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.

(a) Is $2^{n+1} = O(2^n)$?

(a) Is $2^{2n} = O(2^n)$?

Problem 2. Solve the following problems directly.

(a) What is the best-case number of moves in Insertion Sort? Justify.

(b) What is the worst-case number of moves in Insertion Sort? Justify.

(c) What is the average-case number of moves in Insertion Sort? Justify.

Problem 3. Consider the following recurrences:

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

You may assume n is "nice".

(a) Which of the above problems can be solved using the tree method? Solve those exactly using the tree method.

(b) Which of the above problems can be solved using the "Master Theorem" derived in class. Solve those exactly using the "Master Theorem".

Problem 4. 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 5. 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.

Problem 6. Assume the you have a heap of size $n = 2^m - 1$ (stored in an array). (The largest element is at the root.) You may describe the following algorithms in high level language, but they *must* be clear. Give your answers as a function of n. You may assume n is large. You may use extra memory.

(a) Give an algorithm, minimizing the number of comparisons, to find the smallest element in the heap. Exactly how many comparisons does it take.

(b) Give an algorithm, minimizing the number of comparisons, to find the second smallest element in the heap. Exactly how many comparisons does it take.

(c) Give an algorithm, minimizing the number of comparisons, to find the third smallest element in the heap. Exactly how many comparisons does it take.

Problem 7. Show that

(a)

$$\frac{1}{2} \leq \sum_{j=1}^{\infty} \frac{1}{j2^j} \leq 1$$

(b)

$$1 \leq \sum_{j=1}^{\infty} \frac{1}{j^2} \leq 2$$