

 POLITECNICO DI MILANO

Dipartimento di  
Elettronica e Informazione

# Maschere, ricorsione e struct

Matteo Ferroni  
[matteo.ferroni@polimi.it](mailto:matteo.ferroni@polimi.it)

15/01/2016



POLITECNICO  
DI MILANO



# Agenda

---

(30') Maschere di bit (ordinamento v2.0) (ex Es5)

(30') Ricerca binaria (ricorsiva)

(30') Calcolo derivata (ricorsiva)

(30') Struct (rilievi altimetrici) (ex Es6)

(30') Struct (film) (ex Es7)

(30') Vettori, vettori everywhere! (ex Es1.2)

# Da tenere sempre a mente...

---

Scorsa  
esercitazione

# Da tenere sempre a mente...

---

In MATLAB si ragiona con un paradigma diverso dal C:

# Da tenere sempre a mente...

---

Scorsa  
esercitazione

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;

# Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;
- gli **scalari** sono un caso particolare di array multidimensionale;

# Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;

# Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)

# Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)
- i **for** sono quasi sempre da **evitare**

# Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un **array**;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)
- i **for** sono quasi sempre da **evitare**
- le maschere di bit tornano molto comode quando dovete **filtrare** in qualche modo i dati;

# Esercizio: Maschere di bit (ordinamento)

---

- Si ordini un array di  $n$  elementi facendo uso delle istruzioni messe a disposizione da matlab.

Scorsa  
esercitazione

# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione


- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

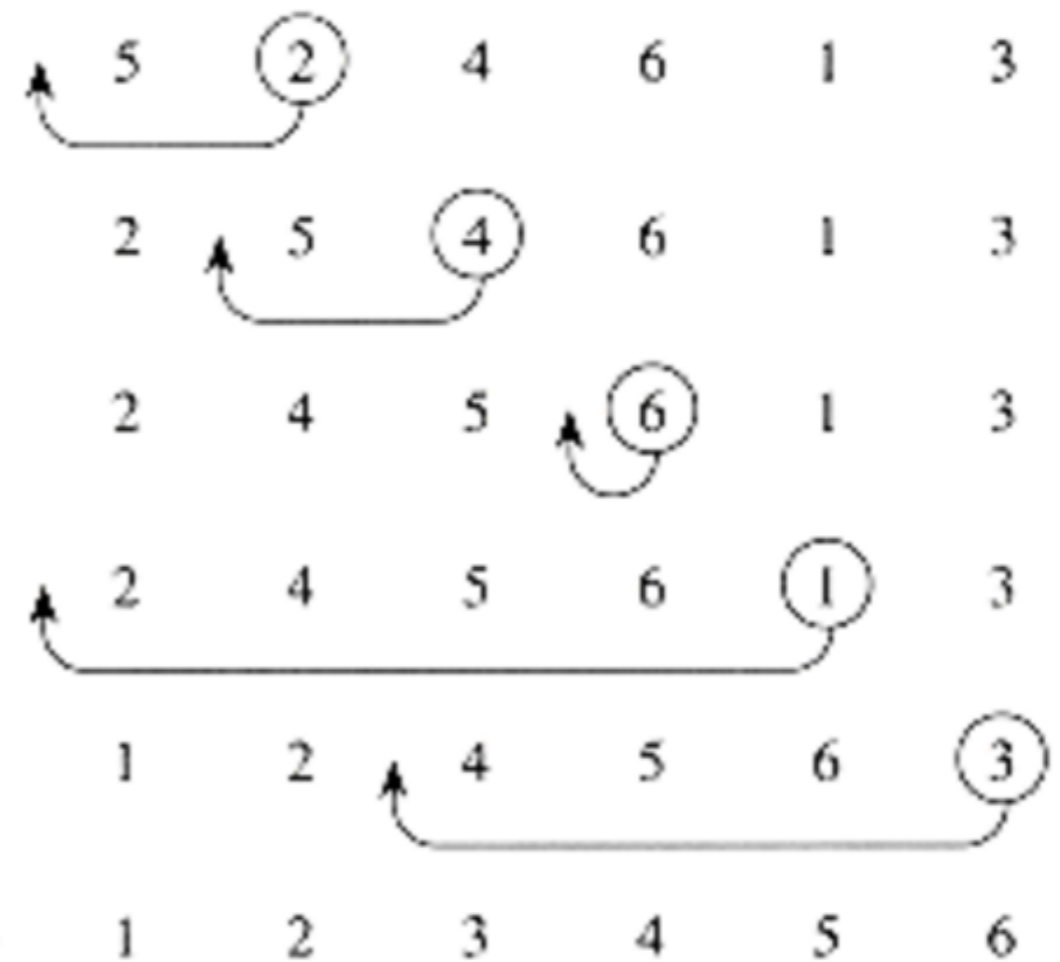
*Algoritmo di ordinamento: "Insertion sort"*  
([https://it.wikipedia.org/wiki/Insertion\\_sort](https://it.wikipedia.org/wiki/Insertion_sort))



# Esercizio: Maschere di bit (ordinamento)


Scorsa  
esercitazione

a  [5 2 4 6 1 3]

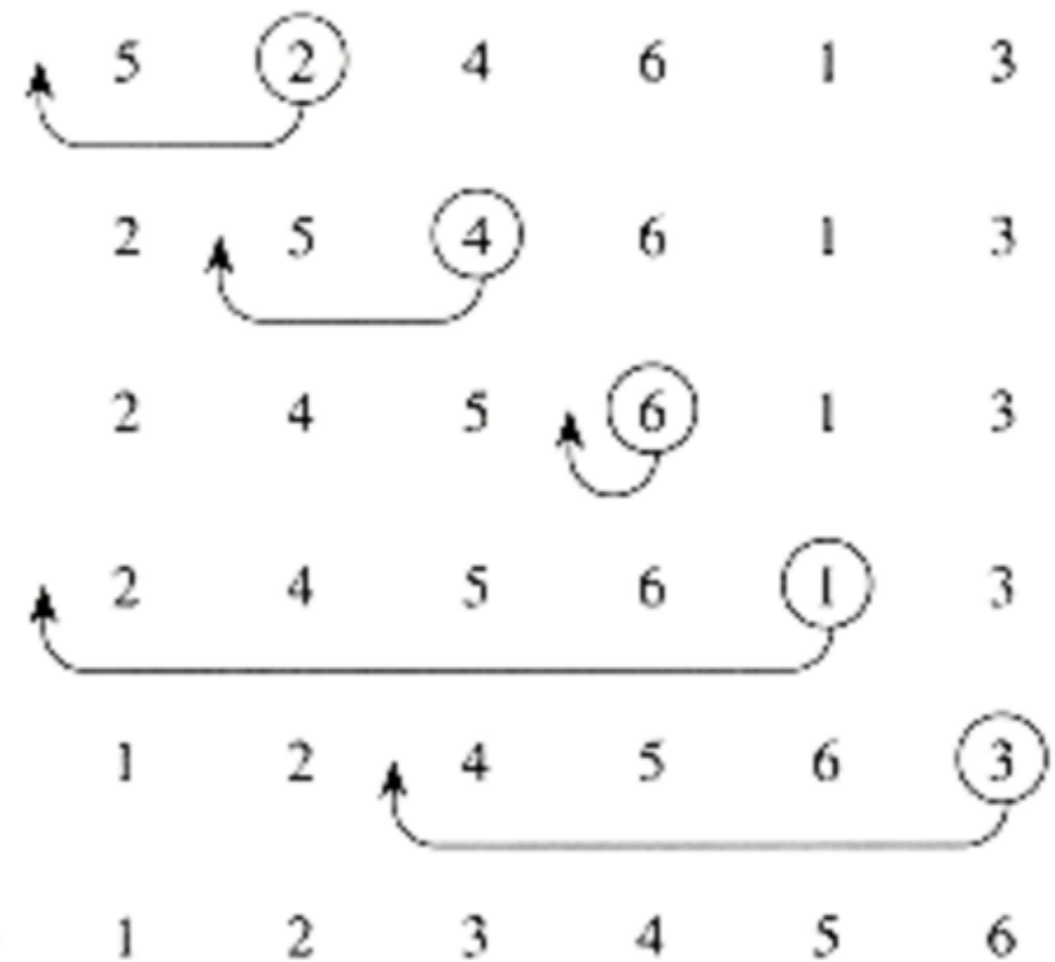


# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a  [5 2 4 6 1 3]

a\_ord = a(1);



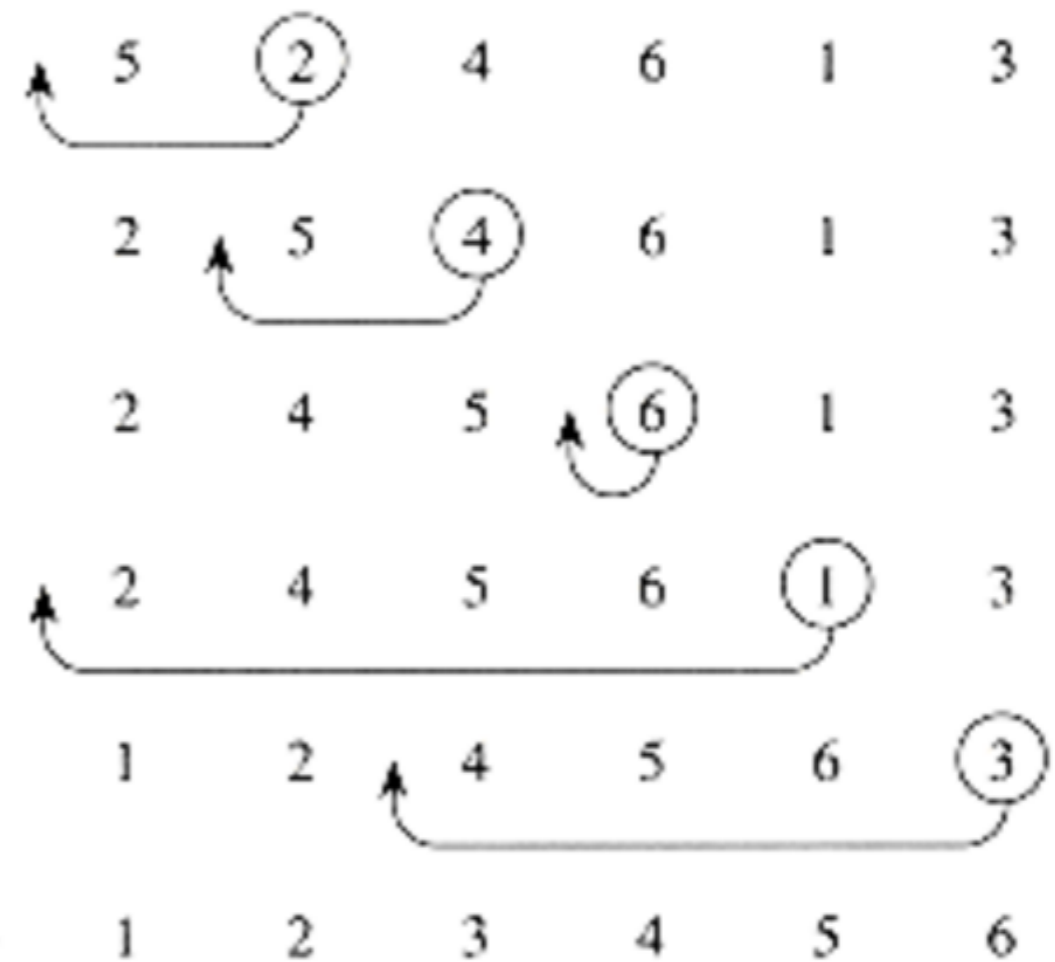
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

a\_ord = a(1);

for i=2:length(a)



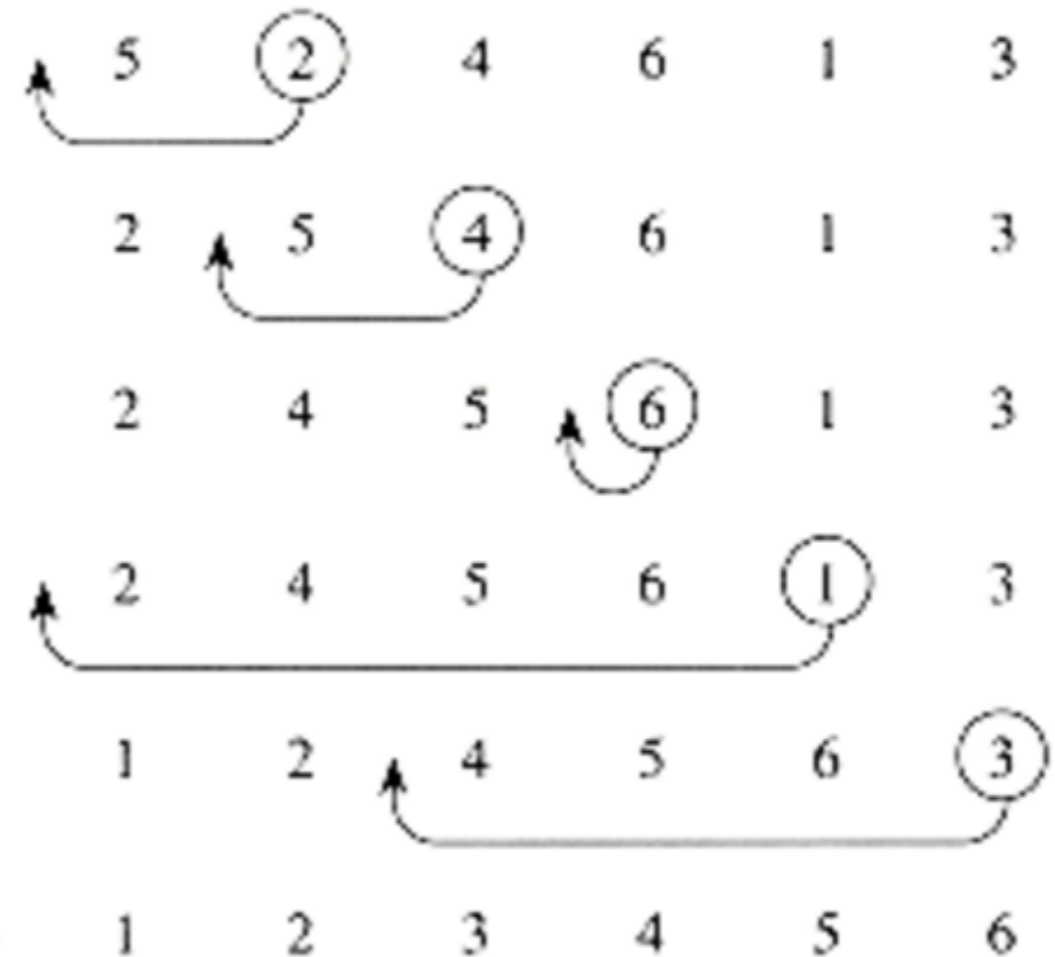
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord,a(i));  
end
```

a\_ord



# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

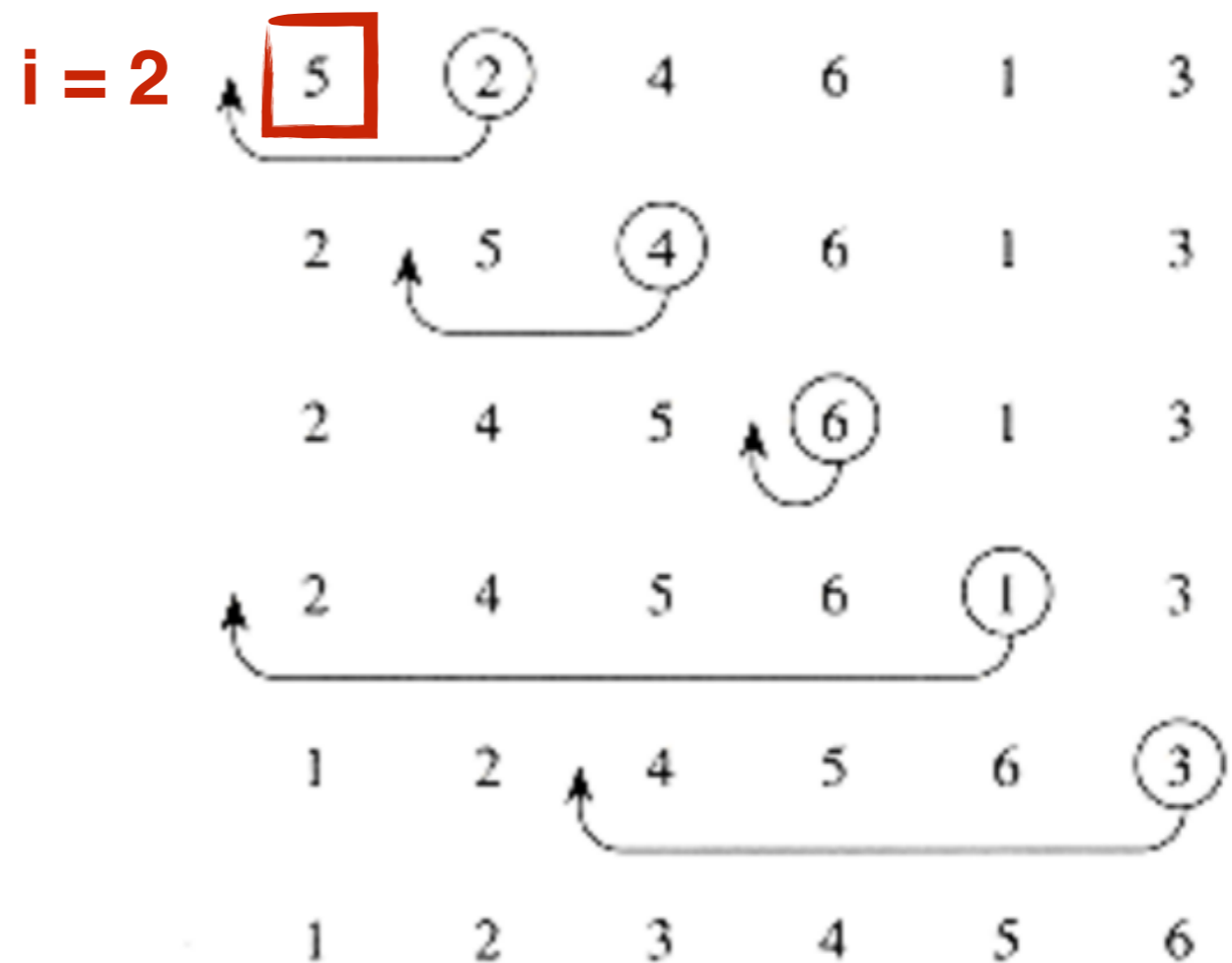
```
a_ord = a(1);
```

```
for i=2:length(a)
```

```
    a_ord = inserisci(a_ord, a(i));
```

```
end
```

a\_ord



# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

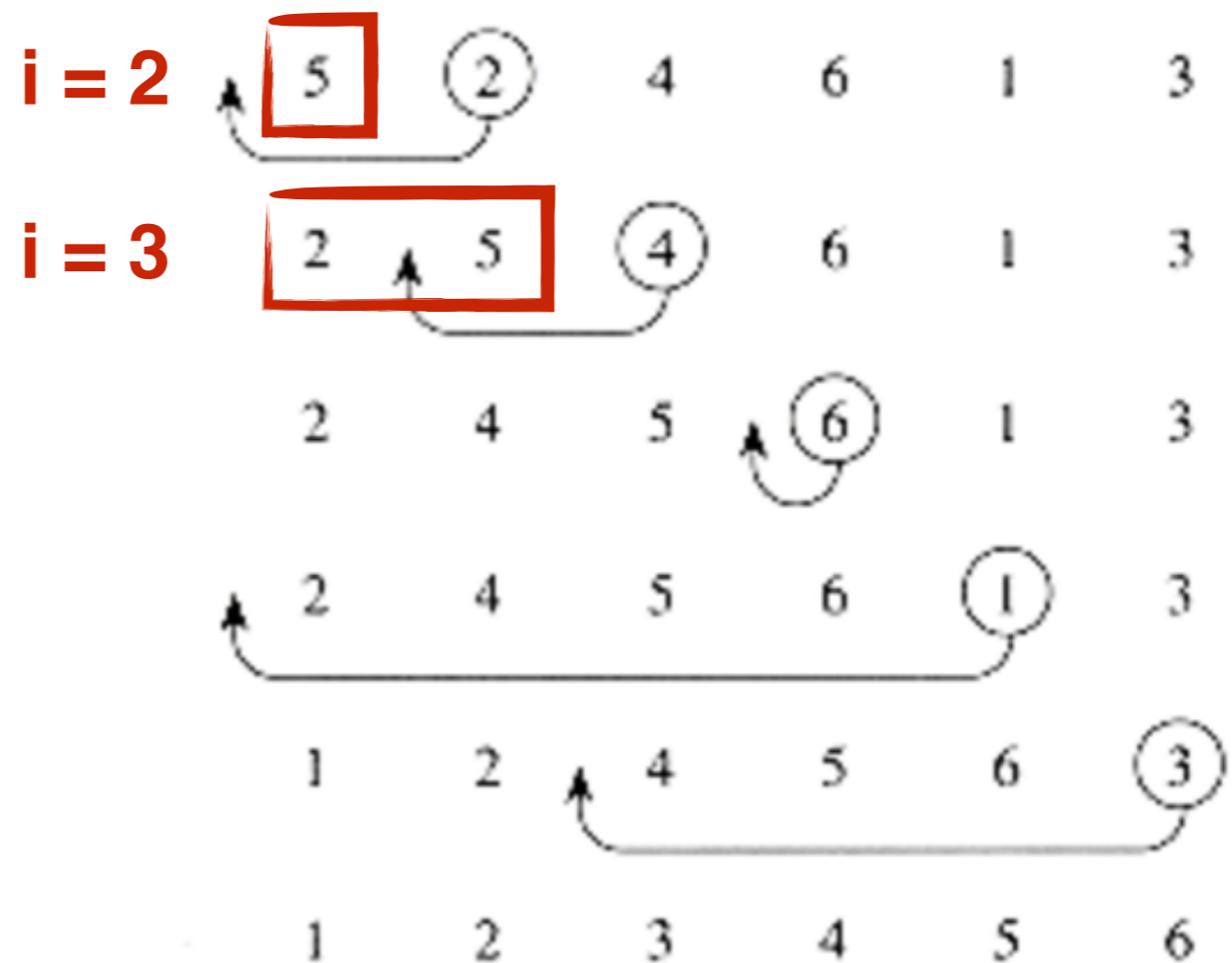
a = [5 2 4 6 1 3]

```
a_ord = a(1);
```

```
for i=2:length(a)  
    a_ord = inserisci(a_ord, a(i));
```

```
end
```

a\_ord



# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

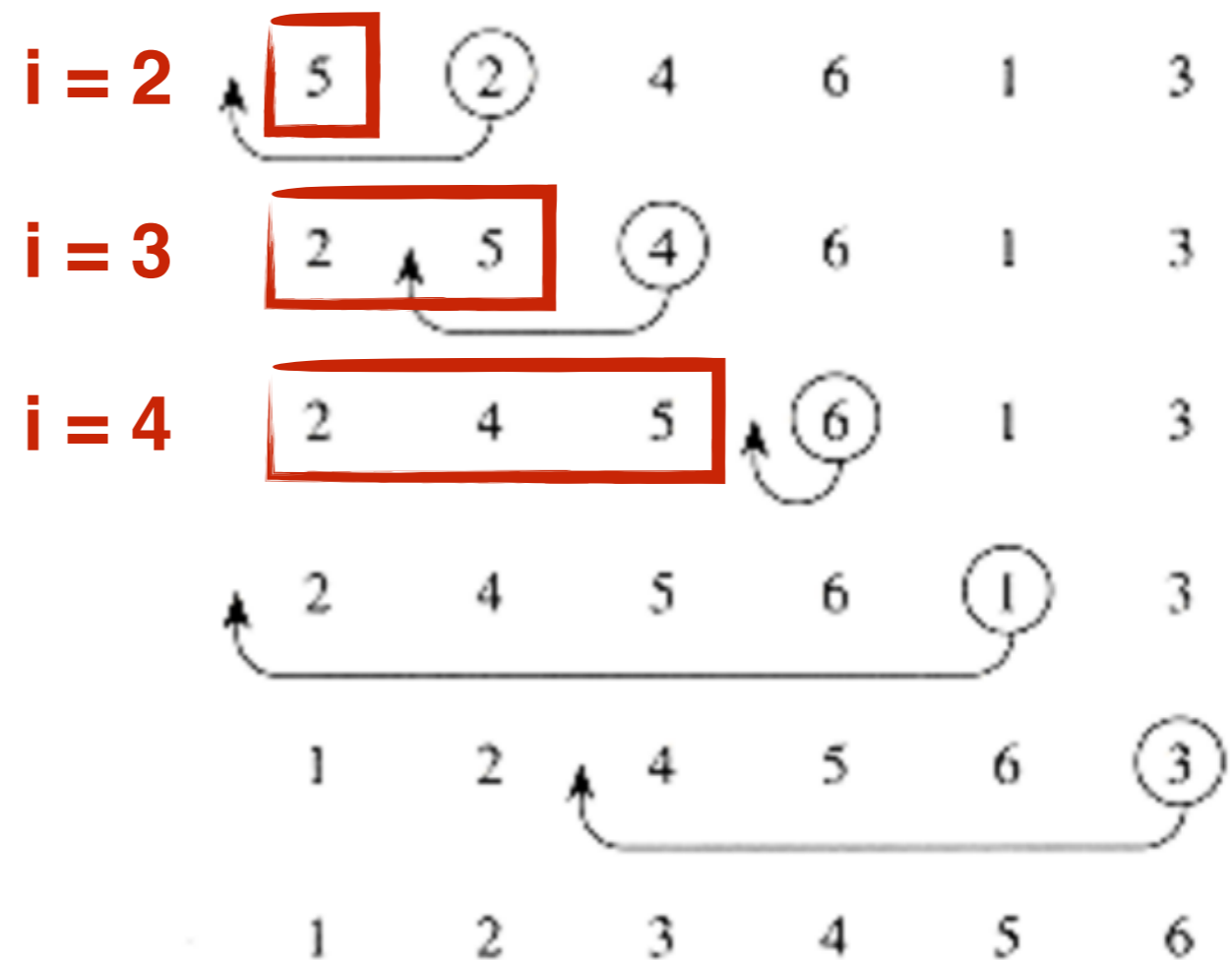
```
a_ord = a(1);
```

```
for i=2:length(a)
```

```
    a_ord = inserisci(a_ord, a(i));
```

```
end
```

a\_ord



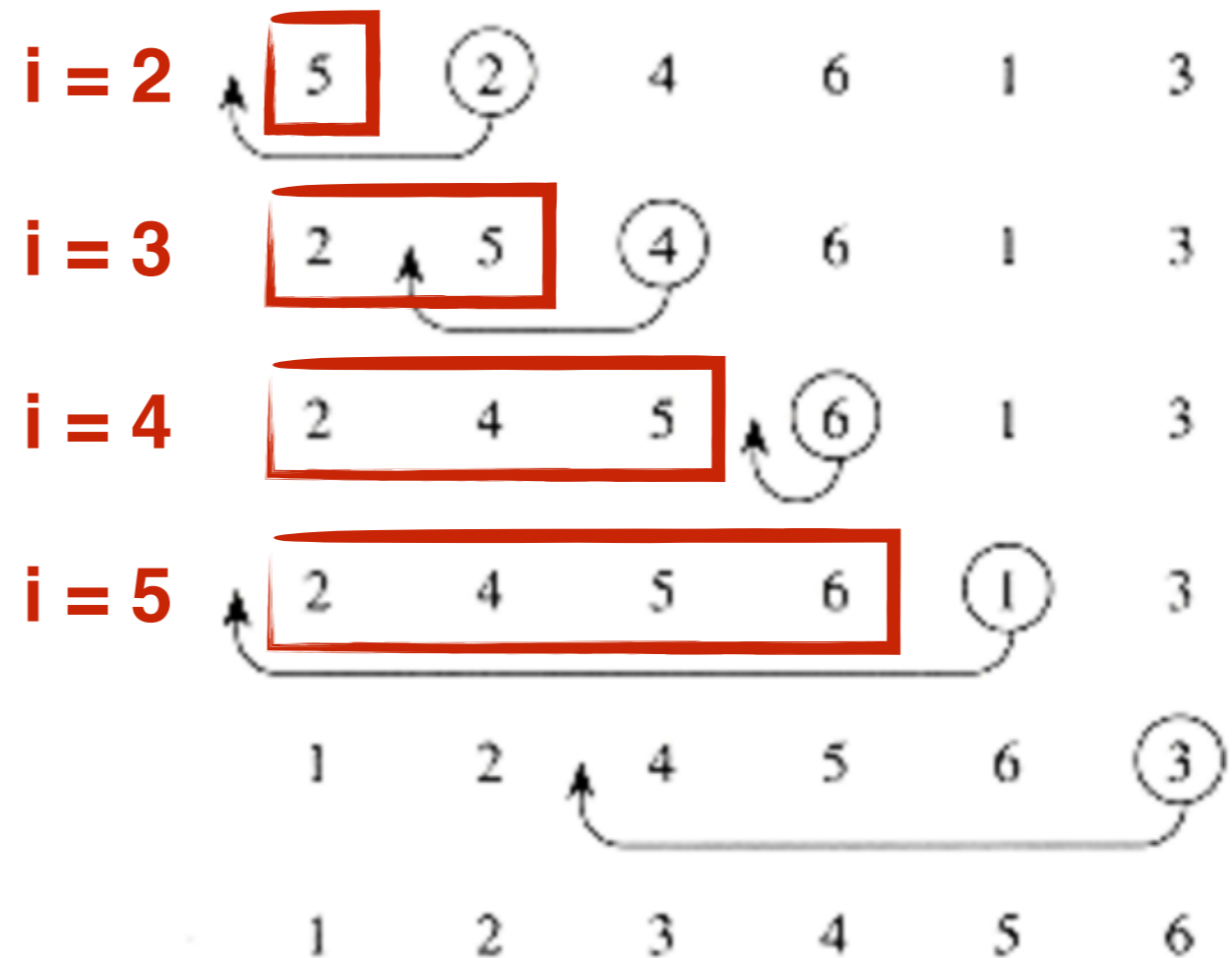
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord, a(i));  
end
```

a\_ord



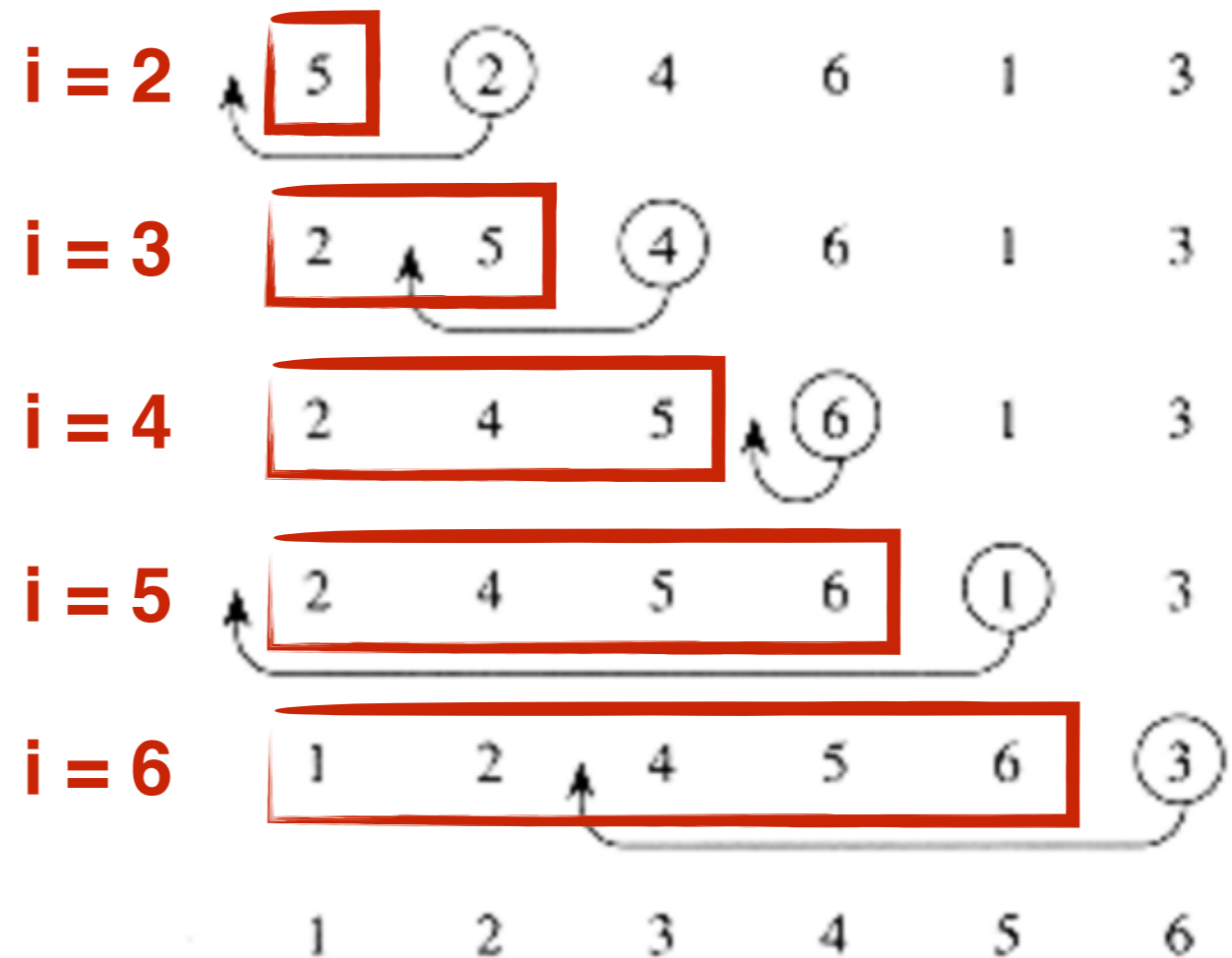
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord, a(i));  
end
```

a\_ord



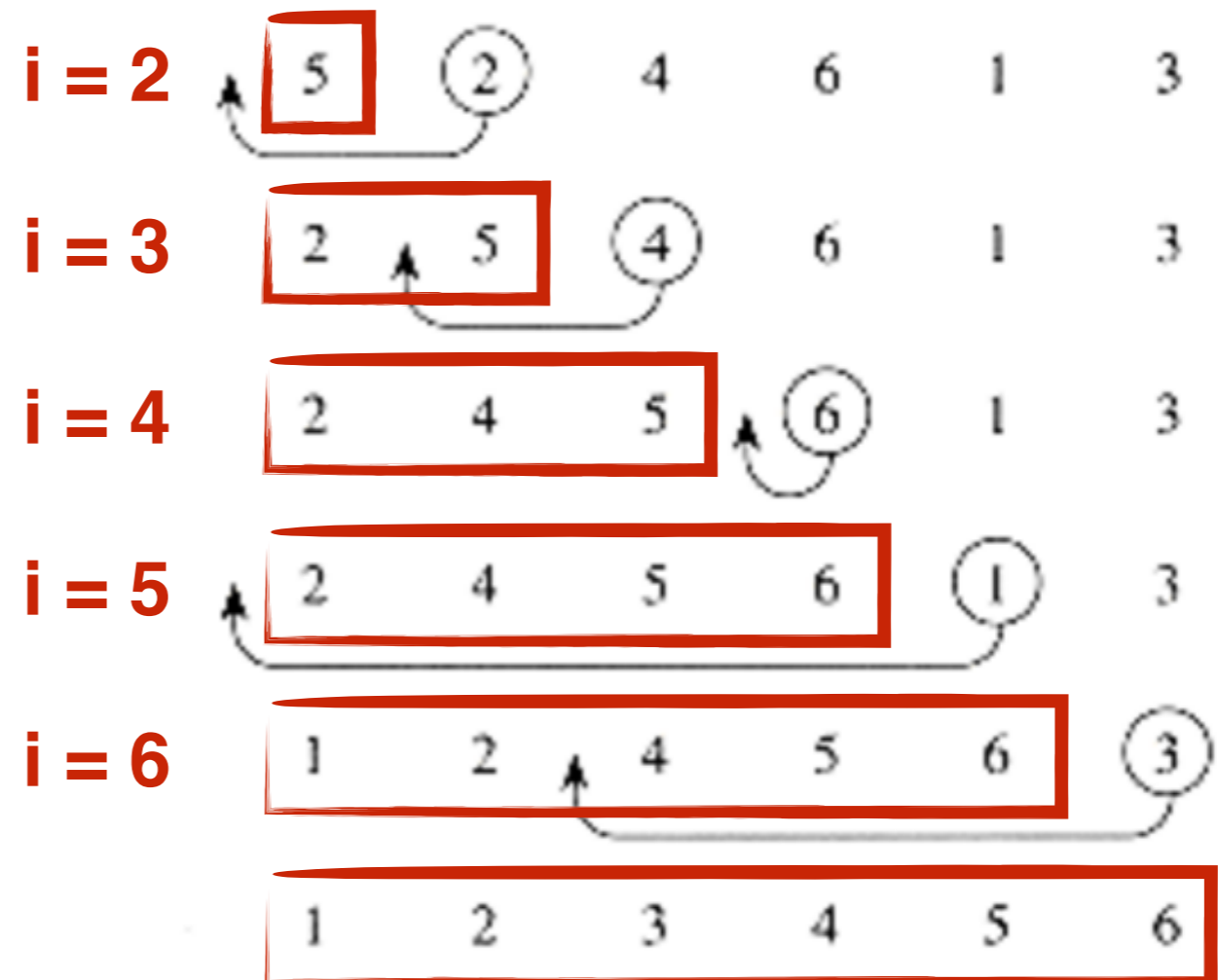
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord, a(i));  
end
```

a\_ord



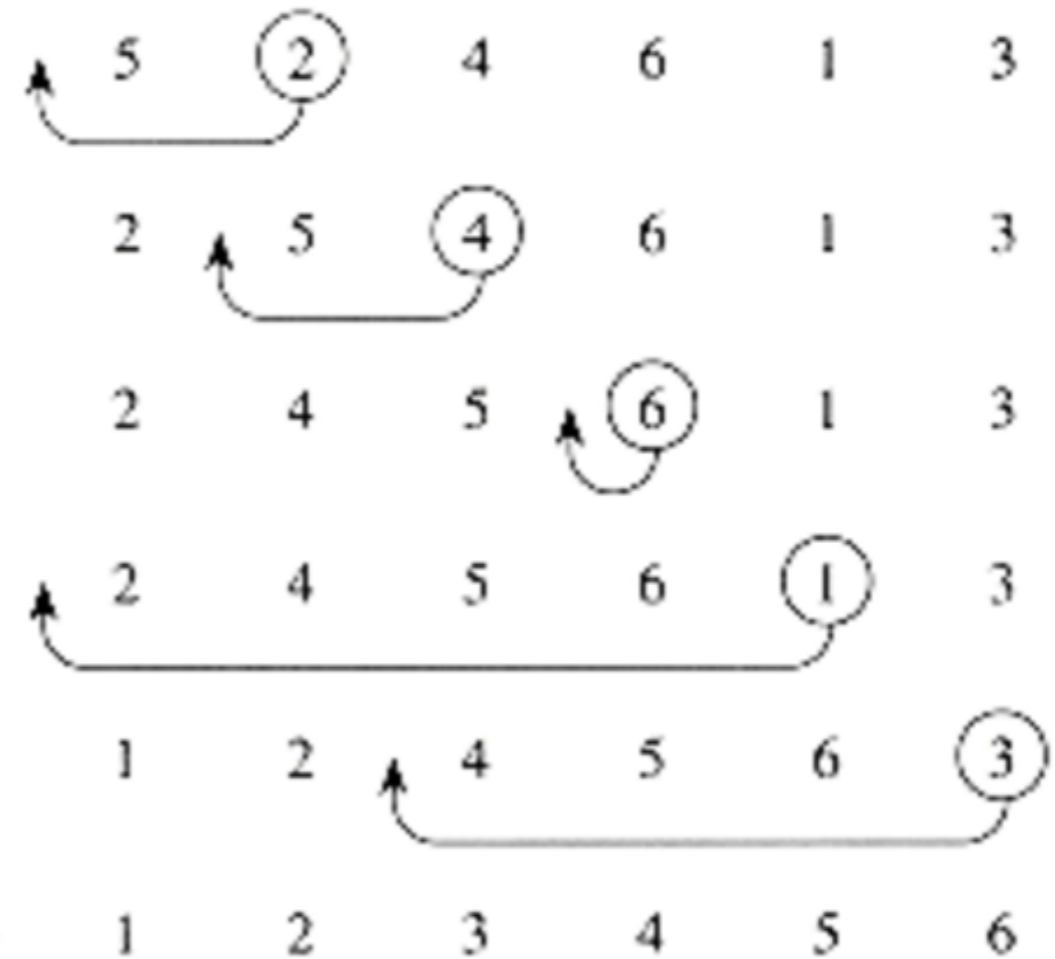
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord,a(i));  
end
```

a\_ord



# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

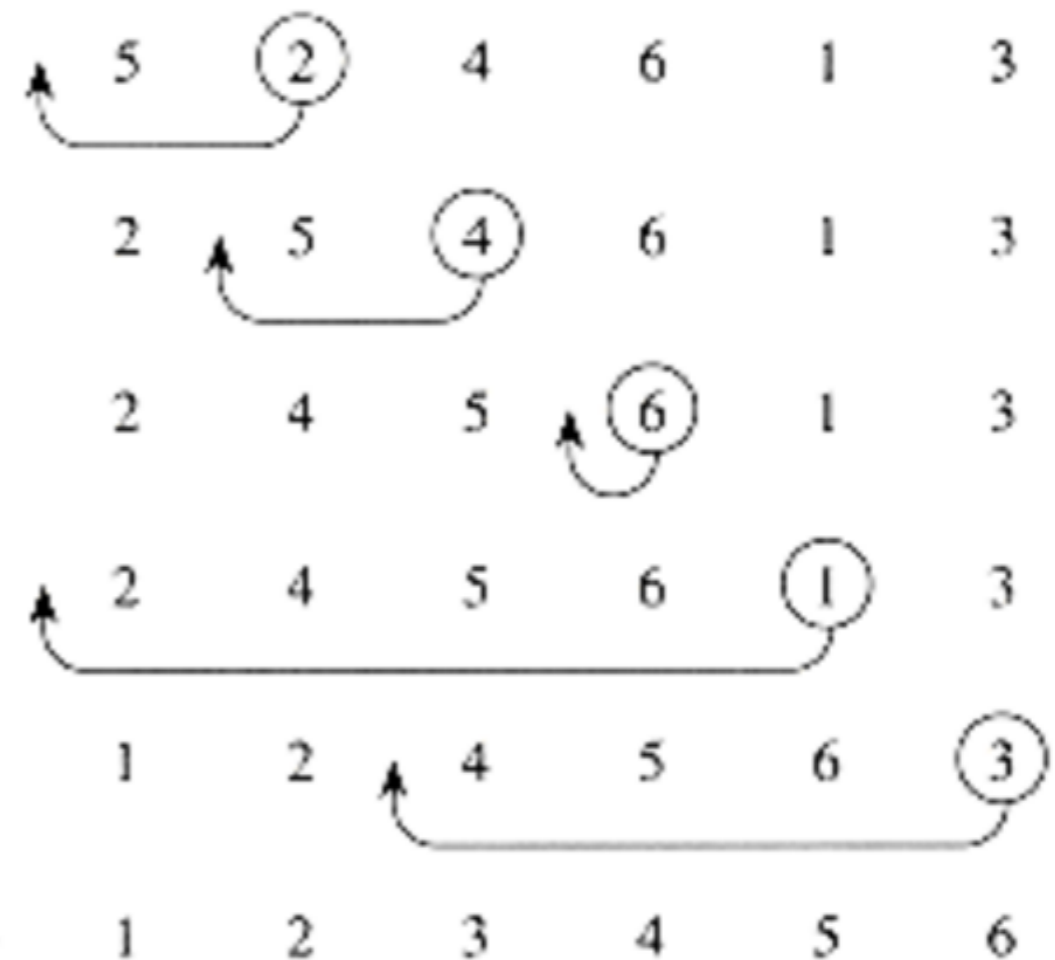
a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord,a(i));  
end
```

a\_ord

↓

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



# Ripasso: maschera, selezione, concatenazione

---

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

# Ripasso: maschera, selezione, concatenazione

---

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

# Ripasso: maschera, selezione, concatenazione

---

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

**maschera**

# Ripasso: maschera, selezione, concatenazione

---

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

**maschera**

Vettore di **bit**:

[0 0 **1** 0 ... 0 **1 1 1** 0 0]

Vettore di **bit**:

[**1 1** 0 **1** ... **1** 0 0 0 **1 1**]

# Ripasso: maschera, selezione, concatenazione

---

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

**maschera**

Vettore di **bit**:

[0 0 **1** 0 ... 0 **1 1 1** 0 0]

Vettore di **bit**:

[**1 1** 0 **1** ... **1** 0 0 0 **1 1**]

**1** se:  **$v(i) \leq e$**

0 altrimenti

# Ripasso: maschera, selezione, concatenazione

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

**maschera**

Vettore di **bit**:

[0 0 **1** 0 ... 0 **1 1 1** 0 0]

**1** se:  $v(i) \leq e$   
0 altrimenti

Vettore di **bit**:

[**1 1** 0 **1** ... **1** 0 0 0 **1 1**]

**1** se:  $v(i) > e$   
0 altrimenti

## Ripasso: maschera, **selezione**, concatenazione

---

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

# Ripasso: maschera, **selezione**, concatenazione

---

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

# Ripasso: maschera, **selezione**, concatenazione

---

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

**selezione**

# Ripasso: maschera, **selezione**, concatenazione

---

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

**selezione**

**v**([0 0 **1** 0 ... 0 **1 1 1** 0 0])    **v**([**1 1** 0 **1** ... **1** 0 0 0 **1 1**])

# Ripasso: maschera, **selezione**, concatenazione

---

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

**selezione**

Two red arrows originate from the boxed expressions in the code above. One arrow points from `v(v<=e)` to the first vector expression below, and the other points from `v(v>e)` to the second vector expression below.

**v**([0 0 **1** 0 ... 0 **1 1 1** 0 0])    **v**([**1 1** 0 **1** ... **1** 0 0 0 **1 1**])

Crea un **vettore**  
che contiene gli **elementi** di **v**  
**solo** nelle **posizioni degli 1**

# Ripasso: maschera, selezione, **concatenazione**

---

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

# Ripasso: maschera, selezione, **concatenazione**

---

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

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

**array 1**

Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

array 1

array 2

Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

array 1

array 2

array 3

Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

# Ripasso: maschera, selezione, **concatenazione**

---

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

**concatenazione**

[ array 1 array 2 array 3 ]

Crea un **vettore**  
che contiene gli **elementi** dei **vettori**  
**nell'ordine** specificato

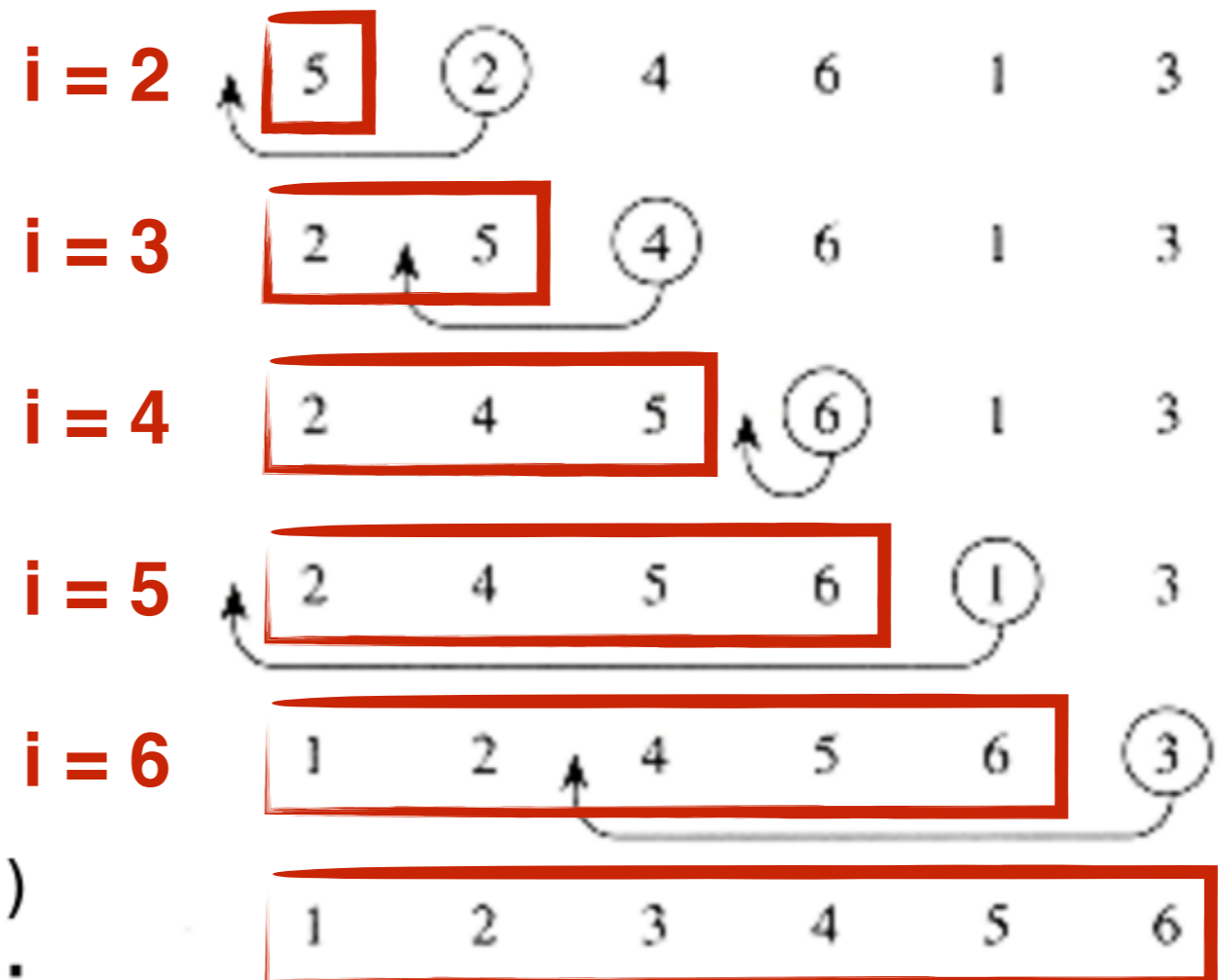
# Esercizio: Maschere di bit (ordinamento)

Scorsa  
esercitazione

a = [5 2 4 6 1 3]

```
a_ord = a(1);  
for i=2:length(a)  
    a_ord = inserisci(a_ord, a(i));  
end
```

a\_ord



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

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

(30') Ricerca binaria (ricorsiva)

(30') Calcolo derivata (ricorsiva)

(30') Struct (rilievi altimetrici) (ex Es6)

(30') Struct (film) (ex Es7)

(30') Vettori, vettori everywhere! (ex Es1.2)

# Esercizio: ricerca binaria (ricorsiva)

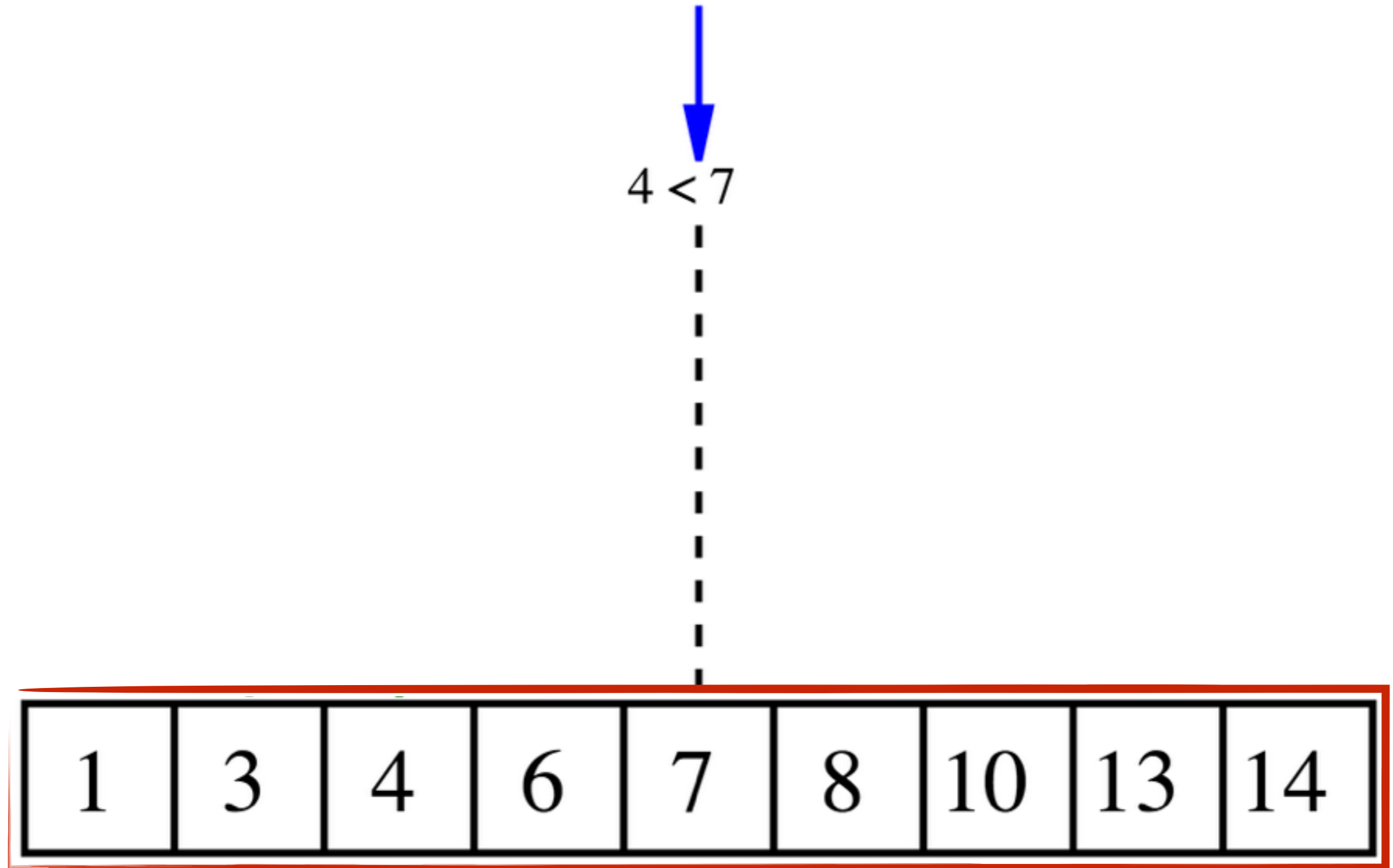
---

- Effettuare una **ricerca binaria** di un valore all'interno di un array ordinato. Il valore  $F$  é intero e l'array  $a$  é di interi.

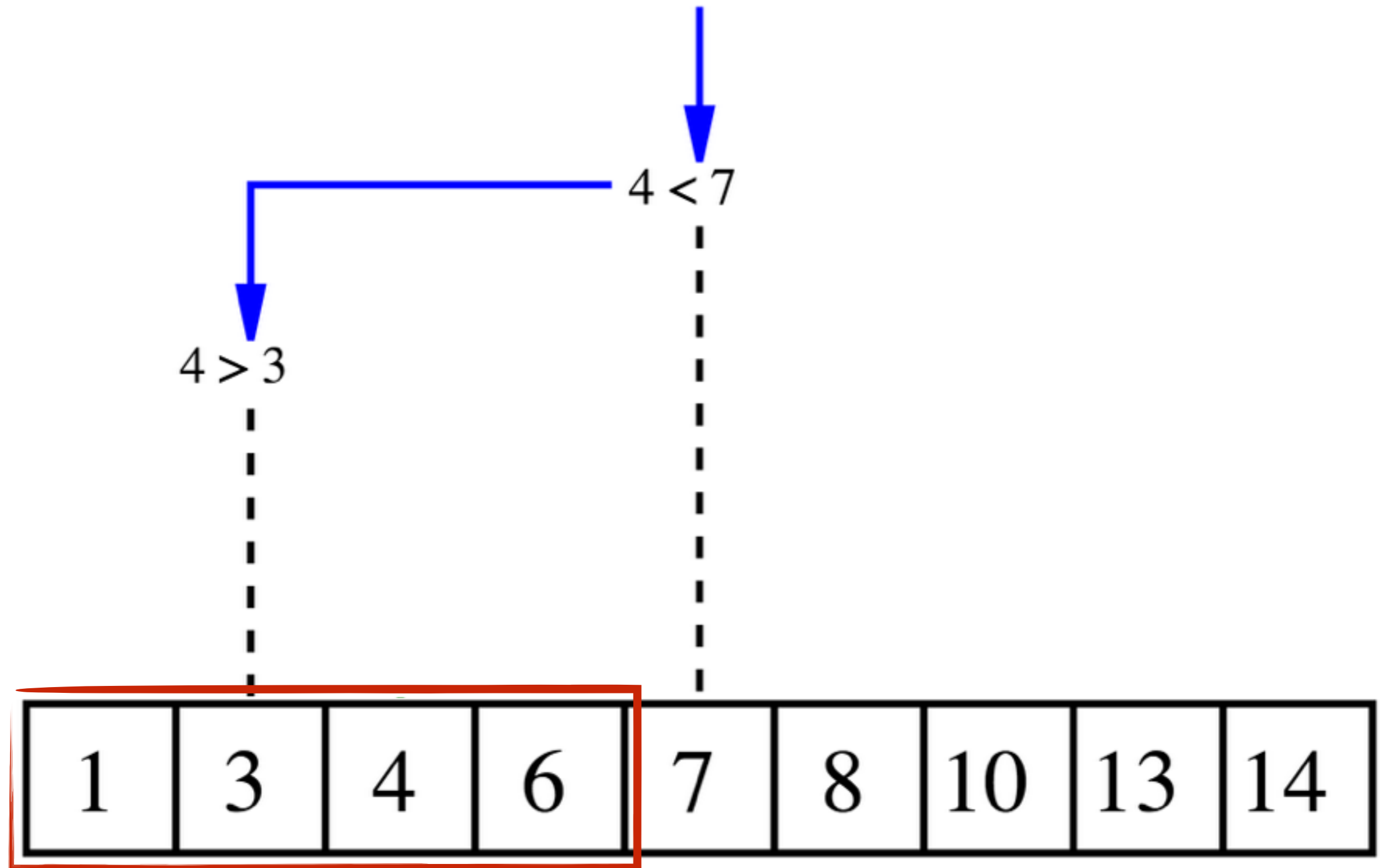
*RICERCA BINARIA: di volta in volta cerco nel **punto medio** di un intervallo che si dimezza in lunghezza ad ogni invocazione dell'algoritmo, **tenendo** opportunamente **la parte destra o sinistra** a seconda che il **punto medio fosse a destra o a sinistra** del valore  **$F$***

# Esercizio: ricerca binaria (ricorsiva)

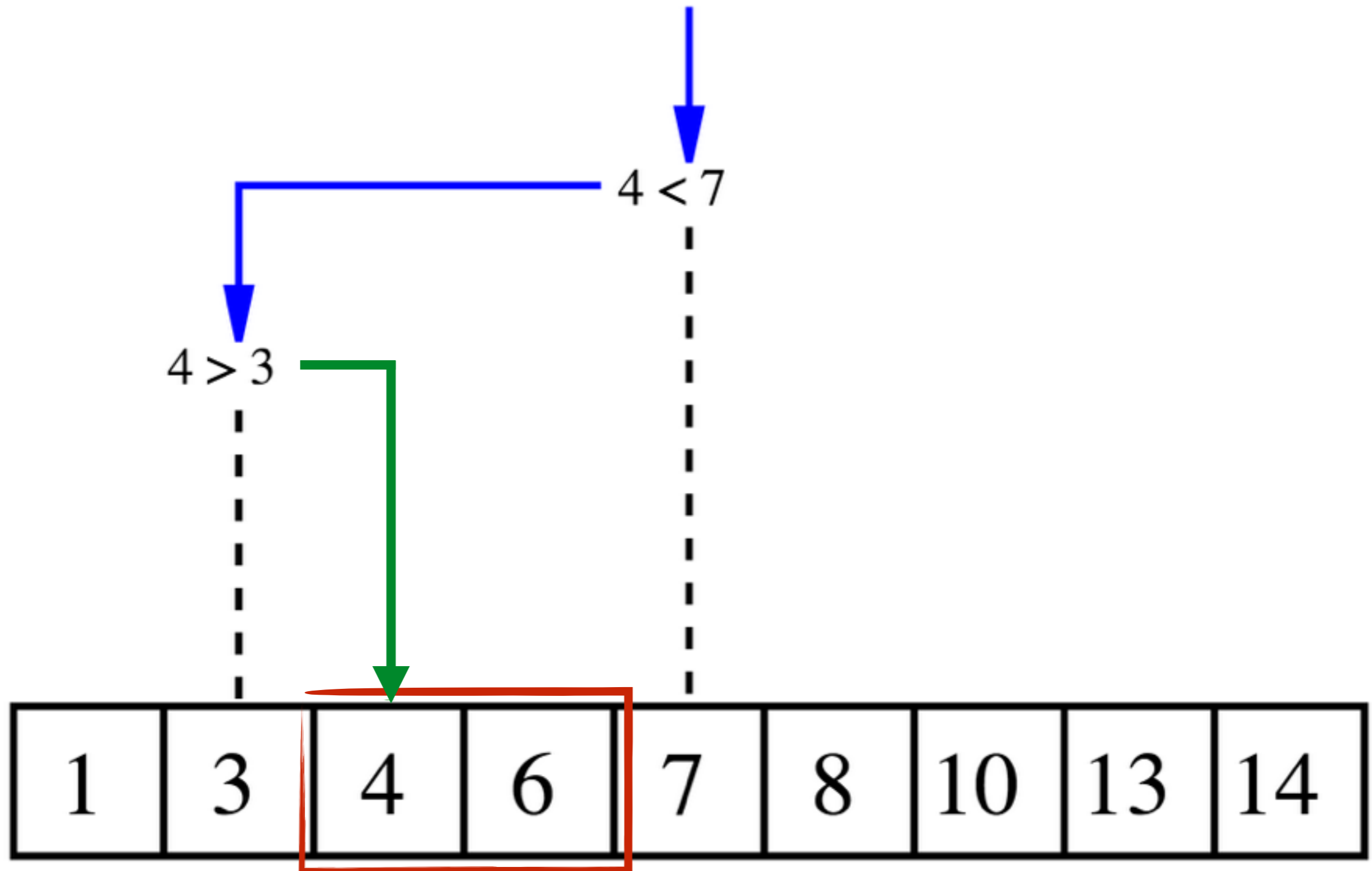
---



# Esercizio: ricerca binaria (ricorsiva)



# Esercizio: ricerca binaria (ricorsiva)



# Esercizio: ricerca binaria (ricorsiva)

---

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);

    if (v(elementoMedio) == cercato)
        val = elementoMedio;
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);

    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else

        val = ricercaBinaria(v, cercato, limSinistra, limDestra);

    end
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);

    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else
        if (v(elementoMedio) > cercato)
            limDestra = elementoMedio;

        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
    end
end
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);

    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else
        if (v(elementoMedio) > cercato)
            limDestra = elementoMedio;
        else
            limSinistra = elementoMedio;
        end

        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
    end
end
```

# Esercizio: ricerca binaria (ricorsiva)

---

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);

    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else
        if (v(elementoMedio) > cercato)
            limDestra = elementoMedio;
        else
            limSinistra = elementoMedio;
        end
        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
    end
end
```

**Caso generale**

# Esercizio: ricerca binaria (ricorsiva)

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);
```

```
    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else
```

**Caso particolare**

```
        if (v(elementoMedio) > cercato)
            limDestra = elementoMedio;
        else
            limSinistra = elementoMedio;
        end
```

**Caso generale**

```
        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
```

```
end
```

# Esercizio: ricerca binaria (ricorsiva)

---

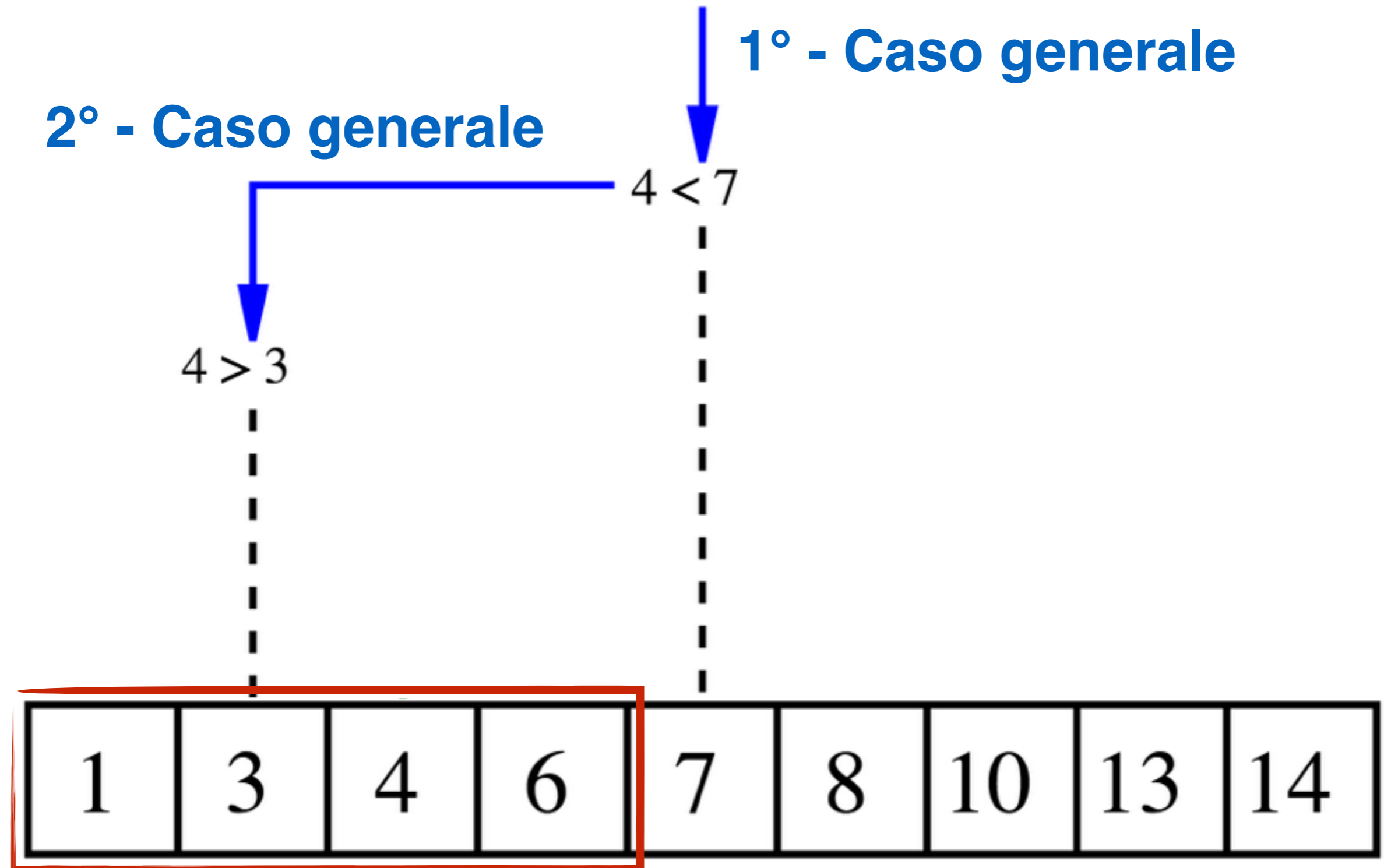
**1° - Caso generale**

4 < 7

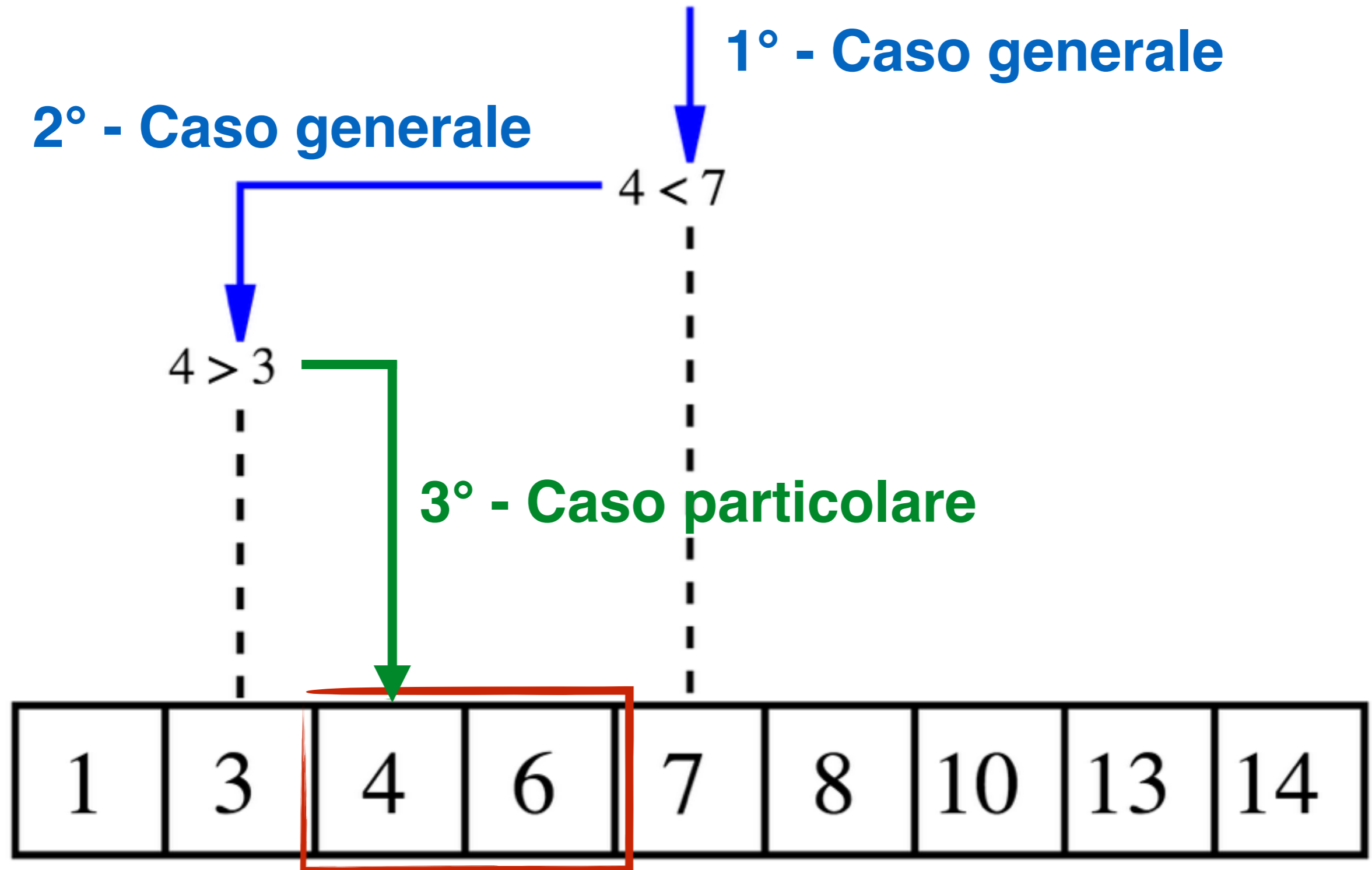


1	3	4	6	7	8	10	13	14
---	---	---	---	---	---	----	----	----

# Esercizio: ricerca binaria (ricorsiva)



# Esercizio: ricerca binaria (ricorsiva)



# Esercizio: ricerca binaria (ricorsiva)

```
function val = ricercaBinaria(v, cercato, limSinistra, limDestra)

    elementoMedio = floor((limDestra + limSinistra)/2);
```

```
    if (v(elementoMedio) == cercato)
        val = elementoMedio;
    else
```

**Caso particolare**

```
        if (v(elementoMedio) > cercato)
            limDestra = elementoMedio;
        else
            limSinistra = elementoMedio;
        end
```

**Caso generale**

```
        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
```

```
    end
```

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

~~(30') Ricerca binaria (ricorsiva)~~

(30') Calcolo derivata (ricorsiva)

(30') Struct (rilievi altimetrici) (ex Es6)

(30') Struct (film) (ex Es7)

(30') Vettori, vettori everywhere! (ex Es1.2)

# Esercizio: calcolo derivata (ricorsiva)

---

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.  
*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$
  2. Restituisca un vettore che rappresenta la derivata  $n$ -esima del polinomio

# Esercizio: calcolo derivata (ricorsiva)

---

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$
  2. Restituisca un vettore che rappresenta la derivata  $n$ -esima del polinomio
- Per calcolare la **derivata prima** del polinomio applicare la comune regola di derivazione per i polinomi.

# Esercizio: calcolo derivata (ricorsiva)

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.  
*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*   
*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde al vettore  $[3 \ 0 \ 5 \ 2 \ 7]$  (lunghezza 5)*
- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$
  2. Restituisca un vettore che rappresenta la derivata  $n$ -esima del polinomio
- Per calcolare la **derivata prima** del polinomio applicare la comune regola di derivazione per i polinomi.
- Calcolare la **derivata  $n$ -esima** come la **derivata prima della derivata  $(n-1)$ -esima**

# Esercizio: calcolo derivata (ricorsiva)

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$
  2. Restituisca un vettore che rappresenta la derivata  $n$ -esima del polinomio

- Per calcolare la **derivata prima** del polinomio applicare la comune regola di derivazione per i polinomi. **Caso particolare**

- Calcolare la **derivata  $n$ -esima** come la **derivata prima della derivata  $(n-1)$ -esima**

# Esercizio: calcolo derivata (ricorsiva)

- Rappresentiamo un **polinomio** con un **vettore** contenente i suoi **coefficienti**, dal termine di grado massimo a quello di grado minimo.

*Si noti che un polinomio di grado  $n$  corrisponde a un vettore di lunghezza  $n+1$*

*Esempio:  $3x^4 + 5x^2 + 2x + 7$  (grado 4) corrisponde  
al vettore **[3 0 5 2 7]** (lunghezza 5)*

- Scrivere una funzione ricorsiva di nome **derivata** che:
  1. Riceva in ingresso un vettore che rappresenta un polinomio e un valore  $n$
  2. Restituisca un vettore che rappresenta la derivata  $n$ -esima del polinomio

- Per calcolare la **derivata prima** del polinomio applicare la comune regola di derivazione per i polinomi. **Caso particolare**

- Calcolare la **derivata  $n$ -esima** come la **derivata prima della derivata  $(n-1)$ -esima** **Caso generale**

# Esercizio: calcolo derivata (ricorsiva)

---

```
function[der] = derivata (pol, n)
```

# Esercizio: calcolo derivata (ricorsiva)

---

```
function[der] = derivata (pol, n)

% Caso base: calcolo della derivata prima
if n==1

else

    % Passo ricorsivo: derivata (n-1)-esima della derivata prima

end
```

# Esercizio: calcolo derivata (ricorsiva)

---

```
function[der] = derivata (pol, n)

% Caso base: calcolo della derivata prima
if n==1

else

    % Passo ricorsivo: derivata (n-1)-esima della derivata prima
    der = derivata(derivata(pol, n-1), 1)

end
```

# Esercizio: calcolo derivata (ricorsiva)

---

```
function[der] = derivata (pol, n)

% Caso base: calcolo della derivata prima
if n==1

    % Vettore con gli esponenti dei singoli monomi
    % Es. per  $2x^3 + x^2 + 3 \rightarrow \text{esp} = [3 \ 2 \ 1]$ 
    esp = [length(pol)-1 : -1 : 1]

else

    % Passo ricorsivo: derivata (n-1)-esima della derivata prima
    der = derivata(derivata(pol, n-1), 1)

end
```

# Esercizio: calcolo derivata (ricorsiva)

```
function[der] = derivata (pol, n)

% Caso base: calcolo della derivata prima
if n==1

    % Vettore con gli esponenti dei singoli monomi
    % Es. per  $2x^3 + x^2 + 3 \rightarrow \text{esp} = [3 \ 2 \ 1]$ 
    esp = [length(pol)-1 : -1 : 1]

    % Calcolo della derivata prima:
    % Es. per il polinomio precedente:
    %  $[3 \ 2 \ 1].*[2 \ 1 \ 0] = [6 \ 2 \ 0]$ 
    der = esp.*pol(1:length(pol)-1)

else

    % Passo ricorsivo: derivata (n-1)-esima della derivata prima
    der = derivata(derivata(pol, n-1), 1)

end
```

# Esercizio: calcolo derivata (ricorsiva)

```
function[der] = derivata (pol, n)

% Caso base: calcolo della derivata prima
if n==1

    % Vettore con gli esponenti dei singoli monomi
    % Es. per  $2x^3 + x^2 + 3 \rightarrow \text{esp} = [3 \ 2 \ 1]$ 
    esp = [length(pol)-1 : -1 : 1]

    % Calcolo della derivata prima:
    % Es. per il polinomio precedente:
    %  $[3 \ 2 \ 1].*[2 \ 1 \ 0] = [6 \ 2 \ 0]$ 
    der = esp.*pol(1:length(pol)-1)

    Caso particolare

else

    % Passo ricorsivo: derivata (n-1)-esima della derivata prima
    der = derivata(derivata(pol, n-1), 1)

end
```

# Esercizio: calcolo derivata (ricorsiva)

```
function[der] = derivata (pol, n)
```

```
% Caso base: calcolo della derivata prima
```

```
if n==1
```

```
% Vettore con gli esponenti dei singoli monomi
```

```
% Es. per  $2x^3 + x^2 + 3 \rightarrow \text{esp} = [3 \ 2 \ 1]$ 
```

```
esp = [length(pol)-1 : -1 : 1]
```

```
% Calcolo della derivata prima:
```

```
% Es. per il polinomio precedente:
```

```
%  $[3 \ 2 \ 1].*[2 \ 1 \ 0] = [6 \ 2 \ 0]$ 
```

```
der = esp.*pol(1:length(pol)-1)
```

**Caso particolare**

```
else
```

```
% Passo ricorsivo: derivata (n-1)-esima della derivata prima
```

```
der = derivata(derivata(pol, n-1), 1)
```

**Caso generale**


```
end
```

# Esercizio: calcolo derivata (ricorsiva)

---


# Esercizio: calcolo derivata (ricorsiva)

---

```
% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)
```

# Esercizio: calcolo derivata (ricorsiva)


---

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

# Esercizio: calcolo derivata (ricorsiva)


---

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

# Esercizio: calcolo derivata (ricorsiva)


% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)



# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)


derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

der = derivata(**derivata(pol, 1)**, 1)

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)


der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

der = derivata(**derivata(pol, 1)**, 1)



# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)


der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

der = derivata(**derivata(pol, 1)**, 1)

**derivata(pol, 1)** 

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)


**derivata(pol, 2)** 

der = derivata(**derivata(pol, 1)**, 1)

**derivata(pol, 1)** 

esp = [5 4 3 2 1]

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$  (grado 5)  
corrisponde al vettore **[5 4 3 2 1 1]** (lunghezza 6)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 


der = derivata(**derivata(pol, 1)**, 1)

**derivata(pol, 1)** 

esp = [5 4 3 2 1]

der = **[25 16 9 4 1]**

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)


derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

**der = derivata([25 16 9 4 1], 1)**

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)


derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

**der = derivata([25 16 9 4 1], 1)**  


# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)


der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

**der = derivata([25 16 9 4 1], 1)**

 derivata([25 16 9 4 1], 1)

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)


**derivata(pol, 2)** 

**der = derivata([25 16 9 4 1], 1)**

 derivata([25 16 9 4 1], 1)

esp = [4 3 2 1]

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$  (grado 5)  
corrisponde al vettore **[5 4 3 2 1 1]** (lunghezza 6)

derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 


**der = derivata([25 16 9 4 1], 1)**

 derivata([25 16 9 4 1], 1)

esp = [4 3 2 1]

der = **[100 48 18 4]**

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)


derivata(pol, 3)

der = derivata(**derivata(pol, 2)**, 1)

**derivata(pol, 2)** 

der = [**100 48 18 4**]

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)


**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = **derivata**(**[100 48 18 4]**, **1**)



# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$  (grado 5)  
corrisponde al vettore **[5 4 3 2 1 1]** (lunghezza 6)


derivata(pol, 3)

der = **derivata([100 48 18 4], 1)**



**derivata([100 48 18 4], 1)**

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$  (grado 5)  
corrisponde al vettore **[5 4 3 2 1 1]** (lunghezza 6)

derivata(pol, 3)


der = **derivata([100 48 18 4], 1)**



**derivata([100 48 18 4], 1)**

esp = [3 2 1]

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$  (grado 5)  
corrisponde al vettore **[5 4 3 2 1 1]** (lunghezza 6)

derivata(pol, 3)

der = **derivata([100 48 18 4], 1)**




**derivata([100 48 18 4], 1)**

esp = [3 2 1]

der = **[300 96 18]**

# Esercizio: calcolo derivata (ricorsiva)

% Passo ricorsivo: derivata (n-1)-esima della derivata prima  
der  derivata(derivata(pol, n-1), 1)

**5** $x^5$  + **4** $x^4$  + **3** $x^3$  + **2** $x^2$  + **1** $x^1$  + **1** (*grado 5*)  
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

derivata(pol, 3)

der = [**300 96 18**]

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

~~(30') Ricerca binaria (ricorsiva)~~

~~(30') Calcolo derivata (ricorsiva)~~

(30') Struct (rilievi altimetrici) (ex Es6)

(30') Struct (film) (ex Es7)

(30') Vettori, vettori everywhere! (ex Es1.2)

# Esercizio: Struct (rilievi altimetrici)

Scorsa  
esercitazione

- Si sviluppi un programma in matlab che acquisisce da tastiera i dati relativi a rilievi altimetrici e stampa a video l'altitudine media di tutti quelli che hanno latitudine compresa tra 10 e 80 e longitudine tra 30 e 60

# Esercizio: Struct (rilievi altimetrici)

Scorsa  
esercitazione

```
more = input('vuoi inserire valori altimetrici? (s/n)');
```

# Esercizio: Struct (rilievi altimetrici)

Scorsa  
esercitazione

```
more = input('vuoi inserire valori altimetrici? (s/n)');  
ii=1;  
while more=='s'  
  
    ii = ii+1;  
    more = input('vuoi inserire altri valori altimetrici? (s/n)');  
end
```

# Esercizio: Struct (rilievi altimetrici)

Scorsa  
esercitazione

```
more = input('vuoi inserire valori altimetrici? (s/n)');  
ii=1;  
while more=='s'  
    arch(ii).altitudine = input('altitudine ');  
    arch(ii).longitudine = input('longitudine ');  
    arch(ii).latitudine = input('latitudine ');  
    ii = ii+1;  
    more = input('vuoi inserire altri valori altimetrici? (s/n)');  
end
```

# Esercizio: Struct (rilievi altimetrici)

```

more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'
    arch(ii).altitudine = input('altitudine ');
    arch(ii).longitudine = input('longitudine ');
    arch(ii).latitudine = input('latitudine ');
    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end

jj=1;
for ii=1:length(arch)

end

disp(['la media degli elementi selezionati e` ' num2str(mean(elemSelez))])

```

# Esercizio: Struct (rilievi altimetrici)

Scorsa  
esercitazione

```
more = input('vuoi inserire valori altimetrici? (s/n)');
ii=1;
while more=='s'
    arch(ii).altitudine = input('altitudine ');
    arch(ii).longitudine = input('longitudine ');
    arch(ii).latitudine = input('latitudine ');
    ii = ii+1;
    more = input('vuoi inserire altri valori altimetrici? (s/n)');
end

jj=1;
for ii=1:length(arch)
    % attenzione: la condizione deve essere scritta sulla stessa linea...
    if arch(ii).latitudine>=10&&arch(ii).latitudine<=80 &&
        arch(ii).longitudine>=30&&arch(ii).longitudine<=60
        elemSelez(jj) = arch(ii).altitudine;
        jj=jj+1;
    end
end
disp(['la media degli elementi selezionati e` ' num2str(mean(elemSelez))])
```

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

~~(30') Ricerca binaria (ricorsiva)~~

~~(30') Calcolo derivata (ricorsiva)~~

~~(30') Struct (rilievi altimetrici) (ex Es6)~~

(30') Struct (film) (ex Es7)

(30') Vettori, vettori everywhere! (ex Es1.2)

# Esercizio: Struct (film)

Scorsa  
esercitazione

- Si vuole compilare la pagella dei propri film preferiti. Per farlo:
  - Scrivere un programma che chieda di inserire i film. Ogni film e' caratterizzato da un anno, un titolo e un voto;
  - Scrivere il codice che permetta di visualizzare il numero totale di film con voto superiore a 6;
  - Scrivere il codice che permetta di visualizzare i titoli ed i voti dei film prodotti tra il 2000 e il 2005;

# Esercizio: Struct (film)

---

% Inserimento films

Scorsa  
esercitazione

# Esercizio: Struct (film)

```
% Inserimento films
```

```
stopInput='S';
```

```
count=1;
```

```
while(stopInput=='S')
```

```
end
```

# Esercizio: Struct (film)

```
% Inserimento films
```

```
stopInput='S';  
count=1;
```

```
while(stopInput=='S')
```

```
    films(count).titolo = input('Inserisci il titolo: ');  
    films(count).anno = input('Inserisci anno: ');  
    films(count).voto = input('Inserisci il voto: ');
```

```
end
```

# Esercizio: Struct (film)

```
% Inserimento films
```

```
stopInput='S';  
count=1;
```

```
while(stopInput=='S')
```

```
    films(count).titolo = input('Inserisci il titolo: ');  
    films(count).anno = input('Inserisci anno: ');  
    films(count).voto = input('Inserisci il voto: ');
```

```
    count = count + 1;
```

```
    stopInput = input('Vuoi inserire altri film? (S/N) ');
```

```
end
```

# Esercizio: Struct (film)

---

Scorsa  
esercitazione

% Numero totale di film con voto superiore a 6

# Esercizio: Struct (film)

```
% Numero totale di film con voto superiore a 6  
sum([films.voto] > 6)
```

```
% Film prodotti tra il 2002 ed il 2005
```

# Esercizio: Struct (film)

```
% Numero totale di film con voto superiore a 6  
sum([films.voto] > 6)
```

```
% Film prodotti tra il 2002 ed il 2005  
idx = find([films.anno] > 2002 & [films.anno] < 2005);
```

```
% Stampa titoli e voti separatamente
```

# Esercizio: Struct (film)

```
% Numero totale di film con voto superiore a 6  
sum([films.voto] > 6)
```

```
% Film prodotti tra il 2002 ed il 2005  
idx = find([films.anno] > 2002 & [films.anno] < 2005);
```

```
% Stampa titoli e voti separatamente  
films(idx).titolo  
films(idx).voto
```

```
% Stampa titolo e voto affiancati
```

# Esercizio: Struct (film)

```
% Numero totale di film con voto superiore a 6  
sum([films.voto] > 6)
```

```
% Film prodotti tra il 2002 ed il 2005  
idx = find([films.anno] > 2002 & [films.anno] < 2005);
```

```
% Stampa titoli e voti separatamente  
films(idx).titolo  
films(idx).voto
```

```
% Stampa titolo e voto affiancati  
for i=idx  
  
end
```

# Esercizio: Struct (film)

```
% Numero totale di film con voto superiore a 6
sum([films.voto] > 6)

% Film prodotti tra il 2002 ed il 2005
idx = find([films.anno] > 2002 & [films.anno] < 2005);

% Stampa titoli e voti separatamente
films(idx).titolo
films(idx).voto

% Stampa titolo e voto affiancati
for i=idx
    display([films(i).titolo ' ' num2str(films(i).voto)])
end
```

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

~~(30') Ricerca binaria (ricorsiva)~~

~~(30') Calcolo derivata (ricorsiva)~~

~~(30') Struet (rilievi altimetrici) (ex Es6)~~

~~(30') Struet (film) (ex Es7)~~

(30') Vettori, vettori everywhere! (ex Es1.2)

# Esercizio: Vettori, vettori everywhere.

NON  
FATTO

- 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.

*NOTA: Si continui a chiedere all'utente di inserire dei vettori fino a quando questi non soddisfano le condizioni necessarie per risolvere l'esercizio.*

# Esercizio: Vettori, vettori everywhere.

---

NON  
FATTO

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

# Esercizio: Vettori, vettori everywhere.

---

NON  
FATTO

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

# Esercizio: Vettori, vettori everywhere.

---

NON  
FATTO

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

# Esercizio: Vettori, vettori everywhere.

NON  
FATTO

```
while
```

```
    v1 = input('Inserisci il primo vettore (tra []) : ');  
    v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
end
```

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

```
v1  
v2
```

# Esercizio: Vettori, vettori everywhere.

NON  
FATTO

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
```

end

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

v1  
v2

# Esercizio: Vettori, vettori everywhere.

NON  
FATTO

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;  
vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
```

end

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

v1  
v2

# Esercizio: Vettori, vettori everywhere.

NON  
FATTO

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo 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: Vettori, vettori everywhere.

NON  
FATTO

```
while(pari == false || vettore == false || numerico == false)
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo 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: Vettori, vettori everywhere.

NON  
FATTO

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

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

# Agenda

---

~~(30') Maschere di bit (ordinamento v2.0) (ex Es5)~~

~~(30') Ricerca binaria (ricorsiva)~~

~~(30') Calcolo derivata (ricorsiva)~~

~~(30') Struet (rilievi altimetrici) (ex Es6)~~

~~(30') Struet (film) (ex Es7)~~

~~(30') Vettori, vettori everywhere! (ex Es1.2)~~