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

Felicia Fredriksson

October 4, 2021

1 Comparison of ***InsertionSort***, ***QuickSort*** and ***HeapSort***

Each of these three algorithms are "in place" algorithms and is operating in the input array.
Heapsort have the same time complexity for Best case, Average case and Worst case, $T(n) = \Theta(nlog(n)n)$, and Quicksort has the same time complexity for its Best case and Average case . But the worst case for Quicksort is $\Theta(n^2)$.
InsertionSort has $T(n) = \Theta(n^2)$ for Worst case and Average case, but the Best case is linear, $T(n) = \Theta(n)$.
In lecture 4 we learned that QuickSort can be very heavy to perform for arrays with small size. In those cases it is better to implement InsertionSort since it will work very well for smaller arrays.
HeapSort has a better (or equal) time complexity for the Worst case and Average case than the other two algorithms. It is only in the best case of InsertionSort that this algorithm will grow slower with the size of the array than HeapSort. Thus, HeapSort is good to implement for cases of really large arrays (since then we can guarantee the time complexity does not become larger than $nlog(n)$). QuickSort can be faster than HeapSort, but then we take a risk that we might have a "worst case"-scenario and it takes a long time.
Another important difference is to emphasize that QuickSort is the only *Divide-and-Conquer* algorithm of these three. Since it divides the problem each time this speeds up the process quite well. So if speed is really important this algorithm is the best, (but there is a risk with the worst case).