# Homework 2

Due: September 28, 2021 - 14:30 ET

This homework must be typed in LaTeX and handed in via Gradescope.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before you write. Except in the rare cases where it is indicated otherwise, consider every problem as asking you to prove your result.

### Problem 1

3-Mergesort is a variation of Mergesort, such that:

- In the divide phase: the input is split into 3 sub-sequences on which 3-Mergesort is recursively invoked;
- In the conquer -merge- phase: the three sorted lists are merged in a single sorted list, using a variation of the Merging algorithm called 3-Merge.

For the sake of simplicity, assume that the size of the sequence of elements to be sorted is a perfect power of 3.

1. Provide a detailed, but concise, presentation of 3-Mergesort.
2. Consider a k-way mergesort. Provide a generalization of the number of comparisons executed in the worst case. Please count the *actual* number of comparisons.
3. Is 3-Mergesort faster than standard Mergesort? Is it asymptotically faster? (**Hint:** you may want to think about the actual constant terms vs the asymptotic notation).

**Problem 2**

1. Assume that we want to sort sequences such that each element is placed in the input at most $k$ locations-away from its placement in the ordered sequence. Design a modification of Selection sort which runs in time O(nk).

2. Under the same assumption, what can we say about the worst-case runtime of Insertion Sort?

3. For input sequences exhibiting this property, which among Mergesort, Insertion Sort and Standard Selection sort offered better performance? Motivate your answer.

4. Assume now that we are looking to sort input sequences in which there is a subsequence of size $k$ which is already sorted. Show that using one among Selection, Insertion, or Bubble sort it is possible to sort such inputs in O(n(n-k)) steps.

**Problem 3**

1. Say we are given sorted list S with n items, i.e. n entries of the form (key, value) whose keys are strictly increasing with the indexing. Consider a "scrambled" version of it $S'$ in which elements may have been moved by at most two locations: that is, elements previously in index $i$ may be indexed within the range $[i-2, i+2]$. Provide a searching algorithm that still provides the functionality of the searching algorithm discussed in class with runtime O(log n). Provide proof for the correctness and the runtime.