Exercise 1

Mergesort improvements. Write a program that implements the three improvements to mergesort that are described in the text: add a cutoff from small subarrays, test whether the array is already in order, and avoid the copy by switching arguments in the recursive code.

Test your program on a large random array and compare the running time of the improved version with the original version.

Exercise 2

Inversions. Develop and implement a linearithmic ($\Theta(n \log n)$) algorithm for computing the number of inversions in a given array (the number of exchanges that would be performed by insertion sort for that array).

Recall that an inversion is a pair of indices i and j such that $i < j$ but $a[i] > a[j]$. E.g., the array $2, 3, 8, 6, 1$ has five inversions: $(2, 1)$, $(3, 1)$, $(8, 6)$, $(8, 1)$, and $(6, 1)$.

This quantity is related to the *Kendall tau distance* in statistics, which is a measure of how close two rankings are to each other.

Hint: Modify mergesort to count inversions. While merging the two subarrays, count the number of inversions in which the right element is smaller than the left element.

Exercise 3

ShellSort. The goal of this exercise is to understand the impact of increment sequences on the performance of Shellsort and to find the best increment sequence based on the number of comparisons.