EECS 376 Final **SOLUTIONS**

The multiple-choice portion of the exam consists of the 8 questions in the "Multiple Choice" section below. The written portion of the exam consists of the 4 questions in the "Written Answer" section below. These two portions are released as a single Canvas quiz. You are to submit the answers to the multiple-choice portion on Canvas. Submit your written solutions to Gradescope as you would for a homework assignment. **You will not submit anything on Canvas for the written part.**

Both portions are to be submitted before 9pm Eastern Time, but the written part will have a 15-minute grace period until 9:15pm.

Logistics:

- The exam will take place on Wednesday, December 14, 7pm - 9pm Eastern Time.
- You must **submit your multiple-choice answers on Canvas prior to 9pm Eastern Time**. There is no grace period for the multiple-choice questions.
- The deadline to **submit your written answers to Gradescope is 9pm Eastern Time. There will be a grace period until 9:15pm.** However, you must start the process of preparing your submission by 8:45pm.
- You may use any **course** resources for the exam, including the textbooks, lecture slides, online notes, discussion materials, etc.
  You may **not** use any **non-course** resources, such as search engines (e.g. Google) or calculators (e.g. a physical calculator, WolframAlpha, etc.).
- You are prohibited from searching for answers to any of the exam questions online.
- You are prohibited from soliciting help from anyone, whether in person, over text/chat, on StackOverflow, making public Piazza posts, or any other means.
- Your solutions must be entirely your own work.
- If you have clarification questions, make a private post on Piazza, and a staff member will respond as soon as possible.
- If you run into technical issues or have an emergency, contact the staff (eecs376f22@umich.edu) right away. Do **not** contact a fellow student for help.
- Each multiple-choice question has only a single correct answer.

Any deviation from these rules will constitute an Honor Code violation. In addition, the staff reserves the right **not** to grade any exam taken in violation of this policy.

Attest to the following honor pledge by signing your name below.

Honor pledge:

I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.
I will not discuss the exam with anyone who has not already taken it.
I am taking the exam at the time I was assigned by the staff.

Signature: _____

Recall the following languages (unless otherwise specified, all graphs are simple and undirected):

- $L_{\text{ACC}} = \{\langle M, x\rangle : M \text{ is a Turing machine that accepts the input } x\}$
- $L_{\text{HALT}} = \{\langle M, x\rangle : M \text{ is a Turing machine that halts on input } x\}$
- $\overline{L} = \{x \in \Sigma^* : x \notin L\}$
- $\text{SAT} = \{\phi : \phi \text{ is a satisfiable Boolean formula}\}$
- $\text{3SAT} = \{\phi : \phi \text{ is a satisfiable 3CNF formula}\}$
- $\textsc{Clique} = \{(G, k) : G \text{ is a graph with a clique of size (at least) } k\}$
- $\textsc{Vertex-Cover} = \{(G, k) : G \text{ is a graph with a vertex cover of size (at most) } k\}$
- $\textsc{Independent-Set} = \{(G, k) : G \text{ is a graph with an independent set of size (at least) } k\}$
- $\textsc{Hamiltonian-Cycle} = \{G : G \text{ is a graph with a Hamiltonian cycle}\}$
- $\textsc{Hamiltonian-Path} = \{(G, s, t) : G \text{ is a graph with a Hamiltonian path from } s \text{ to } t\}$
- $\text{TSP} = \{(G, k) : G \text{ is a weighted graph with a tour of weight at most } k\}$
- $\textsc{Max-Cut} = \{(G, k) : G \text{ is a graph with a cut of size at least } k\}$

Multiple Choice (5 points each)

1. Suppose that a language B is NP-Complete, and we demonstrate that $B \leq_p A$ for some language A. Which of the following can we **always** conclude about the language A?

   - ● **A is NP-Hard**
   - ○ A is NP-Complete
   - ○ A is in NP
   - ○ None of the other choices are valid facts we can conclude.

   Solution: Since B is NP-Complete, it is NP-Hard, and by transitivity, A is also NP-Hard. We cannot conclude that A is in NP – for instance, 3SAT $\leq_p L_{\text{HALT}}$, but L_{HALT} is not in NP as all NP languages are decidable in exponential time. Thus, we also cannot conclude that A is NP-Complete.

2. Suppose that a language A is in P and a language B is NP-Complete. Then it must be the case that $A \leq_p B$.

   - ● **True**
   - ○ False
   - ○ Unknown

   Solution: True. By definition, all languages in NP are polytime reducible to an NP-Complete language. Since P $\subseteq$ NP, we have $A \in$ NP, so it can by polytime reduced to B.

3. Suppose a language L is **not** in the class P. Then L must be undecidable.

   - ○ True
   - ● **False**
   - ○ Unknown

   Solution: False. If P $\neq$ NP, then NP-Complete problems such as SAT would not be in P. However, we know that all languages in NP can be decided in exponential time, and thus SAT is decidable.

   The statement is also false if P = NP, as there are decidable languages known to not be in P. We saw such a language on the homework:

   $$\textsc{BoundedHalt} = \{(\langle M \rangle, x, k) : k \text{ is a binary number and } M \text{ halts on } x \text{ within } k \text{ steps}\}$$

4. Which of the following is an **invalid** public key (n, e) for the RSA encryption scheme?

   - ● **(55, 40)**
   - ○ (65, 13)
   - ○ (77, 13)
   - ○ (85, 27)
   - ○ All of the other choices are valid public keys.

Solution: $(55, 40)$ is invalid – since $55 = 5 \cdot 11$, $(p-1)(q-1) = 40$, and 40 does not have an inverse mod 40.

The other choices are valid. For $n = 65 = 5 \cdot 13$, $(p-1)(q-1) = 48$, and 13 is coprime to 48. For $n = 77 = 7 \cdot 11$, $(p-1)(q-1) = 60$, and 13 is coprime to 60. For $n = 85 = 5 \cdot 17$, $(p-1)(q-1) = 64$, and 27 is coprime to 64.

5. Suppose that we have $(n, e) = (51, 11)$ as an RSA public key, and we wish to sign the message $m = 5$ using the RSA **signature** scheme. What is the correct value for the signed message s?

- ◯ 11
- ◯ 13
- ◯ 14
- ● **23**
- ◯ 49

Solution: Since $n = 51 = 3 \cdot 17$, $(p-1)(q-1) = 32$, we need to compute the inverse of 11 mod 32. We can do so either via the extended Euclidean algorithm or by inspection to get $d \equiv e^{-1} \equiv 3 \pmod{32}$. To sign a message m, we compute $s \equiv m^d \equiv 5^3 \equiv 125 \equiv 23 \pmod{51}$. We verify this is correct:

$$
\begin{aligned}
s^e \equiv 23^{11} &\equiv 23^1 \cdot 23^2 \cdot 23^8 \pmod{51} \\
23^2 &\equiv 529 \equiv 19 \pmod{51} \\
23^4 \equiv (23^2)^2 &\equiv 19^2 \equiv 361 \equiv 4 \pmod{51} \\
23^8 \equiv (23^4)^2 &\equiv 4^2 \equiv 16 \pmod{51} \\
23^{11} &\equiv 23 \cdot 19 \cdot 16 \equiv 23 \cdot 304 \equiv 23 \cdot -2 \equiv -46 \equiv 5 \pmod{51}
\end{aligned}
$$

6. Suppose that X is a non-negative random variable with an expectation equal to 1. Which of the following is **always** a correct upper bound on $\Pr[X \geq 3/4]$?

- ◯ 0.001
- ◯ $e^{-3/4}$
- ◯ 3/4
- ● **1**

Solution: The only bound we can apply is Markov's inequality, as we don't know $\text{Var}(X)$, nor do we know that X is the sum of independent random variables. We wish to compute an upper bound on $\Pr[X \geq 3/4 \cdot \mathbb{E}[X]]$. By Markov's inequality, we have

$$\Pr\left[X \geq \frac{3}{4}\mathbb{E}[X]\right] \leq \frac{\mathbb{E}[X]}{3/4 \cdot \mathbb{E}[X]} = 4/3$$

However, an upper bound of 1 is also valid, since a probability cannot be more than 1. Thus, the correct answer is 1.

In fact, no tighter bound than 1 is possible for an arbitrary non-negative RV X. It may be that $\Pr[X = 1] = 1$ (i.e. X only takes on a single value), in which case $\Pr[X \geq 3/4] = 1$.

7. Assuming P $\neq$ NP, the CLIQUE language is **not** NP-Complete.

- ◯ True
- ● **False**
- ◯ Unknown

Solution: False. A language L is NP-Complete if every language in NP is polytime reducible to L. This is not conditional on whether or not P = NP. Since we showed that SAT $\leq_p$ 3SAT $\leq_p$ CLIQUE and that all languages in NP are polytime reducible to SAT, we demonstrated that they are all polytime reducible to CLIQUE, and therefore CLIQUE is NP-Complete.

8. Define #SAT to be the problem of determining how many satisfying assignments a Boolean formula ϕ has. Define the decision version as follows:

$$L_{\#\text{SAT}} = \{(\phi, k) : \phi \text{ is a Boolean formula with} \geq k \text{ satisfying assignments}\}$$

Suppose we have an efficient decider D for the decision problem $L_{\#\text{SAT}}$. Then the following algorithm would *efficiently* solve the #SAT problem:

$A =$ "On input ϕ:
1. $k \leftarrow 0$
2. While $D(\phi, k+1)$:
3. $\quad k \leftarrow k+1$
4. Return k"

- ◯ True
- ● **False**
- ◯ Unknown

Solution: False. A Boolean formula over n variables can have up to 2^n satisfying assignments. If each variable only appears a constant number of times in ϕ, then ϕ would have length $O(n)$. However, $A(\phi)$ can do 2^n iterations of the loop, for a runtime of $\Omega(2^n)$. This is not polynomial with respect to the input size $O(n)$.

A specific example of a worst-case formula is the following:

$$(x_1 \vee \neg x_1) \wedge (x_2 \vee \neg x_2) \wedge \cdots \wedge (x_n \vee \neg x_n)$$

This formula has size $2n$ and 2^n satisfying assignments.

Written Answer (15 points each)

9. A blood test for a rare disorder is being performed on n people. Each person can be tested individually, but this is expensive. To decrease the cost, the n people are divided into n/k groups, $G_1, G_2, \ldots, G_{n/k}$, of k people each (assume that k divides n); the blood samples of the people in a group are then pooled and tested together, so that only n/k tests are used initially. If the test for group G_i is negative, none of the members of G_i have the disorder, so they don't need to be tested individually. On the other hand, if the test for group G_i is positive, at least one person in the group has the disorder, and the entire group must be retested individually.

   Suppose each person independently has a probability p of having the disorder. Compute each of the following quantities as a simplified expression in terms of n, k, and p. Justify your answers.

   (a) The probability that a specific group G_i needs individual testing for its members.

   (b) The expected number of groups that need individual testing.

   (c) The expected number of total tests required over all groups, including both group and individual tests.

Solution:

(a) A group G_i does not need individual testing when all its members are free of the disorder. Since the probability of each person not having the disorder is independent, the probability that all k of them are free of the disorder is $(1-p)^k$. Then the probability that at least one person has the disorder is $1-(1-p)^k$.

(b) Let X_i be an indicator for whether or not group G_i needs individual testing. Then $\mathbb{E}[X_i] = 1-(1-p)^k$. Let $X = X_1 + \cdots + X_{n/k}$ be the number of groups that need individual testing. By linearity of expectation, $\mathbb{E}[X] = \frac{n}{k}(1-(1-p)^k)$.

(c) No matter what, n/k group tests are required. Then for each group that needs individual testing, k additional tests are required. Thus the total number of tests is $Y = n/k + kX$, where X is defined as above. By linearity of expectation,

$$\begin{aligned}\mathbb{E}[Y] &= \frac{n}{k} + k \cdot \mathbb{E}[X] \\ &= \frac{n}{k} + k \cdot \frac{n}{k}(1-(1-p)^k) \\ &= \frac{n}{k}(1 + k - k(1-p)^k)\end{aligned}$$

Equivalent expressions:

$$\begin{aligned}\mathbb{E}[Y] &= \frac{n}{k}(1 + k - k(1-p)^k) \\ \mathbb{E}[Y] &= \frac{n}{k} + n(1-(1-p)^k) \\ \mathbb{E}[Y] &= \frac{n}{k} + n - n(1-p)^k\end{aligned}$$

Alternate Solution:
Let Y_i be the number of tests required for group G_i. If no one in G_i has the disorder, $Y_i = 1$,

otherwise $Y_i = k+1$ (the group test plus k individual tests). By the definition of expectation, we have

$$\begin{aligned}\mathbb{E}[Y_i] &= 1 \cdot \Pr[Y_i = 1] + (k+1) \cdot \Pr[Y_i = k+1] \\ &= 1 \cdot (1-p)^k + (k+1) \cdot (1-(1-p)^k) \\ &= (1-p)^k + (k - k(1-p)^k + 1 - (1-p)^k) \\ &= k + 1 - k(1-p)^k\end{aligned}$$

Let $Y = Y_1 + \cdots + Y_{n/k}$ be the total number of tests. By linearity of expectation,

$$\mathbb{E}[Y] = \frac{n}{k}(k + 1 - k(1-p)^k)$$

10. We define the following language:

$$\textsc{Big-Clique} = \{G = (V, E) : G \text{ is an undirected graph with a clique of size } |V| - 1\}$$

Prove that Big-Clique is in P.

Solution: There are $\binom{|V|}{|V|-1} = |V|$ subsets of V of size $|V|-1$. A decider just needs to check each of these subsets to see if any are a clique. The following does so:

D = "On input $G = (V, E)$:
1. For $i = 1$ to $|V|$:
2. For $j = 1$ to $|V|$:
3. For $k = j + 1$ to $|V|$:
4. If $i \neq j$ and $i \neq k$ and $(v_j, v_k) \notin E$:
5. Continue to the next iteration of the loop on line 1
6. **Accept**
7. **Reject**"

Correctness Analysis:

- If $G = (V, E) \in$ Big-Clique, then there is some subset of $|V| - 1$ vertices in G that form a clique C, and only a single vertex $v_m \in V \setminus C$. When the loop on line reaches $i = m$, it checks in lines 2-4 whether the rest of the vertices all have edges between them. Since that is indeed the case, execution will reach line 6, and the decider accepts.
- If $G = (V, E) \notin$ Big-Clique, then no subset of $|V| - 1$ vertices is a clique. Any such subset has a missing edge between two vertices v_a and v_b, for $a < b$. When $j = a$ and $k = b$, the decider will see in line 4 that there is no edge (v_a, v_b) in E and therefore will move on to the next subset. Once all subsets of size $|V| - 1$ are checked, execution will reach line 7 and the decider rejects.

Runtime Analysis:
The code has a triply nested loop, each with at most $|V|$ iterations, for a total of at most $|V|^3$ iterations. We can check whether or not an edge exists efficiently (e.g. in constant time for an adjacency-matrix representation, or linear time for an adjacency list). Since the number of iterations is polynomial in $|G|$ and each iteration takes polynomial time, the total time is polynomial in $|G|$.

Since D efficiently decides Big-Clique, we conclude that Big-Clique $\in$ P.

11. You are planning to drive from Michigan to Southern California for the CFP National Championship game, and there are a number of sights you'd like to see along the way (e.g. Yellowstone, The Grand Canyon, Death Valley). You'd like to see as many of these places as possible, without having to backtrack (traverse a cycle) during your trip.

    Formally, we define the following language:

$$\textsc{Road-Trip} = \left\{ (G = (V, E), s, t, S, k) : \begin{array}{r} G \text{ has a simple path from } s \text{ to } t \text{ that} \\ \text{visits at least } k \text{ of the vertices in } S \end{array} \right\}$$

    (A *simple path* is a path without a cycle.)

    Given an efficient decider D for Road-Trip, design and analyze (both correctness and runtime) an efficient algorithm that given $(G = (V, E), s, t, S)$, finds an actual simple path (i.e. a path without a cycle) from s to t that goes through as many vertices of S as possible. The path should be returned as a set of edges (the edges do **not** have to be ordered). You may assume that $\{s, t\} \subseteq V$ and $S \subseteq V$.

Solution: We follow the standard process of first finding the size of an optimal object, then finding the object itself. On input (G, s, t, S), we first run the decider on (G, s, t, S, k) for each k between 0 and $|S|$ to find the largest k for which the decider accepts.

Second, we determine which edges are essential for the maximal path. For each edge e, we use the decider to determine whether removing e from the graph causes the graph to no longer have a path that goes through the maximal number of vertices of S – if so, the edge is part of that path. Otherwise, it is not, and we can safely remove e and move on to the next edge.

The full search algorithm is as follows:

$\mathcal{A}$ = "On input $(G = (V, E), s, t, S)$:

1. $k \leftarrow 0$
2. While $D(G, s, t, S, k + 1)$ accepts:
3. $\quad k \leftarrow k + 1$
4. For each edge $e \in E$:
5. $\quad E' \leftarrow E \setminus \{e\}$
6. $\quad$ If $D(G' = (V, E'), s, t, S, k)$ accepts:
7. $\quad\quad E \leftarrow E'$
8. Return E"

Correctness Analysis:
The algorithm tries every possible value of k to find the maximal value for which the decider accepts. Thus, we know that after the loop on line 3 completes, k will have the maximal valid value.

The algorithm then proceeds to find the actual set of edges that comprise a maximal path. For each edge, it queries the decider to determine whether removing it permits a solution of size k. After considering every edge in the graph, we will have retained only the edges e that must be in the maximal path. The algorithm returns the set of these edges.

Runtime Analysis:
The loop on line 2 does at most $|V|$ iterations, since there are at most $|V|$ vertices of S that a simple path can visit. Since all it does is invoke D, which is assumed to be efficient, the loop as a whole is efficient.

The loop on line 4 does at most $|E|$ iterations. Each iteration removes an edge, which can be done efficiently, and invokes the efficient decider D on inputs that are (slightly) smaller than the inputs to $\mathcal{A}$. Thus, each iteration is efficient, and since there are only a polynomial number, the loop as a whole is efficient.

Thus, the algorithm as a whole is efficient.

12. *Note: The language here is the same as the one defined in the previous question.*

You are planning to drive from Michigan to Southern California for the CFP National Championship game, and there are a number of sights you'd like to see along the way (e.g. Yellowstone, The Grand Canyon, Death Valley). You'd like to see as many of these places as possible, without having to backtrack (traverse a cycle) during your trip.

Formally, we define the following language:

$$\textsc{Road-Trip} = \left\{ (G = (V, E), s, t, S, k) : \begin{array}{r} G \text{ has a simple path from } s \text{ to } t \text{ that} \\ \text{visits at least } k \text{ of the vertices in } S \end{array} \right\}$$

(A *simple path* is a path without a cycle.)

(a) Provide an efficient verifier for Road-Trip. You do **not** need to provide analysis for this part.

(b) Prove that Road-Trip is NP-Hard.

Solution:

(a) The following is an efficient verifier V for Road-Trip.

$V =$ "On input $((G = (V, E), s, t, S, k), (v_1, \ldots, v_m))$:
1. If $v_1 \neq s$ or $v_m \neq t$ or $\{v_1, \ldots, v_m\} \not\subseteq V$ or $k \notin [0, |V|]$, **reject**
2. If there are any duplicates in $(v_1, \ldots, v_m)$, **reject**
3. $count \leftarrow 0$
4. For $i = 1$ to $m - 1$:
5. If $(v_i, v_{i+1}) \notin E$, **reject**
6. If $v_i \in S$, $count \leftarrow count + 1$
7. If $v_m \in S$, $count \leftarrow count + 1$
8. If $count \geq k$, **accept**
9. Else **reject**"

Analysis (optional):
We show that V is a valid polynomial verifier for Road-Trip.

V runs in polynomial time. Step 1 takes constant time for the first two checks and linear time for the fourth check. The third check can be done in at most quadratic time. Step 2, checking that there are no duplicates, can be done efficiently by inserting the items $v_1, \ldots, v_m$ into a set and checking that the size of the resulting set is m. The loop in step 3 does at most $|V| - 1$ iterations, each of which checks one edge and one set membership of S, so it is efficient. The remaining work (arithmetic, checking membership in S on line 7) can be done efficiently.

($\Rightarrow$) Suppose $(G, s, t, S, k) \in$ Road-Trip. Then, by definition, there exists a simple path $(v_1, \ldots, v_m)$ such that $v_1 = s$, and $v_m = t$, and $S \cap \{v_1, \ldots, v_m\} \geq k$. Then, using this path as our certificate, $V(\langle G, s, t, S, k\rangle, (v_1, \ldots, v_m))$ accepts – the verifier checks that the path is valid and that the number of vertices in S that are encountered is at least k, both of which are true by definition.

($\Leftarrow$) Suppose $(G, s, t, S, k) \notin$ Road-Trip. Then, by definition, there does not exist a path $(v_1, \ldots, v_m)$ such that $v_1 = s$, and $v_m = t$, and $S \cap \{v_1, \ldots, v_m\} \geq k$. Thus, any certificate passed to the verifier either won't start at s and end at t, or will not go through k of the vertices in S, or will not be a simple path, or will utilize edges not in the graph. V rejects certificates that don't start at s and end at t in the first step, certificates that are not simple

paths in the second step, certificates that utilize missing edges in step 5, and certificates that do not go through k vertices of S in step 8. Thus, V will reject all certificates.

V is a valid, efficient verifier for ROAD-TRIP, so ROAD-TRIP $\in$ NP.

(b) Next, we show ROAD-TRIP is NP-Hard. To do this, we will show HAMILTONIAN-PATH $\leq_p$ ROAD-TRIP. Recall that

$$\text{HAMILTONIAN-PATH} = \{(G, s, t) : G \text{ is a graph with a Hamiltonian path from } s \text{ to } t\}$$

and that HAMILTONIAN-PATH is NP-Complete. We construct the following mapping function.

f = "On input $(G = (V, E), s, t)$:
1. Return $(G, s, t, V, |V|)$."

We now show f is a valid polynomial mapping function from HAMILTONIAN-PATH to ROAD-TRIP.

f runs in polynomial time, as it just copies the input to the output, with the addition of $|V|$ which can be computed efficiently.

($\Rightarrow$) Suppose $(G = (V, E), s, t) \in$ HAMILTONIAN-PATH. Then by definition, there is a simple path $(s, v_2, \ldots, v_{n-1}, t)$ that goes through all $|V|$ vertices in G. Thus, G has a simple path from s to t that goes through $|V|$ vertices of V, so $(G, s, t, V, |V|) \in$ ROAD-TRIP.

($\Leftarrow$) Suppose $(G = (V, E), s, t, V, |V|) \in$ ROAD-TRIP. Then by definition, there is a simple path $(s, v_2, \ldots, v_{n-1}, t)$ that goes through $|V|$ vertices of V. This path goes through all of the vertices of G, so it is a Hamiltonian path from s to t. Therefore $(G, s, t) \in$ HAMILTONIAN-PATH.

f is a valid, efficient mapping function, so ROAD-TRIP is NP-Hard