CSCI 3333 Homework: Sorting

1 Algorithm Analysis

Problem 1. (Based on Book Exercise 7.1) Show the step-by-step result of performing insertion sort on the following array: 3, 5, 2, 1, 5, 6, 0.

Problem 2. (Based on Book Exercise 7.2) Give the worst-case asymptotic running time of the following sorting algorithms for input array of \( n \) identical items:

- Insertion sort.
- Mergesort.
- Quicksort (assume pivot is first element).

Problem 3. Mergesort (as seen in class) is actually just one algorithm in a family of mergesort algorithms that partition the input list into 2, 3, 4, etc. equal-sized sublists; these algorithms are called 2-way, 3-way, 4-way, etc. mergesort. Analyze the running time of 3-way mergesort (give the recurrence relation, find a closed form, prove the closed form correct).

Problem 4. Suppose you will run quicksort only on inputs that are almost sorted (i.e., \( A[i] \leq A[j] \) for all but at most \( O(1) \) index pairs \( i, j \) with \( i < j \)). For these inputs, give the worst-case running time of the following sorting algorithms:

- Insertion sort.
- Mergesort.
- Quicksort (selecting the first element as pivot).

2 Algorithm Design

A sequence of numbers \( A_1, A_2, \ldots, A_n \) is sign-sorted provided that for every pair of elements \( A_i, A_j \) with \( i < j \):

- \( A_i, A_j > 0 \Rightarrow A_i \leq A_j \).
- \( A_i, A_j < 0 \Rightarrow A_i \leq A_j \).

Problem 5. Give a \( O(n) \)-time algorithm that determines whether an array \( A \) (of length \( n \)) is sign-sorted.

Problem 6. Give a \( O(n) \)-time algorithm that sorts a sign-sorted array \( A \) (of length \( n \)).

A sequence of numbers \( A_1, A_2, \ldots, A_n \) is half-sorted provided that for at least half of the consecutive pairs \( A_i, A_{i+1}, A_i \leq A_{i+1} \).
Problem 7. Give an algorithm for the following “half-sorting” problem:

- Input: an array \( A \) of length \( n \).
- Output: an array containing a permutation of the items in \( A \) such that at most half of the indices have the property that \( A[i] > A[i+1] \).

Now try to give a \( O(n) \)-time algorithm for the problem. (Hint: if a sequence of things has no two consecutive “bad” things, then at most half of the things are “bad”).

3 Lower Bounds

Problem 8. (Based on Book Exercise 7.52) Prove that any comparison-based sorting algorithm has \( \Omega(n \log(n)) \) average-case running time.