# Algoritmi di Ordinamento

Framework numerico di valutazione della complessità

Codifica di Algoritmi di Ordinamento Generici

Valutazione Numerica del caso ottimo, medio e pessimo

# Algoritmi di Ordinamento

- Un algoritmo di ordinamento prende in input una sequenza di elementi di un certo insieme per cui esiste una relazione totale di ordinamento
- Esso deve restituire in output una permutazione della sequenza che risulti ordinata secondo la relazione di ordinamento assegnata
- Esistono diversi algoritmi di ordinamento
- Tipicamente lo pseudocodice viene dato su sequenze di numeri interi rappresentate come array

# Complessità Computazionale

- La complessità computazionale in tempo e in spazio viene calcolata analiticamente nel caso peggiore, migliore e, in alcuni casi, medio (assumendo delle distribuzioni di probabilità)
- Per la complessità in tempo l'unità di misura è il numero di confronti tra gli elementi
- Se gli algoritmi hanno una complessità in spazio lineare su n, dove n è il numero di elementi nella sequenza, $\Omega(n \log n)$ è il limite inferiore di complessità computazionale in tempo

# Complessità Computazionale

- Oltre all'analisi asintotica è possibile valutare statisticamente il tempo di esecuzione
- Basta generare molti esempi per diverse lunghezze e far girare l'algoritmo di interesse
- Si possono misurare il numero di confronti e il tempo effettivo di esecuzione
- Faremo un framework numerico di valutazione di diversi algoritmi di ordinamento

# Parametri del Framework

```
/** Lunghezza minima delle sequenze da generare
     */
    public static int MIN_LENGTH = 30;
    /**
     * Lunghezza massima delle sequenze da generare
     */
    public static int MAX_LENGTH = 500;
    /**
     * Numero di sequenze da generare per lunghezza
     */
    public static int NUMBER_OF_SAMPLES_PER_LENGTH = 1000; */
```

# Generico Algoritmo di Ordinamento

- Creiamo una interface **`SortingAlgorithm<E extends Comparable<E>>`**
- Il contratto richiede di implementare un metodo

```
public SortingAlgorithmResult<E> sort(List<E> l);
```

- Il metodo lavora sulla lista in ingresso e la ordina in base all'ordinamento naturale della classe **`E`**
- Il risultato è un oggetto che contiene la lista ordinata e il numero di confronti effettuati

# Generico Algoritmo di Ordinamento

- Si veda il codice allegato per un esempio di implementazione della versione di base del Bubble Sort
- Si veda anche la classe di Test per la generazione di numeri casuali e il report dei risultati in termini di tempo di esecuzione
- Esercitazione in classe: implementare Merge Sort, Insertion Sort e Quick Sort nel framework