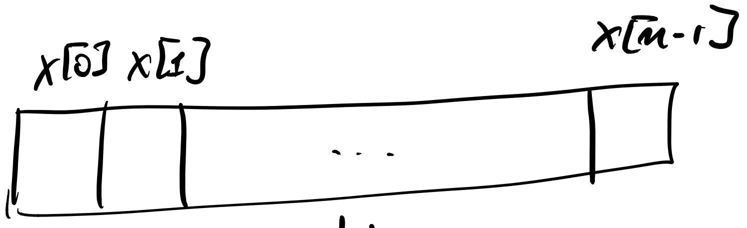
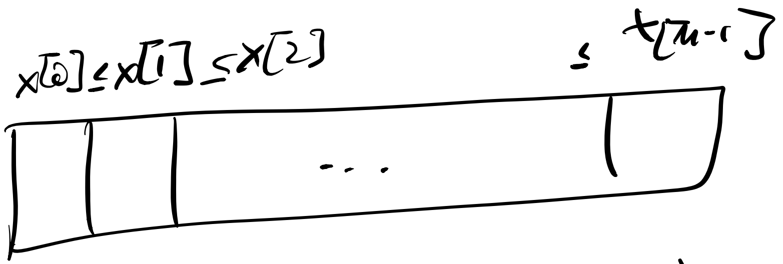
ORDINAMENTO DI SLICE

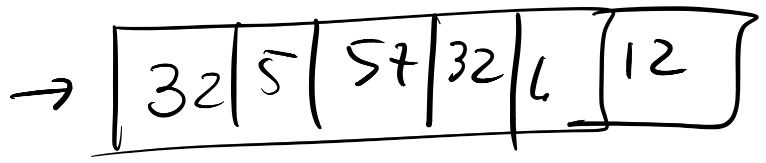
var x []int



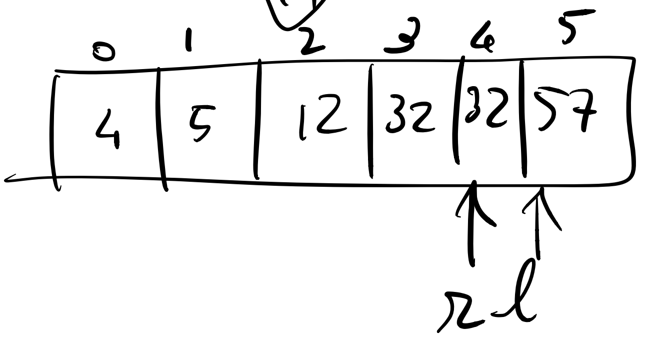
⇓



Donald Knuth
TAOCP



⇓



36

RICERCA

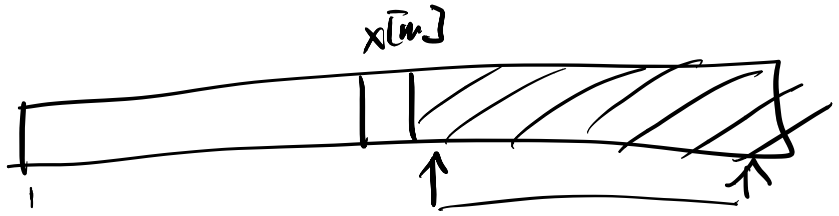
```
var  x  []int
var  y  int

func  search (x []int, y int) int {
        for   i, e := range x {
                  if  e == y {
                        return i
                  }
        }
        return -1
}
```

RICERCA LINEARE

slice lunga n

CASO PEGGIORE $\rightarrow$ n confronti
(quando l'elemento non c'è)



```
func  binary Search (x []int, y int) int{
      var left, right  int
      left = 0
      right = len(x) - 1
      for  left <=right {
              m:=(left+right)/2
              if  x[m] == y {
                      return  m
              } else   x[m] <y {
                      left = m+1

              } else  {
                      right  = m -1

              }
      }
      return  -1
}
```

Dato n, qual è il valore k per cui

$$\left.\frac{\left(\frac{\left(\frac{n}{2}\right)}{2}\right)}{2}\right\} k \text{ volte} \quad < 1$$

$$\Rightarrow \quad \frac{n}{2^k} < 1$$

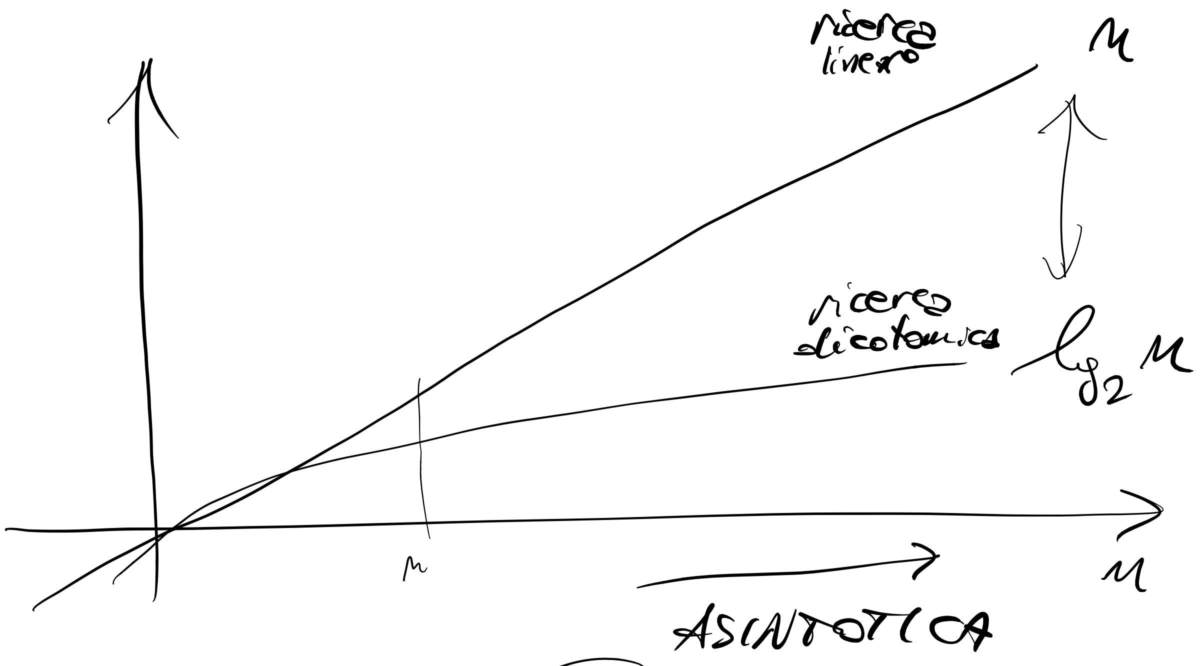
$$n < 2^k$$

$$\log_2 n < \log_2 2^k = k \log_2 2 = k$$

$$\log_2 n < k$$

$$\lceil \log_2 n \rceil$$

CASO PEGGIORE $\rightarrow \lceil \log_2 n \rceil$ CONFRONTI

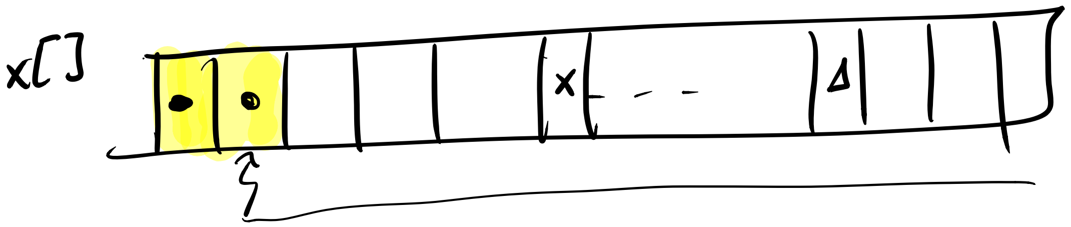


$$\lim_{n \to \infty} \frac{\log n}{n} = 0$$

ORDINAMENTO

- in-place
- non in-place

SELECTION SORT



```
func selectionSort(x []int) {
    // Posizione da sistemare
    for i=0; i<n; i++ {
        imin=i
        min = x[i]
        for j=i+1; j<n; j++ {
            if x[j]<min {
                imin=j
                min = x[j]
            }
        }
        x[i], x[imin] = x[imin], x[i]
    }
}
```

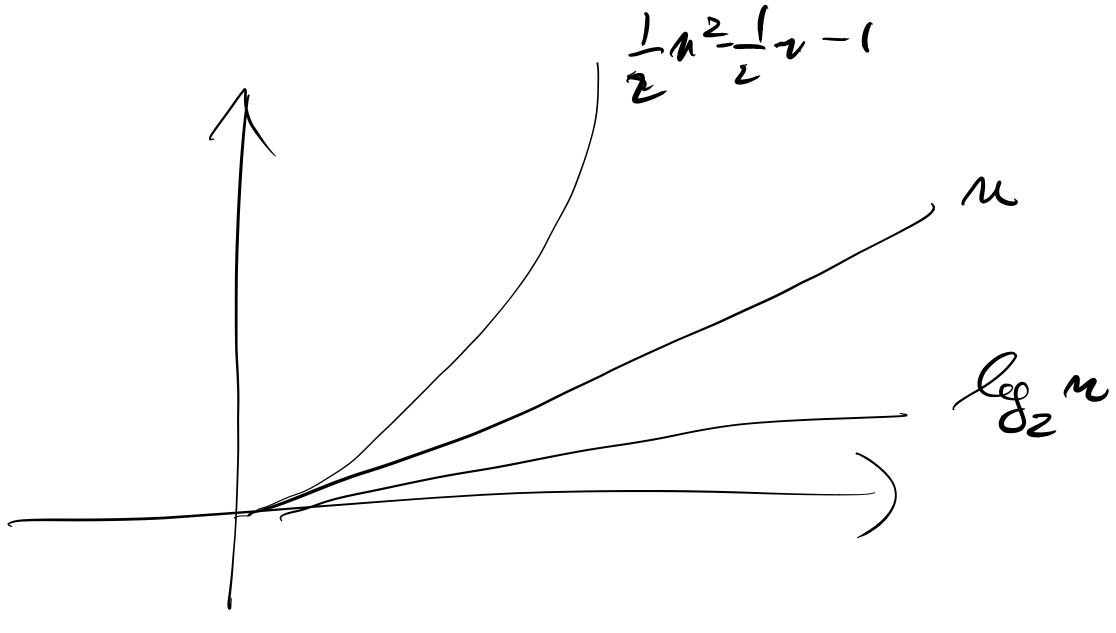
$n-1 \quad n-2 \quad n-3 \quad \dots \quad 2$

$$2+3+\dots+(n-1) = \sum_{i=1}^{n-1} i - 1 =$$

$$= \frac{(n-1)n}{2} - 1 = \frac{1}{2}n^2 - \frac{1}{2}n - 1$$

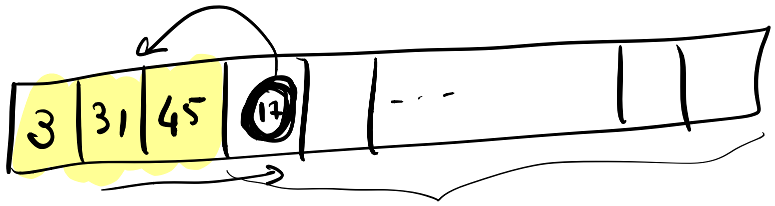
CASO PEGGIORE: n° confronti

$$\frac{1}{2}n^2 - \frac{1}{2}n - 1 = O(n^2)$$



INSERTION SORT

x[]



```
func insertionSort(x []int){
    ~~var~~ k int
    n = len(x)
    for i:= 1; i<n; i++ {
        j:=i+1   //Da inserire fra i primi
        for k:= 0; k<i; k++ {
            if x[k] > x[j]{
                break
            }
        }
        temp:= x[j]
        for h:= i; h>=k; h--{
            x[h+1]= x[h]
        }
        x[k] = temp
    }
}
```

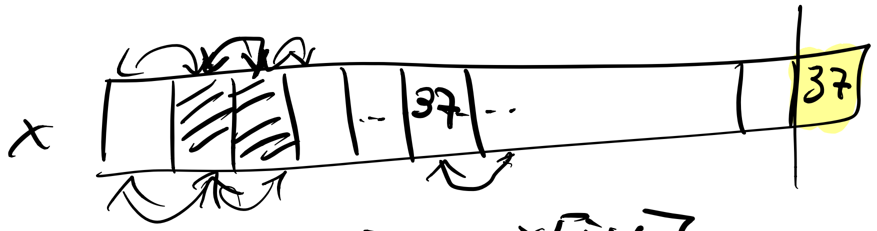
$$\underbrace{1 \quad 2 \quad 3 \quad \ldots \quad n-1}$$

$$1+2+3+\ldots+n-1 = \sum_{i=1}^{n-1} i = \frac{(n-1)(n)}{2} =$$

$$= \frac{1}{2}n^2 - \frac{1}{2}n$$

$$\boxed{\text{CASO PEGGIORE: } n^\circ \text{ assegnamenti } \quad \frac{1}{2}n^2 - \frac{1}{2}n}$$

BUBBLESORT



$\rightarrow$ x[i] <= x[i+1]

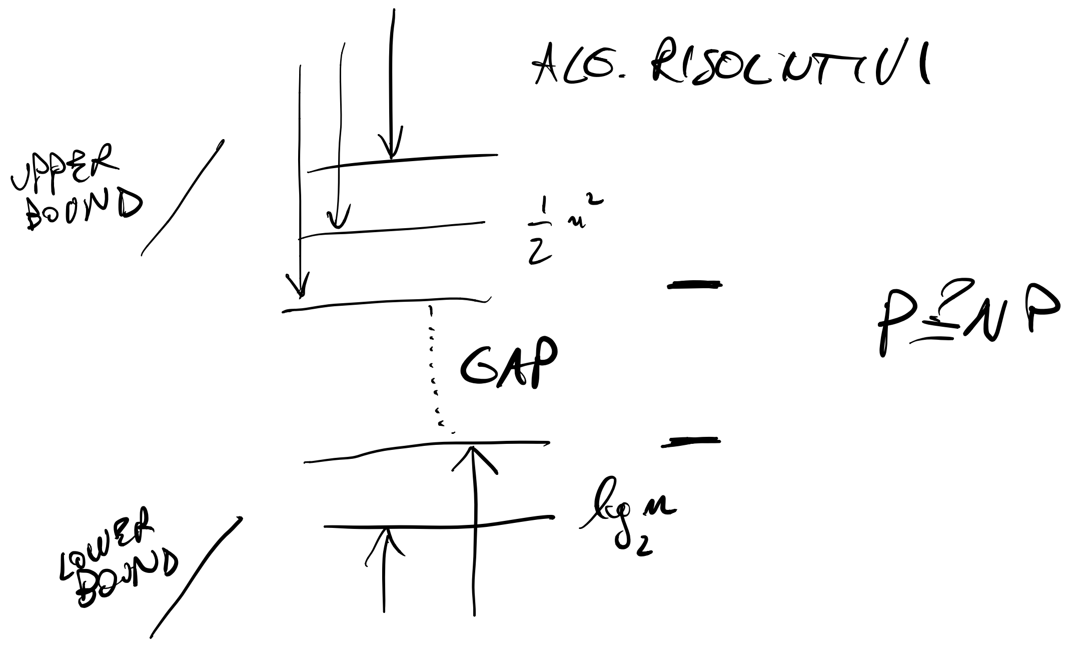
```
func bubbleSort(x []int) {
    for (j=n; j>0; j--) {
        for (i=0; i<j-1; i++) {
            if x[i] > x[i+1] {
                x[i], x[i+1] = x[i+1], x[i]
            }
        }
    }
}
```

$$(n-1) + (n-2) + \dots + 1 + 0 = $$

$$= \sum_{i=1}^{n-1} i = \frac{(n-1)\,n}{2} = \frac{1}{2} n^2 - \frac{1}{2} n$$

$$\boxed{\text{CASO PEGGIORE} \quad \frac{1}{2}n^2 - \frac{1}{2}n}$$

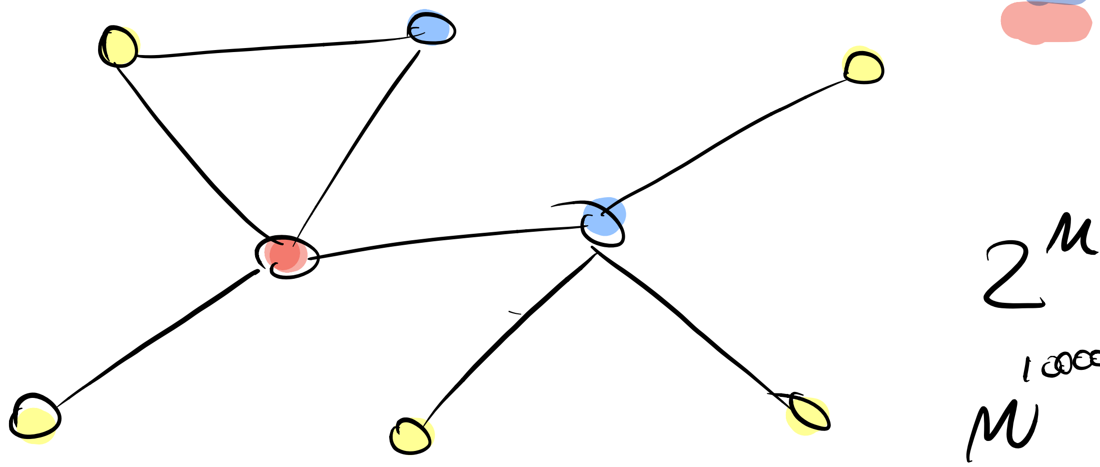
LOWER/UPPER BOUND



- ORDINAMENTO CON SCAMBI E CONFRONTI RICHIEDE $\Omega(n \log n)$
- ESISTONO ALGORITMI $O(n \log n)$
  - HEAPSORT
  - QUICKSORT

ALGORITMI NAIF $O(n^2)$

COLORABILITY ~ NP-completi

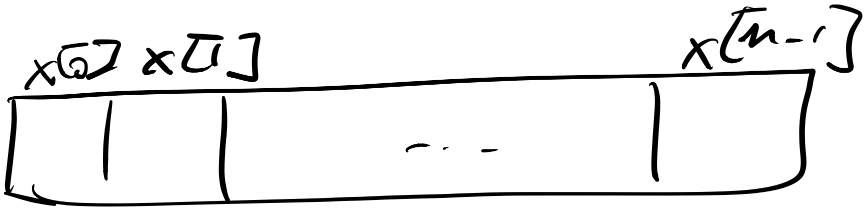


2^n

$n^{1000000}$

---

x [ ]



type persona struct{

nome, cognome string

dataDiNascita data

redditoAnnuo int

}

var x [ ]persona

$$\begin{tikzcd}
 & \text{stabile} \\
\text{algoritmi di ordinamento} \arrow[ur, dash] \arrow[dr, dash] & \\
 & \text{instabile}
\end{tikzcd}$$