Programmazione





Ordinamento

Algoritmi di ordinamento

Scopo: ***ordinare una sequenza di elementi*** **in base a una certa**
relazione d'ordine (es: crescente, decrescente)

lo scopo finale è ben definito
→ ***algoritmi equivalenti***

diversi algoritmi possono avere ***efficienza assai diversa***

Ipotesi:
gli elementi siano memorizzati in un array

Principali algoritmi di ordinamento

- ***naïve sort*** (semplice, intuitivo, poco efficiente)
- **bubble sort** (semplice, un po' più efficiente)
- ***insert sort*** (intuitivo, abbastanza efficiente)
- ***merge sort*** (non intuitivo, molto efficiente)
- ***quick sort*** (non intuitivo, molto efficiente)

Per "misurare le prestazioni" di un algoritmo, conteremo quante volte viene svolto il ***confronto fra elementi dell'array.***

NAÏVE SORT

- **Molto intuitivo e semplice, è il primo che viene in mente**

  Specifica (sia `n` la dimensione dell'array `v`)

  `while` (<*array non vuoto*>) {

  <*trova la posizione* `p` *del massimo*>

  `if (p<n-1)` <*scambia* `v[n-1]` *e* `v[p]` >

  /* **ora v[n-1] contiene il massimo** */

  <*restringi l'attenzione alle prime* `n-1` *caselle dell' array, ponendo* `n' = n-1` >

  }

Esempio NAÏVE SORT

Iterazione 1 while dim=4

Iterazione 2 while dim=3

Iterazione 3 while dim=2

Il numero di *confronti* necessari (per la ricercar del massimo) vale *sempre (ad ogni Passo)*:

(N-1) + (N-2) + (N-3) + ... + 2 + 1 = = N*(N-1)/2 =

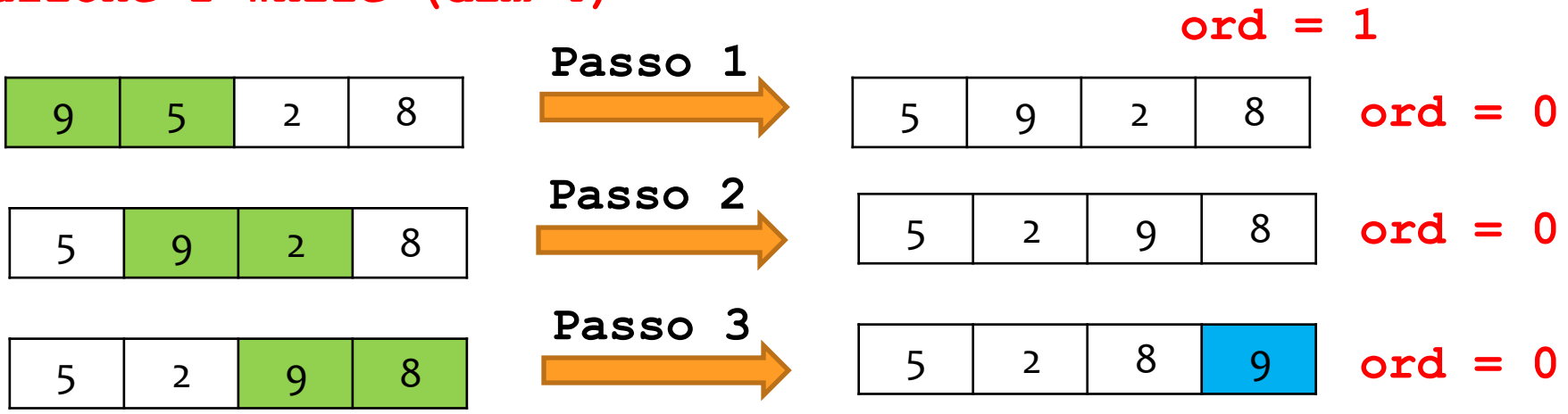
$O(N_2/2)$

BUBBLE SORT

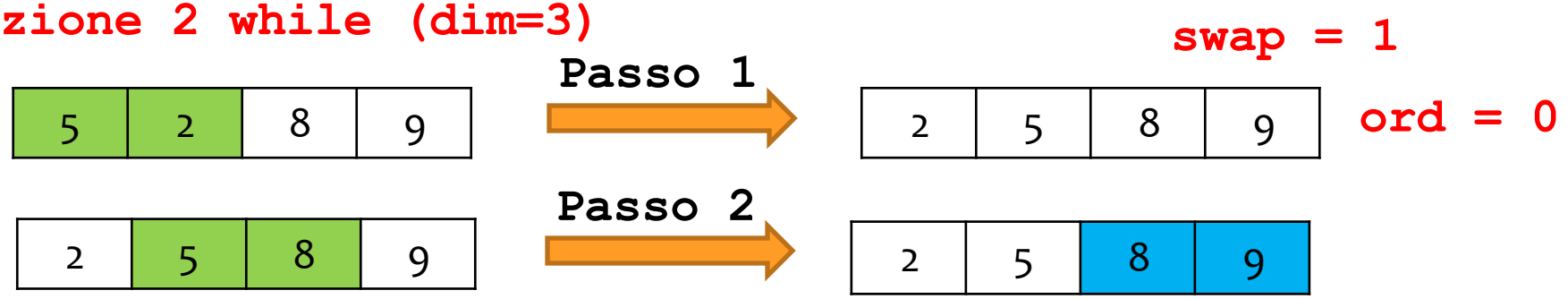
- **Corregge il difetto principale del naïve sort: quello di *non accorgersi se l'array, a un certo punto, è già ordinato.***
- **Opera per *"passate successive"* sull'array:**
  - **a ogni iterazione, considera una ad una *tutte le possibili coppie di elementi adiacenti*, scambiandoli se risultano nell'ordine errato**
  - **così, dopo ogni iterazione, l'elemento massimo è in fondo alla parte di array considerata**
- **Quando non si verificano scambi, l'array è ordinato, e l'algoritmo termina.**

Esempio Bubble Sort

Iterazione 1 while (dim=4)



Iterazione 2 while (dim=3)



Esempio Bubble Sort

Iterazione 3 while (dim=2)

ord = 1

In questa iterazione non è stato fatto alcuna sostituzione: Posso fermarmi, l'array è ordinato!

Ottimizzazione: ad ogni iterazione ho confrontato tutte le coppie di elementi adiacenti da 0 a *n*-1.
L'algoritmo garantisce che all'iterazione *i*, l'*i*-esimo elemento più grande è nella sua posizione corretta → all'iterazione *i* posso fermarmi all'elemento *n-i*.

Caso peggiore: numero di *confronti* identico al precedente □ **O(N_2/2)**
***Nel caso migliore, però, basta una sola* passata → O(N)**