# Sorting and selection – handout 1

November 9, 2017

In class (November 2 and 9) we discussed the deterministic sorting algorithm *insertion sort* that uses roughly $n \log n$ comparisons (which is nearly best possible) and the randomized sorting algorithm *quicksort*, for which we showed a proof that the expected number of comparisons is roughly $2n \ln n$.

We presented a simple randomized algorithm for selecting the median (*quickselect*) that makes $O(n)$ comparisons in expectation, and a more complicated one that makes roughly $\frac{3n}{2}$ comparisons. In passing we encountered some important principles for probabilistic analysis, namely, linearity of expectation, and concentration bounds for independent random variables.

We also presented a deterministic median selection algorithm that uses $O(n)$ comparisons, and showed that every such algorithm needs to make at least $\frac{3n}{2}$ comparisons.

Other principles encountered are *divide and conquer algorithms* (the use of the splitting item for quicksort), and *Yao's principle* for lower bounding the expected running time of randomized algorithms, and the *principle of deferred decisions* for analysing randomized algorithms.

**No class on November 16**.

**Homework – hand in (either in Hebrew or English) by November 23**.

1) We have seen in class a deterministic algorithm for selecting the median based on partitioning the items into groups of size 5. The number of comparisons it uses is at most $18n$. Design a similar algorithm based on partitioning the items into groups of size 7, and prove an upper bound better than $18n$ (e.g., $17n$ or $16n$) on the number of comparisons that it makes. (When analysing the algorithm, you may assume for simplicity that all relevant numbers are divisible by 7.)

2) We have seen a proof that every deterministic comparison-based median selection algorithm needs to make at least $3n/2$ comparisons (up to low order terms) in the worst case. Prove for some $\delta > 0$ of your choice (say, $\delta = \frac{1}{10}$) that for every randomized algorithm there is some input on which the expected number of comparisons that it makes is at least $(1 + \delta)n$ (up to low order terms). [Hints: use Yao's principle. Then, rather than fixing a random permutation in advance, use the principle of deferred decisions: whenever the algorithm makes a comparison that involves an item that has not been involved in any previous comparison, at that point the item is given a random value in $\{1, 2, \ldots, n\}$, among the values not given to previous items.]