UNIVERSITÀ DEGLI STUDI DI TRIESTE

MODULO 2: Algoritmi di ordinamento (parte II)

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 7 maggio 2026

Agenda di oggi

- Bubblesort
- Quicksort
- Counting sort

Agenda di oggi

- Bubblesort
- *Quicksort*
- *Counting sort*

ALGORITMO

Input: un elenco di numeri da ordinare (come una fila di carte)

1. **Confronta** i primi due numeri dell'elenco:
   - Se il primo è maggiore del secondo, scambiali di posto.
   - Altrimenti, lasciali così.
2. Spostati alla coppia successiva (secondo e terzo numero) e fai lo stesso confronto e possibile scambio.
3. Continua così, confrontando ogni coppia adiacente, fino alla fine dell'elenco.
4. **Dopo il primo giro, il numero più grande si sarà spostato alla fine** (come una bolla d'aria che sale in superficie — da qui il nome *Bubble Sort*).
5. Ripeti lo stesso procedimento, ignorando l'ultimo elemento già in posizione.
6. Continua a ripetere i passaggi, ogni volta confrontando una coppia in meno, *finché non ci sono più scambi da fare (oppure per n-1 volte).*
7. A quel punto, l'elenco è completamente ordinato in ordine crescente.

ESEMPIO

Array da ordinare

Iterazione completa #1

Iterazione completa #2

Iterazione completa #3

Iterazione completa #4

Iterazione completa #5

Bubble Sort non ottimizzato: fa comunque tutte le passate previste, cioè fino a n - 1.

Bubble Sort ottimizzato: si ferma quando una passata termina con 0 scambi.

PSEUDOCODICE*

```
BUBBLE-SORT(A,n)              //input: array A di lunghezza n
for i=1 to n-1
    flag_scambi = Falso       //flag che verifica se sono stati fatti scambi
    for j = 1 to n-i          //faccio confronti e scambi escludendo gli
                              //elementi in coda già ordinati nelle i iterazioni
                              //precedenti
        if A[j] > A[j+1]      //confronto
            scambia A[j] con A[j+1]      //scambio
            flag_scambi = Vero           //verifica: ci sono stati scambi?
    if flag_scambi = Falso               //no scambi → ho finito
        return
```

*ottimizzato rispetto al libro di testo

PRESTAZIONI: CALCOLO NUMERO ITERAZIONI

```
BUBBLE-SORT(A,n)
for i=1 to n-1                          Al massimo n-1 iterazioni
    flag_scambi = Falso
    for j = 1 to n-i                    Al massimo n-i iterazioni
        if A[j] > A[j+1]
            scambia A[j] con A[j+1]
            flag_scambi = Vero
    if flag_scambi = Falso
        return
```

Caso peggiore

COSTI UNITARI	COSTO SINGOLA **ITERAZIONE CICLO ESTERNO**	COSTO COMPLESSIVO per istruzione
c_1	c_1	$c_1 \cdot (n-1)$
c_2	c_2	$c_2 \cdot (n-1)$
c_3	$c_3 \cdot (n-i)$	$c_3 \cdot [(n-1)+(n-2)+\ldots+1]$
c_4	$c_4 \cdot (n-i)$	$c_4 \cdot [(n-1)+(n-2)+\ldots+1]$
c_5	$c_5 \cdot (n-i)$	$c_5 \cdot [(n-1)+(n-2)+\ldots+1]$
c_6	$c_6 \cdot (n-i)$	$c_6 \cdot [(n-1)+(n-2)+\ldots+1]$
c_7	c_7	$c_7 \cdot (n-1)$
c_8	c_8	c_8

Nota. c1, c2, .. sono delle costanti che quantificano il costo di esecuzione della relativa istruzione. Per **esempio c1 rappresenta il costo dell'istruzione for** e include l'inizializzazione dell'indice (i=1), il controllo della condizione (i≤n-1) e l'incremento a ogni iterazione.

PRESTAZIONI: CALCOLO NUMERO ITERAZIONI

COSTI UNITARI	COSTO SINGOLA **ITERAZIONE CICLO ESTERNO**	COSTO COMPLESSIVO per istruzione
c_1	c_1	$c_1 \cdot (n-1)$
c_2	c_2	$c_2 \cdot (n-1)$
c_3	$c_3 \cdot (n-i)$	$c_3 \cdot [(n-1)+(n-2)+...+1]$
c_4	$c_4 \cdot (n-i)$	$c_4 \cdot [(n-1)+(n-2)+...+1]$
c_5	$c_5 \cdot (n-i)$	$c_5 \cdot [(n-1)+(n-2)+...+1]$
c_6	$c_6 \cdot (n-i)$	$c_6 \cdot [(n-1)+(n-2)+...+1]$
c_7	c_7	$c_7 \cdot (n-1)$
c_8	c_8	c_8

Calcolo costo complessivo

$$T(n) = c_1(n-1) + c_2(n-1) + ...$$

$$... + (c_3 + c_4 + c_5 + c_6)[(n-1) + (n-2) + ... + 1] + ...$$

$$... + c_7(n-1) + c_8$$

Si nota che:

$$(c_3 + c_4 + c_5 + c_6)[(n-1) + (n-2) + ... + 1] = \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} (c_3 + c_4 + c_5 + c_6)$$

Tenendo solo i costi dominanti:

Somma dei primi n-1 numeri interi

$$T(n) = O(n) + \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} O(1) = O(n) + \sum_{i=1}^{n} i = O(n) + \frac{n(n+1)}{2} = O(n^2)$$

Progressione artimetica notevole

CORRETTA RISPETTO ALLA LEZIONE

CONFRONTO CON ALTRI ALGORITMI DI ORDINAMENTO

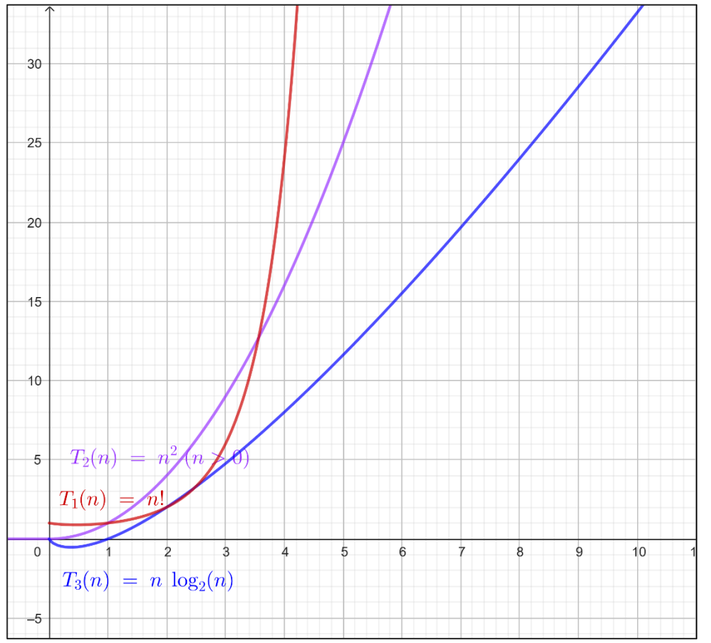
Caso migliore (array già ordinato): O(n)

Caso medio/peggiore: $O(n^2)$

Algoritmo	Caso peggiore	Caso migliore
Bogosort	O(n!)	O(n)
Insertion sort	$O(n^2)$	O(n)
Merge sort	$O(n\log_2 n)$	$O(n\log_2 n)$
Bubble sort	$O(n^2)$	O(n)

Insertionsort

Bogosort Bubblesort Mergesort



Note

- Insertion Sort: nel caso migliore, ogni elemento viene confrontato con il precedente e non serve spostare nulla. Quindi fa circa n-1 confronti: O(n).
- Merge Sort: divide comunque il vettore fino ai sottovettori di dimensione 1 e poi fa tutti i merge. Quindi anche nel caso migliore resta O(n log n).

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE