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

Problem Set 9

Harvard SEAS - Fall 2025 *Due: Wed Nov. 19, 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 to reinforce the definitions and basic theory of our complexity classes and practice NP-completeness proofs and related reductions.

1. (Complexity Classes and Reductions) Consider the computational problem $\Pi = (\mathcal{I}, \mathcal{O}, f)$ where $\mathcal{I} = \mathcal{O} = \mathbb{N}$ and $f(x) = \mathbb{N}$ for all $x \in \mathbb{N}$. Show that $\Pi \in \mathsf{P}_{\mathsf{search}}$ but $\Pi \notin \mathsf{NP}_{\mathsf{search}}$. Thus, $\mathsf{P}_{\mathsf{search}} \not\subseteq \mathsf{NP}_{\mathsf{search}}$.

2. ($\mathsf{NP}_{\mathsf{search}}$-completeness) Consider the following variant of k-SAT.

Input: A k-CNF formula $\varphi(x_0, \ldots, x_{n-1})$

Output: An assignment $\alpha \in \{0,1\}^n$ such that $\varphi(\alpha) = 1$ and $\varphi(\neg\alpha) = 1$ (if one exists), where $\neg\alpha$ is the bitwise negation of α

Computational Problem Flip k-SAT

Your proofs for this problem should all include correctness (in both directions) and runtime, following the outline of proofs done in lecture and the textbook.

(a) Prove that Flip 4-SAT is $\mathsf{NP}_{\mathsf{search}}$-complete by reduction from 3-SAT. (Hint: Add one new variable y and replace each clause $(\ell_0 \vee \ell_1 \vee \ell_2)$ by $(\ell_0 \vee \ell_1 \vee \ell_2 \vee y)$.)

(b) Flip 4-SAT can be reduced to Flip 3-SAT by the same method we used to reduce SAT to 3-SAT, and thus Flip 3-SAT is also $\mathsf{NP}_{\mathsf{search}}$-complete. Using this fact, prove that Graph 3-Coloring is $\mathsf{NP}_{\mathsf{search}}$-complete. (Hint: use the same construction as we used in the reduction from 3-SAT to Independent Set, except add one extra vertex that's connected to all of the variable-gadget-vertices.)

(c) (optional[1]) Fill in the omitted details of the reduction from Flip 4-SAT to Flip 3-SAT, with its proof of correctness.

3. (Breaking Crypto if $\mathsf{P} = \mathsf{NP}$) Every time your browser or one of your apps makes a secure connection to a website or server, it uses cryptography to make sure that eavesdroppers to the network cannot decipher the information you are transmitting. Specifically, the following happens at the initiation of a connection between two parties, Alice and Bob.

   (a) Alice chooses a uniformly random secret key $SK \in \{0,1\}^n$.

   (b) Alice applies a polynomial-time algorithm `Gen` to it to generate a public key $PK = \texttt{Gen}(SK)$, and sends PK to Bob.

   (c) Now whenever Bob wants to send a message $m \in \{0,1\}^\ell$ to Alice, Bob generates a uniformly random $nonce \in \{0,1\}^n$ ($nonce$ means "number used only once") and runs a polynomial-time encryption algorithm `Enc` to generate a ciphertex $c = \texttt{Enc}(PK, m, nonce) \in \{0,1\}^{\ell+n}$.

   (d) When Alice receives a ciphertext c from Bob, she runs a polynomial-time decryption algorithm to obtain the message $m = \texttt{Dec}(SK, c)$.

   The correctness of the public-key encryption scheme says Alice will always decrypt correctly, regardless of the random choices: for every $SK, nonce \in \{0,1\}^n$ and $m \in \{0,1\}^\ell$, we have

$$\texttt{Dec}(SK, \texttt{Enc}(\texttt{Gen}(SK), m, nonce)) = m.$$

   As an example, in the probabilistic *Rabin* cryptosystem, SK consists of two random $n/2$-bit prime numbers p and q such that $p, q \equiv 3 \pmod 4$ and $PK = pq$. $nonce$ is interpreted a random number in $\{1, 2, \ldots, PK - 1\}$ that is not divisible by either p or q. Then for a 1-bit message $m \in \{0,1\}$,

$$\texttt{Enc}(PK, m, nonce) = (nonce^4 \bmod PK, \mathrm{lsb}(nonce^2 \bmod PK) \oplus m),$$

   where lsb denotes the least-significant bit and $\oplus$ denotes XOR. It turns out that given the prime factors p and q of PK, from $nonce^4 \bmod PK$ it is possible to efficiently compute $nonce^2 \bmod PK$,[2] and hence decrypt.

   For security, it is *conjectured* that recovering m from c and PK, or even learning any partial information about m, takes time exponential in n (even in the average case). In the case of the Rabin cryptosystem, this conjecture is known to be true if factoring numbers generated like PK takes time exponential in n (which also a conjecture).

[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.

[2] Specifically $nonce^2 = (nonce^4)^{((p-1)(q-1)+4)/8} \bmod PK$. Modular exponentiation can be computed in polynomial time (in the bitlength of the base, exponent, and modulus), by a "repeated square and multiply" algorithm that we have not covered in cs1200.

Prove that if $\mathsf{P} = \mathsf{NP}$, this conjecture is false. Specifically, there is a polynomial-time algorithm E (for "Eve," the common name cryptographers use for the eavesdropper) such that $E(PK, c) = m$ whenever PK and c are generated as in Steps (3a)–(3d) above. Your solution should apply to an arbitrary encryption scheme, not only the example Rabin scheme shown above. (Hint: define an appropriate $\mathsf{NP}_{\mathsf{search}}$ problem Π such that solving Π in polynomial time enables decrypting m. There are multiple ways to do this! When proving correctness of your solution, recall that an algorithm solving Π is only guaranteed to produce *some* valid output on each input.)

4. (reflection) Describe one theoretical idea from this course that you have found beautiful, and explain why it is beautiful to you. Your answer should: (1) explain the idea in a way that could be understood by a classmate who has taken classes CS20 and CS50 but has not yet taken this class and (2) address how this beauty is similar to or different from other kinds of beauty that human beings encounter.

   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.

5. 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!