CMSC 351: Practice Questions for Final Exam

These are practice problems for the upcoming final 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.** Describe how to modify quicksort so that it has a $\Theta(n \log n)$ worst case (rather than just average case). Is this a good idea? Why or why not?

**Problem 2.** Let $A[1, \ldots, n]$ 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 3.** Let $A[1, \ldots, n]$ be an array of $n$ numbers (some positive and some negative).

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

(b) Analyze its running time.

**Problem 4.** Assume that you developed an algorithm to find the (index of the) $n/3$ smallest element of a list of $n$ elements in $2n$ comparisons.

(a) Using the algorithm (as a black box), give an algorithm, efficient in the worst case, to find the $k$th smallest element of a list.

(b) Write down a recurrence for (a bound on) the number of comparisons it executes in the worst case.

(c) Solve the recurrence (using constructive induction). Find the high order term exactly (but you do not need any low order terms).

(d) Using the (black box) algorithm for finding the $n/3$ smallest element and using the ideas and results of Parts (a), (b), and (c), give an efficient algorithm to find (the index of) two elements, the $k_1$th smallest and the $k_2$ smallest (for inputs $k_1$ and $k_2$). The algorithm description can be very high level and brief.

(e) How many comparisons does it use? Find the high order term exactly (but you do not need any low order terms). Give a brief justification.

**Problem 5.**

(a) Give a lower bound for sorting $n$ integers in range $1, \ldots, k$ using a *comparison-based algorithm*. You may assume $k \leq n$. [We know that counting sort can solve this problem in time $\Theta(n + k)$, but it uses basic operations other than comparisons.]

(b) Give an efficient comparison-based algorithm for sorting $n$ integers in range $1, \ldots, k$.

(c) Analyze your algorithm.

(d) Compare the upper and lower bounds.
Problem 6. A graph is tripartite if the vertices can be partitioned into three sets so that there are no edges internal to any set. The complete tripartite graph, $K(a,b,c)$, has three sets of vertices with sizes $a$, $b$, and $c$ and all possible edges between each pair of sets of vertices. $K(3,2,3)$ is pictured below. A Hamiltonian cycle in a graph is a cycle that traverses every vertex exactly once.

(a) For which values of $n$ does $K(1,1,n)$ have a Hamiltonian cycle. Justify your answer.
(b) For which values of $n$ does $K(1,n,n)$ have a Hamiltonian cycle. Justify your answer.
(c) For which values of $n$ does $K(n,n,n)$ have a Hamiltonian cycle. Justify your answer.

Problem 7. Let $G = (V,E)$ be an undirected graph. A triangle is a set of three vertices such that each pair has an edge.

(a) Give an efficient algorithm to find all of the triangles in a graph.
(b) How fast is your algorithm?

Problem 8. Answer each of the following clearly but briefly.

(a) What is the class $P$?
(b) What is the class $NP$?
(c) What is the class $NP$-complete?
(d) Name a problem that is $NP$-complete.
(e) Name a problem that is not $NP$-complete. Why not?