These are practice problems for the upcoming midterm exam. You will be given a sheet of notes for the exam. Also, go over your homework assignments. **Warning:** This does not necessarily reflect the length, difficulty, or coverage of the actual exam.

Problem 1. For this problem, assume that you are using Insertion Sort with a sentinel (as done in class). Assume that the first, second, and third smallest elements are next to each other (somewhere in the input), the fourth, fifth, and sixth smallest elements are next to each other, the 7th, 8th, and 9th elements are next to each other, etc. For example,

80, 70, 90, 40, 50, 60, 30, 10, 20, 120, 110, 100

(The algorithm does not know this, and executes without this extra information.) Assume the problem size n is a multiple of 3. For each part show your work.

(a) Assume each of the above triples are in reverse order (so that the third smallest element comes before the second smallest which comes before the smallest, the sixth smallest comes before the fifth smallest which comes before the fourth smallest, etc.). For example,

90, 80, 70, 60, 50, 40, 30, 20, 10, 120, 110, 100

What is the exact number of comparisons in the best case?

(b) Assume each of the above triples are in order (so that the smallest element comes before the second smallest which comes before the third smallest, the fourth smallest comes before the fifth smallest which comes before the sixth smallest, etc.). For example,

70, 80, 90, 40, 50, 60, 10, 20, 30, 100, 110, 120

What is the exact number of comparisons in the worst case?

(c) Assume each of the above triples are in reverse order (as in part (a)), and $n = 6$. Calculate the exact number of comparisons in the average case?

Problem 2. Solve the following recurrences exactly using the tree method. You may assume n is "nice". Prove your answers using mathematical induction.

(i) $T(n) = 2T(n/2) + n^3$, $T(1) = 1$.

(ii) $T(n) = T(\sqrt{n}) + 1$, $T(2) = 1$.

(iii) $T(n) = 2T(n/2) + n \lg n$, $T(1) = 1$.

(iv) $T(n) = T(n-3) + 5$, $T(1) = 2$.

Problem 3. Assume you have an array of numbers, where each value occurs *at most* twice. We consider sums of contiguous numbers in the array. But we only consider such sums whose two endpoints have the same value. The sum includes the two equal values themselves. So if the two equal numbers are at index i and index j ($i < j$) in array A, then we sum all the values $A[i], A[i+1], \ldots, A[j]$.

(a) Give an algorithm that finds the maximum such sum. Make your algorithm as efficient as possible. Describe the algorithm briefly in English *and* in psuedo code.

(b) Analyze the running time of your algorithm.

Problem 4. Let $A[1, .., n]$ be an array of n numbers (some positive and some negative).

(a) Give an algorithm to find which two numbers have sum closest to zero. Make your algorithm as efficient as possible. Write it in pseudo-code.

(b) Analyze its running time.