Algoritmi di Ordinamento

Framework di valutazione numerica della complessità

Bubble Sort

Insertion Sort

Merge Sort

Algoritmi di Ordinamento

- Presa una sequenza di elementi, restituiscono una permutazione della sequenza
- La sequenza restituita è in “ordine” (crescente o decrescente) rispetto a una relazione di ordinamento (totale) prefissata definita tra gli elementi
- La complessità si valuta in numero di confronti effettuati rispetto alla lunghezza della sequenza
- Limite inferiore di complessità nel caso pessimo $\Omega(n \log n)$

Algoritmi di Ordinamento

- Altro aspetto da considerare è se l'algoritmo lavora "in loco" oppure no
- Data la sequenza (in un array o arraylist), l'algoritmo lavora in loco se in ogni momento dell'esecuzione un numero di elementi al più costante si trova fuori dallo spazio occupato dalla sequenza iniziale data
- Esempi: BubbleSort, InsertionSort, QuickSort, HeapSort
- Esempio di non in loco: MergeSort

Algoritmi di Ordinamento

- Quello basato sui confronti non è l'unico approccio
- Esistono degli algoritmi di ordinamento non basati sul confronto degli elementi che hanno una complessità (in tempo) lineare!
- Questi algoritmi lavorano sfruttando un grande spazio di memoria
- Esempi: Counting Sort, Radix Sort, Bucket Sort

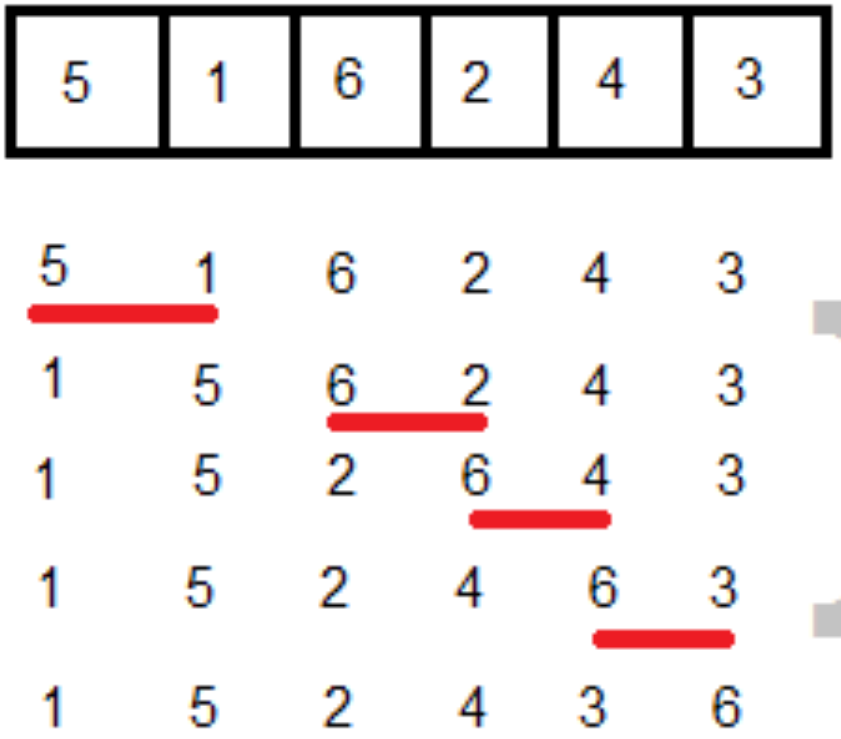
Framework di Valutazione Numerica

- Si tratta di una infrastruttura di classi e interfacce che crea un set crescente di sequenze numeriche e le dà in pasto a un certo numero di algoritmi di ordinamento diversi
- Il risultato sono misurazioni dei tempi di esecuzione e del numero di confronti sulle sequenze generate
- E' possibile confrontare numericamente (e graficamente) gli algoritmi
- Si veda il codice allegato

Bubble Sort

- È uno dei più semplici algoritmi di ordinamento
- Lavora facendo emergere ad ogni iterazione il valore più grande verso la parte alta dell'array
- Come "bolle" che salgono verso l'alto in un liquido

Bubble Sort


5 1 6 2 4 3
Lets take this Array.
5 1 6 2 4 3
1 5 6 2 4 3
1 5 2 6 4 3
1 5 2 4 6 3
1 5 2 4 3 6
Here we can see the Array after the first iteration.
Similarly, after other consecutive iterations, this array will get sorted.

Insertion Sort

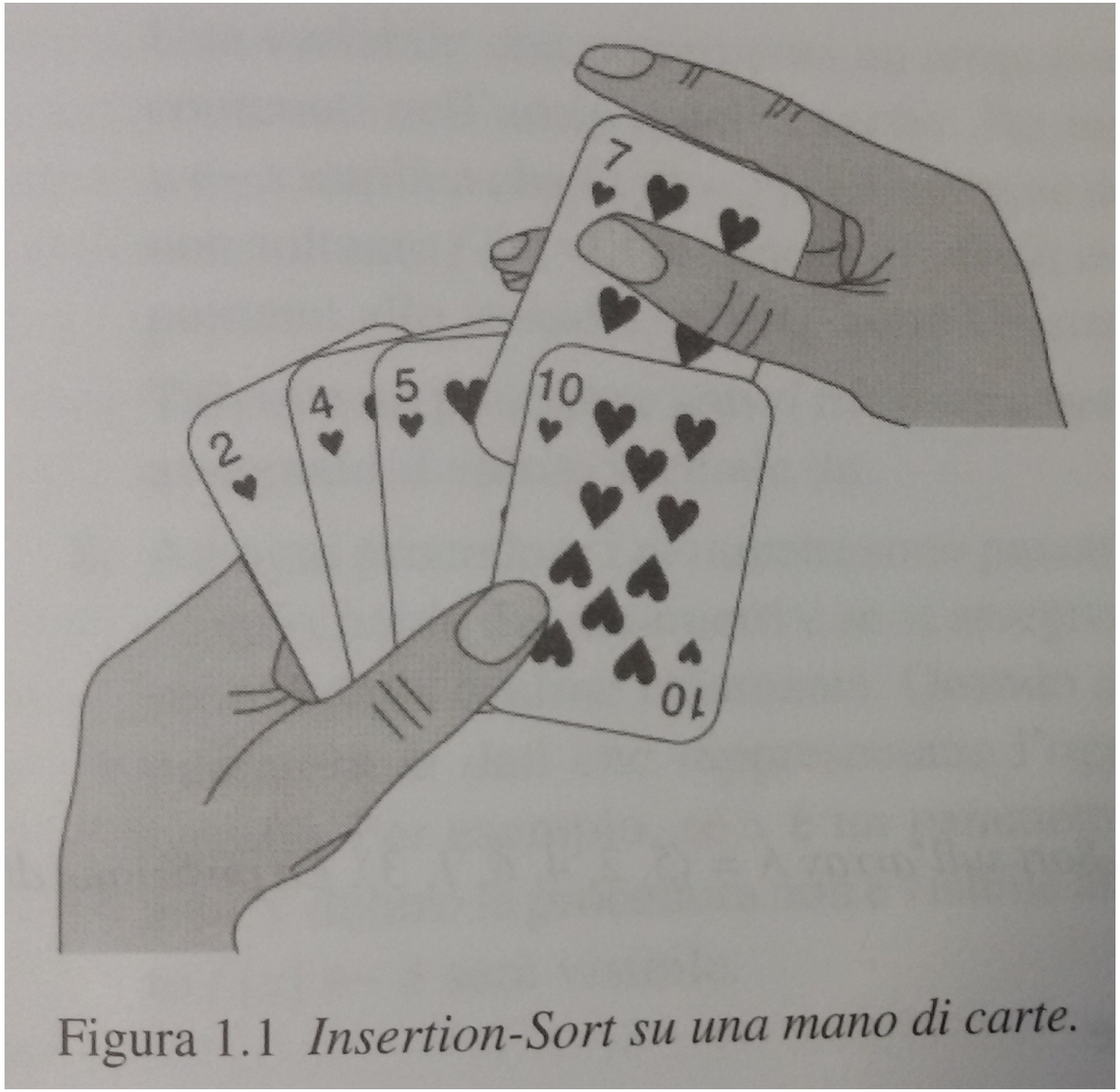
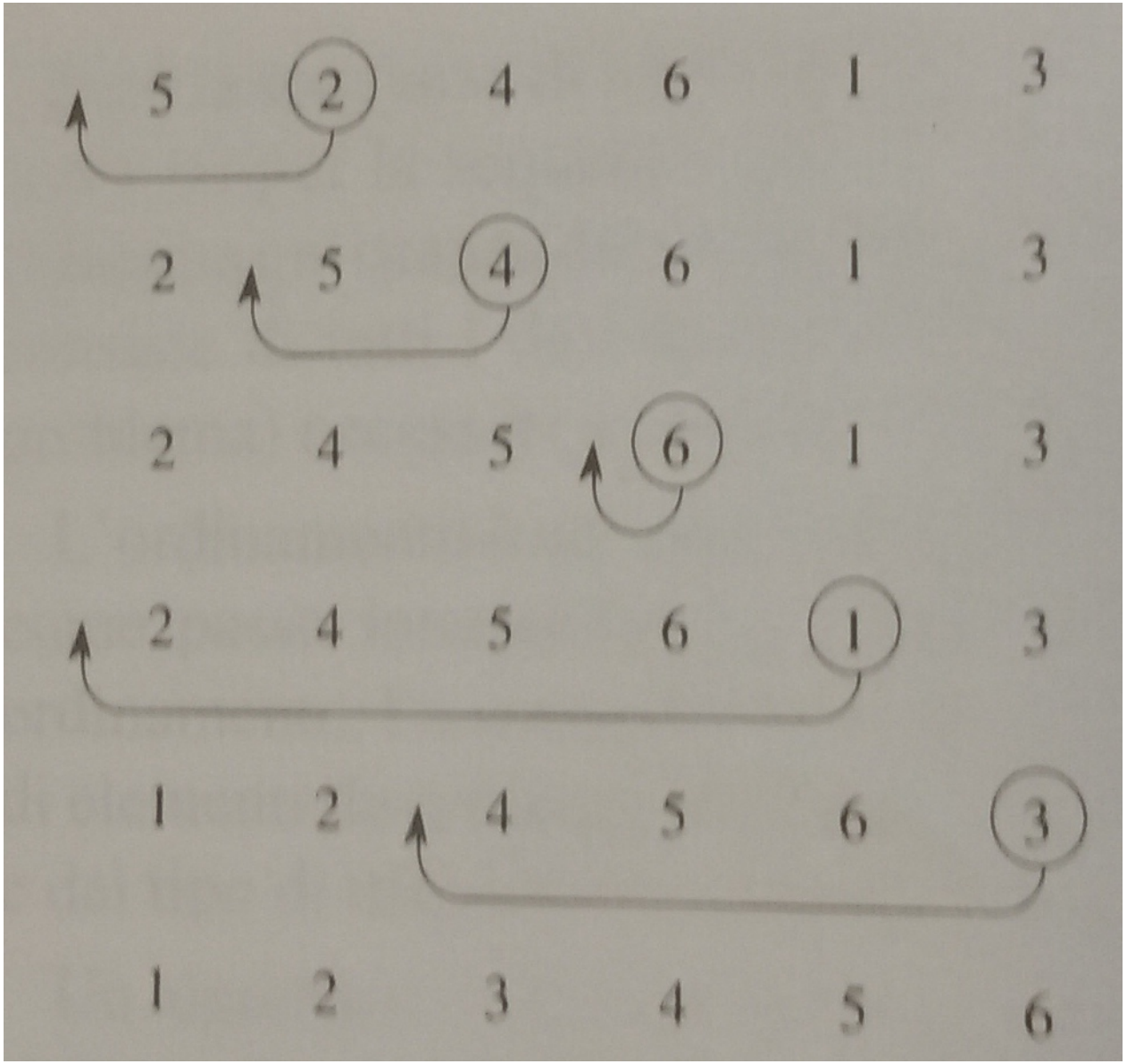


Figura 1.1 *Insertion-Sort su una mano di carte.*

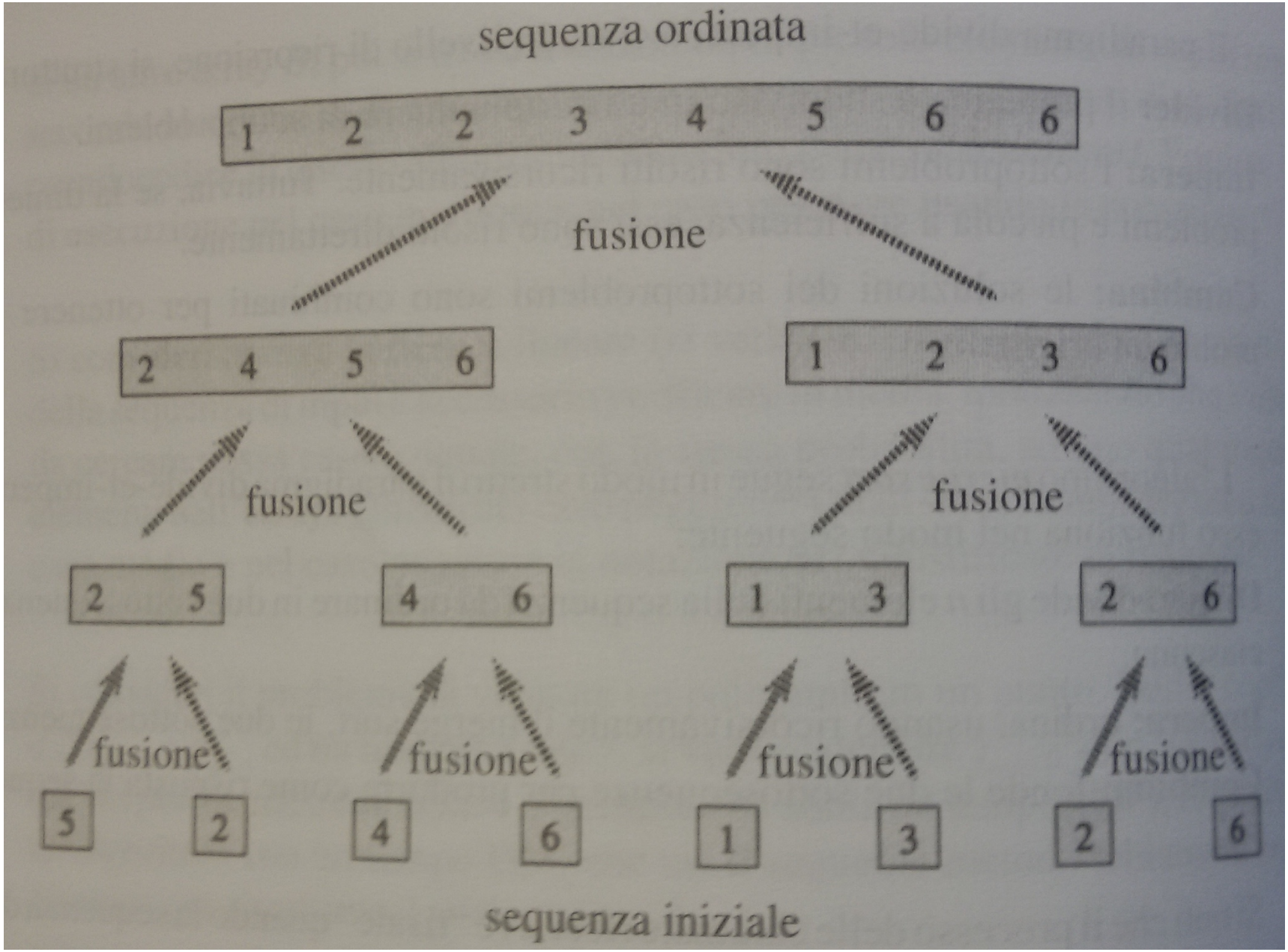
Insertion Sort



Merge Sort

- Non lavora in loco
- E’ ricorsivo
- Strategia Divide et Impera
- È asintoticamente ottimo O(n log n)
- Richiede una procedura accessoria di merge

Merge Sort


sequenza ordinata
1 2 2 3 4 5 6 6
fusione
2 4 5 6
1 2 3 6
fusione
fusione
2 5
4 6
1 3
2 6
fusione
fusione
fusione
fusione
5
2
4
6
1
3
2
6
sequenza iniziale

Algoritmi di Ordinamento

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduzione agli Algoritmi e Strutture Dati, McGraw-Hill, 2005

- Pagg. 2-13
- Pagg. 128-154