The answers must be word-processed or typed. You may substitute formulae and figures with hand-writings. Your submitted algorithms should be in the pseudo-code, not in any specific programming language. Answers deviating from these requirements will be returned without grading.

The answers must be the student’s own work. Idea sharing and referencing to others’ work are not allowed. Plagiarism and other forms of academic dishonesty will be handled within the guidelines of the Student Handbook and reported to the University.

There are 150 points in total (130 points for undergraduates)

1. (25 points) Problem Min Vertex Cover is defined as
   
   Input: a graph $G = (V, E)$;

   Output: a subset $C \subseteq V$, called vertex cover, such that
   (1) $\forall (u, v) \in E$, either $u \in C$ or $v \in C$, and
   (2) $|C|$ is the minimum.

   Design an exhaustive search algorithm to solve problem Min Vertex Cover such that it runs in time $O(1.619^n)$ or better.

2. (25 points) Optimization problem 0-1 Knapsack is defined as:
   
   Input: a set of $n$ items of values $v_i$ and sizes $s_i$, $i = 1, 2, \ldots, n$ and knapsack size $B$;

   Output: a subset $A \subseteq \{1, 2, \ldots, n\}$ such that
   
   $$\sum_{i \in A} s_i \leq B \quad \text{and} \quad \sum_{i \in A} v_i \text{ is the maximum}$$

   We formulate a decision problem 0-1 Knapsack Decision as follows:
Input: a set of $n$ items of values $v_i$ and sizes $s_i$, $i = 1, 2, \ldots, n$, bag size $B$, and value $k$;

Output: “yes” if and only if there is a packing of items into the bag with total value $\geq k$.

Prove that if there is a polynomial time algorithm for 0-1 Knapsack Decision, then there is a polynomial time algorithm for 0-1 Knapsack.

3. (10 points) Will the claim in Q2 still hold if the inequality “$\geq k$” in the definition of problem 0-1 Knapsack Decision is replaced with inequality “$\leq k$”? How about being replaced with equality “$= k$”? 

4. (20 points) Prove that problem 0-1 Knapsack Decision is in the class NP.

5. (10 points) Give a simple reason to explain that if a reduction function $f$ holds for $L_1 \leq L_2$, $f^{-1}$ usually does not hold for $L_2 \leq L_1$, where $f^{-1}$ is the inverse function of $f$.

6. (20 points, Graduate students only, bonus for undergraduates) Prove that if $L_1 \leq_p L_2$ and $L_2$ admits an algorithm that runs in time $O(n^{O(\log n)})$, then $L_1$ can be solved in time $O(n^{O(\log n)})$ as well.

7. (20 points) Assume two languages $L$ and $L'$ can be polynomially reduced to each other, i.e., $L \leq_p L'$ and $L' \leq_p L$. Prove that, if $L$ is NP-complete, so is $L'$.

8. (20 points) Prove that if some NP-hard problem can be solved in deterministic polynomial time, then $\mathcal{P} = \mathcal{NP}$.