Assignment 2 Algorithms and Data Structures 1 (1DL210) 2021

Simon Pettersson Palestro

Niclas Söderlund

Viktor Evestam

Joel Thyberg

Insertionsort:

+ Easy to implement compared to quicksort and insertionsort which are more complex algorithms. For a small array insertionsort can be the fastest among the algorithms as it does not contain any overhead from recursive function calls.
+ Linear best-time complexity when the list is already sorted, which then makes it quicker than the two other algorithms. A sorted list as input is actually the worst case for the type of Quicksort we implemented where the last element is chosen as pivot.
- Worst average time complexity n^2 compared to nlogn which is average for the two other algorithms. So for a large array insertionsort can be very slow compared to the other algorithms.

Heapsort:

+ Fixed time complexity nlogn means that it suits for large arrays and that the worst-case for this algorithm is faster than the worst case of the two other algorithms which are both n^2. This algorithm can be useful in situations where there are strict requirements on running time, where the two others can't guarantee being fast enough (even though Quicksort most likely would).
- Maintaining the Heaps (constant factor for Heapsort) cost more time than doing the partition (constant factor for Quicksort).

Quicksort:

+ This algorithm can avoid doing as many swaps as heapsort (which always swaps 100% of the elements) which makes Quicksort the fastest in general. In the best case where the pivot element is chosen as the middle value, very few swaps are needed.
+ nlogn average case complexity makes this algorithm suitable for large arrays