CS1200: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Problem Set 10

Harvard SEAS - Fall 2025 *Due: Wed. Dec. 3, 2025 (11:59pm)*

Please see the syllabus for the full collaboration and generative AI policy, as well as information on grading, late days, and revisions.

All sources of ideas, including (but not restricted to) any collaborators, AI tools, people outside of the course, websites, ARC tutors, and textbooks other than Hesterberg–Vadhan must be listed on your submitted homework along with a brief description of how they influenced your work. You need not cite core course resources, which are lectures, the Hesterberg–Vadhan textbook, sections, SREs, problem sets and solutions sets from earlier in the semester. If you use any concepts, terminology, or problem-solving approaches not covered in the course material by that point in the semester, you must describe the source of that idea. If you credit an AI tool for a particular idea, then you should also provide a primary source that corroborates it. Github Copilot and similar tools should be turned off when working on programming assignments.

If you did not have any collaborators or external resources, please write 'none.' Please remember to select pages when you submit on Gradescope. A problem set on the border between two letter grades cannot be rounded up if pages are not selected.

Your name:
Collaborators and External Resources:
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. (recognizing solvability and unsolvability) Which of the following computational problems about programs are solvable? Justify your answers, for example by giving an algorithm to solve the problem or using a reduction to prove unsolvability.

   (a) **Input:** Word-RAM programs P and Q that both solve Graph 3-Coloring, and a graph G, and a word length w

   Output: `yes` if the running time of $P[w]$ on G is smaller than the running time of $Q[w]$ on G, `no` otherwise

   Computational Problem Which Algorithm Is Faster On This Input

(b)

Input: A *polynomial-time* Turing machine M

Output: `yes` if there is a *polynomial-time* Word-RAM program P that solves all of the same computational problems as M, `no` otherwise

Computational Problem TM vs. Word-RAM

(c)

Input: A RAM program P

Output: `yes` if on at least one of the inputs $x \in \{\texttt{Vladimir}, \texttt{Estragon}\}$, $P(x)$ eventually halts and outputs `Godot`, `no` otherwise

Computational Problem Waiting For Godot

2. (undecidability of arithmetic overflows) An *arithmetic overflow* in the execution of a Word-RAM program $P[w]$ is when the result of an arithmetic operation (addition or multiplication) results in a value at least 2^w, where w is the word size, so the result has to be capped at $2^w - 1$. In the 11/25 lecture, 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 x,

   - There exists a word length w such that $P'[w]$ halts on x without crashing if and only if P halts on x,
   - For all word lengths w such that $P'[w]$ halts without crashing on x, its output $P'[w](x)$ equals the output $P(x)$, and
   - For all word lengths w, whenever $P'[w](x)$ carries out an operation $\texttt{var}_i = \texttt{var}_j \ \texttt{op} \ \texttt{var}_k$, the result is always smaller than 2^w.

   Do not worry about the efficiency of your simulation (in contrast to Theorem 7.5 from the Hesterberg-Vadhan textbook, which does a fairly involved simulation in order to obtain the $O(T(x))$ runtime of the Word-RAM program; something much simpler suffices here). (Hint: Try to make $P'[w]$ crash or go into an infinite loop if an overflow would happen.)

   (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 there is a word length w such that $P[w]$ has an arithmetic overflow when run on input ε, `no` otherwise

   Computational Problem Arithmetic Overflow

   (c) Show that the ArithmeticOverflow problem is solvable if we fix the word length w:

   Input: A Word-RAM program P and a word length w

   Output: `yes` if $P[w]$ has an arithmetic overflow when run on input ε, `no` otherwise

Computational Problem Arithmetic Overflow On Fixed Word Length

(Hint: consider the *state* of the computation of $P[w]$ when run on input ε, namely the current size and contents of memory, the values of all the variables, and the current line number. Note that if the state is ever repeated prior to halting, then P will never halt.)

(d) Discuss why your algorithm from Item 2c is not practical for real-world programs with word length $w = 64$. This (and the following item) motivate the use of SMT Solvers to find bugs like arithmetic overflows.

(e) (optional[1]) Show that Arithmetic Overflow On Fixed Word Length is NP-hard. (In fact, it is much harder than NP-hard and is provably not in P, but this fact is beyond the scope of this course.)

Note that the input length here is $N = |P| + \log_2 w$, so really the complexity is doubly exponential in the bitlength of w. So this problem is actually complete for doubly-exponential space!

3. (reflection) How have the ideas of this course enlarged your sense of what it means to do computer science? Be specific, pointing to the particular course content that made this change for you.

   Quick note on grading: Good responses are usually about a paragraph, with something like 7 or 8 sentences. Most importantly, please make sure your answer is specific to this class and your experiences in it. If your answer could have been edited lightly to apply to another class at Harvard, points will be taken off.

4. Once you're done with this problem set, please fill out this survey so that we can gather students' thoughts on the problem set, and the class in general. It's not required, but we really appreciate all responses!

[1]This problem won't make a difference between N, L, R-, and R grades. As this problem is purely extra credit, course staff will deprioritize questions about this problem at office hours and on Ed.