There are 5 problems, accounting for 100% in total.

Problem 1 (15%)**.** Consider the Knapsack problem. Let $\mathcal{I}$ be the set of items, a_i, b_i be the size and the profit of item i for any $i \in \mathcal{I}$, and W be the knapsack size. Consider the following greedy algorithm:

1. Sort the items in $\mathcal{I}$ in non-increasing order of b_i/a_i for all $i \in \mathcal{I}$.
2. Consider the items in the sorted order. For each item considered, pick the item in the knapsack whenever possible.

Show that the above greedy algorithm can perform arbitrarily bad. That is, for any given constant $\epsilon > 0$, provide an instance such that the solution produced by the above algorithm on this instance has a total value at most ϵ times the value of the optimal solution.

Problem 2 (20%)**.** Consider the following four linear programs.

1. $\min\{\ cx \mid Ax \leq b\ \}$
2. $\max\{\ cx \mid Ax \leq b\ \}$
3. $\max\{\ cx \mid Ax \leq b,\ x \geq 0\ \}$
4. $\max\{\ cx \mid Ax = b,\ x \geq 0\ \}$

Show that they can be reduced to each other. That is, each of the four linear programs can be rewritten as another linear program such that they have the same optimal value. Conclude that it makes sense to assume that LPs are always written in canonical forms.

Problem 3 (20%)**.** Consider the Knapsack problem again. For any $0 \leq i \leq n$ and any $s \geq 0$, define $A'(i, s)$ to be the maximum profit that can be achieved by combinations that have size exactly s and that use only the first i items.

1. Derive a recurrence formula for $A'(i, s)$ for any i and s. Briefly justify your answer.
2. Does this recurrence formula lead to an FPTAS for the Knapsack problem?

   If so, describe how it can be done and justify your answer. If not, describe your reason.

Problem 4 (20%)**.** Let $G = (V, E)$ be a given graph. We have seen that, any maximal independent set for G^2 dominates G^2 and lower-bounds any dominating set for G in size.

Consider the following greedy algorithm, which computes a dominating set for G^2 that is *minimal* is size.

Algorithm 1 GREEDY-ALGO-4-DOMINATING-SET

1: $U \longleftarrow V$.
2: **while** there exists $v \in U$ such that $U \setminus \{v\}$ still dominates V in G^2 **do**
3: Remove v from U.
4: **end while**
5: Output U.

Prove or disprove that, the set U output by the greedy algorithm lower-bounds the size of the optimal dominating set for any given graph G.

Problem 5. Let $G = (V, E)$ be an undirected graph. For any subset $A, B \subseteq V$, let $\mathcal{C}(A, B)$ denote the set of edges between A and B, i.e.,

$$\mathcal{C}(A, B) \ := \ \{\, e \in E : \ e \text{ connects some vertex in } A \text{ and some vertex in } B \,\}.$$

Note that, $\mathcal{C}(A, B)$ is also referred to as the set of *cut edges* between A and B.

In the cardinality maximum cut problem, the goal is to compute a partition of V into two sets S and $\overline{S}$ such that the cardinality of $\mathcal{C}(A, B)$ is maximized.

- (10%) Consider the following *randomized algorithm* for max-cut.

  1. For each $v \in V$, pick $x_v \in \{0, 1\}$ independently and uniformly at random.

     Let $S_0 := \{v \in V : x_v = 0\}$ and $S_1 := V - S_0$.
  2. Output (S_0, S_1).

  Prove that, for any $e \in E$, $\Pr[e \in \mathcal{C}(S_0, S_1)] = 1/2$ and establish that $\mathcal{C}(S_0, S_1)$ is a randomized $(1/2)$-approximation for the max-cut problem, i.e., the expected number of edges in $\mathcal{C}(S_0, S_1)$ is at least one-half of the number in any optimal cut.

- (10%) Consider the following algorithm.

  1. Let $A \leftarrow \{v_1\}$, $B \leftarrow \{v_2\}$.
  2. For each $v \in V = \{v_3, v_4, \ldots, v_n\}$, do
     - If $|\mathcal{C}(\{v\}, A)| \leq |\mathcal{C}(\{v\}, B)|$, then add v to A.

       Otherwise, add v to B.
  3. Output (A, B).

Show that the above algorithm also computes a (1/2)-approximation for the cardinality maximum cut problem.

Hint: For any vertex in $\{v_3, \ldots, v_n\}$, consider its incident edges to A and to B when it is processed by step 2.

- (5%) What are the upper-bounds you use for the optimal cuts in the analysis of the above two algorithms?