

Algoritmo Bubble Sort

L'algoritmo **Bubble Sort** risolve il problema dell'ordinamento seguendo la seguente strategia:

- ✓ 1) Si scorre la sequenza $A[1], \dots, A[n]$, eliminando inversioni contigue (in tal modo si porta il massimo degli n elementi considerati nella posizione $A[n]$);
- ✓ 2) Si scorre la sequenza $A[1], \dots, A[n-1]$, eliminando inversioni contigue (in tal modo si porta il massimo degli $n-1$ elementi considerati nella posizione $A[n-1]$);
- ✓ $n-1$) Si scorre la sequenza $A[1], A[2]$, eliminando inversioni contigue (in tal modo si porta il massimo dei due elementi considerati nella posizione $A[2]$).

Definizione

Sia $A = \langle a_1, a_2, \dots, a_n \rangle$ una sequenza di n numeri. La coppia (a_i, a_j) è chiamata **inversione** se $i < j$ ed $a_i > a_j$.

Algoritmo Bubble Sort - Versione 1

```
Bubble-Sort-1(A)
limite ← length(A)
for h ← 1 to (length(A)-1)
  do { for i ← 1 to (limite-1)
      do { if (A[i] > A[i+1])
          then { k ← A[i]
                A[i] ← A[i+1]
                A[i+1] ← k
          }
        }
    }
  limite ← limite - 1
```

L'algoritmo e' corretto?

L'algoritmo termina sempre;

Per ogni input, l'output e' corretto.

(**esercizio**: mostrare, ragionando per induzione, la correttezza del Bubble Sort)

Qual e' la complessita' temporale dell'algoritmo?

- linea (o il blocco di linee di codice) eseguito piu' volte

if (A[i] < A[i+1]) - operazione di confronto

- N. volte che viene eseguita

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

Notare- il numero di volte che questa linea viene eseguita non dipende dall' input

T(n)=Θ(n²) indep. dall'input.

Esercizio: determinare la compl. spaziale del BS

```

Bubble-Sort-1(A)
limite ← length(A)
for h ← 1 to (length(A)-1)
  do { for i ← 1 to (limite-1)
        do { if (A[i] > A[i+1])
              then { k ← A[i]
                    A[i] ← A[i+1]
                    A[i+1] ← k
              }
        }
    }
  limite ← limite - 1

```

L'efficienza del bubble sort puo' essere migliorata se consideriamo che:

-Allo step h:

La posizione j dell'ultima inversione e' diversa da $n-h$ se e solo se la sottosequenza $A[j+1].....A[n]$ verifica:

$$\max(A[1].....A[j+1]) \leq A[j+1] \leq A[j+2] \leq \leq A[n]$$

infatti:

Per un generico h, quando $i = j$:

$$A[j+1] = \max(A[1].....A[j+1])$$

Allora:

$$\max(A[1].....A[j+1]) \leq A[j+2] \leq \leq A[n] \quad \Rightarrow \quad \text{No inversioni per } i > j$$

$$\text{No inversioni per } i > j \quad \Rightarrow \quad \max(A[1].....A[j+1]) \leq A[j+2] \leq \leq A[n]$$

Algoritmo Bubble Sort - Versione 2

```
Bubble-Sort(A)
limite ← length(A)
while (limite > 1)
  do { j ← 1
      for i ← 1 to (limite-1)
        do { if (A[i] > A[i+1])
            then { k ← A[i]
                  A[i] ← A[i+1]
                  A[i+1] ← k
                  j ← i
            }
          }
      }
  limite ← j
```

Abbiamo modificato strategia precedente nel modo seguente:

- o Ogni volta che scorriamo la sequenza memorizziamo in una variabile j la posizione dell'ultima inversione che incontriamo. Da $j+1$ ad n la sequenza e' gia ordinata e non deve essere piu' toccata;
- o Allo step successivo esaminiamo solamente la sequenza $A[1], \dots, A[j]$;
- o Quando $j=1$ la sequenza e' ordinata e l'algoritmo termina.

```

Bubble-Sort(A)
limite ← length(A)
while (limite > 1)
    do j ← 1
        for i ← 1 to (limite-1)
            do if (A[i] > A[i+1])
                then k ← A[i]
                    A[i] ← A[i+1]
                    A[i+1] ← k
                    j ← i
        limite ← j

```

L'algoritmo e' corretto?

Verifica di terminazione -

Ogni volta che il blocco **while** viene eseguito

$\text{limite} \leftarrow j \leq \text{limite}-1 \Rightarrow$ Il ciclo while viene
eseguito, al piu', n-1 volte

Qual e' la complessita' temporale dell'algoritmo?

-linea (o il blocco di linee di codice) eseguita piu' volte

if (A[i] < A[i+1]) - operazione di confronto

- N. volte che viene eseguita \rightarrow **dipende dall'input**

caso peggiore -
A ordinata non crescente

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

caso migliore -
A ordinata non decrescente

n-1

Riassumendo:

L'algoritmo di Bubble Sort ha complessità temporale $T(n)$:

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

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

Il comportamento asintotico dell'algoritmo Bubble Sort è simile al comportamento asintotico dell'algoritmo Insertion Sort.