# 6. (optional): Advanced Sorting

## 6.1. Objectives

- Merge sort
- Shell sort
- Quick sort

## 6.2. Problem 1: PartitionApp.java

- Add counters for the number of comparisons and swaps and display them after partitioning.
- Investigate the relationship between the index of partitioning, the number of comparison, and the number of swaps.
- Do the previous exercise with different pivots:
    - beginning, end, or middle of the interval;
    - selected at random from the interval or from a larger interval;
    - last item in the array.
- Compute the average number of comparisons and swaps over 100 runs.

## 6.3. Merge sort, Shell sort, Quick sort

Code the following program as in the text book:

- mergeSort.java (listing 6.6, page 288)
- shellSort.java (listing 7.1, page 321)
- quickSort3.java (listing 7.5, page 351)

In each sorting algorithm, add counters for the number of comparisons, copies, and swaps and display them after sorting.

Create an array of integer numbers, fill the array with random data and print the number of **comparisons, copies, and swaps** made for sorting 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 50000 items and fill in the table below. Analyze the trend for the three different algorithms.

Compare with the same table in Lab 2 with simple sorting algorithms.

| **COPIES/ COMPARISONS/ SWAPS** | | | |
|---|---|---|---|
| | **Merge Sort** | **Shell Sort** | **Quick Sort** |
| **10000** | | | |
| **15000** | | | |
| **20000** | | | |
| **25000** | | | |
| **30000** | | | |
| **35000** | | | |
| **40000** | | | |
| **45000** | | | |
| **50000** | | | |