Problem Set 7: $\mathsf{P} \subsetneq \mathsf{NP}$?

Due: **7:29pm Tuesday, 26 November 2019**

This assignment focuses on the material in Classes 17–21 and textbook Chapter 11, Chapter 12, Chapter 13, and Chapter 14. This is a lot of material (although we only cover a few small parts of Chapter 11 and 12), so this problem set is longer than previous problem sets (and you have more time for it).

Collaboration Policy: You should do this assignment by yourself and submit your own answers. You may discuss the problems with anyone you want and it is also fine to get help from anyone on problems with LaTeX or debugging your Turing machine code. You are permitted to use any resources you find for this assignment. You should note in the *Collaborators and Resources* box below the people you collaborated with and any external resources you used.

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

This problem set does not include any Jupyter notebook components. As with previous problem sets, the first thing you should do in ps7.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 id, e.g., `\submitter{Stephen Cook (sac0np)}`. Before submitting your PDF, also remember to (1) list your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}`, and (2) replace the second line in ps7.tex, `\usepackage{uvatoc}` with `\usepackage[response]{uvatoc}` so the directions do not appear in your final PDF.

Problem 1 *Rice's Theorem* (Deferred from Problem Set 6)

For each subproblem, indicate whether Rice's Theorem applies. If it applies, answer if the problem is computable or uncomputable. If it does not apply, just indicated that it doesn't apply (it is not necessary to determine whether or not it is computable if Rice's theorem does not apply).

(a) Given the description of a Turing Machine, does that machine always return 0.

(b) Given the description of a Turing Machine, does that machine always return 1 when it receives no input.

(c) Given the description of a Turing Machine, does that machine ever use more than 10,000 cells on its tape.

(d) Given the description of a Turing Machine, is the language of that machine recognizable.

(e) Given the description of a Turing Machine, does the Turing machine have exactly 50 states.

Problem 2 *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 $x+1$.

(a) Characterize using asymptotic notation the *average* running time cost for solving *INCREMENT*, where cost is the number of steps required by a standard Turing Machine.

(b) 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.)

(c) Does your answer to either sub-question change if the input and output are represented using *most significant bit first*?

Problem 3 $TIME_{TM}$ *to increment*

Define the Inc function as $\forall x, y \in \mathbb{N} : Inc(x,y) = (y = x+1)$ where x and y are represented as binary numbers using most significant bit first, separated by a # symbol.

For example, $Inc(0\#1) = 1$, $Inc(1\#11) = 0$, $Inc(101111000100\#101111000101) = 1$.

We use the definition of $TIME_{TM}$ from Definition 12.1 in the TCS book, so the n in the equations below is from that definition.

1. Define a $f(n)$ such that $Inc \in TIME_{TM}(f(n))$ but $Inc \notin TIME_{TM}(\frac{1}{2}\sqrt{f(n)})$. Support your answer with a convincing argument.
2. ($\star\star$) Does there exist a function, $f(n)$, such that, for sufficiently large n, $Inc \in TIME_{TM}(f(n))$ but $Inc \notin TIME_{TM}(f(n)-1)$?

Problem 4 *TIME for RAM and TM*

Theorem 12.5 shows that $TIME_{TM}(T(n)) \subseteq TIME_{RAM}(10 \cdot T(n))$ for all functions $T : \mathbb{N} \to \mathbb{N}$ such that $\forall n : T(n) \geq n$ and an additional constraint on the ability of a TM to compute T in $O(T(n))$. Show that for *some* $T(n)$ that satisfies the necessary constraints, the stronger property that $TIME_{TM}(T(n)) \subsetneq TIME_{RAM}(T(n))$ is true. That is, show that there is a function that can be computed by a RAM machine within $T(n)$ steps, but cannot be computed by any TM within $T(n)$ steps.

Problem 5 *Polynomial-Time Reductions*

For each sub-problem, indicated if the state proposition is **True** or **False**, and provide a brief (one or two sentences) justification for your answer.

We use the notations in the book: let $F, G : \{0,1\}^* \to \{0,1\}^*$. We say that F reduces to G, denoted by $F \leq_p G$, if there is a polynomial-time computable $R : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $F(x) = G(R(x))$.

(a) $F \leq_p G$ and $G \in \mathbf{P}$ implies $F \in \mathbf{P}$.

(b) $F \leq_p G$ and $F \in \mathbf{P}$ implies $G \in \mathbf{P}$.

(c) $F \leq_p G$ and $G \in \mathbf{EXP}$ implies $F \in \mathbf{EXP}$.

(d) $F \leq_p G$ and $G \leq_p F$ implies $F \in \mathbf{P}$.

(e) $F \leq_p G$ and G is computable implies F is computable.

(f) $F \leq_p G$ and F is computable implies G is computable.

(g) $F \leq_p G$ and $G \in \mathit{TIME}_{TM}(O(n^2))$ implies $F \in \mathit{TIME}_{TM}(O(n^2))$.

(h) $F \leq_p F$ (Hint: use the definition of $\leq_p$ to show this is always True for any F.)

Problem 6 *Silly Reductions*

Consider the $\mathit{SORTING}$ and $\mathit{MINIMUM}$ 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 $\mathit{MINIMUM}$ and $\mathit{SORTING}$ are equivalently hard problems?

Problem 7 *Jeffersonian Paths*

The Hamlitonian 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.

Problem 8 *Genome Assembly*

In order to assemble a genome, it is necessary to combine snippets from many reads into a single sequence. The input is a set of n genome snippets, each of which is a string of up to k symbols. The output is the smallest single string that contains all of the input snippets as substrings. For example, if the input is ACCAGAATACC, TCCAGAATAA, TACCCGTGATCCA the output should be ACCAGAATACCCGTGATCCAGAATAA:

```
ACCAGAATACC
                TCCAGAATAA
       TACCCGTGATCCA
```

(a) Prove that the genome assembly problem is in NP.

(b) Prove that the genome assembly problem is NP-Complete.

(c) On June 26, 2000, President Clinton, Francis Collins (CLAS 1970), and Craig Venter announced completion of the Human Genome project and a complete sequence of the human genome. The human genome is about 3 Billion base pairs long. Readers at the time were able to read about 700 bases per read fragment, and sequencing the genome involved aligning about 30 million fragments. What problem had they actually solved?

Problem 9 *Simpson's Math*

Watch this video of Simon Singh talking about P and NP: https://www.youtube.com/watch?v=dJUEkjxylBw.

Clearly identify at least one clear and fundamental technical misunderstanding in his explanation.

Problem 10 *Turing Machines and Concrete Computers*

(a) Give three reasons why a Turing Machine is better than the computer you are typing on now.

(b) Give three reasons why the computer you are typing on now is better than a Turing Machine.

End of Problem Set 7

