Question 1: Greedy. A coloring of a graph $G = (V, E)$ is an assignment of colors to the vertices of V. We say a graph is 2-colorable if it is possible to color the vertices using only two colors such that adjacent vertices have different colors.

- Give a polynomial time algorithm for the graph coloring problem.
- Argue that your algorithm runs in time polynomial in n, where $n = |V|$ is the number of vertices in G.
- Argue that your algorithm is correct.
- Give an example of a graph that is not 2-colorable.

Question 2: Divide and Conquer. Given is n particles placed at regular intervals along a straight line; each particle j has a charge q_j (either positive or negative). A computational prediction of the force F_j on particle j, defined by Coulomb's Law, is equal to:

$$F_j = \sum_{i<j} \frac{Cq_iq_j}{(i-j)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2},$$

where C is a constant. Assume that computing a term $\frac{Cq_iq_j}{(i-j)^2}$ can be done in constant time.

- Give a trivial algorithm that computes the force F_j for all particles j in $O(n^2)$ running time.
- Devise a divide and conquer algorithm that does the above job in $O(n \log n)$ running time.
- Argue the running time and the correctness.

Question 3: Dynamic Programming. We are given a set $\{x_1, x_2, \ldots, x_n\}$ of n integers and want to know if there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in S} x_i = \sum_{i \notin S} x_i.$$

- Devise a dynamic programming algorithm for the partitioning problem that runs in time polynomial with respect to n and W where $W = \sum_{i=1}^{n} x_i$.

- Argue the running time and the correctness.

Question 4: Dynamic Programming. You have q kids and they have received a long candybar for sharing. You are now going to cut the bar into q pieces by cutting it at $q-1$ places so that the smallest piece is as large as possible. You can, however, only cut the candybar at certain pre-specified places.

Formally, given is an array $A[1:n]$ of of n positive integers. The weight of a subarray $A[i:j]$ is the sum $A[i]+A[i+1]+\cdots+A[j]$, where $1 \leq i \leq j \leq n$. You want to cut the array into q subarrays such that the smallest weight among subarrays become maximized (as large as you can).

- What would bet the running time for an exhaustive search method to find the solution.

- Devise a dynamic programming algorithm that runs faster than exhaustive search method.

- Make a non-trivial example for which your algorithm works.

- Argue the running time and the correctness.

Question 5: NP-completeness. We say that a clause is proper if it contains each variable at most once, and if it does not contain a variable and its negation. For example, the clause $x_1 \vee \overline{x_2} \vee x_3$ is proper, while the clauses $x_1 \vee \overline{x_1} \vee x_2$ and $x_1 \vee x_1 \vee x_2$ are not proper. The decision problem PROPER-k-SAT takes as input a formula ϕ with proper clauses each of length k. The problem is to decide whether ϕ is satisfiable.

- Prove that PROPER-3-SAT is NP-complete (note $k = 3$).