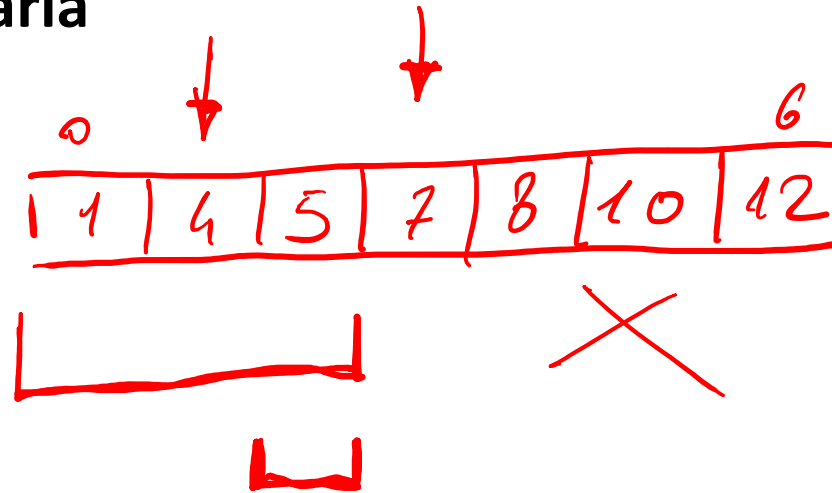


## Ricerca binaria



find (4)

find (5)

`int` binSearch\_it(`int` A[], `int` x, `int` l, `int` r)

{

`int` m = (l+r)/2;

`while` A[m] != x

{

`if` (x < A[m]) ~~sx~~

r = m-1;

`else` // x > A[m]

l = m+1;

`if` (l > r)

`return` -1;

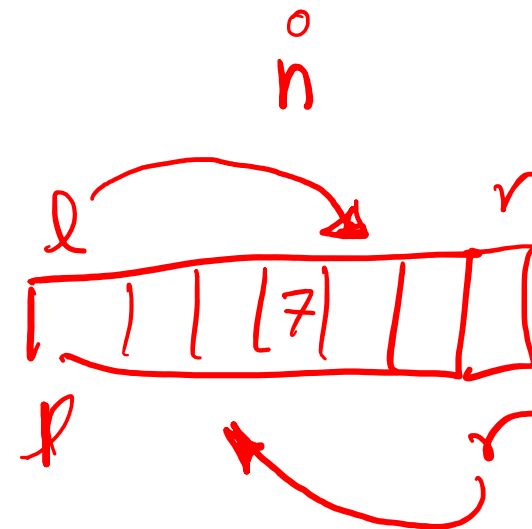
m = (l+r)/2;

}

`return` m;

}

## Ricerca Binaria (versione iterativa)



$$\begin{array}{ccc} n/2 & n/4 & n/8 \\ 1 & 2 & 3 \end{array}$$

$$i \rightarrow \frac{n}{2^i} = 1 \rightarrow i = \log(n)$$

$$\text{Order \& Search} = n^2 + \log(n) \rightarrow O(n^2)$$

## Ricerca Binaria (versione iterativa)

```
int binSearch_it(int A[], int x, int l, int r)
{
    int m = (l+r)/2;
    while( A[m] != x )
    {
        if(x < A[m])
            r = m-1;
        else // x > A[m]
            l = m+1;
        if(l > r)
            return -1;
        m = (l+r)/2;
    }
    return m;
}
```

- caso migliore  $O(1)$
- caso peggiore: al passo  $i$ , array ha dimension  $n/2^i$ .  
nel caso peggiore  $l=r \rightarrow n/2^i = 1 \rightarrow i = \log n$

## Esercizio 1

$$m = f(n)$$

$$E_{2a} \quad E_{2b} = O(f(n))$$

$$J \leq m = f(n)$$

Calcolare la complessità in funzione di  $n > 0$  del seguente frammento di programma

$\rightarrow$   $\{$   $\text{for } (\text{int } j=1 ; j \leq f(n) ; j++)$   $\}$   
 $\quad a += n$

con la seguente definizione di f:

```
int f (int n)
```

```
{
```

```
    int a=0;  $O(1)$ 
```

```
    for (int j=1; j<=n ; j++)
```

```
        a+=n;
```

```
    return a;  $O(1)$ 
```

```
}
```

$a = 0$   
 $a += n \Rightarrow n$   
 $a += n \Rightarrow 2n$   
 $a += n \Rightarrow 3n$

$n \cdot n$

$$C[E_1] + C[E_2] + \{C[C] + C[E_2] + C[E_3]\} \cdot f(n)$$

$$\downarrow$$

$$O(1) + O(f(n)) + \{O(1) + O(f(n)) + O(1)\} \cdot R(f(n))$$

$$C(\text{for}) = O(n) \rightarrow C(f(n)) = O(n)$$

$$R(f(n)) = O(n^2)$$

$$O(1) + O(n) + \{O(1) + O(n) + O(1)\} \cdot O(n^2)$$

$$O(n) + O(n^3) \Rightarrow \in O(n^3)$$

# Esercizio 1

Calcolare la complessità in funzione di  $n > 0$  del seguente frammento di programma

```
for (int j=1 ; j<=f(n) ; j++)  
    a+=n
```

con la seguente definizione di f:

```
int f (int n)  
{  
    int a=0;  
    for (int j=1; j<=n ; j++)  
        a+=n;  
    return a;  
}
```

$C [ \text{for} ( E1; E2; E3 ) C ] =$

$C [ E1 ] + C [ E2 ] + ( C [ C ] + C [ E2 ] + C [ E3 ] )$

$O( g(n) )$

- numero di iterazioni del for =  $[n(n+1)]/2 = \text{Risultato}[f(n)]$   
 $O(n^2)$
- complessità di una iterazione del for = Complessità della chiamata a f
- complessità di  $f = O(1) + O(n) + O(1) = O(n)$
- complessità del for:
  - numero iterazioni \* complessità di una iterazione =  $O(n^2) * O(n) = O(n^3)$

## Esercizio 2

Dire, per ogni coppia di funzioni fra quelle definite sotto, se una è  $O$  dell'altra oppure no.

$$f(n) \neq O(g(n))$$

$$f(n) = \begin{cases} 3n^3 + 3n & \text{se } n \text{ \u00e9 primo} \\ n & \text{altrimenti} \end{cases}$$

$$f(n) = O(g(n))$$

$$[n_0=1, c=6]$$

$$f(n) = O(h(n))$$

$$[n_0=51, c=6]$$

$$g(n) = \begin{cases} 4n^3 & \text{se l'ultima cifra di } n \text{ \u00e9 } 0 \text{ o } 5 \\ n^3 & \text{altrimenti} \end{cases}$$

$$g(n) \neq O(f(n))$$

esistono infiniti numeri composti

$$g(n) = O(h(n))$$

$$[n_0=51, c=4]$$

$$h(n) = \begin{cases} 4n^2 & \text{se } n \text{ \u00e9 divisor di } 50 \\ n^3 & \text{altrimenti} \end{cases}$$

$$f(n) = O(h(n))$$

$$n_0 = 51$$

$$h(n) \neq O(f(n))$$

esistono infiniti numeri composti

$$h(n) = O(g(n))$$

$$[n_0=1, c=1]$$

## Esercizio 3

Calcolare la complessità in funzione di  $n \geq 0$  della seguente funzione:

```
int g (int n) {
```

```
    int a=n;
```

```
    if (n<=500)
```

```
        for (int i=1 ; i<=n; i++)
```

```
            for (int j=1 ; j<=n ; j++)
```

```
                a+=n;
```

```
    else
```

```
        for (int i=1 ; i<=n ; i++)
```

```
            a+=n;
```

```
    return a;
```

```
}
```

$\left[ \begin{array}{l} \text{for } (i=1 ; i \leq n ; i++) \\ \text{for } (j=1 ; j \leq n ; j++) \\ a+=n; \end{array} \right] n \quad O(n^2) \quad \left\{ \begin{array}{l} 0 \dots 500 \end{array} \right.$

$\left[ \begin{array}{l} \text{for } (i=1 ; i \leq n ; i++) \\ a+=n; \end{array} \right] n \quad O(n) \quad \left\{ \begin{array}{l} 501 \dots \text{INT} \end{array} \right.$

$g(n) \in O(n) \quad n_0 = 501$



## Esercizio 4

Dato il seguente frammento di programma:

```
i=n;
while (i>=1)
{
    for (int j=1 ; j<=n ; j++)
        a++;
    i=i-1;
}
```

a)  $i = n ; i \geq 1 ; i -= 1$

$\left. \begin{array}{l} \text{for loop} \\ \text{while loop} \end{array} \right\} n$

calcolare la complessità in funzione di  $n > 0$  nei casi

- a)  $E = i-1 \rightarrow O(n^2)$
- b)  $E = i-n \rightarrow O(n)$
- c)  $E = i/2 \rightarrow O(n \log n)$

$$i = n$$


$$i = n/2$$

$$i = n/4$$

$$i = n/2^k$$




## Esercizio 5.a

Date le seguenti dichiarazioni di funzione, calcolare la complessità in funzione di  $n > 0$  della chiamata  $P(F(n), y)$ . 

```
int f(int n)
{
    int b;
    int a=0;
    for (int i=1 ; i<=n ; i++)
        for (int j=1 ; j<=n ; j++)
            a++;
    b= a;
    return 2*b;
}

void P (int m, int &x) {
    for (int i=1 ; i<=m*m ; i++)
        x+=3;
}
```



## Esercizio 5.b

Date le seguenti dichiarazioni di funzione, calcolare la complessità in funzione di  $n > 0$  della chiamata  $P(F(n), y)$ .

```
int f(int n)
{
    int b;
    int a=0;
    for (int i=1 ; i<=n ; i++)
        for (int j=1 ; j<=n ; j++)
            a++;
    [b= a/n;
    return 2*b;
}

void P (int m, int &x) {
    for (int i=1 ; i<=m*m ; i++)
        x+=3;
}
```

# Moltiplicazione fra matrici

```
void matrixMult(int A[N][N], int B[N][N], int C[N][N])
{
    for (int i=0; i < N; i++)
        for (int j=0; j < N; j++)
        {
            C[i][j] = 0;
            for (int k=0; k < N; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
}
```