Exercise Set 8: NP or not NP?

The first thing you should do in exercise8.tex is set up your name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your name and UVA email id, e.g., `\submitter{Grace Hopper (gmh1a)}`.

Before submitting , also remember to:

– List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. (Remember to update this before submitting if you work with more people.)

– Replace the second line in exercise8.tex, `\usepackage{uvatoc}` with `\usepackage[response1]{uvatoc}`, `\usepackage[response2]{uvatoc}`, etc. for the appropriate problem submission.

Collaborators and Resources: TODO: replace this with your collaborators and resources (if you did not have any, replace this with *None*)

Exercise 8-1: Silly Reductions

Consider the *SORTING* and *MINIMUM* problems defined below:

SORTING
Input: A list of n natural numbers, $x_1, x_2, x_3, \ldots, x_n$.

Output: An ordering of the input list, $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ where $\{i_1\} \cup \{i_2\} \cup \ldots \{i_n\} = \{1, 2, \ldots, n\}$ and for all $k \in \{1, 2, \ldots, n-1\}$, $x_{i_k} \leq x_{i_{k+1}}$.

MINIMUM
Input: A list of n natural numbers, $x_1, x_2, x_3, \ldots, x_n$.

Output: A member, x_m, such that $x_m \in \{x_1, x_2, \ldots, x_n\}$ and for all $k \in \{1, 2, \ldots, n\}$, $x_m \leq x_k$.

(a) Show that $\mathit{MINIMUM} \leq_p \mathit{SORTING}$.

(b) Show that $\mathit{SORTING} \leq_p \mathit{MINIMUM}$.

(c) Does this mean that *MINIMUM* and *SORTING* are equivalently hard problems?

Exercise 8-2: Jeffersonian Paths

The Hamiltonian Path problem (not named after Alexander), defined below, is known to be NP-complete.

HAMILTONIAN PATH
Input: A description of an undirected, finite graph, $G = (V, E)$.

Output: **True** if there exists a path in G that visits each vertex in V exactly once; otherwise **False**.

Despite his declarations to the contrary, Jefferson does not consider all vertices equal, and defines the *JEFFERSONIAN PATH* problem as:

> *JEFFERSONIAN PATH*
> **Input:** A description of an undirected, finite graph, $G = (V, E)$, and a partitioning of V into two subsets V_1 and V_2 such that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$.
>
> **Output: True** if there exists a path in G that visits each vertex in V exactly once, where all vertices in V_1 are visited before any vertex in V_2; otherwise **False**.

Prove *JEFFERSONIAN PATH* is NP-complete.

Exercise 8-3: TAUT

Definition 1 (CNF) We say that a formula is in conjunctive normal form (CNF for short) if it is an AND of ORs of variables or their negations. E.g. $(x_7 \vee \overline{x_{22}} \vee x_{15}) \wedge (x_{37} \vee x_{22} \vee \overline{x_7})$ is in CNF. We say that it is k-CNF if there are exactly k variables per clause (group of variables combined with OR).

Definition 2 (DNF) We say that a formula is in disjunctive normal form (DNF for short) if it is an OR of ANDs of variables or their negations. E.g. $(x_7 \wedge \overline{x_{22}} \wedge x_{15}) \vee (x_{37} \wedge x_{22} \wedge \overline{x_7})$ is in DNF. We say that it is k-DNF if there are exactly k variables per clause (group of variables combined with AND).

We know that 3-SAT is $\mathrm{NP-Complete}$, where 3-SAT requires determining whether there exists at least one way to assign Boolean values to each variable in a 3-CNF formula so that the formula evaluates to True.

For this question we will show 3-TAUT is $\mathrm{NP-Hard}$. This problem requires determining whether *every* assignment causes a 3-DNF formula to evaluate to True (i.e., no assignments will cause the formula to evaluate to False).

1. Show 3-TAUT is $\mathrm{NP-Hard}$.
2. 3-TAUT is not known to belong to NP. Give an intuitive reason why it is difficult to show that 3-TAUT belongs to NP.

Exercise 8-4: Running Time Analysis

Consider the *INCREMENT* problem defined below:

> **Input:** A natural number, x, encoded using binary representation with *least significant bit first*.
>
> **Output:** A binary encoding, least significant bit first, of $x + 1$.

Characterize using asymptotic notation the *worst-case* running time cost for solving *INCREMENT*, where cost is the number of steps required by a standard Turing Machine. (You should be able to get a tight bound using Θ notation. Use n to represent the size of the input in your answer.)

Does this change if the input is encoded using binary representation with *most significant bit first*?