1:middle_point));
8      v2 = merge_sort(v(middle_point+1:end));
9
10     %Impera
11     len_v1 = length(v1);
12     len_v2 = length(v2);
13     ii = 1;
14     jj = 1;
15     v = [];
16     while ii <= len_v1 && jj <= len_v2
17         if v1(ii) < v2(jj)
```

```
18              v = [v v1(ii)];
19              ii = ii + 1;
20          else
21              v = [v v2(jj)];
22              jj = jj + 1;
23          end
24      end
25      if ii <= len_v1
26          v = [v v1(ii:end)];
27      else
28          v = [v v2(jj:end)];
29      end
30      assert(length(v) == len_v1 + len_v2)
31  end
```