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

Polina Barsuk, Javier Eguíluz Romero, Jonas Frankemölle, Erik Jarvis

October 4, 2021

Comparison of Sorting algorithms

Insertion Sort

When all elements of an array are already sorted or equal, Insertion sort has an advantage over the two other algorithms. The complexity of sorting the described array using Insertion Sort will be in best case $\mathcal{O}(n)$, which is faster than the best case complexity of Quicksort ($\Theta(nlog(n))$) or Heapsort ($\mathcal{O}(n \log n)$). Also, Insertion Sort is better to use than other algorithms when the number of elements in an array is small, since it is easy algorithm and fast to implement. However, its average and worst case complexity is slower than the ones of Quicksort and Heapsort.

Quicksort

In comparison to Insertion Sort, Quicksort has a faster average and worse case runtime which makes it generally a faster algorithm than Insertion Sort. Compared to Heapsort, Quicksort has a slower worst-case $\Theta(n^2)$. Average-case and best-case are bounded by $\Theta(nlog(n))$, whereas Heapsort is bounded by $\mathcal{O}(nlog(n))$, which suggests that Heapsort is faster in these cases. However, Quicksort has advantages over Heapsort when the array is already sorted, or at least part of it: in this case, Quicksort is more efficient since it does not have to swap elements - unlike Heapsort, which has to swap the elements (all in best-case). This makes Quicksort much faster than Heapsort.

Heapsort

Heapsort is slower than Insertion Sort in best-case ($\mathcal{O}(nlog(n))$, but is faster in average- and worst-case. See the above section for the differences to Quicksort. The strength of Heapsort is the worst-case run time - here it is much faster than the other 2 algorithms, with a run time of $\mathcal{O}(nlog(n))$. Insertion sort and Quicksort only guarantee $\Theta(n^2)$. Also, data structures like max-heap allow us to easily extract the maximum value (as it is the root), insert an element into the array, remove the maximum value, and find an element with particular key and increase the value of it.