Problem 1. Quicksort

Do Weiss problem 7.19. Please follow the algorithm given by the book. You should show the results of all steps that involve swapping elements, with enough text so that someone reading your solution can follow what you’re doing. The problem notes that there is a ”cutoff” value - this refers to the idea of running insertion sort on subarrays that are as small as the cutoff value. You don’t need to show any steps of insertion sort; simply assume it sorts the values.

Problem 2. Comparing Sorting Algorithms

Consider the following sorting algorithms: selection, insertion, heap, merge, and quicksort(using the first element as the pivot). For each of the following inputs, state the $\Theta$ (tight bound) complexity for every one of the above algorithms. Assume in each case that the array contains $N$ items. You do not need to show your work, but it may be helpful for partial credit.

(a) input is sorted already (smallest to largest).

(b) input is reverse sorted (largest to smallest).

(c) For each algorithm, what does the worst-case input look like? A one-sentence, general description will suffice for each.

Problem 3. Some searching, some sorting, and some sums

Suppose you are given as input $n$ positive integers and a number $k$. Our goal is to develop an algorithm to determine if there are any four of them, repetitions allowed, that sum to $k$. As an example, if $n = 7$, the input numbers are 6, 1, 7, 12, 5, 2, 14 and $k = 15$, the answer should be YES because $6 + 5 + 2 + 2 = 15$.

(a) First solve the simpler problem of determining if there are any two numbers in a list that sum to $k$ (repetitions allowed). Describe, in words, an algorithm that will achieve this goal in $O(n \log n)$ worst case time. Explain how your algorithm achieves this complexity bound.

(b) Now observe that the sum of four numbers is the sum of two pairs of numbers. Carefully explain how we can solve the original problem (i.e., whether a four number sum exists). For full credit, show that your algorithm runs in worst case time $O(n^2 \log n)$.