Problem Solving

- Metodo per la soluzione di un problema al computer
- Esempi di applicazione

Soluzione di un problema

1. Definire il problema e gli obiettivi
2. Identificare le informazioni a disposizione (input) e i risultati desiderati (outputs)
3. Scegliere un approccio (ipotesi, approssimazioni, principi scientifici e matematici…)
4. Scrivere il programma
5. Testare il programma (caso semplice)

Esempio 1: approssimazione lineare

Dati n coppie di valori (x_i,y_i), trovare la migliore approssimazione lineare:

$$y = m\,x + b$$

$m \equiv$ coeff.angolare,

$b \equiv$ intercetta y

Metodo di soluzione:

1. Definiamo il problema: trovare la migliore approssimazione lineare dei dati
2. Identifichiamo input ed output:

   $(x_i,y_i) \rightarrow (m,b)$
3. Algoritmo da utilizzare: minimi quadrati
4. Scriviamo il programma
5. Testiamo il programma

Metodo dei minimi quadrati:

$$m = \frac{\left(\sum_{j=1}^{N} x_j y_j\right) - \frac{1}{N}\left(\sum_{j=1}^{N} x_j\right)\left(\sum_{j=1}^{N} y_j\right)}{\left(\sum_{j=1}^{N} x_j^{\,2}\right) - \frac{1}{N}\left(\sum_{j=1}^{N} x_j\right)^2}$$

$$b = \frac{1}{N}\left(\sum_{j=1}^{N} y_j\right) - \frac{m}{N}\left(\sum_{j=1}^{N} x_j\right)$$

Flusso del programma:

1. Leggiamo I dati:
   - Quanti punti?
   - Leggiamo le coordinate
2. Calcoliamo *m* e *b*
3. Formiamo l'output:
   - Valori numerici di *m* e *b*
   - Presentazione grafica


```
% Comments on the program
…
% Get number of pairs, n_points
n_points = input('How many (x,y) pairs? ');

% Read in the input data x, y
for ii = 1:n_points
    temp = input('Enter [x y]: ');
    x(ii) = temp(1);
    y(ii) = temp(2);
end
```

Usiamo un ciclo, ma potevamo fare immettere all'utente direttamente i vettori!

$$m = \frac{\left(\sum_{j=1}^{N} x_j y_j\right) - \frac{1}{N}\left(\sum_{j=1}^{N} x_j\right)\left(\sum_{j=1}^{N} y_j\right)}{\left(\sum_{j=1}^{N} x_j^{\,2}\right) - \frac{1}{N}\left(\sum_{j=1}^{N} x_j\right)^2}$$

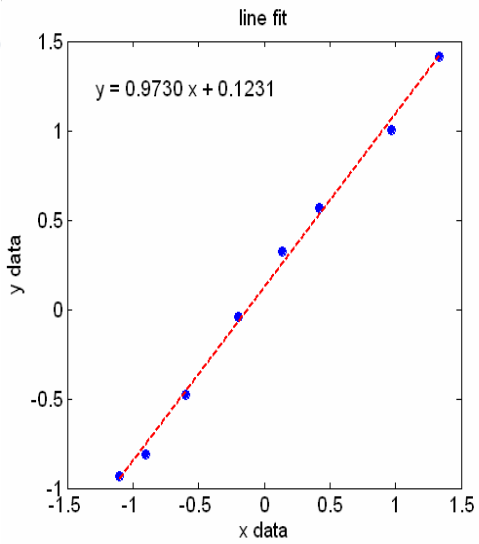
$$b = \frac{1}{N}\left(\sum_{j=1}^{N} y_j\right) - \frac{m}{N}\left(\sum_{j=1}^{N} x_j\right)$$

```
% Calculate m and b
m = ( sum(x.*y) – sum(x)*sum(y)/n_points )/( …
    sum(x.^2) – sum(x)^2/n_points );
b = ( sum(y) – m*sum(x) )/n_points;
```

```
% Plot the data
plot(x,y,'bo')
hold on
twopts = [ min(x), max(x) ];
plot( twopts, m*twopts+b ,'r-','Linewidth',2)
hold off
% Add labels, etc.
gtext([' y = ',num2str(m), ' x + ',num2str(b)])
xlabel('x data'), ylabel('y data'), title('line fit')
```


Tipico risultato

Proviamo su un caso base (solo 2 punti)



Esempio 2: ordinamento

Dati N numeri (x_1,…x_N), ordinarli dal minore al maggiore (senza usare la sort !!!)

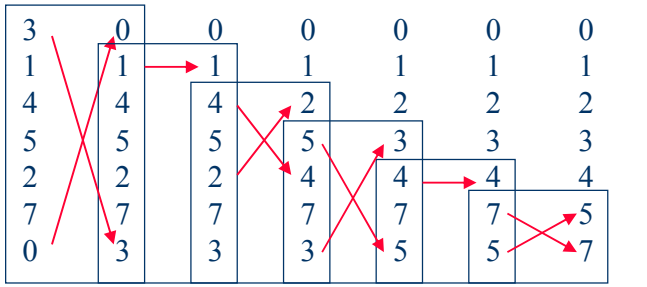
3 1 4 5 2 7 0

0 1 2 3 4 5 7

Algoritmo:

- Abbiamo N numeri, x_1…x_N:
  - Troviamo il più piccolo tra x_1…x_N e lo scambiamo con x_1
  - Troviamo il più piccolo tra x_2…x_N e lo scambiamo con x_2
  - Continuiamo per N-1 volte

start



done

Il programma utilizza cicli annidati!!!

Flusso del programma:

1. Consideriamo i dati come parametri della funzione
2. Ciclo esterno che esclude ogni volta il primo numero della lista (già a posto)
3. Ciclo interno per trovare il minimo e scambiarlo se necessario


```
function array = simpsort(array)
% function to sort from smallest to largest
for k = 1:length(array)-1
    loc = k;
    for k2 = k:length(array)
        if array(k2)<array(loc)
            loc = k2;
        end
    end
    if loc ~= k
        array([k,loc ]) = array([loc,k]);
    end
end
```

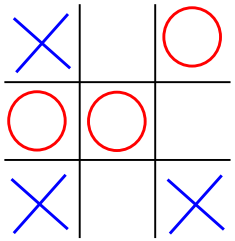
Ciclo Est.

Ciclo interno per trovare la posizione (loc) del minimo

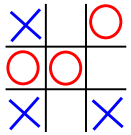
Scambio I valori, se necessario

Esempio 3: tic-tac-toe

- Giochiamo tic-tac-toe, umano contro umano:
  - Tracciamo le mosse
  - Vediamo lo schema
  - Fine del gioco



- Matrice 3x3 per lo schema. Valori:
  - Cella vuota = 0
  - X = +1
  - O = – 1
- Fine del gioco:
  - C'è un vincitore se la somma di una riga, colonna o diagonale = +3 or –3
  - Pareggio se non ci son più zeri
- Ciclo di gioco:
  - Mossa iniziale (X) + 4 turni
  - Fine se c'è un vincitore



Flusso del programma:

1. Inizializziamo lo schema
2. Mossa del primo giocatore
3. Ciclo 4 volte:
   - Mossa del secondo, verifico se vince
   - Mossa del primo, verifico se vince
4. Vedere se c'è un pareggio

Inizializzazione e prima mossa (X)

```
function tictactoe
% tic-tac-toe PFS 3/06
%
clc
board = zeros(3,3);
%
% get first move
move = input('enter player 1 move [r,c] ');
board(move(1),move(2)) = 1
%
```


Inizio ciclo, muove il secondo (O)

```
% loop until done
for turn = 1:4
    % player 2 next
    move = input('enter player 2 move [r,c] ');
    board(move(1),move(2)) = -1
    result = check(-board);
    if result == 1
        disp('player 2 wins!')
        break
    end
```

Fine del ciclo, muove il primo (X)

```
    % back to player 1
    move = input('enter player 1 move [r,c] ');
    board(move(1),move(2)) = 1
    result = check(board);
    if result == 1
        disp('player 1 wins!')
        break
    end
end
```


Prova del pareggio

```
% check for draw
if sum(abs(board(:)))==9 & result==0
    disp('nobody wins')
end
```


Come vedere una vittoria

```
function result = check(board)
% check for 3 ones in a row, column, or diag
result = sum(board);
result(4:6) = sum(board');
result(7) = board(1,1) + board(2,2) + board(3,3);
result(8) = board(1,3) + board(2,2) + board(3,1);
result = max(result)==3;
```


Tipico risultato

```
enter player 1 move [r,c] [2 2]

board =

     0     0     0
     0     1     0
     0     0     0

enter player 2 move [r,c] [1 2]

board =

     0    -1     0
     0     1     0
     0     0     0
```

(esempi)

```
enter player 1 move [r,c] [2 1]

board =

     0    -1     0
     1     1     0
     0     0     0

enter player 2 move [r,c] [1 3]

board =

     0    -1    -1
     1     1     0
     0     0     0

enter player 1 move [r,c] [2 3]

board =

     0    -1    -1
     1     1     1
     0     0     0

player 1 wins!
>>
```