

 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Ricorsione ed ordine superiore

Matteo Ferroni
matteo.ferroni@polimi.it

17/01/2017

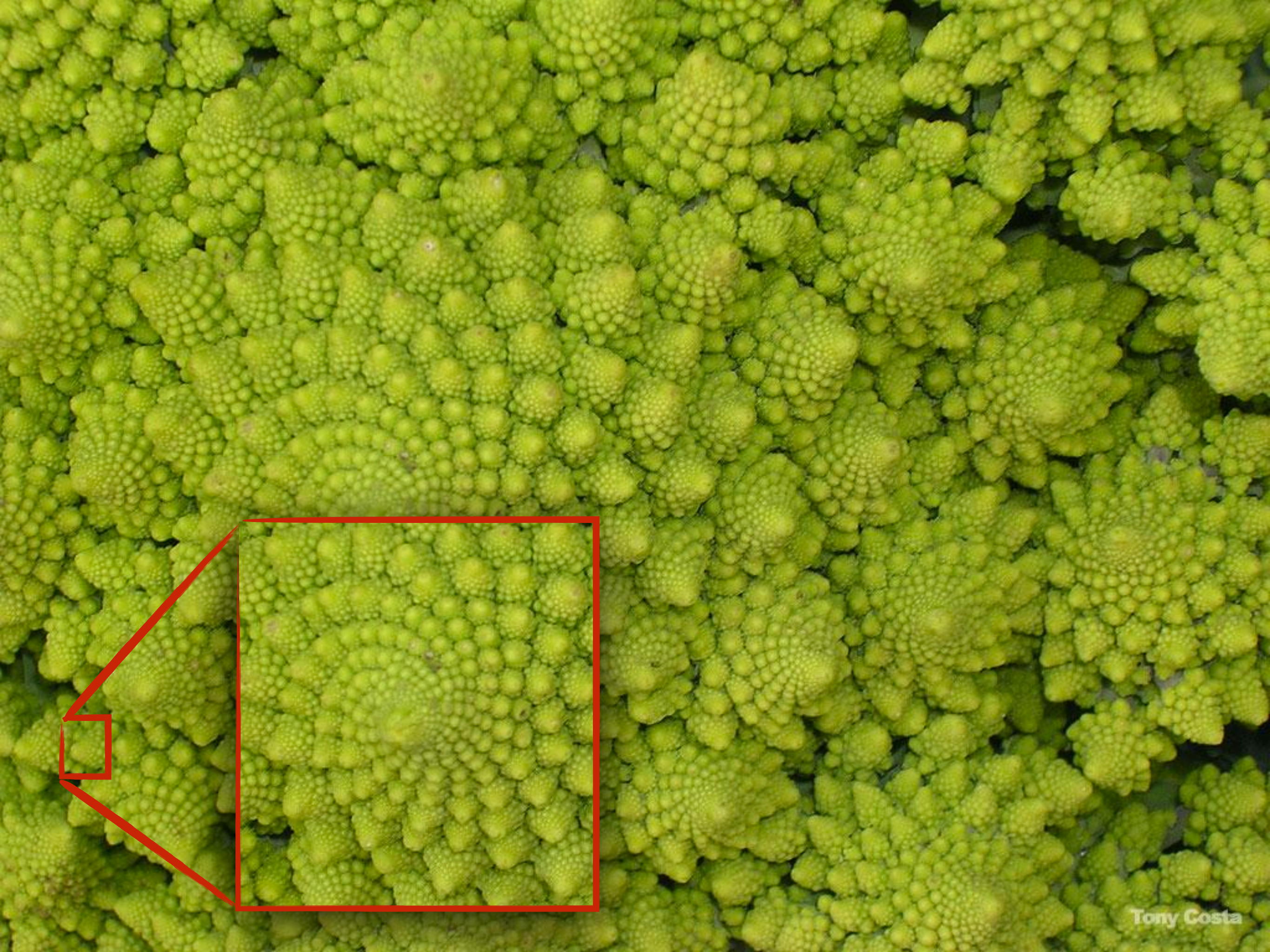


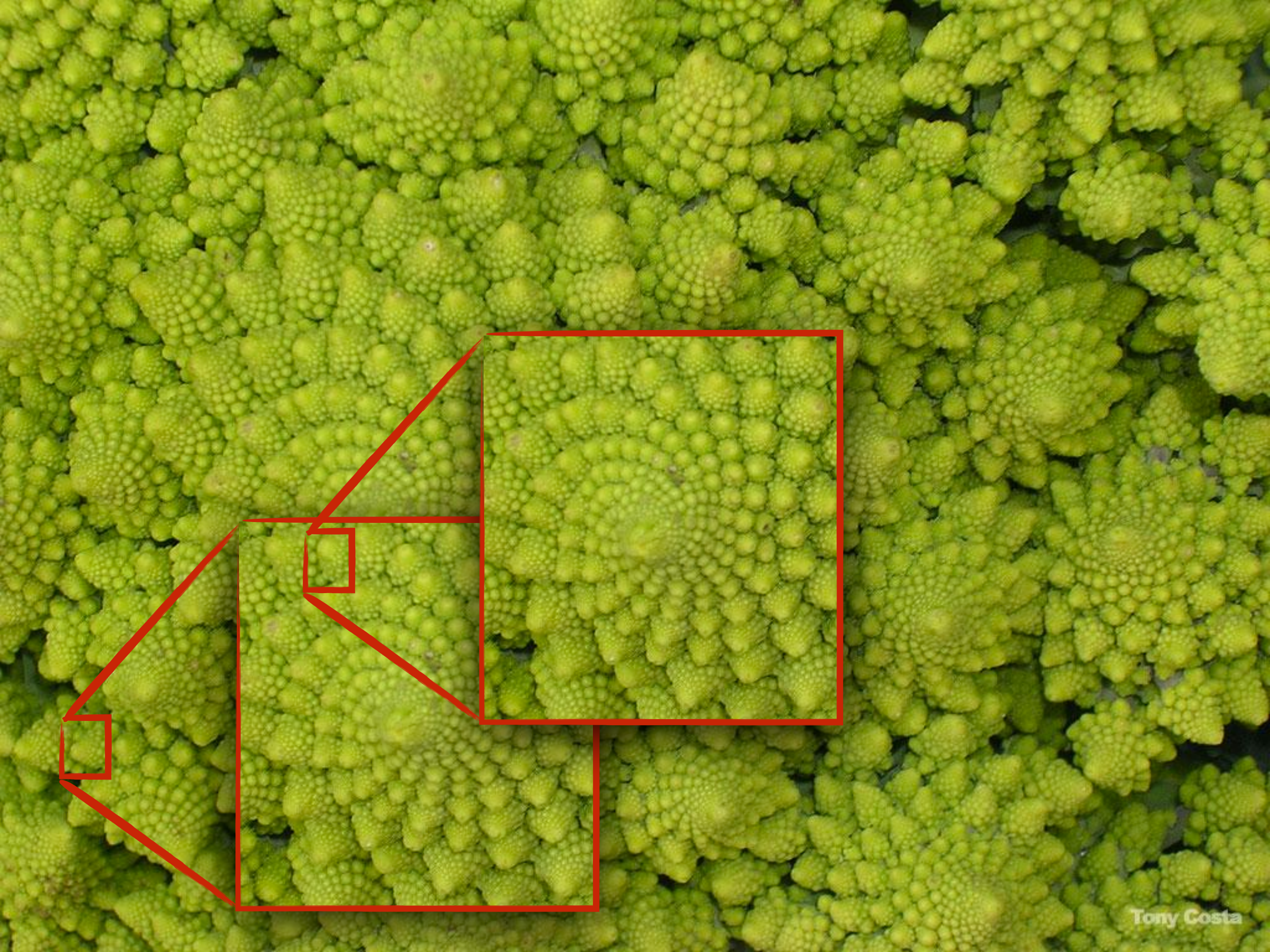
POLITECNICO
DI MILANO

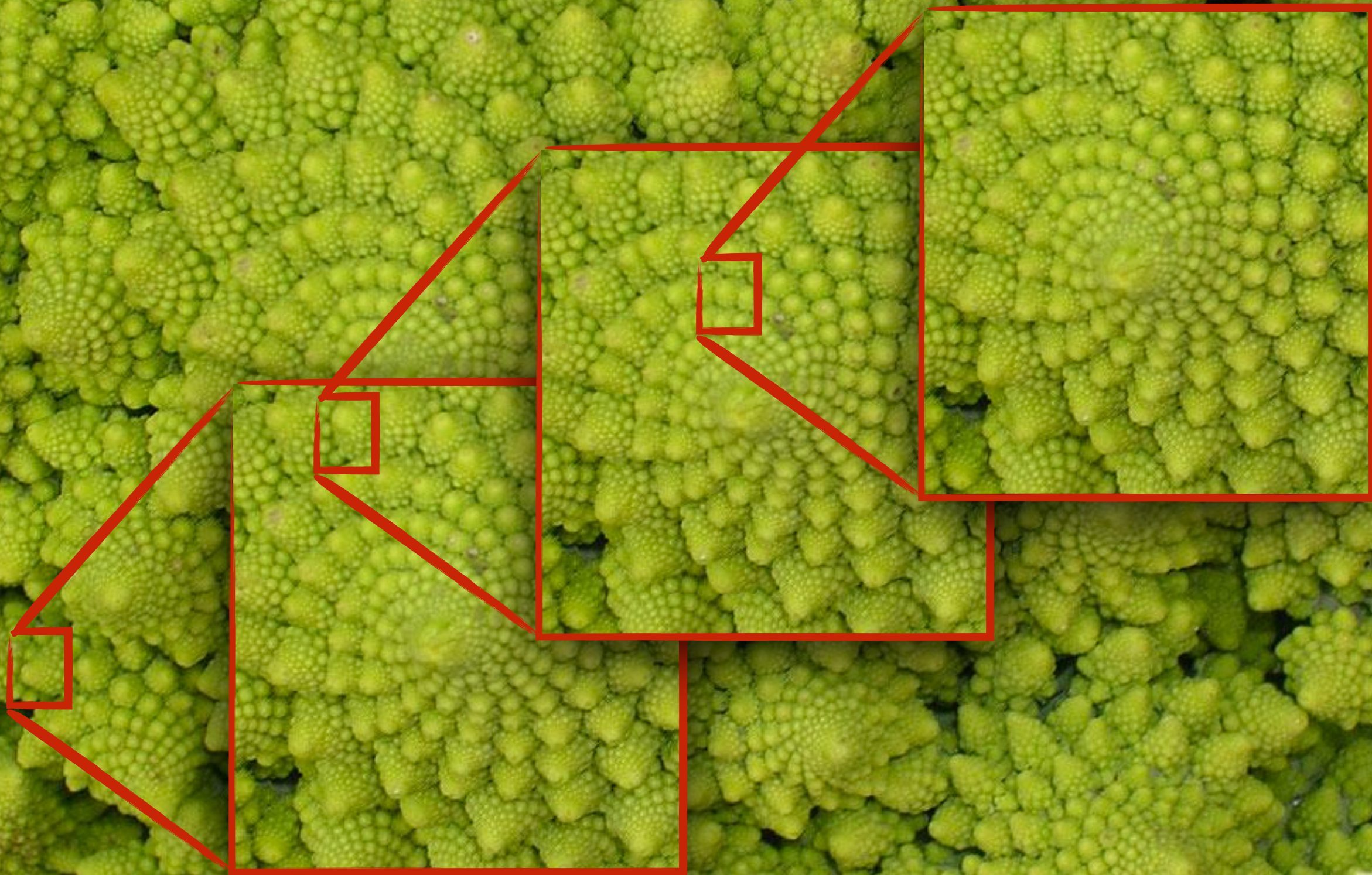
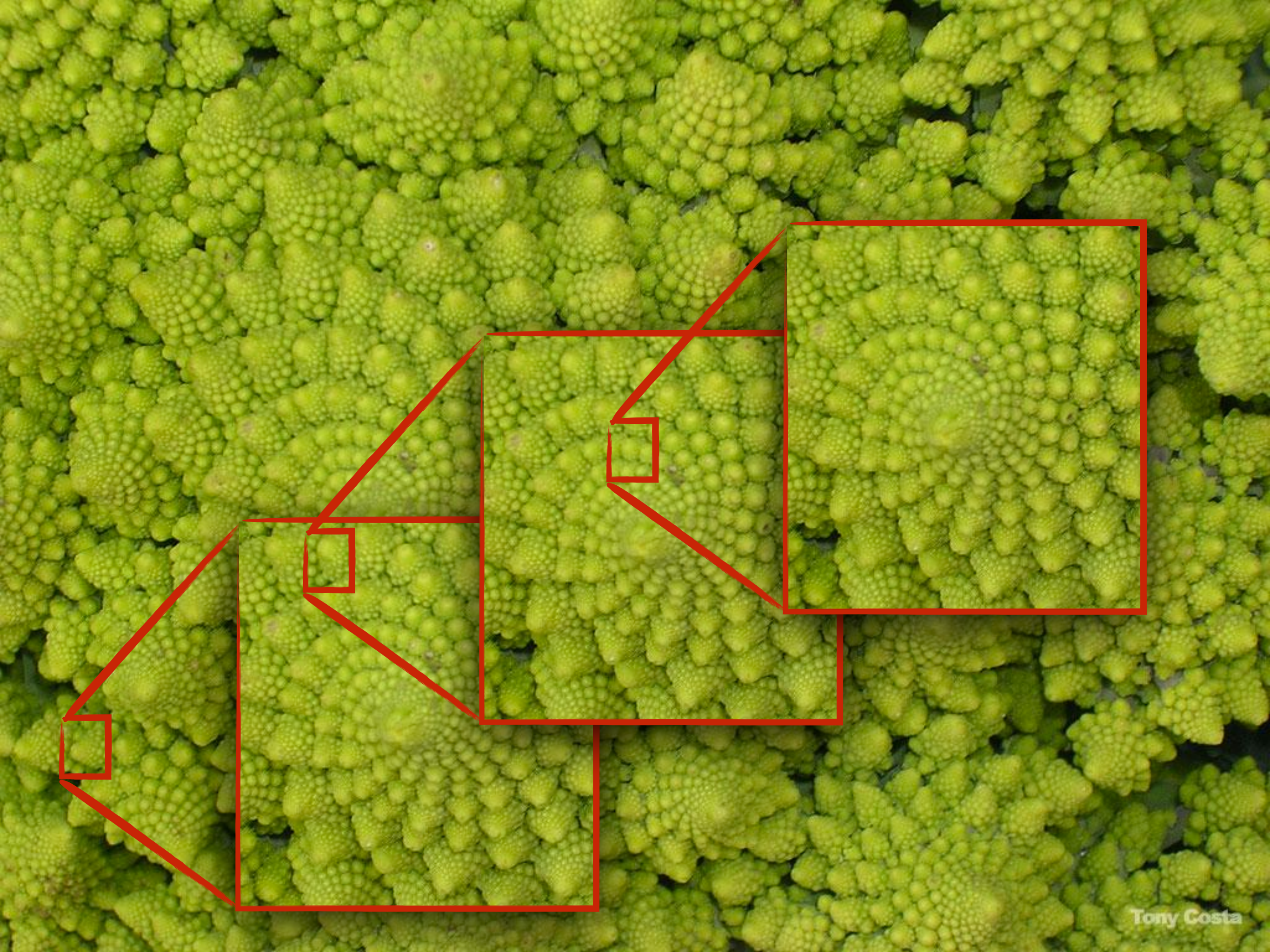












Agenda

(20') Sottomatrici, da un altro punto di vista

(30') Ricerca binaria (ricorsiva)

(30') Calcolo derivata (ricorsiva)

(30') Numero primo, in tutte le salse

(30') Funzioni di ordine superiore - *mag*

Sottomatrici, da un altro punto di vista

- Creare una matrice di dimensione **$2n \times 2n$** che nel suo centro contiene un quadrato 2×2 che contiene il valore 1 e, andando verso l'esterno, i valori 2, 3, .. fino a **n** nella cornice più esterna.

Esempio:

$$\begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 6 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 6 \\ 6 & 5 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 3 & 3 & 3 & 3 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 2 & 2 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 3 & 3 & 3 & 3 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 5 & 6 \\ 6 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 6 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

[illegible]

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

[illegible]

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

```
if(n == 1)
```

```
M = ones(2,2);
```

[illegible]

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

```
if(n == 1)
```

```
M = ones(2,2);
```

```
else
```

```
M = sottoMatr(n-1);
```

[illegible]

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

```
if(n == 1)
```

```
M = ones(2,2);
```

```
else
```

```
M = sottoMatr(n-1);
```

```
r = ones(1, size(M, 2)) * n;
```

$$M = [r; M; r];$$
[illegible]

Sottomatrici, versione 1

```
function [M] = sottoMatr(n)
```

```
if(n == 1)
```

```
    M = ones(2,2);
```

```
else
```

```
    M = sottoMatr(n-1);
```

```
    r = ones(1, size(M, 2)) * n;
```

```
    M = [r; M ; r];
```

```
    c = ones(size(M, 1), 1) * n;
```

```
    M = [c M c];
```

```
end
```

6	6	6	6	6	6	6	6	6	6	6	6
6	5	5	5	5	5	5	5	5	5	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6

Sottomatrici, versione 2

6	6	6	6	6	6	6	6	6	6	6	6
6	5	5	5	5	5	5	5	5	5	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6

```
function [matrRis]=sottoMatrici2(n)
```


Sottomatrici, versione 2

6	6	6	6	6	6	6	6	6	6	6	6
6	5	5	5	5	5	5	5	5	5	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6

```
function [matRis]=sottoMatrici2(n)
```

```
if n==1
```

```
    matRis=ones(2);
```


Sottomatrici, versione 2

6	6	6	6	6	6	6	6	6	6	6	6
6	5	5	5	5	5	5	5	5	5	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6

```
function [matRis]=sottoMatrici2(n)
```

```
if n==1
```

```
    matRis=ones(2);
```

```
else
```

```
    matRis=n*ones(2*n);
```


Sottomatrici, versione 2

6	6	6	6	6	6	6	6	6	6	6	6
6	5	5	5	5	5	5	5	5	5	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	1	1	2	3	4	5	6
6	5	4	3	2	2	2	2	3	4	5	6
6	5	4	3	3	3	3	3	3	4	5	6
6	5	4	4	4	4	4	4	4	4	5	6
6	5	5	5	5	5	5	5	5	5	5	6
6	6	6	6	6	6	6	6	6	6	6	6

```
function [matRis]=sottoMatrici2(n)
```

```
if n==1
```

```
    matRis=ones(2);
```

```
else
```

```
    matRis=n*ones(2*n);
```

```
    matRis(2:2*n-1,2:2*n-1)=sottoMatrici2(n-1);
```

```
end
```


Agenda

~~(20') Sottomatrici, da un altro punto di vista~~

(30') Ricerca binaria (ricorsiva)

(30') Calcolo derivata (ricorsiva)

(30') Numero primo, in tutte le salse

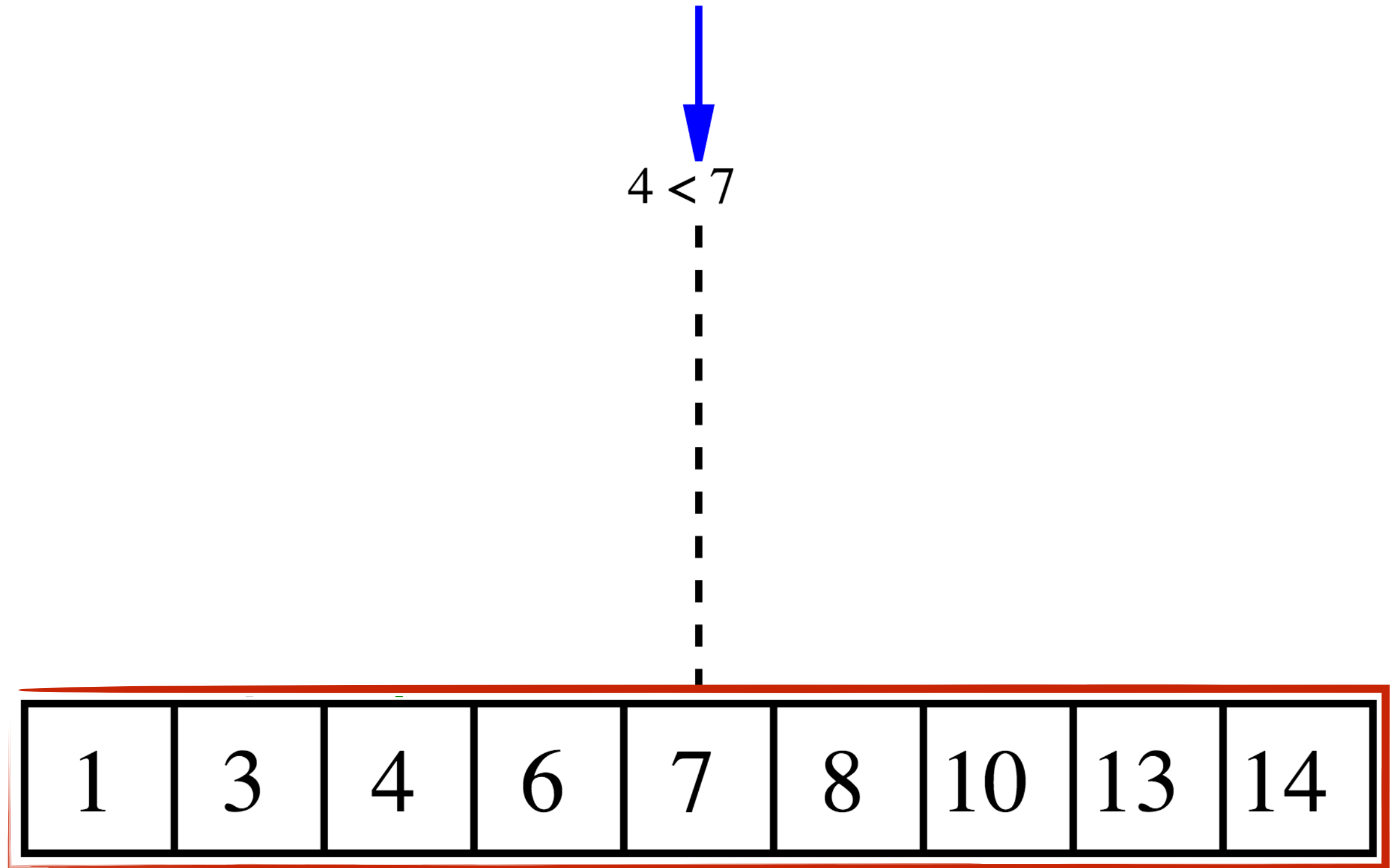
(30') Funzioni di ordine superiore - *mag*

Esercizio: ricerca binaria (ricorsiva)

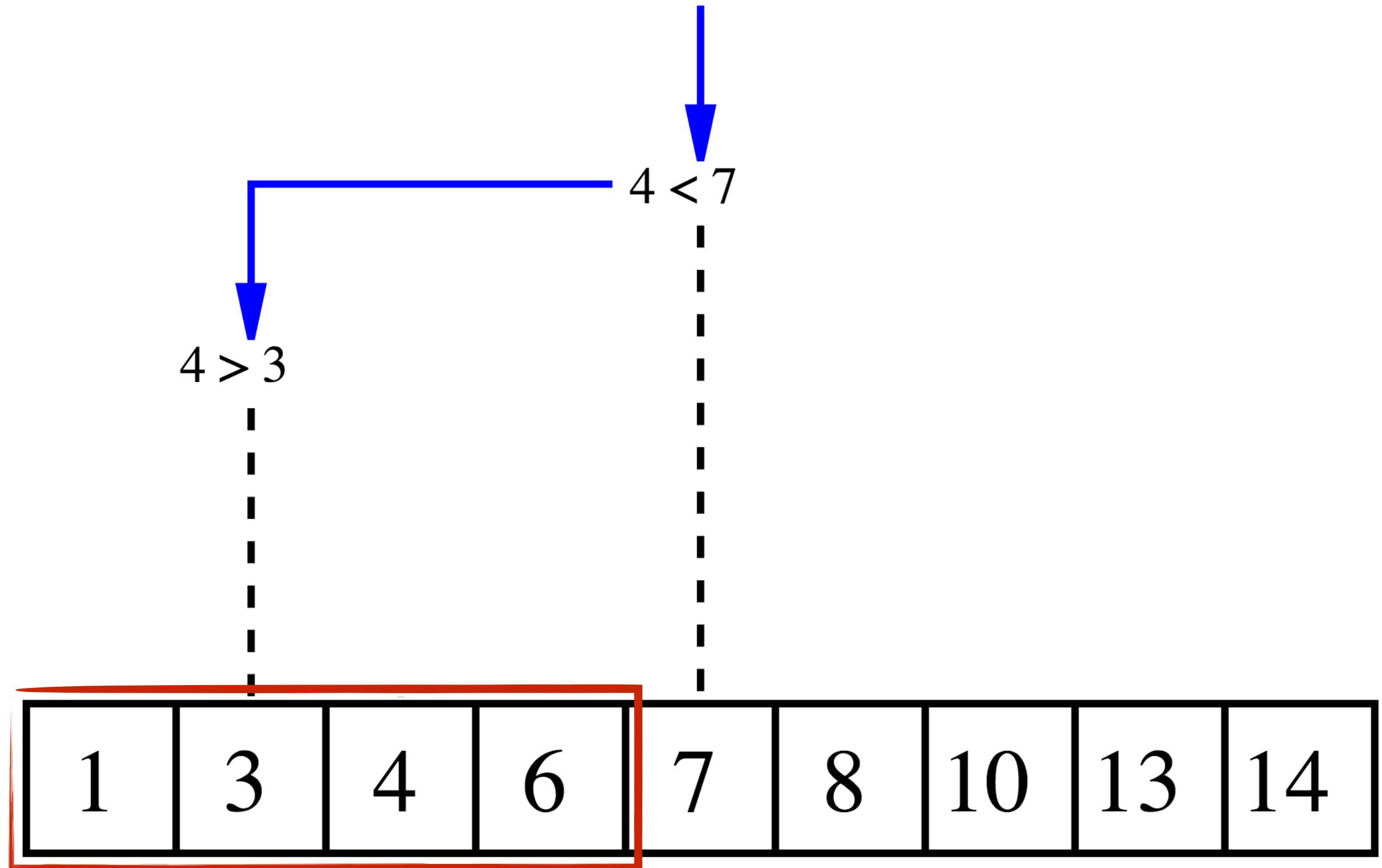
- Effettuare una **ricerca binaria** di un valore all'interno di un array ordinato. Il valore F é intero e l'array a é di interi.

*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***

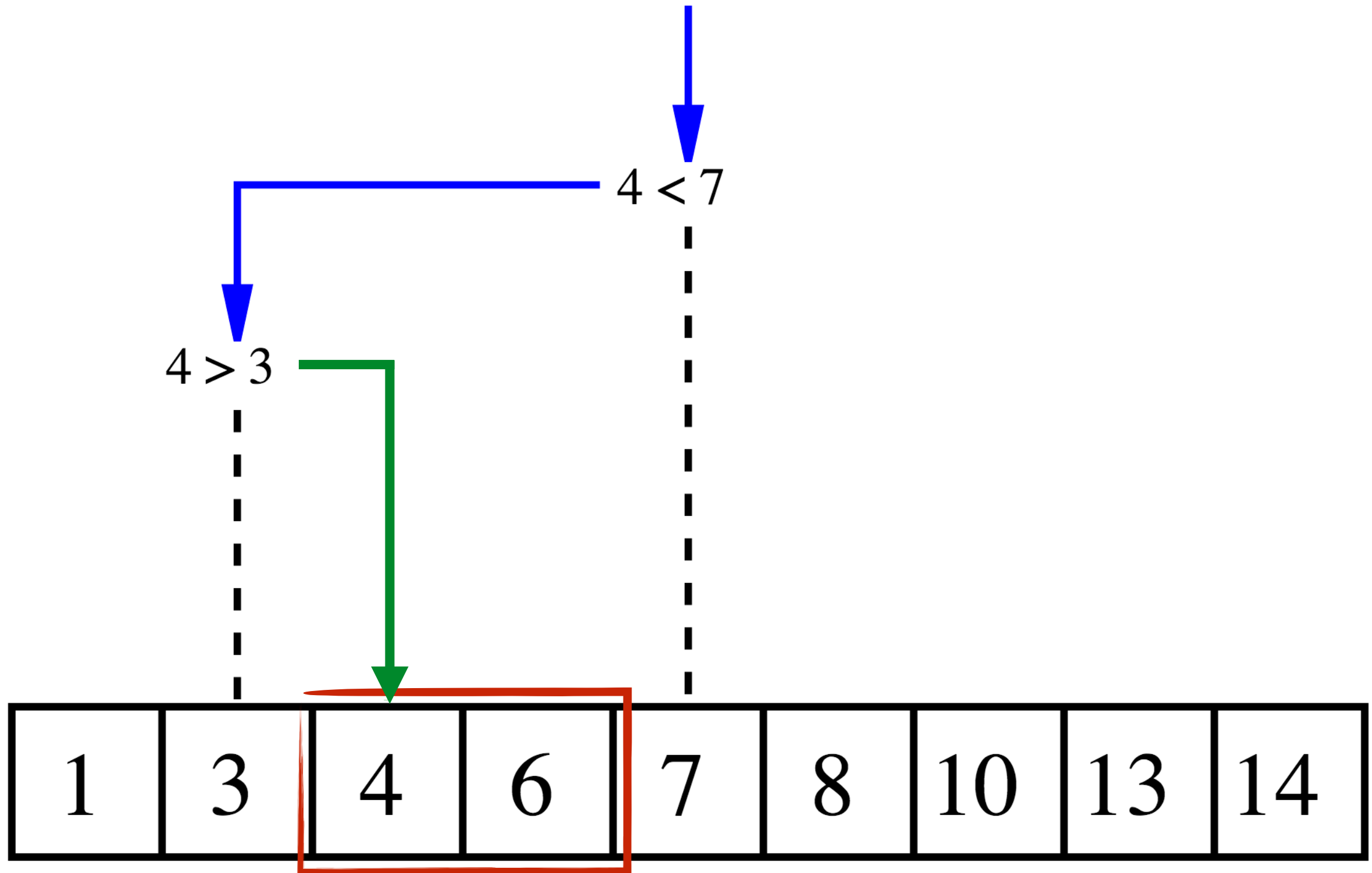
Esercizio: ricerca binaria (ricorsiva)



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
```


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

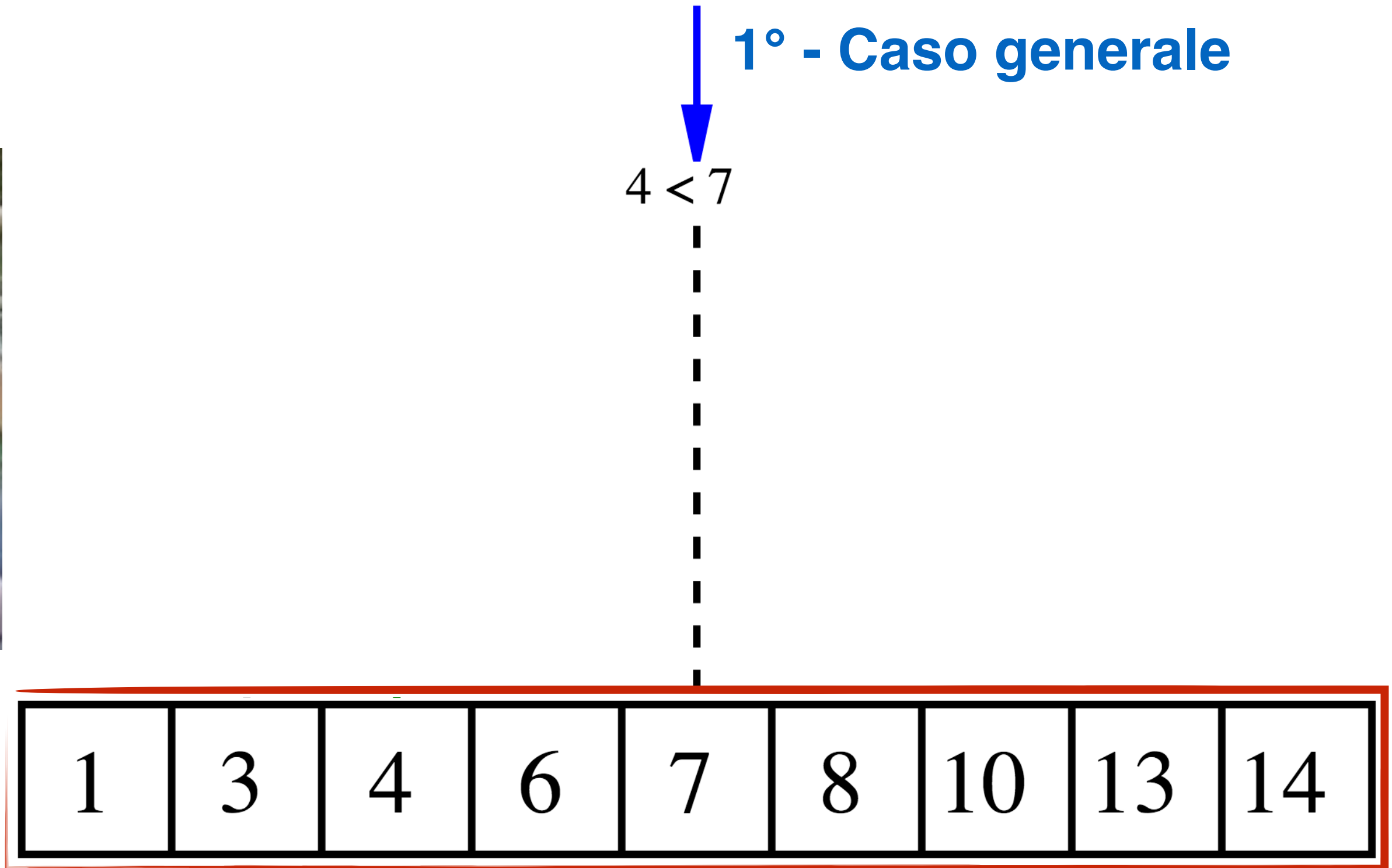
```
        val = ricercaBinaria(v, cercato, limSinistra, limDestra);
```

```
    end
```


Esercizio: ricerca binaria (ricorsiva)

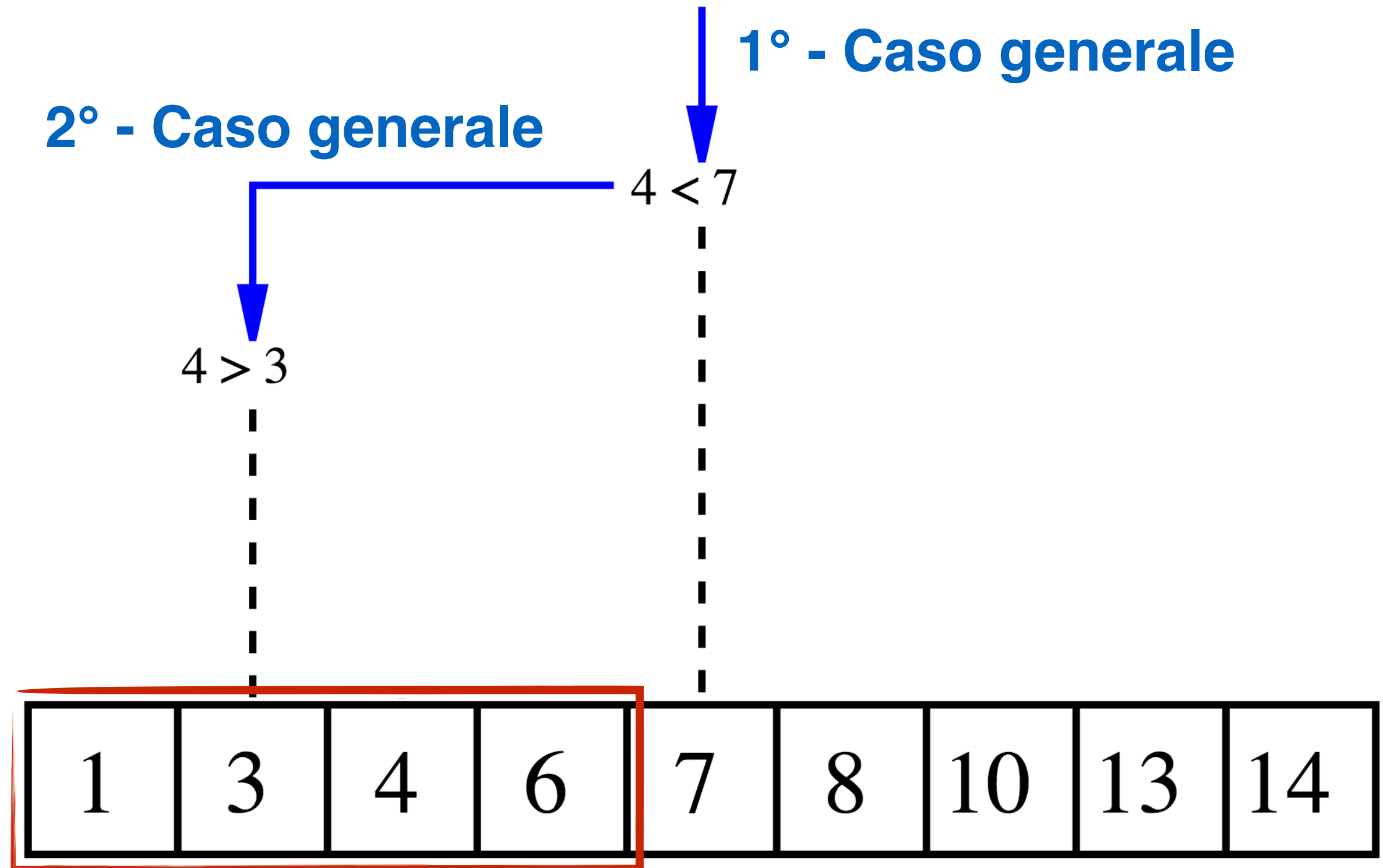
1° - Caso generale

4 < 7

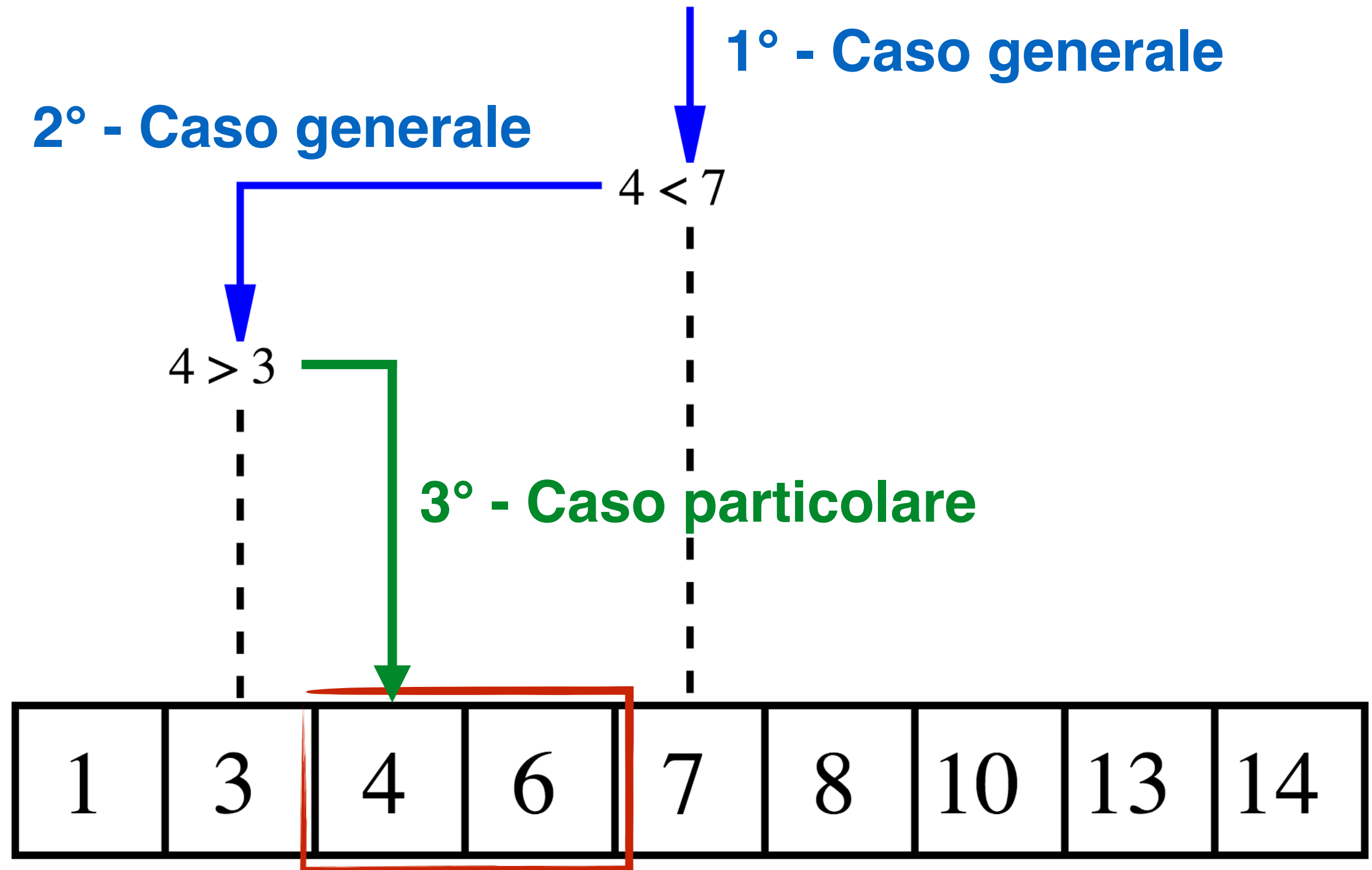


1	3	4	6	7	8	10	13	14
---	---	---	---	---	---	----	----	----

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

~~(20') Sottomatrici, da un altro punto di vista~~

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

(30') Calcolo derivata (ricorsiva)

(30') Numero primo, in tutte le salse

(30') Funzioni di ordine superiore - *mag*

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)

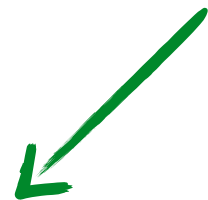
5 x^5 + **4** x^4 + **3** x^3 + **2** x^2 + **1** x^1 + **1** (*grado 5*)
corrisponde al vettore [**5 4 3 2 1 1**] (*lunghezza 6*)

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)


$5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x^1 + 1$ (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)

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]

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

~~(20') Sottomatrici, da un altro punto di vista~~

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

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

(30') Numero primo, in tutte le salse

(30') Funzioni di ordine superiore - *mag*

Numero primo, in tutte le salse

Numero primo, in tutte le salse

- Dato un numero intero positivo inserito dall'utente, dire se tale numero è primo (stampa a video 1 se primo, 0 altrimenti).

Numero primo, in tutte le salse

- Dato un numero intero positivo inserito dall'utente, dire se tale numero è primo (stampa a video 1 se primo, 0 altrimenti).
- Un numero è primo se è divisibile solo per 1 e se stesso.

Numero primo, in tutte le salse

- Dato un numero intero positivo inserito dall'utente, dire se tale numero è primo (stampa a video 1 se primo, 0 altrimenti).
- Un numero è primo se è divisibile solo per 1 e se stesso.

Esempio:

- $7 \longrightarrow 1$
- $9 \longrightarrow 0$
- $9871 \longrightarrow ?$

Numero primo, in tutte le salse

- Dato un numero intero positivo inserito dall'utente, dire se tale numero è primo (stampa a video 1 se primo, 0 altrimenti).
- Un numero è primo se è divisibile solo per 1 e se stesso.

Esempio:

- $7 \longrightarrow 1$
- $9 \longrightarrow 0$
- $9871 \longrightarrow ?$

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
```


Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)

    for y=2:sqrt(x)

    end
```

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)

    end
```

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)

    end
```

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
    end
```

← un numero è **primo**
fino a **prova contraria!**

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

← un numero è **primo**
fino a **prova contraria!**

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

← un numero è **primo**
fino a **prova contraria!**

Numero primo, soluzione *iterativa*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

un numero è **primo**
fino a **prova contraria!**

basta che sia **0** ad una iterazione qualsiasi
per aver trovato la “**prova contraria**”

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```


```
function [ris] = numeroPrimoRicorsiva(x,y)
```

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
```

```
    ris = (~(rem(x,y)))*numeroPrimoRicorsiva(x,y+1);
```



Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
```

```
    ris = 1;
```

```
    ris = (~(rem(x,y)))*numeroPrimoRicorsiva(x,y+1);
```

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
    if y>sqrt(x)
        ris = 1;
    else
        ris = (~(rem(x,y)))*numeroPrimoRicorsiva(x,y+1);
    end
```

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
    if y>sqrt(x)
        ris = 1;
    else
        ris = (~(rem(x,y)))*numeroPrimoRicorsiva(x,y+1);
    end
```


Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
    if y>sqrt(x)
        ris = 1;
    else
        ris = ( ~(rem(x,y)) ) * numeroPrimoRicorsiva(x,y+1);
    end
```

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
    if y>sqrt(x)
        ris = 1;
    else
        ris = (rem(x,y) == 0) * numeroPrimoRicorsiva(x,y+1);
    end
```

Numero primo, soluzione *ricorsiva*

```
function [ris] = numeroPrimoIterativa(x)
    ris=1;
    for y=2:sqrt(x)
        ris = ris*(~(rem(x,y)));
    end
```

```
function [ris] = numeroPrimoRicorsiva(x,y)
    if y>sqrt(x)
        ris = 1;
    else
        ris = ( ~(rem(x,y)) ) * numeroPrimoRicorsiva(x,y+1);
    end
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)  
    divisori = 2:sqrt(x);
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)  
    divisori = 2:sqrt(x);
```


Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```

Numero primo, soluzione *vettoriale*

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```

Numero primo, soluzione *vettoriale* (compatta)

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```



Equivalente a

```
function [ris] = numeroPrimoVettoriale(x)
    ris = ~any(rem(x,2:sqrt(x)) == 0);
```

Numero primo, soluzione *vettoriale* (compatta)

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```



Equivalente a

```
function [ris] = numeroPrimoVettoriale(x)
    ris = ~any(rem(x,2:sqrt(x)) == 0);
```


Numero primo, soluzione *vettoriale* (compatta)

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```



Equivalente a

```
function [ris] = numeroPrimoVettoriale(x)
    ris = ~any(rem(x,2:sqrt(x)) == 0);
```

Numero primo, soluzione *vettoriale* (compatta)

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```



Equivalente a

```
function [ris] = numeroPrimoVettoriale(x)
    ris = ~any(rem(x,2:sqrt(x)) == 0);
```

Numero primo, soluzione *vettoriale* (compatta)

```
function [ris] = numeroPrimoVettoriale(x)
    divisori = 2:sqrt(x);
    resti = rem(x,divisori);
    maschera = (resti == 0);
    ris = ~any(maschera);
```

Equivalente a

```
function [ris] = numeroPrimoVettoriale(x)
    ris = ~any(rem(x,2:sqrt(x)) == 0);
```

Esempio di invocazione

```
numeroDaControllare = 9871;  
numeroPrimoIterativa(numeroDaControllare)  
numeroPrimoRicorsiva(numeroDaControllare,2)  
numeroPrimoVettoriale(numeroDaControllare)
```

Agenda

~~(20') Sottomatrici, da un altro punto di vista~~

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

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

~~(30') Numero primo, in tutto le salse~~

(30') Funzioni di ordine superiore - *mag*

Funzioni di ordine superiore - *mag*

Funzioni di ordine superiore - *mag*

- Si scriva una ***funzione di ordine superiore mag*** che riceve in ingresso un *handle f* ed un vettore *v*.

Funzioni di ordine superiore - *mag*

- Si scriva una ***funzione di ordine superiore mag*** che riceve in ingresso un *handle f* ed un vettore *v*.
- L'***handle f*** è una funzione che prende in ingresso un valore numerico e restituisce un valore numerico.

Funzioni di ordine superiore - *mag*

- Si scriva una **funzione di ordine superiore *mag*** che riceve in ingresso un *handle f* ed un vettore *v*.
- L'***handle f*** è una funzione che prende in ingresso un valore numerico e restituisce un valore numerico.
- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.

Funzioni di ordine superiore - *mag*

- Si scriva una **funzione di ordine superiore *mag*** che riceve in ingresso un *handle f* ed un vettore *v*.
- L'***handle f*** è una funzione che prende in ingresso un valore numerico e restituisce un valore numerico.
- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.
- Si mostri inoltre un esempio di uso della suddetta funzione *mag*.

Funzione *mag*

```
function r = mag(f, v)
```

- Si scriva una ***funzione di ordine superiore mag*** che riceve in ingresso un *handle f* ed un vettore *v*.

Funzione *mag*

```
function r = mag(f, v)
```

- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.

Funzione *mag*

```
function r = mag(f, v)

    for i=1:length(v)

    end

end
```

- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.

Funzione *mag*

```
function r = mag(f, v)
    r = v(1)
    for i=1:length(v)

        end

    end
```

- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.

Funzione *mag*

```
function r = mag(f, v)
    r = v(1)
    for i=1:length(v)
        if (f(v(i)) > f(r))
            r = v(i)
        end
    end
end
```

- *mag* applica la funzione *f* ad ogni elemento di *v* e restituisce *quello tra questi elementi che massimizza f*.

Esempio di *handle* f

$f = @(x) \log(x);$

Esempio di *handle* f

```
f = @(x) log(x);
```



```
function risultato = f(x)  
    risultato = log(x);  
end
```

Esempio di uso di *mag*

```
% Esempio di uso di mag
```

```
v = 2:100;
```

```
f = @(x) log(x);
```

```
risultato = mag(f,v)
```

Esempio di uso di *mag*

```
% Esempio di uso di mag
```

```
v = 2:100;
```

```
f = @(x) log(x);
```

```
risultato = mag(f,v)
```



risultato =

100

Esempio di uso di *mag*

```
% Esempio di uso di mag
```

```
v = 2:100;
```

```
f = @(x) log(x);
```

```
risultato = mag(f,v)
```



risultato =

100

```
g = @(x) -x^2 + 40*x;
```

```
risultato = mag(g,v)
```


Esempio di uso di *mag*

```
% Esempio di uso di mag
```

```
v = 2:100;
```

```
f = @(x) log(x);
```

```
risultato = mag(f,v)
```

risultato =

100

```
g = @(x) -x^2 + 40*x;
```

```
risultato = mag(g,v)
```

risultato =

20

Agenda

~~(20') Sottomatrici, da un altro punto di vista~~

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

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

~~(30') Numero primo, in tutto le salse~~

~~(30') Funzioni di ordine superiore *mag*~~