# Assignment 2

## Group 2:

Emma Lindell, Hannah Liljestam, Kerstin Wärja, Max Norberg

## 1 Comparison

One difference between the three algorithms is that Quick sort is a divide and conquer algorithm, Insertion sort operates incrementally and heapsort operates by using a heap.

There are differences in Best- and Worst-Case performance between the three algorithms, as can be seen in the figure below.

Table 1: Running time

| Algorithm | Worst-Case | Average-Case | Best-Case |
|---|---|---|---|
| Insertion Sort | **ϴ**(n$^2$) | **ϴ**(n$^2$) | **ϴ**(n) |
| Quick Sort | **ϴ**(n$^2$) | **ϴ**(nlog(n)) | **ϴ**(nlog(n)) |
| Heapsort | **O**(nlog(n)) | **O**(nlog(n)) | **O**(nlog(n)) |

For Best-Case we note that Insertion Sort is faster than Quicksort and Heapsort, and Insertion Sort is thus the best algorithm in a Best-Case situation. For Worst-Case we note that Heapsort is faster than Insertion Sort and Quicksort, and Heapsort is thus the best algorithm in a Worst_case situation.

If the input array is very large and inverse sorted, Heapsort is the most efficient sorting algorithm, but if the input is sorted and small, Insertion Sort is more efficient. In an average case, Quicksort and Heapsort have the same upper bound, while Quicksort becomes a little more efficient because it does not make as many unnecessary swaps as Heapsort.

## 2 Instructions to test our program

To test our sorting programs, follow the instructions below:

1. Call on the function *rangen(n)*
2. Call on the sorting program you wish to test by using one of the three functions:
    a. *run_insertionsort()*
    b. *run_quicksort()*
    c. *run_heapsort()*