

# *Programmazione I*

A.A. 2002-03

---

## *Funzioni*

( *Lezione XXI* )

### *Algoritmi notevoli per gli array*

---

***Prof. Giovanni Gallo***

***Dr. Gianluca Cincotti***

Dipartimento di Matematica e Informatica

Università di Catania

**e-mail :** { gallo, cincotti } @dmi.unict.it

## *Il problema dell'ordinamento*

---

➤ Sia dato un array di elementi scelti in un universo *totalmente ordinato*, il **problema dell'ordinamento** consiste nel sistemare gli elementi dell'array in un ordine particolare:

- in senso crescente (non decrescente), oppure
- in senso decrescente (non crescente).

# *Gli algoritmi di ordinamento*

---

➤ Esistono numerosi algoritmi di ordinamento.

- BubbleSort
- SelectionSort
- InsertionSort
- QuickSort
- MergeSort
- HeapSort
- ShellSort

## *BubbleSort*

---

➤ Algoritmo *BubbleSort*

- Si passano in rassegna tutti i valori dell'array diverse volte. Durante ogni scansione vengono confrontate tutte le coppie di elementi contigui;
  - se in una coppia il primo valore è minore o uguale al secondo, gli elementi vengono lasciati al loro posto;
  - se, invece, il primo valore è maggiore del secondo i due elementi vengono scambiati di posto nell'array;
- ripetere finché l'intero array è ordinato.

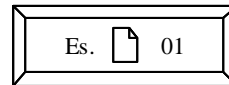
## BubbleSort (cont.)

Array da ordinare in ordine crescente:

3	8	5	1	2
3	8	5	1	2
3	5	8	1	2
3	5	1	8	2
3	5	1	2	8
3	5	1	2	8

3	1	5	2	8
3	1	2	5	8
1	3	2	5	8
1	2	3	5	8
1	2	3	5	8

Variante: Se durante una scansione non avvengono scambi l'algoritmo termina.



## SelectionSort

### ➤ Algoritmo SelectionSort

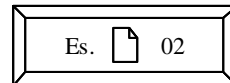
- ricerca il più piccolo valore dell'array,
- scambia tale elemento con l'elemento nella prima posizione dell'array,
- ricerca il successivo valore più piccolo nell'array,
- scambia tale elemento con quello nella seconda posizione dell'array,
- ripetere finché l'intero array è ordinato.

## SelectionSort (cont.)

---

➤ Array da ordinare in ordine crescente:

3	8	5	1	2
1	8	5	3	2
1	2	5	3	8
1	2	3	5	8
1	2	3	5	8



## InsertionSort

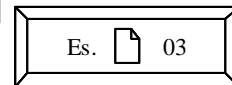
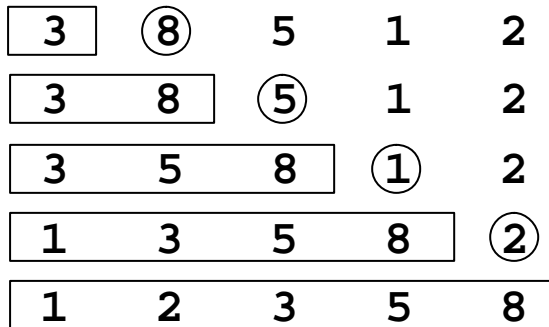
---

➤ Algoritmo InsertionSort

- Ogni elemento dell'array viene spostato in un sotto-array ordinato (è la maniera di ordinare le carte da gioco! )
  - Considerando un elemento alla volta, si inserisca ciascun elemento al posto “giusto” tra quelli già considerati (mantenendo quest'ultimi ordinati).
- Più in dettaglio:
  - il primo elemento nella prima posizione dell'array costituisce un sotto-array ordinato di lunghezza uno,
  - inserire il secondo elemento al posto giusto nel sotto-array, facendo shiftare se necessario il primo elemento di una posizione a destra per fare spazio per l'inserimento,
  - inserire il terzo elemento nel sotto-array ordinato di lunghezza due, shiftando se necessario gli elementi per mantenere l'ordinamento,
  - ripetere finché l'intero array è ordinato.

## InsertionSort (cont.)

➤ Array da ordinare in ordine crescente:



## Confronto tra gli algoritmi

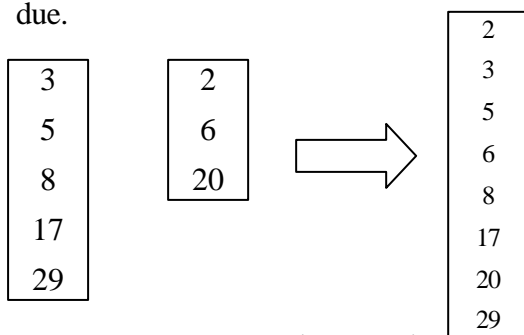
➤ L'efficienza di un algoritmo di ordinamento può essere stabilita teoricamente contando il **numero di confronti** effettuati.

- Gli ordinamenti visti sono *simili* per efficienza.
  - Presentano un ciclo esterno che esegue più volte un ciclo interno che scandisce quasi tutti gli elementi.
    - ❖ Si eseguono approssimativamente  $n^2$  confronti per ordinare un array di  $n$  elementi.
    - ❖ In questo caso si dice che gli algoritmi sono di *ordine*  $n^2$ 
      - Esistono algoritmi più efficienti, dell'*ordine* di  $n \log_2 n$

## Il problema della fusione

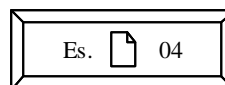
### ➤ *Natural merge*

- Dati due array *ordinati* in input, restituire in output un terzo array *ordinato* contenente gli elementi dei primi due.



## Ricerca dicotomica (o binaria)

- La ricerca binaria in un array *ordinato* consiste nel confrontare l'elemento centrale  $M$  dell'array con la chiave di ricerca e
  - se sono uguali, l'elemento è stato trovato;
  - se la chiave di ricerca è minore di  $M$ , la ricerca prosegue iterativamente nella prima metà di array;
  - se la chiave di ricerca è maggiore di  $M$ , la ricerca prosegue iterativamente nella seconda metà di array.
- Osservazione : Ad ogni iterazione, l'algoritmo prosegue la ricerca solo su metà degli elementi considerati al passo precedente.



---

# *Fine*