Algorithms and Data Structures

Summer Semester 2025

For discussion on Wednesday, July 9, 2025

1. Is the array-based implementation of merge-sort given in

   `https://github.com/mjwestcott/Goodrich/blob/master/ch12/merge_array.py`

   stable? Explain why or why not.

2. Is the linked-list-based implementation of merge-sort given in

   `https://github.com/mjwestcott/Goodrich/blob/master/ch12/merge_queue.py`

   stable? Explain why or why not.

3. (GTG Exercise R-12.8) Suppose we modify the deterministic version of the quick-sort algorithm so that, instead of selecting the last element in an n-element sequence as the pivot, we choose the element at index $\lfloor n/2 \rfloor$. What is the running time of this version of quick-sort on a sequence that is already sorted?

4. (GTG Exercise R-12.7) Suppose we are given two n-element sorted sequences A and B each with distinct elements, but potentially some elements that are in both sequences. Describe an $O(n)$-time method for computing a sequence representing the union $A \cup B$ (with no duplicates) as a sorted sequence.

5. (GTG Exercise R-12.19) Suppose S is a sequence of n values, each equal to 0 or 1. How long will it take to sort S with the merge-sort algorithm? What about quick-sort?

6. (GTG Exercise R-12.20) Suppose S is a sequence of n values, each equal to 0 or 1. How long will it take to sort S stably with the bucket-sort algorithm?

7. (GTG Exercise R-12.21) Given a sequence S of n values, each equal to 0 or 1, describe an in-place method for sorting S.

8. (GTG Exercise R-12.10) Show that the best-case running time of quick-sort on a sequence of size n with distinct elements is $\Omega(n \log n)$.

9. (GTG Exercise C-12.37) Show that any comparison-based sorting algorithm can be made to be stable without affecting its asymptotic running time.

10. (GTG Exercise C-12.39) Given an array A of n integers in the range $[0, n^2 - 1]$, describe a simple method for sorting A in $O(n)$ time using at most $O(n)$ extra space.