CS120: Intro. to Algorithms and their Limitations Hesterberg & Vadhan

Problem Set 9

Harvard SEAS - Fall 2022 *Due: FRI. Dec. 2, 2022 (11:59pm)*

Your name:
Collaborators:
No. of late days used on previous psets:
No. of late days used after including this pset:

The purpose of this problem set is to practice proving that problems are unsolvable via reduction, and gain more intuition for the kinds of problems about programs that are unsolvable (through examples). Throughout this problem set, you may use some pseudocode in describing RAM and Word-RAM programs (like for loops), but be sure that the pseudocode can be implemented using actual RAM/Word-RAM commands that satisfy the constraints of the given problem (e.g. having no arithmetic overflows or being write-free).

1. ($\mathsf{P}_{\mathsf{search}} \not\subseteq \mathsf{NP}_{\mathsf{search}}$) Recall that in lecture, we commented that there are artificial problems in $\mathsf{P}_{\mathsf{search}}$ that are not in $\mathsf{NP}_{\mathsf{search}}$. Here you will see one. Specifically, consider the computational problem $\Pi = (\mathcal{I}, \mathcal{O}, f)$ given as follows:

$$\begin{aligned} \mathcal{I} &= \{P : P \text{ is a RAM program}\} \\ \mathcal{O} &= \{0,1\} \\ f(P) &= \begin{cases} \{0,1\} & \text{if } P \text{ halts on } \varepsilon \\ \{0\} & \text{otherwise} \end{cases} \end{aligned}$$

   Prove that Π is in $\mathsf{P}_{\mathsf{search}}$ and that Π is not in $\mathsf{NP}_{\mathsf{search}}$.

2. (Undecidability of arithmetic overflows) An *arithmetic overflow* in the execution of a Word-RAM program is when the result of an arithmetic operation (addition or multiplication) results in a value larger than 2^w, where w is the current word size, so the result has to be taken modulo 2^w. In Lecture 24, you will see how SMT Solvers are able to find a bug due to arithmetic overflow in Binary Search truncated to two levels of recursion. In this problem, you will see that finding arithmetic overflow errors in general programs is an unsolvable problem.

   (a) Give an algorithm that converts any RAM program P into an equivalent Word-RAM program P' that never has arithmetic overflow. That is, for all inputs (arrays of natural numbers) x, P' halts on x iff P halts on x, and if they do halt, then $P'(x) = P(x)$, and whenever P' carries out an operation $\texttt{var}_i = \texttt{var}_j \texttt{ op } \texttt{var}_k$, the result is always smaller than 2^w, where w is the current word size. Do not worry about the efficiency of your simulation (in contrast to Theorem 7.1.1 of Lecture 7, which does a fairly involved simulation in order to obtain the runtime as described; something much simpler suffices here).

   (b) Using Part 2a and the undecidability of HaltOnEmpty for RAM programs, prove that the following computational problem is unsolvable:

Input	**:** A Word-RAM program P
Output	**:** `yes` if P has an arithmetic overflow when run on input ε, `no` otherwise

Computational Problem ArithmeticOverflow

3. (optional:[1] Memory-Free RAM Programs)

   (a) A *memory-free RAM Program* $P = (V, C_0, \ldots, C_{\ell-1})$ has no commands to read or write to memory, has a fixed set of input variables $x_0, x_1, \ldots, x_{n-1}$, and a fixed set of output variables $y_0, \ldots, y_{m-1}$. It begins its computation with the input x placed in the variables $(x_0, \ldots, x_{n-1})$ and if it halts, its output $P(x)$ is defined to be the values in $(y_0, \ldots, y_{m-1})$. Using the unsolvability of Diophantine Equations (as stated without proof in Lecture 23), prove that the Halting Problem for Memory-Free RAM Programs is unsolvable.

   (b) A memory-free *Word*-RAM is defined similarly to a memory-free RAM, except it has a finite word size, which is initialized to be $w = \lceil \log_2 \max\{x_0, x_1, \ldots, x_{n-1}\} \rceil$ and never changes throughout the computation. Show that, in contrast to memory-free RAMs, the Halting Problem for memory-free *Word*-RAMs is solvable. That is, there is an algorithm that given a memory-free Word-RAM Program $P = (V, C_0, \ldots, C_{\ell-1})$ an input x, decides whether $P(x)$ ever halts. (Hint: The entire state of a computation of a memory-free (Word-)RAM program is determined by the current line number and the values of all variables. Use this to compute a threshold $T(P, x)$ such that if P runs for more than $T(P, x)$ steps, then P must be in an infinite loop.)

4. (optional challenge:[1] Artemis' Party)

   Adam is hosting a birthday party for his cat, Artemis. In a shocking turn of events, Artemis' cat-rival Simetra is also having a birthday party on the same day! Adam gathered some intel and found out that Simetra's party will have k guests. So as not to be shown up, Artemis is now feeling the pressure to have k guests as well. The only problem as that a lot of Artemis' friends are enemies with each other. To plan out his invite list, Artemis represents these relationships in a graph, where vertices are his cat-friends and an edge represents an enemy relationship. Given this graph and k, will Artemis be able to invite at least k non-hostile friends to his party, or will his party lose out to his nemesis Simetra?

Input	**:** A graph G of cats where edges are enemy relationships and k is the number of cats Artemis must invite (at least) in order to retain his honor.
Output	**:** `yes` if Artemis can successfully create an invite list of at least k invitees; `no` otherwise

Computational Problem ArtemisBirthdayParty

   Prove or disprove: ArtemisBirthdayParty is in $\mathsf{P}_{\mathsf{search}}$.

[1]This problem is meant to be done based on your enjoyment/interest and only if you have time. It won't make a difference between N, L, R-, and R grades, and course staff will deprioritize questions about this problem at office hours and on Ed.