CPSC 413-Fall 2020

Problem Set 7 — Complexity Theory

Total marks: 50

1. **(5 marks)** For some arbitrary problem P let x be the input and y the output. Assume that function f solves this problem. In other words, when calling $f(x)$ it will return a correct value for y.

   For another problem P' let x' be the input and y' the output. The following is a function that solves P' :

   $f'(x)$
   $x = \textbf{transform1}(x')$
   $y = f(x)$
   $y' = \textbf{transform2}(y)$
   return y'

   The functions **transform1** and **transform2** both run in polynomial time. (In other words, $P' \leq_p P$.)

   It is known that there is no polynomial time solution for P'. Is it possible that there exists a polynomial time solution for P? (In other words, is it possible that $f(x)$ runs in polynomial time?) Explain your answer.

2. **(5 marks)** Question 1 on page 505 of the textbook. You may assume the following facts:

   - INTERVAL SCHEDULING $\in \mathcal{P}$
   - INTERVAL SCHEDULING $\leq_P$ INDEPENDENT SET
   - INDEPENDENT SET $\in \mathcal{NP}$-COMPLETE
   - VERTEX COVER $\in \mathcal{NP}$-COMPLETE

In the remaining problems, you are asked to prove that a particular decision problem B is $\mathcal{NP}$-COMPLETE. The following are required in your solutions:

- Proof that $B \in \mathcal{NP}$:
  - State the input to the verification algorithm (input to the problem and a certificate).
  - Show that certificate size is polynomial in size to the remaining input.
  - Give the verification algorithm.
  - Prove that the verification algorithm is correct.
  - Show that the verification algorithm runs in polynomial time.

- Proof that $A \leq_P B$ for some problem $A \in \mathcal{NP}$-COMPLETE :
  - Give an algorithm to transform the input to A to an input to B.
  - Prove that the transformation algorithm runs in polynomial time.
  - Let s be an input to A and s' the transformed input to B. Prove that s is a "yes" instance of A *if and only if* s' is a "yes" instance of B.

3. **(20 marks)** Consider the following decision problems:

   PARTITION

   - Pre-condition: a sequence of positive integers $m_1, m_2, \ldots, m_n$.
   - Post-condition: output "yes" if there exists a subset $A \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in A} m_i = \sum_{i \notin A} m_i$; "no" otherwise.

   BIN PACKING

   - Pre-condition: a sequence of real numbers $s_1, s_2, \ldots, s_n \in [0, 1]$ and an integer K.
   - Post-condition: output "yes" if it is possible to place items with sizes $s_1, s_2, \ldots, s_n$ into at most K unit-size bins; "no" otherwise.

   Assuming that PARTITION is $\mathcal{NP}$-COMPLETE, prove that BIN PACKING is $\mathcal{NP}$-COMPLETE.

4. **(20 marks)** Question 8 on page 507-508 of the textbook. You may assume that the 3-DIMENSIONAL MATCHING problem is $\mathcal{NP}$-COMPLETE (see p.481 of the textbook for more discussion of this problem). The relevant problem definitions are:

   MAGNETS

   - Pre-condition:
     - set $M = \{m_1, \ldots, m_l\}$ of l magnets, where each magnet $m_i \in S = \{s_1, \ldots, s_n\}$ (one of the available symbols)
     - set W, where each element is in S^* (i.e., a string with characters taken from S representing a word that Madison knows)
   - Post-condition: "Yes" if there exists words $w_1, \ldots, w_k \in W$ such that the multi-set of symbols in these words is equal to M (i.e., the magnets can make up all the words with none left over), "no" otherwise.

   3-DIMENSIONAL MATCHING

   - Pre-condition:
     - disjoint sets X, Y, Z each of size n,
     - set $T \subseteq X \times Y \times Z$
   - Post-condition: "Yes" if there exists a set of n triples in T so that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples, "no" otherwise