

Introduzione agli Array e all'Ordinamento Naïve

Prof. Fedeli Massimo - IIS Fermi Sacconi Cpia

Cos'è un array?

Immagina di dover memorizzare i voti di una classe in informatica: ad esempio, i voti di cinque studenti sono 7, 5, 9, 6, 8. Potresti usare cinque variabili diverse, tipo `voto1`, `voto2`, ecc., ma questo diventa scomodo se i voti sono centinaia.

Un **array** (in italiano “vettore” o “tabella”) è una struttura dati che permette di conservare molti valori dello stesso tipo in un'unica variabile, accessibili tramite un **indice**.

In pratica, un array è come una fila di cassette numerate: ogni cassetto contiene un valore, e puoi aprire il cassetto numero 0, 1, 2, ecc. per leggere o modificare il contenuto.

Esempio pratico

Supponiamo di avere un array chiamato `voti` con i seguenti elementi:

Indice	0	1	2	3	4
Valore	7	5	9	6	8

Per accedere al terzo voto (cioè al numero 9), scriviamo `voti[2]`, perché in informatica si parte a contare da zero!

Gli array sono utili perché:

- Consentono di gestire tanti dati con un solo nome.
- Permettono di scorrere tutti gli elementi con un ciclo (**for**).
- Sono la base per molti algoritmi, come quelli di ricerca e ordinamento.

Ordinamento “naïve” (o “a forza bruta”)

A volte vogliamo riordinare i valori di un array, ad esempio dal più piccolo al più grande (ordine crescente). Esistono tanti algoritmi per farlo; uno dei più semplici — anche se non il più efficiente — è l'**ordinamento naïve**, detto anche “ordinamento per confronto esaustivo”.

L'idea è questa: confrontiamo ogni elemento con tutti gli altri, e ogni volta che troviamo un elemento più piccolo, li scambiamo di posto.

Come funziona passo dopo passo

Prendiamo l'array: [7, 5, 9, 6, 8].

1. Partiamo dal primo elemento (7) e lo confrontiamo con tutti gli altri.
2. Se troviamo un numero più piccolo (es. 5), lo scambiamo con 7.
3. Ora l'array diventa [5, 7, 9, 6, 8].
4. Poi passiamo al secondo elemento (7) e lo confrontiamo con i successivi (9, 6, 8).

5. Troviamo 6, che è più piccolo: scambiamo $\rightarrow [5, 6, 9, 7, 8]$.
6. Continuiamo così fino alla fine.

Questo metodo assomiglia molto al famoso **Bubble Sort** (ordinamento a bolle), ma è ancora più diretto: non sfrutta alcuna ottimizzazione.

Codice di esempio in Python

Ecco una versione semplice dell'algoritmo in Python:

```
1  def ordina_naive(arr):
2      n = len(arr)
3      for i in range(n):
4          for j in range(i + 1, n):
5              if arr[i] > arr[j]:
6                  # Scambia arr[i] e arr[j]
7                  arr[i], arr[j] = arr[j], arr[i]
8      return arr
9
10 # Esempio d'uso
11 voti = [7, 5, 9, 6, 8]
12 ordinati = ordina_naive(voti)
13 print(ordinati)  # Stampa: [5, 6, 7, 8, 9]
14
```

Listing 1: Ordinamento naïve in Python

Perché “naïve”?

Il termine “naïve” (ingenuo) indica che l'algoritmo fa il minimo indispensabile senza cercare scorciatoie. È facile da capire, ma non è veloce per array grandi: infatti, con n elementi, deve fare circa n^2 confronti. Per 1000 numeri, servono quasi un milione di operazioni!

Tuttavia, è un ottimo punto di partenza per imparare a ragionare sugli algoritmi.

Conclusione

Gli array sono una delle strutture dati fondamentali in informatica: ti permettono di organizzare tanti dati in modo ordinato e accessibile. L'ordinamento naïve, pur essendo lento, è un algoritmo semplice che aiuta a comprendere come si possono manipolare gli array per ottenere risultati utili, come elenchi ordinati.

Man mano che proseguirai nello studio dell'informatica, incontrerai algoritmi più efficienti (come il *Merge Sort* o il *Quick Sort*), ma ricorda: ogni grande edificio comincia dalle fondamenta!