COMP 3804 — Winter 2026 — Problem Set 7

Question 1: The *set cover problem* is defined as follows:

$$\textsc{SetCover} = \{(B, S_1, S_2, \ldots, S_m, K) : \begin{array}{l} B \text{ is a finite set; } m \text{ is an integer;} \\ S_1, S_2 \ldots, S_m \text{ are sets with } \cup_{i=1}^{m} S_i = B; \\ K \text{ is an integer;} \\ \text{there exists a subset } I \subseteq \{1, 2, \ldots, m\} \text{ of} \\ \text{size } K, \text{ such that } \cup_{i \in I} S_i = B \ \}. \end{array}$$

The *(0-1)-integer programming problem* is defined as follows:

$$\textsc{IntProg} = \{(A, K) : \begin{array}{l} A \text{ is an integer } n \times m \text{ matrix all of whose entries} \\ \text{are in } \{0, 1\}; \ K \text{ is an integer;} \\ \text{there exists a binary column vector } x \text{ of length } m \text{ with} \\ \text{exactly } K \text{ ones, such that } Ax \geq \mathbf{1} \\ \text{(componentwise) } \}, \end{array}$$

where $\mathbf{1}$ denotes the column vector of length n, all of whose entries are equal to 1.

Prove that $\textsc{SetCover} \leq_P \textsc{IntProg}$, i.e., in polynomial time, SetCover can be reduced to IntProg.

Question 2: The *subset sum problem* is defined as follows:

$$\textsc{SubsetSum} = \{(a_1, a_2, \ldots, a_m, b) : \begin{array}{l} m, a_1, a_2, \ldots, a_m \text{ are positive integers,} \\ b \text{ is a non-negative integer, and} \\ \exists I \subseteq \{1, 2, \ldots, m\} \text{ such that } \sum_{i \in I} a_i = b \ \}. \end{array}$$

Assume you have a polynomial-time algorithm A that decides, for any input sequence $(a_1, a_2, \ldots, a_m, b)$, whether or not $(a_1, a_2, \ldots, a_m, b) \in \textsc{SubsetSum}$. Note that this algorithm only returns YES or NO; it does not return anything else.

Design a polynomial-time algorithm B that takes an arbitrary sequence $(a_1, a_2, \ldots, a_m, b)$ as input.

- If $(a_1, a_2, \ldots, a_m, b) \in \textsc{SubsetSum}$, then B returns a subset I of $\{1, 2, \ldots, m\}$ such that $\sum_{i \in I} a_i = b$.
- If $(a_1, a_2, \ldots, a_m, b) \notin \textsc{SubsetSum}$, then B returns NO.

Your algorithm B may use algorithm A as a black box. As always, justify your answer.

Question 3: The *Hamilton cycle problem* is defined as follows:

$$\textsc{HamiltonCycle} = \{G : \ G \text{ is an undirected graph that has a Hamilton cycle}\}.$$

Let φ be a Boolean formula in the variables $x_1, x_2, \ldots, x_n$. We say that φ is in *conjunctive normal form* (CNF) if it is of the form

$$\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

where each C_i, $1 \leq i \leq m$, is of the following form:

$$C_i = l_1^i \vee l_2^i \vee \ldots \vee l_{k_i}^i.$$

Each l_j^i is a *literal*, which is either a variable or the negation of a variable.

The *satisfiability problem* is defined as follows:

$$\textsc{Sat} = \{\varphi : \varphi \text{ is in CNF-form and is satisfiable}\}.$$

Prove that $\textsc{HamiltonCycle} \leq_P \textsc{Sat}$, i.e., in polynomial time, HamiltonCycle can be reduced to Sat.
Hint: Let $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$. Use n^2 Boolean variables x_{ij}, $1 \leq i \leq n$, $1 \leq j \leq n$, where

$$x_{ij} = \textit{true} \Leftrightarrow \text{ vertex } v_i \text{ is the } j\text{-th vertex in the Hamilton cycle.}$$

Question 4: Justin Bieber is intrigued by the **P** versus **NP** problem. While taking a shower, Justin suddenly realizes that the problem is actually not very difficult; he comes up with a proof of what is now known as

Bieber's Theorem: P = NP.

Let φ be a Boolean formula in the variables $x_1, x_2, \ldots, x_n$.

We say that φ is in *conjunctive normal form* (CNF) if it is of the form

$$\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

where each C_i, $1 \leq i \leq m$, is of the following form:

$$C_i = l_1^i \vee l_2^i \vee \ldots \vee l_{k_i}^i.$$

Each l_j^i is a *literal*, which is either a variable or the negation of a variable.

We say that φ is in *disjunctive normal form* (DNF) if it is of the form

$$\varphi = C_1 \vee C_2 \vee \ldots \vee C_m,$$

where each C_i, $1 \leq i \leq m$, is of the following form:

$$C_i = l_1^i \wedge l_2^i \wedge \ldots \wedge l_{k_i}^i.$$

Again, each l_j^i is a literal.

We define the following two decision problems:

$$\textsc{Sat} = \{\varphi : \varphi \text{ is in CNF-form and is satisfiable}\}$$

and

$$\text{DNFSAT} = \{\varphi : \varphi \text{ is in DNF-form and is satisfiable}\}.$$

(4.1) Justin starts by proving that DNFSAT $\in$ **P**. Your task is to present a proof of this fact.

(4.2) Here is Justin's argument to complete the proof of Bieber's Theorem:

- Let φ be an arbitrary Boolean formula in CNF-form. We can use the basic rules of logic (such as De Morgan's Law) to rewrite φ as an equivalent Boolean formula φ' in DNF-form. Therefore,
$$\textsc{Sat} \leq_P \text{DNFSAT}.$$
- We have seen in **(4.1)** that DNFSAT $\in \mathbf{P}$.
- Since $\textsc{Sat} \leq_P$ DNFSAT and DNFSAT $\in \mathbf{P}$, we have $\textsc{Sat} \in \mathbf{P}$.
- Justin remembers from COMP 3804 that Sat is **NP**-complete.
- Thus, the **NP**-complete problem Sat belongs to **P**.
- Therefore, $\mathbf{P} = \mathbf{NP}$.

Is Justin's proof of Bieber's Theorem correct? As always, justify your answer.