Problem 1. 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 2. 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 3. 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 4. Show that you can convert a formula in Conjunctive Normal Form (CNF) where every clause has at most three literals, into a new formula where every clause has exactly three literals, so that the new formula is satisfiable if and only if the original formula is satisfiable. No variable may occur twice in the same clause.

Problem 5. In a graph $G = (V, E)$ edge $(x, y)$ touches vertices $x$ and $y$.

A newtonian cluster in a graph $G = (V, E)$ is a subset of the edges such that every vertex is touched by at least one edge. The size of a newtonian cluster is the number of edges in the subset.

(a) Give an example of a graph that has a newtonian cluster of size four but not of size three.

(b) Let $C$ be a newtonian cluster of $G$. What can you say about the minimum size of $C$ as a function of the number of vertices $n$? Justify.

(c) A minimal newtonian cluster is a newtonian cluster such that if any edge is removed it is no longer a newtonian cluster. Let $C$ be a minimal newtonian cluster of $G$. What can you say about the maximum size of $C$ as a function of the number of vertices $n$? Justify.

(d) The (decision version of) newtonian cluster problem is given a graph $G = (V, E)$ and an integer $k$ does $G$ have a newtonian cluster of size (at most) $k$. Show that the newtonian cluster problem is in NP. What is the certificate?

Problem 6. A vertex cover in a graph $G = (V, E)$ is a subset of vertices such every edge is incident on at least one vertex of the subset. The Weighted Vertex Cover Problem (WVCP) is, given a graph $G = (V, E)$ with integer weights on the vertices, find a vertex cover whose sum of weights is as small as possible. You can assume that the weights are between 1 and $n$ (inclusive).

(a) WVCP is an optimization problem. Define a decision version of WVCP.

(b) Show that the decision version is in NP. Make sure to state the certificate and give the pseudo code.

(c) Show that if you could solve the optimization version in polynomial time that you could also solve the decision version in polynomial time.

(d) Show that if you could solve the decision version in polynomial time that you could also solve the optimization version in polynomial time. HINT: First find the weight of an optimal weighted vertex cover.

Problem 7. This problem is more open-ended than you would see on an exam: If you do not know how to play Sudoku, look it up. Normally, Sudoku is played on a $9 \times 9$ grid.

(a) Generalize Sudoku to larger grids.

(b) State the (generalized) Sudoku game as a decision problem.

(c) Show that the decision version of (generalized) Sudoku is in NP.

(d) Show that if you can solve the decision version of (generalized) Sudoku in polynomial time, you can solve a (generalized) Sudoku puzzle in polynomial time.