Algorithms and Data Structures Assignment 2

Michelle Pap, Oskar Bäcklin, Andreas Elenborg

September 2021

1 Comparison

1.1 Insertion sort

Insertion sort is a simple algorithm, it divides the array in two parts, one sorted and one unsorted. The algorithm takes values from the unsorted part one by one and places them in the sorted part until the whole array is sorted. One advantage with insertion sort compared the other sorting algorithms, is that it is simple to implement. An other advantage is that the algorithm is stable compared to the other ones. Which means that the algorithm does not change the original order of elements with identical keys after it has been sorted. However, quick sort and heap sort can be modified and become stable algorithms. One situation where insertion sort might be preferable over the other methods is for small arrays, as the size of the array won't impact the running time enough to justify using a method that is much more complicated in implementing.

1.2 Quicksort

Quick sort uses partition, it first selects a pivot element and then it divides the other elements into two sub-lists deepening on the size of the element compared to the pivot element. With recursion, the sub-lists are then sorted. One advantage with quick sort is that the algorithm is more efficient when working with real-world data. The pivot element can be chosen in order to optimize the running time. Therefore, the worst case of the data does not occur often. A situation where this method might be preferable to the others is if you are working with real data, as it is often more efficient in these cases.

1.3 Heapsort

Heapsort relies on the concept of heaps, a form of binary tree, in sorting. The principle is to create a max-heap, a heap where no child-node has a larger value than its parent-node, and then switch the root with the element of last index. As the root will contain the largest value in a max-heap, the last index now containing the root can then be ignored as the sorted part of the array, while the remaining array is again made into a max-heap and the process repeats.

With each iteration, the largest remaining value will be added to the sorted values, and so resulting in a sorted array once all of the heap has been sorted. The major advantage of this method is that it is very fast, however the main disadvantage is that it is considerably more complicated to implement than the other methods mentioned here. So for a very large array this method might be preferred over the others, as it will be faster, but for small arrays one of the other might be preferable. One situation where heap sort is more preferable to use is for the worst case scenarios. Heap sort has a running time of $\mathcal{O}(n \log n)$ while insertion sort and quick sort has a running time of $\mathcal{O}(n^2)$.