Comparison sorting methods

Grupp 18: Caroline Bark, Nils Risberg och Emelie Åsbrink

Insertion sort: The time complexity for the best case with insertion sort is Θ(n), where n is the length of the list. This is the quickest of all methods, which is an advantage to insertion sort. However, the needed iterations rise rapidly when the list becomes more unsorted, both the average case and the worst case have a factor of n^2 for the running time which is significantly larger than quicksort and heapsort.

Quicksort: This sorting method is better than insertion sort for more randomized lists and has for almost all cases a running time with a factor of nlog(n). The downside is the unlucky case of choosing a pivot element that fails to change the list efficiently. For this case quicksort has quadratic time complexity, which is the same as insertion sort's worst case. However, this case can be avoided by choosing the pivot element in a smart way. For example, one way is to take a random number instead of picking the last element.

Heapsort: For heapsort, best case, worst case and average case all have nlog(n) time complexity. Compared to quicksort, heapsort has a better worst case running time and the same average case and best case. However, heapsort requires a heavy datastructure and is slower than quicksort in that sense. Therefore in cases where you only want to sort the array it is preferable to use quicksort if the pivot element is chosen in a good way. The advantage with heapsort is that operations such as adding a new element or deleting an element is much more efficient. So in cases where you want to do more than just sort your list it is worth it to implement the max-heap datastructure and use heapsort.