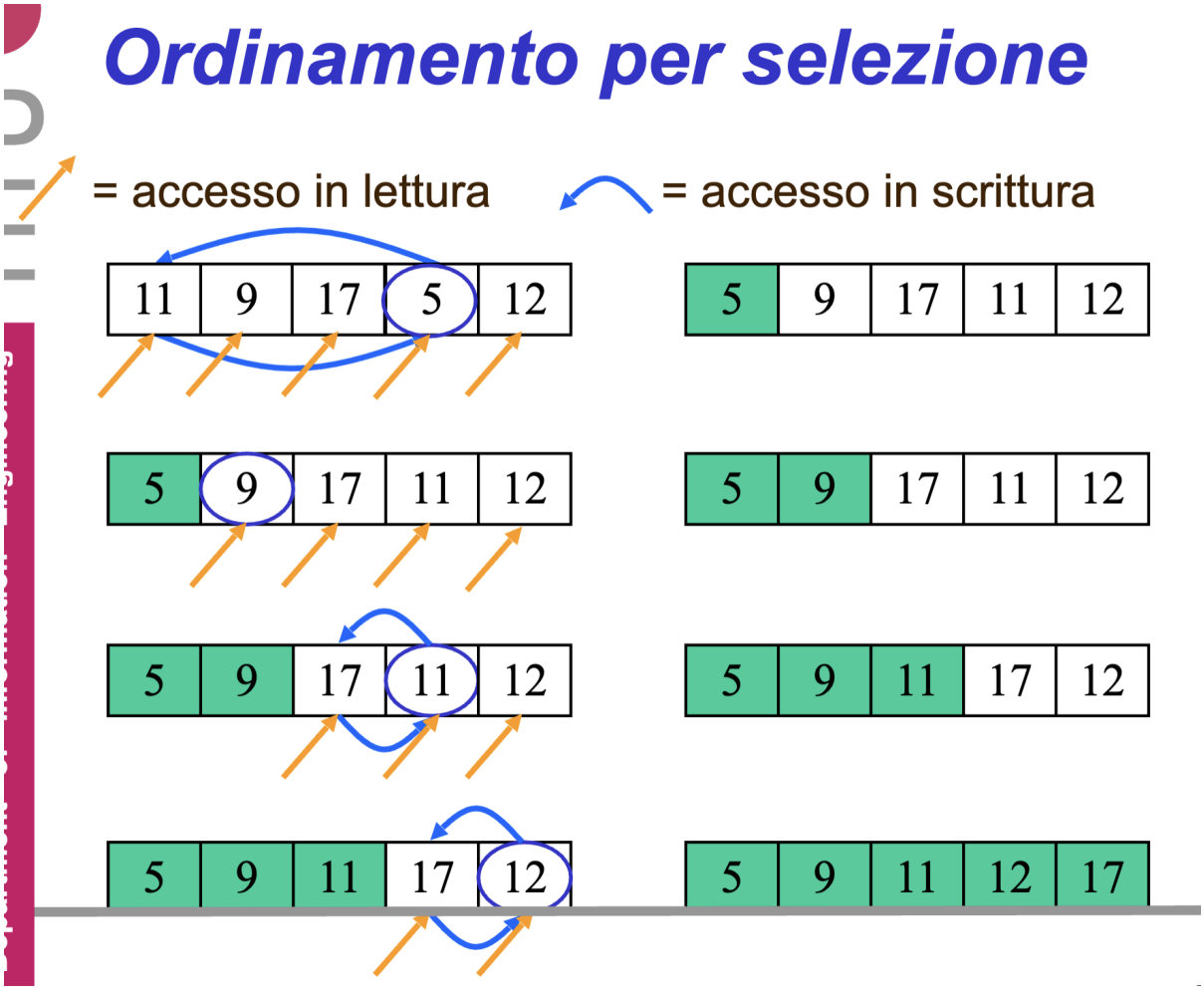
ALGORITMI DI ORDINAMENTO

ORDINAMENTO PER SELEZIONE

- Esempio: ordinare insieme numeri interi
- Cerco valore minimo
  - lo scambio al posto del primo valore (così non serve creare nuovo array)
- Proseguo con lo stesso procedimento nella parte non ordinata dell'array



IMPLEMENTAZIONE IN JAVA

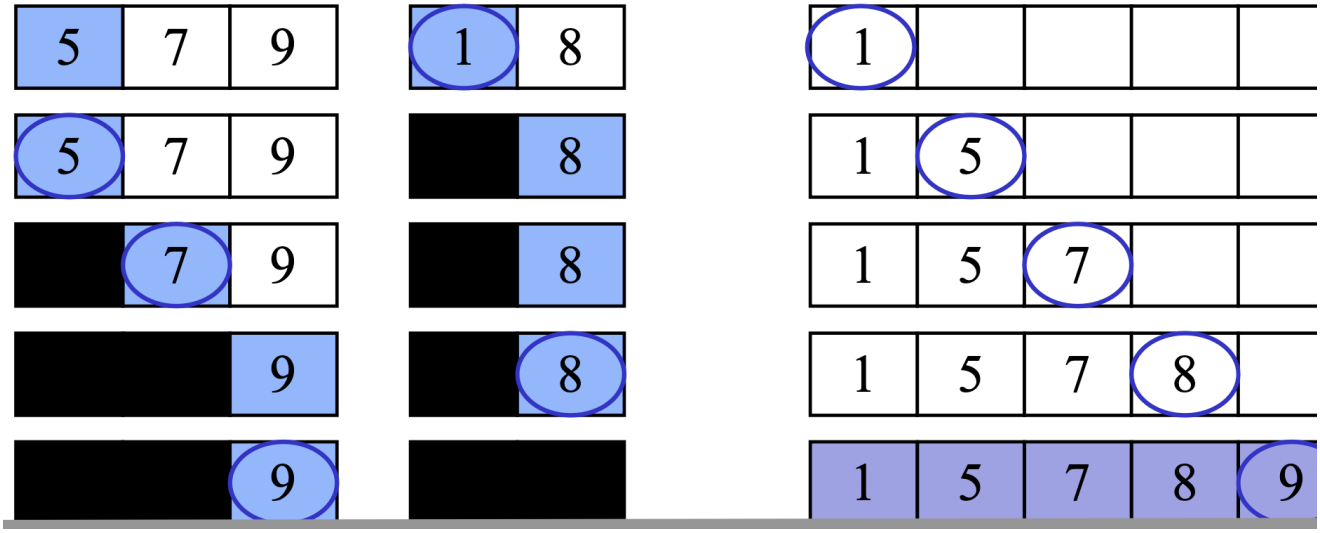
```
public static void selectionSort(int[] v, int vSize) {
    for (int i = 0; i < vSize - 1; i++) {
        int minPos = findMinPos(v, i, vSize - 1);
        if (minPos != i)
            swap(v, minPos, i);
    }
}
```

PRESTAZIONI

- $T(n) = O(n^2)$

ORDINAMENTO PER FUSIONE (MERGE SORT)

- Idea: dividere l'array in piccole parti, ordinare prima quelle e poi unirle
- Algoritmo
  - caso base: array contiene 1 elemento -> è già ordinato
  - passo ricorsivo:
  - **divido** array iniziale in due parti (circa) uguali
  - **ordino** ciascuna delle due parti separatamente
  - chiamando il metodo merge sort ricorsivamente
  - **fusione**(merge) delle due parti in modo ordinato
  - prendo **primo** elemento da uno dei due vettori scegliendo il più piccolo



IMPLEMENTAZIONE IN JAVA

```java
public static void mergeSort(int[] arrayFull, int arraySize) {
    if (arraySize < 2)
        return;

    int midPosition = arraySize / 2;

    int[] arrayLeftPart = new int[midPosition];
    int[] arrayRightPart = new int[arraySize - midPosition];

    System.arraycopy(arrayFull, 0, arrayLeftPart, 0, arrayLeftPart.length);
    System.arraycopy(arrayFull, midPosition, arrayRightPart, 0, arrayRightPart.length);

    mergeSort(arrayLeftPart, arrayLeftPart.length);
    mergeSort(arrayRightPart, arrayRightPart.length);

    merge(arrayFull, arrayLeftPart, arrayRightPart);
}

private static void merge(int[] arrayFull, int[] arrayLeftPart, int[] arrayRightPart) {
    int arrayFullIndex = 0, arrayLeftPartIndex = 0, arrayRightPartIndex = 0;
    while (arrayLeftPartIndex < arrayLeftPart.length && arrayRightPartIndex < arrayRightPart.length) {
        if (arrayLeftPart[arrayLeftPartIndex] < arrayRightPart[arrayRightPartIndex]) {
            arrayFull[arrayFullIndex] = arrayLeftPart[arrayLeftPartIndex];
            arrayFullIndex++;
            arrayLeftPartIndex++;
        } else {
            arrayFull[arrayFullIndex] = arrayRightPart[arrayRightPartIndex];
            arrayFullIndex++;
            arrayRightPartIndex++;
        }
    }

    while (arrayLeftPartIndex < arrayLeftPart.length) {
        arrayFull[arrayFullIndex] = arrayLeftPart[arrayLeftPartIndex];
        arrayFullIndex++;
        arrayLeftPartIndex++;
    }

    while (arrayRightPartIndex < arrayRightPart.length) {
        arrayFull[arrayFullIndex] = arrayRightPart[arrayRightPartIndex];
        arrayFullIndex++;
        arrayRightPartIndex++;
    }
}
```

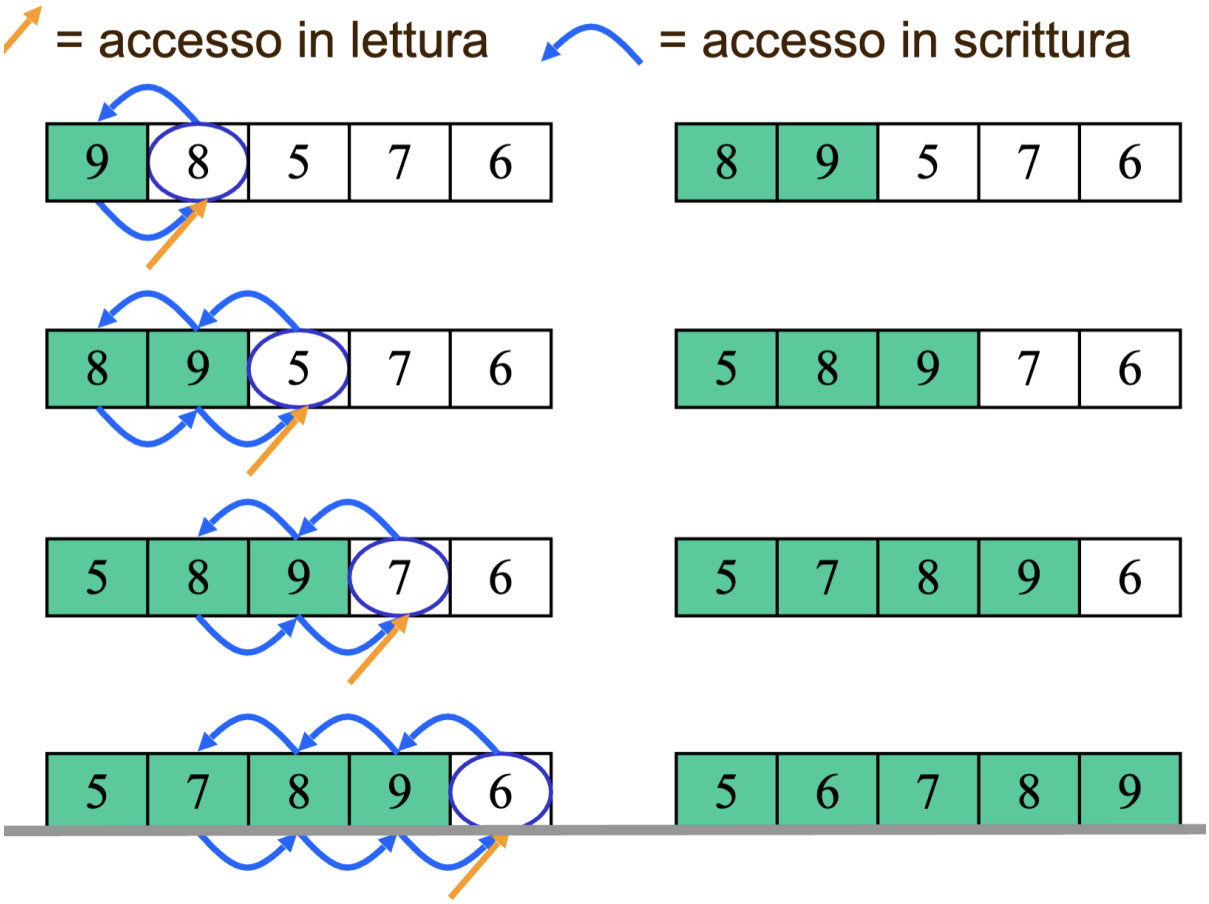
PRESTAZIONI

- Ordine $T(n) = O(n \log n)$

ORDINAMENTO PER INSERIMENTO

- Si inizia osservando **sottoarray** di lunghezza unitaria -> già **ordinato**

- Prendo primo elemento dell'array non ordinato
  - inserisco elemento nel **sottoarray ordinato** nella posizione corretta



IMPLEMENTAZIONE IN JAVA

```java
public static void insertionSort(int[] arrayFull, int arraySize) {
    // il ciclo inizia da 1 perché il primo elemento non richiede attenzione

    for (int i = 1; i < arraySize; i++) {
        int newElementToAdd = arrayFull[i];

        int j; // definisco fuori perché dopo devo utilizzarlo

        for (j = i; j > 0 && arrayFull[j - 1] > newElementToAdd; j--) {
            // sposto a destra ogni elemento maggiore di newElementToAdd
            arrayFull[j] = arrayFull[j - 1];
        }

        // inserisco nella posizione lasciata libera newElementToAdd
        arrayFull[j] = newElementToAdd;
    }
}
```

ORDINAMENTO DI OGGETTI

- Si possono applicare algoritmi di ordinamento a oggetti, non solo a numeri interi
- Gli algoritmi rimangono quasi invariati: l'unica differenza è l'utilizzo di `compareTo()`

ALGORITMI DI ORDINAMENTO DI LIBRERIA

- Package: `java.util.Arrays`
- Metodo `sort`
  - utilizza algoritmo QuickSort
  - ordina l'array di dati fondamentali e di Comparable
- Metodo `binarySearch`
  - restituisce posizione oggetto come numero intero