ORDINAMENTO : limite inferiore

n elementi $a_1, a_2, \dots, a_n$ da ordinare

$a_i : a_j \quad \leq \quad \begin{matrix} < \\ > \\ = \end{matrix}$ unica operazione permessa (a parte copiare e spostare)

▶ Ogni algoritmo (corretto!) di ordinamento tra n elementi usando confronti, ne richiede $\Omega(n \lg n)$

$$\boxed{a_i : a_j \ ?}$$

LED ON

$$\begin{matrix} a_1, a_2, \dots, a_n \\ 18, 7, 22 \\ 3, 1, 7 \end{matrix} \longrightarrow \boxed{\quad A \quad} \longrightarrow \begin{matrix} \mathrm{perm}(a_1, a_2, \dots, a_n) \\ 7, 18, 22 \\ 1, 3, 7 \end{matrix}$$

Oss. Vale l'ordine relativo tra le chiavi $\Rightarrow$
(A usa confronti tra coppie di chiavi)

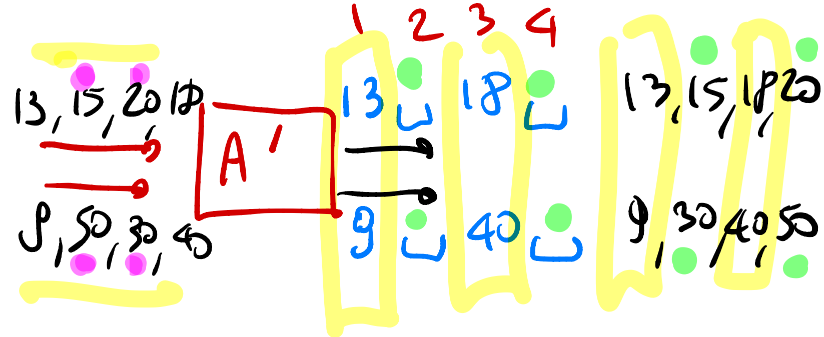
pur essendoci infinite sequenze $a_1, a_2, \dots, a_n$ d'ingresso, ci sono "soltanto" $n!$ ordini relativi

$$18, 7, 22 \qquad \qquad 7, 18, 22$$
$$3, 1, 7 \quad \rightsquigarrow \quad 1, 3, 7$$
$$2 \;\; 1 \;\; 3 \qquad \qquad 1 \;\; 2 \;\; 3$$

▶ Affinchè A sia corretto, deve essere in grado di distinguere tra le permutazioni.

A' "sbagliato"

1 2 4 3 } se confonde
1 4 2 3



▶ due permutazioni distinte hanno sempre una coppia invertita
π_1, π_2

a_i viene prima di a_j in π_1 2 7 3 6 1 8 5 4

a_j " " " a_i in π_2 4 1 5 6 3 7 2 8

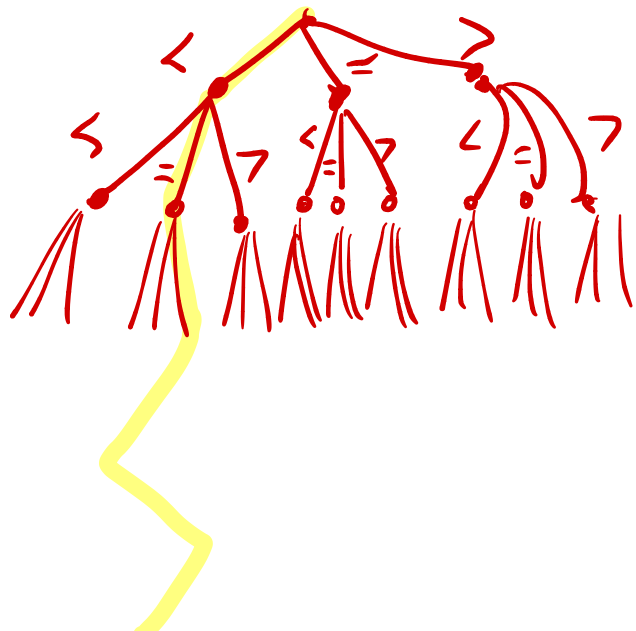
▶ A' confonde π_1 e π_2 ⇒ nell'ordinamento finale o colloca a_i prima di a_j (viola π_2) oppure colloca a_j prima di a_i (viola π_1)

Qualunque algoritmo corretto A che ordina mediante confronti, deve poter discriminare tra tutte le permutazioni di n elementi, che sono appunto $n!$

Algoritmo A

\# confronti

t	max #permutazioni che può distinguere
1	≤ 3
2	$\leq 3^2$
3	$\leq 3^3$
$\vdots$	
t	$\leq 3^t$
$\vdots$	



possibili risposte $= n!$

- certamente, se $3^t < n!$, allora A <u>non</u> può essere corretto

$\Rightarrow$ condizione necessaria $\boxed{3^t \geq n!}$

Prendendo $\underbrace{\log_3 3^t}_{t} \geq \log_3 n! = \Omega(n \lg n)$

- approssimazione di $n!$ con Stirling
- $n! \geq \left(\frac{n}{2}\right)^{n/2}$

```
SelectionSort( a ):                              ⟨pre: la lunghezza di a è n⟩
  FOR (i = 0; i < n-1; i = i+1) {
    minimo = a[i];
    indiceMinimo = i;
    FOR (j = i+1; j < n; j = j+1) {
      IF (a[j] < minimo) {
        minimo = a[j];
        indiceMinimo = j;
      }
    }
    a[indiceMinimo] = a[i];
    a[i] = minimo;
  }
```

minimo in $a[i..n-1]$, $a[0..i-1]$ sono già quelli finali

$O(1)$

$n-i$

$\Theta(n^2)$

$\Theta\left(\sum_{i=1}^{n-1} (n-i)\right)$

$tmp = A[i]$

$A[i] = A[j]$

$A[j] = tmp$

minimo == D

A = D A C G F L

A D C G F L

min

C B A D   A B C D

perm π 3 2 1 4 $\rightsquigarrow$ 1 2 3 4

confront:

C : B , B : A , A : D

A B C D

B : C , B : D

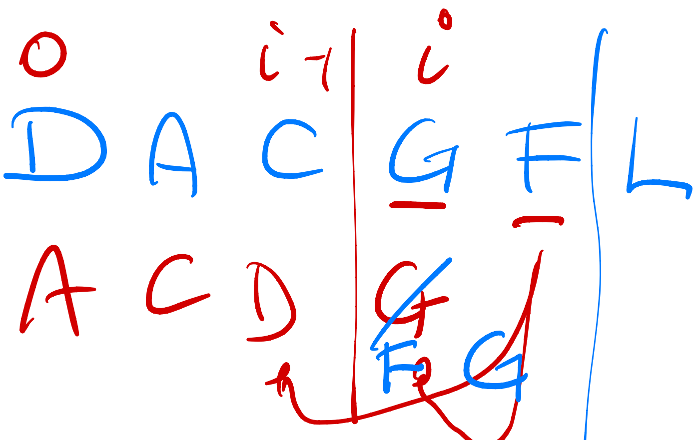
A B C D

C : D

```
InsertionSort( a ):                          ⟨pre: la lunghezza di a è n⟩
  FOR (i = 1; i < n; i = i+1) {
    prossimo = a[i];
    j = i;
    WHILE ((j > 0) && (a[j-1] > prossimo)) {
      a[j] = a[j-1];
      j = j-1;
    }
    a[j] = prossimo;
  }
```

j

a[0..i-1] è già ordinato ma non ci sono necessariamente i minimi

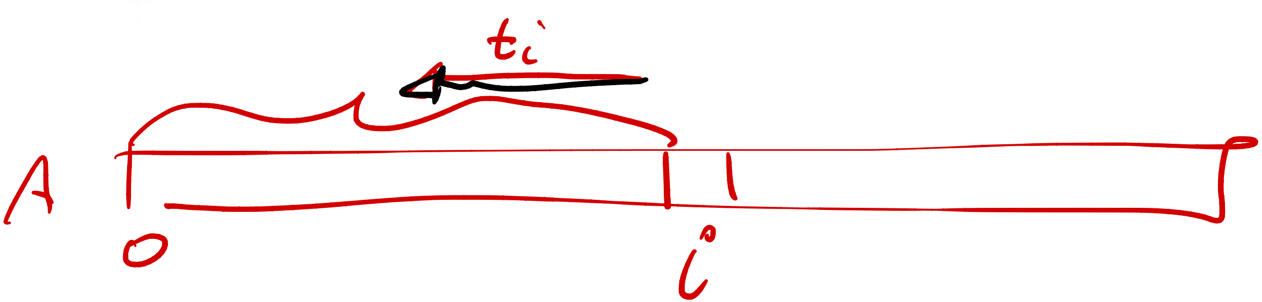


DCBA

```
InsertionSort( a ):                                  ⟨pre: la lunghezza di a è n⟩
  FOR (i = 1; i < n; i = i+1) {
    prossimo = a[i];
    j = i;
    WHILE ((j > 0) && (a[j-1] > prossimo)) {
      a[j] = a[j-1];
      j = j-1;
    }
    a[j] = prossimo;
  }
```

$O(1)$

$0 \le t_i \le i$



$$\text{costo} = O\left(n + \sum_{i=1}^{n-1} t_i\right)$$

caso pessimo $t_i = i \Rightarrow \text{costo} = O(n^2)$

caso ottimo $t_i = 0 \Rightarrow \text{costo} = O(n)$

caso medio $t_i \sim \frac{i}{2} \Rightarrow \text{costo} = O(n^2)$

media su
su $n!$ permutazioni

QUICKSORT

ordinamento 0-1 mediante confronti

$$\begin{array}{lcccccccccccccc} & F & M & B & H & A & Z & C & P & Q & D & T & N & R \\ A = & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}$$

pivot = L

$\boxed{\text{Partition}(A, 'L')}$

i ... j

solo 0 solo 1

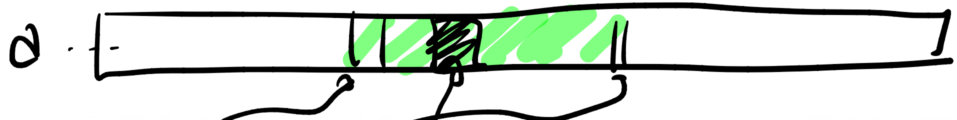
PARTITION

pivot = A[i]

O(n) tempo

0 se $A[i'] < A[i]$

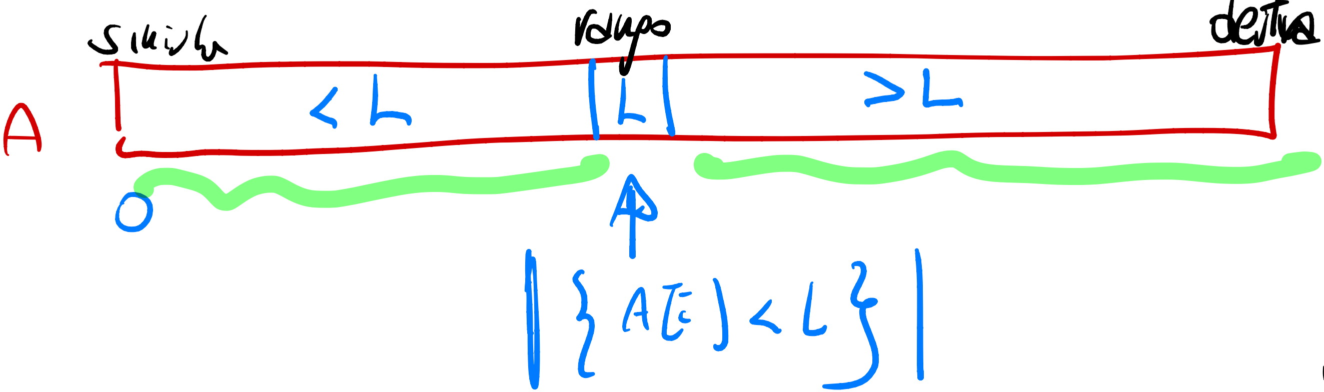
1 se $A[i'] > A[i]$




```
QuickSort( a, sinistra, destra ):
        ⟨pre: 0 ≤ sinistra, destra ≤ n − 1⟩
  IF (sinistra < destra) {
    scegli pivot nell'intervallo [sinistra...destra];
    rango = Distribuzione( a, sinistra, pivot, destra );
    QuickSort( a, sinistra, rango-1 );
    QuickSort( a, rango+1, destra );
  }
```

DIVIDE
ET
IMPERA

DISTRIBUZIONE
PARTIZIONE
con
pivot L
va nella
sua posizione
finale
$\boxed{\text{rango}}$



Esempio

$A = [18, 11, 7, 21, 42, 5, 10, 35]$

value $= A[0] = 18$ $\qquad i = 0$ $\qquad j = A.size - 1$

Pivot $(A, 18) \rightarrow$

value

$A = [\underbrace{7, 10, 5, 11}_{< \text{value}}, \underbrace{18}_{px}, \underbrace{35, 42, 21}_{> \text{value}}]$