

# Analisi di Algoritmi

Continuiamo a discutere il problema dell'ordinamento:

Ordinamento (Sorting)

**INPUT:** Sequenza di  $n$  numeri  $\langle a_1, a_2, \dots, a_n \rangle$

**OUTPUT:** Permutazione

tale che  $\pi \langle a_1, a_2, \dots, a_n \rangle = \langle a'_1, a'_2, \dots, a'_n \rangle$   
 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Algoritmo di **Insertion-Sort**  $\rightarrow$  risolve il problema dell'ordinamento

La **complessita' temporale** (tempo di esecuzione dell'algoritmo) dell' Insertion-Sort verifica:

$T(n) = \Theta(n^2)$  nel caso peggiore e nel caso medio

$T(n) = \Theta(n)$  nel caso migliore

Oltre alla complessita' temporale possiamo studiare la complessita' spaziale di un algoritmo:

**Complessita' spaziale** = spazio di memoria necessario per ospitare le strutture di dati utilizzate dall'algoritmo.

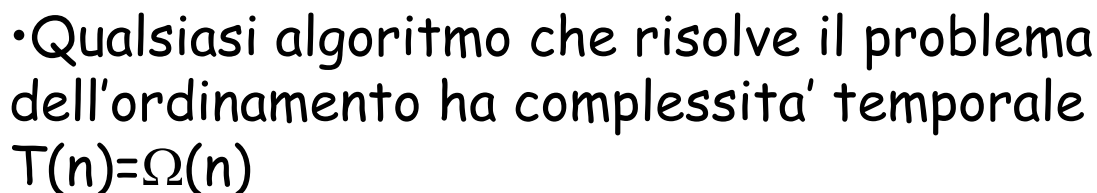
La complessita' spaziale dell'insertion sort e'  $\Theta(n)$

Se la complessità spaziale di un algoritmo è  $\Omega(g(n))$  allora la complessità temporale dell'algoritmo è almeno  $\Omega(g(n))$ .

Per creare strutture dati di dimensione  $\Omega(g(n))$  devo averle scritte almeno una volta. Per fare cio' ho impiegato un tempo almeno  $\Omega(g(n))$ .

[illegible]

- La complessita' spaziale di qualsiasi algoritmo che risolve il problema dell'ordinamento e'  $\Omega(n)$ .



L'algoritmo di **Insertion Sort** e' l' unico algoritmo possibile per risolvere il problema dell'ordinamento?

Ovviamente NO!

L'algoritmo di **Selection Sort** e' definito dalla seguente strategia:

- Si seleziona l'elemento piu' piccolo tra gli  $n$  elementi della sequenza e si pone nella prima posizione (scambiandolo con l'elemento  $A[1]$  );
- Si seleziona l'elemento piu' piccolo tra gli ultimi  $n-1$  elementi di  $A$  e si pone nella seconda posizione (scambiandolo con  $A[2]$  );
- Si seleziona l'elemento piu' piccolo tra gli ultimi  $n-k$  elementi di  $A$  e si pone nella  $k$ -esima posizione (scambiandolo con  $A[k]$ );
- Si seleziona l'elemento piu' piccolo tra gli ultimi 2 elementi di  $e$  e si pone nella penultima posizione (scambiandolo con  $A[n-1]$ ).

## Notare

La strategia descritta prevede ad ogni step la risoluzione dello stesso problema (ricerca del minimo di una sequenza).

Abbiamo risolto il problema dell'ordinamento riconducendo tale problema al piu' semplice problema di ricerca del minimo.

Algoritmo per la ricerca dell'elemento minimo di una sequenza

```
Min_elemento(A)
ind_min ← 1
For i ← 2 to length(A)
    do { if (A[i] < A[ind_min ])
        then { ind_min ← i
Return ind_min
```

Qual e' la complessita' spaziale di questo algoritmo?

$$S(n) = \Theta(n)$$

Qual e' la complessita' temporale di questo algoritmo?

$$T(n) = \Theta(n) \quad \text{per tutti gli input}$$

# Algoritmo di Selection Sort

```
Selection-Sort(A)
For j ← 1 to (length(A)-1)
  do {
    ind_min ← j
    For i ← j+1 to length(A)
      do { if (A[i] < A[ind_min])
          then { ind_min ← i
    k ← A[ind_min]
    A[ind_min] ← A[j]
    A[j] ← k
```

Notiamo che l'algoritmo e' corretto. Infatti:

- L 'algoritmo termina sempre;
- Per ogni istanza di input l'output e' corretto.

La correttezza dell' algoritmo puo' essere mostrata ragionando per **induzione**. Devo mostrare che la seguente affermazione e' vera per  $j=1,2,3,\dots,n-1$ :

"Al j-esimo step l'algoritmo produce una sequenza ordinata non decrescente  $A[1],\dots,A[j]$ , dove  $A[1]$  e' il piu' piccolo elemento,  $A[2]$  e' il secondo elemento piu' piccolo , .....,  $A[j]$  e' il j-esimo elemento piu' piccolo della sequenza iniziale."

- o Verifico la validita' dell'affermazione per  $j=1$ ;
- o Assumo che la affermazione sia valida per un generico valore di  $j=h$ ;
- o Mostro che da cio' segue che la affermazione e' valida per  $j=h+1$ .

## Analisi del Selection Sort

```
Selection-Sort(A)
For j ← 1 to (length(A)-1)
  do { ind_min ← j
      For i ← j+1 to length(A)
        do { if (A[i] < A[ind_min ])
            then { ind_min ← i
          k ← A[ind_min]
          A[ind_min] ← A[j]
          A[j] ← k
```

### Complessita' temporale del Selection Sort

- Qual e' la linea (o il blocco di linee di codice) che viene eseguito piu' volte?

`if (A[i] < A[ind_min ])` - operazione di confronto

- Quante volte viene eseguita?

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

- Il numero di volte che questa linea viene eseguita dipende dall' input?

No

La compl. temp. del selection sort e'  $T(n)=\Theta(n^2)$  (indip. dall'input).