Write your answers in the separate answer booklet.

You have 180 minutes (after you get the answer booklet) to answer seven questions.

1. Recall that a ***run*** in a string $w \in \{0,1\}^*$ is a maximal substring of w whose characters are all equal. For example, the string `00011111110000` is the concatenation of three runs:

$$00011111110000 = 000 \bullet 1111111 \bullet 0000$$

   (a) Let L_a denote the set of all non-empty strings in $\{0,1\}^*$ where the length of the first run is equal to the number of runs. For example, L_a contains the strings `0` and `1100000` and `0001110`, but does not contain `000111` or `100011` or the empty string ε (because it has no first run).

   Prove that L_a is not a regular language.

   (b) Let L_b denote the set of all strings in $\{0,1\}^*$ that contain an even number of odd-length runs. For example, L_b contains the strings `010111` and `1111` and the empty string ε, but does not contain either `0011100` or `11110`.

   - Describe a DFA or NFA that accepts L_b ***and***
   - Give a regular expression that describes L_b.

   (You do not need to prove that your answers are correct.)

2. Aladdin and Badroulbadour are playing a cooperative game. Each player has an array of positive integers, arranged in a row of squares from left to right. Each player has a token, which starts at the leftmost square of their row; their goal is to move *both* tokens onto the rightmost squares at the same time.

   On each turn, *both* players move their tokens *in the same direction*, either left or right. The distance each token travels is equal to the number under that token at the beginning of the turn. For example, if a token starts on a square labeled 5, then it moves either five squares to the right or five squares to the left. If *either* token moves past either end of its row, then both players immediately lose.

   For example, if Aladdin and Badroulbadour are given the arrays

A:	7	5	4	1	2	3	3	2	3	1	4	2
B:	5	1	2	4	7	3	5	2	4	6	3	1

   they can win the game by moving right, left, left, right, right, left, right. On the other hand, if they are given the arrays

A:	2	3	5	1	3
B:	3	4	1	2	1

   they cannot win the game. (The first move must be to the right; then Aladdin's token moves out of bounds on the second turn.)

   Describe and analyze an algorithm to determine whether Aladdin and Badroulbadour can solve their puzzle, given the input arrays $A[1..n]$ and $B[1..n]$.

Problems 3–7 appear on the following pages.

3. Submit a solution to ***exactly one*** of the following problems. Don't forget to tell us which problem you've chosen!

   (a) Let $G = (V, E)$ be an arbitrary undirected graph. A subset $S \subseteq V$ of vertices is *mostly independent* if more than half the vertices of S have no neighbors in S.

   Prove that the following problem is NP-*complete*: Given an undirected graph G and an integer k, decide if G contains a mostly independent set of size at least k.

   (b) ***Prove*** that the following problem is NP-*complete*: Given an undirected graph G and an integer k, decide if G contains two disjoint independent sets of size k.

   (In fact, both of these problems are NP-complete, but we only want a proof for one of them.)

4. Recall that a *palindrome* is any string that is equal to its reversal, like REDIVIDER or POOP.

   (a) Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a palindrome.

   (b) A *double palindrome* is the concatenation of two *non-empty* palindromes, like REFEREE = REFER • EE or POOPREDIVIDER = POOP • REDIVIDER. Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a *double* palindrome. *[Hint: Use your algorithm from part (a).]*

   For both algorithms, the input is an array $A[1..n]$, and the output is an integer. For example, given the input string MAYBEDYNAMICPROGRAMMING, your algorithm for part (a) should return 7 (for the subsequences NMRORMN and MAYBYAM, among others), and your algorithm for part (b) should return 12 (for the subsequence MAYBYAMIRORI).

5. Suppose we are given n intervals $I = \{[a_1, b_1], \ldots, [a_n, b_n]\}$ (*not* necessarily sorted) which are pairwise disjoint (i.e., non-overlapping) along with a positive number L. Our goal is to find a maximum size set $X = \{x_1, \ldots, x_k\}$ of numbers such that each member of X belongs to an interval of I, and each pair of numbers in X at *at least* L apart, i.e.,

$$\min\left\{|x_j - x_i| \;\middle|\; 1 \le i < j \le k\right\} \ge L.$$

   For example, given the intervals $I = \{[1, 4], [3, 8], [11, 12], [13, 17], [19, 23]\}$ and $L = 7$, one valid answer is $X = \{1, 8, 15, 23\}$ which has size $k = 4$. Note that in general, X is allowed to contain multiple numbers from the same interval.

   (a) ***Prove*** the following greedy strategy leads to a correct algorithm for the above problem:

   > Let ℓ be the leftmost number contained in any input interval. Add ℓ to X and compute the remainder of X by recursively working with the intervals I' made by removing all values ~~within~~ distance L of ℓ in each interval of I.

   [Hint: Let X^* be an optimal solution to the problem. Do an exchange argument to show there is an equally large solution X that contains ℓ.]

(b) Describe an algorithm to find a maximum size set of numbers X as described above. An algorithm based on the strategy in part (a) requires no further justification to receive full credit. describe & analyze. no further proof

6. For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No". For example:

- $x + y = 5$

  [ ] Yes [X] No — Suppose $x = 3$ and $y = 4$.

- 3SAT can be solved in polynomial time.

  [ ] Yes [X] No — 3SAT is NP-hard.

- If P = NP, then Emily can beat Super Mario Brothers in four minutes and 53 seconds.

  [X] Yes [ ] No — The hypothesis is false, so the implication is true.

Read each statement *very* carefully; some of these are deliberately subtle!

(a) Which of the following statements are true?

- The solution to the recurrence $T(n) = 4T(n/4) + O(n)$ is $T(n) = O(n \log n)$.
- The solution to the recurrence $T(n) = 4T(n/4) + O(n^2)$ is $T(n) = O(n^2 \log n)$.
- Every directed acyclic graph contains at most one source and at most one sink.
- Depth-first search explores every path from the source vertex s to every other vertex in the input graph.
- Suppose $A[1..n]$ is an array of integers. Consider the following recursive function:

$$Huh(i,j) = \begin{cases} 0 & \text{if } i < 0 \text{ or } j > n \\ \max \begin{Bmatrix} Huh(i, j+1) \\ Huh(i-1, j) \\ A[i] \cdot A[j] + Huh(i-1, j+1) \end{Bmatrix} & \text{otherwise} \end{cases}$$

We can compute $Huh(n, 0)$ by memoizing this function into an array $Huh[0..n, 0..n]$ in $O(n^2)$ time, increasing i in the outer loop and increasing j in the inner loop.

(b) Suppose we want to prove that the following language is undecidable.

$$\textsc{Duck} := \{\langle M \rangle \mid M \text{ accepts } \texttt{GRAPES} \text{ but rejects } \texttt{LEMONADE}\}$$

Professor Canard, your wetlands-ornithology instructor, suggests a reduction from the standard halting language

$$\textsc{Halt} := \{(\langle M \rangle, w) \mid M \text{ halts on input } w\}.$$

Specifically, suppose there is a Turing machine LooksLikeADuck that decides Duck. Professor Canard claims that the following algorithm decides Halt.

```
DecideHalt(⟨M⟩, w):
  Write code for the following algorithm:
    WaddleAway(x):
      run M on input w
      ⟨⟨ignore the output of M⟩⟩
      if x = LEMONADE
          return False
      else
          return True
  return LooksLikeADuck(⟨WaddleAway⟩)
```

Which of the following statements ***must be*** true ***for all*** inputs $(\langle M \rangle, w)$?

- If M accepts w, then WaddleAway accepts GRAPES.
- If M diverges on w, then WaddleAway rejects GRAPES.
- If M accepts w, then LooksLikeADuck accepts $\langle$WaddleAway$\rangle$.
- If M diverges on w, then DecideHalt rejects $(\langle M \rangle, w)$.
- DecideHalt decides the language Halt. (That is, Professor Canard's reduction is correct.)

7. **More of the same:** For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.**

   (a) Which of the following statements are true for ***all*** languages $L \subseteq \{0,1\}^*$?

   - $L^* = (L^*)^*$
   - If L is decidable, then L^* is decidable.
   - L is either regular or NP-hard.
   - If L is undecidable, then L has an infinite fooling set.
   - The language $\{\langle M \rangle \mid M \text{ decides } L\}$ is undecidable.

   (b) Suppose there is a ***polynomial-time*** many-one reduction from some language $A \subseteq \{0,1\}$ to some other language $B \subseteq \{0,1\}$. Which of the following statements are true, assuming P $\neq$ NP?

   - $A \cap B \neq \emptyset$.
   - There is an algorithm to transform any Python program that solves B in polynomial time into a Python program that solves A in polynomial time.
   - If B is NP-hard, then A is NP-hard.
   - If B is decidable, then A is decidable.
   - If a Turing machine M accepts every string in B, the *same* Turing machine M also accepts every string in A.

CS/ECE 374 A ✧ Spring 2026

Practice Final Exam 2

May 6, 2026

Name:	Emily Fox
NetID:	ekfox

- ***Don't panic!***
- You have 180 minutes to answer seven questions. The questions are described in more detail in a separate handout.
- If you brought anything except your writing implements, your two **hand-written** double-sided 8½" × 11" cheat sheets, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- Please clearly print your name and your NetID in the boxes above.
- Please also print your name at the top of every page of the answer booklet, except this cover page. We want to make sure that if a staple falls out, we can reassemble your answer booklet. (It doesn't happen often, but it does happen.)
- Greedy algorithms require formal proofs of correctness to receive any credit, even if they are correct. Otherwise, proofs or other justifications beyond items listed in the standard rubrics are required for full credit if and only if we explicitly ask for them, using the word ***prove*** or ***justify*** in bold italics. (*Problem 5 part (b)* contains a specific exception to this instruction.)

- **Please do not write outside the black boxes on each page.** These indicate the area of the page that our scanners can actually scan. If the scanner can't see your work, we can't grade it.
- If you run out of space for an answer, please use the overflow/scratch pages at the back of the answer booklet, but **please clearly indicate where we should look**. If we can't find your work, we can't grade it.
- **Only work that is written into the stapled answer booklet will be graded**. In particular, you are welcome to detach scratch pages from the answer booklet, but any work on those detached pages will not be graded. We will provide additional scratch paper on request, but any work on that scratch paper will not be graded.
- Please return ***all*** paper with your answer booklet: your question sheet, your cheat sheet, and all scratch paper. **Please put all loose paper *inside* your answer booklet.**
- Breathe in. Breathe out. You've got this.

This sentence contains four and three fourths percent a's, five and one half percent c's, three percent d's, eighteen and one fourth percent e's, four and one half percent f's, three fourths percent g's, five percent h's, two and one half percent i's, two percent l's, twelve and one half percent n's, six percent o's, five percent p's, eight percent r's, seven percent s's, nine and three fourths percent t's, two and one fourth percent u's, one and one half percent v's, one and one fourth percent w's and one half percent x's.

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 1	Name: Emily Fox

Recall that a ***run*** in a string $w \in \{0,1\}^*$ is a maximal substring of w whose characters are all equal.

(a) Let L_a denote the set of all non-empty strings in $\{0,1\}^*$ where the length of the first run is equal to the number of runs. ***Prove*** that L_a is not a regular language. $F = 00^*1^*$?

(b) Let L_b denote the set of all strings in $\{0,1\}^*$ that contain an even number of odd-length runs. Describe a DFA or NFA that accepts L_b ***and*** give a regular expression that describes L_b. (You do not need to prove that your answers are correct.)

(a) Let $F = 1(11)^*$. ← odd # 1s

Let $x \neq y$ be any two distinct members of F.

$x = 1(11)^i \neq y = 1(11)^j$ for some $i \neq j$.

Let $z = (01)^i$

$xz = 1(11)^i(01)^i$ has $2i+1$ runs & first has length $2i+1$,

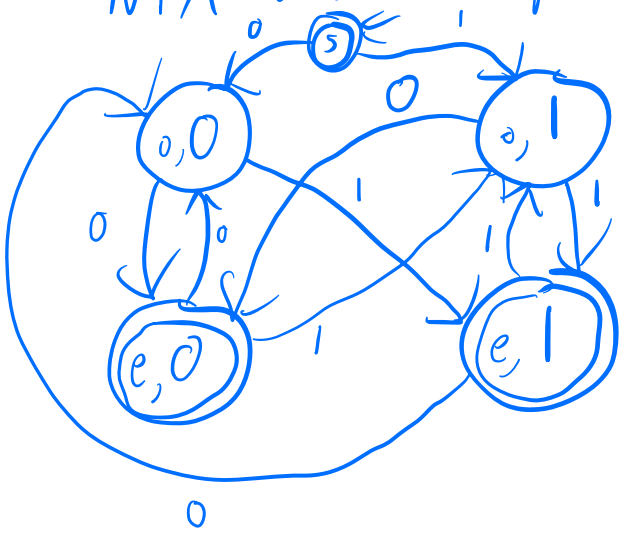
so $xz \in L_a$.

$yz = 1(11)^j(01)^i$ has $2i+1$ runs, but first has length

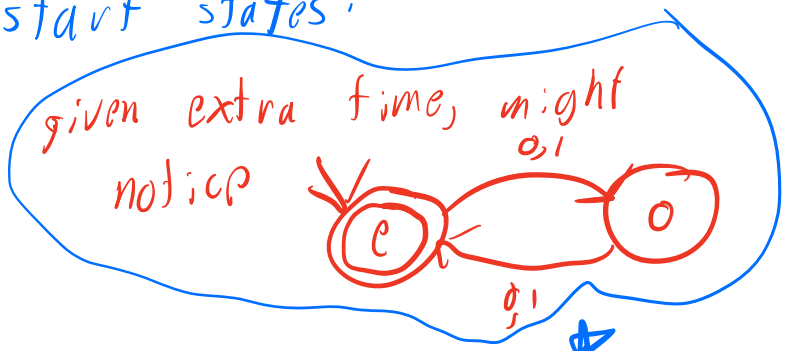
$2j+1 \neq 2i+1$, so $yz \notin L_a$.

$x \neq y$ were arbitrary, so F is an infinite fooling set of L_a.

(b) DFA ~~NFA~~ with multiple start states:



given extra time, might notice



$((0+1)(0+1))^*$

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 2	Name: Emily Fox

Aladdin and Badroulbadour are playing a cooperative game. Each player has an array of positive integers, arranged in a row of squares from left to right. Each player has a token, which starts at the leftmost square of their row; their goal is to move *both* tokens onto the rightmost squares at the same time.

On each turn, *both* players move their tokens *in the same direction*, either left or right. The distance each token travels is equal to the number under that token at the beginning of the turn. If *either* token moves past either end of its row, then both players immediately lose.

Describe and analyze an algorithm to determine whether Aladdin and Badroulbadour can solve their puzzle, given the input arrays $A[1..n]$ and $B[1..n]$.

Build a graph $G = (V, E)$.

$V := \{1, \dots, n\} \times \{1, \dots, n\}$

$E := \{((i,j), (i - A[i], j - B[j])) \mid (i,j) \in V, (i - A[i], j - B[j]) \in V\} \cup \{((i,j), (i + A[i], j + B[j])) \mid (i,j) \in V, (i + A[i], j + B[j]) \in V\}$.

Need to determine if $(1,1)$ can reach (n,n).

Call BFS$((1,1))$ + return if (n,n) is marked.

Time: $O(|V| + |E|) = \underline{O(n^2)}$

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 3	Name: Emily Fox

Submit a solution to ***exactly one*** of the following problems. Don't forget to tell us which problem you've chosen!

(a) Let $G = (V, E)$ be an arbitrary undirected graph. A subset $S \subseteq V$ of vertices is *mostly independent* if more than half the vertices of S have no neighbors in S.

Prove that the following problem is NP-*complete*: Given an undirected graph G and an integer k, decide if G contains a mostly independent set of size at least k.

(b) ***Prove*** that the following problem is NP-*complete*: Given an undirected graph G and an integer k, decide if G contains two disjoint independent sets of size k.

(In fact, both of these problems are NP-complete, but we only want a proof for one of them.)

(a) In NP: The certificate is the mostly independent set.

NP-hard: via reduction from independent set.

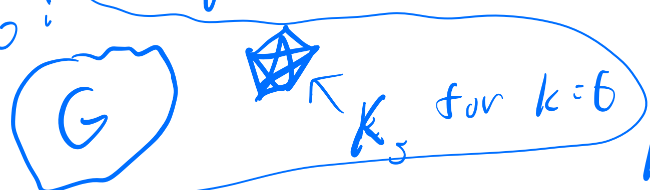
Given undirected graph $G=(V,E)$ & integer k.

Build mostly ind. set instance $G'=(V',E')$ & k'.

If $k=0$, $G' =$ • ← one vertex & $k'=1$. Trivially, Yes & Yes ✓

o.w. G' is a copy of G plus a disjoint clique/complete graph with $k-1$ new vertices.

eg. G':



Takes polynomial $O(|V|+|E|+k^2$ time).

$k' := 2k-1$.

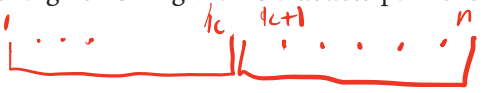
Need to prove G has ind. set of size k iff G' has mostly ind. set of size k'.

cont. on page 8.

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 4	Name: Emily Fox

(a) Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a palindrome.

(b) A *double palindrome* is the concatenation of two *non-empty* palindromes. Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a *double* palindrome. *[Hint: Use your algorithm from part (a).]*



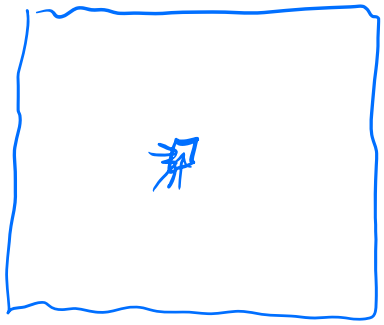
(a) $Palin(i,j)$: length of the longest palindrome subsequence of $A[i..j]$.

Need to compute $Palin(1,n)$.

$$Palin(i,j) = \begin{cases} 0 & \text{if } i > j \\ 1 & \text{if } i = j \\ \max\{Palin(i+1,j), Palin(i,j-1)\} & \text{if } i < j \text{ \& } A[i] \neq A[j] \\ \max\{2 + Palin(i+1,j-1), Palin(i+1,j), Palin(i,j-1)\} & \text{o.w.} \end{cases}$$

Memoize in a 2D array $Palin[1..n+1, 0..n]$

Eval order



- or -

- for i going down &
for j going up }

Time: $O(1) \cdot O(n^2) = O(n^2)$

(b) on page 9.

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 5	Name: Emily Fox

See the question sheets for a full description of the problem and its greedy solution.

(a) ~~X~~ ***Prove*** the greedy strategy described in the question sheets leads to a correct algorithm for the problem.

[Hint: Let X^* be an optimal solution to the problem. Do an exchange argument to show there is an equally large solution X that contains ℓ.]

(b) ~~X~~ Describe an algorithm to find a maximum size set of numbers X as described in the question sheets. An algorithm based on the strategy in part (a) requires no further justification to receive full credit.

(a) Let X^* be an optimal subset. Will prove there is a solution X s.t. $|X| \geq |X^*|$ & $\ell \in X$.

If $\ell \in X^*$, let $X = X^*$.

o.w, let ℓ' be the least/leftmost member of X^*.

Let $X = (X^* \setminus \{\ell'\}) \cup \{\ell\}$. For any $x \in (X^* \setminus \{\ell'\})$, $|x - \ell| > |x - \ell'| \geq L$.

so X is still valid. Also, $|X| = |X^*|$.

Finally, remainder of X should largest subset of numbers $\geq \ell + L$ which is found by induction.

(b) Algorithm:

Sort intervals by lower endpoint. Assume intervals named so $[a_i, b_i]$ is the ith interval after sorting.

If $n = 0$, return.

$X \leftarrow \{a_1\}$

last $\leftarrow a_1$, curInterval $\leftarrow 1$

while curInterval $\leq n$

if $b_i \geq$ last $+ L$

add last $+ L$ to X

last $\leftarrow$ last $+ L$

else, curInterval $\leftarrow$ curInterval $+ 1$

(outside while)

return X

Time: $O(|X| + n \log n)$

↑ loop ↑ sorting

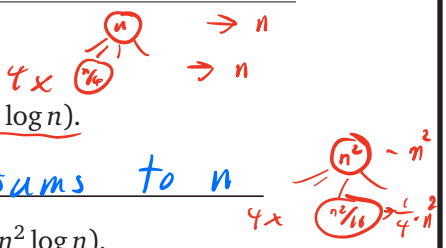
CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 6	Name: Emily Fox

For each statement below, check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No". Read each statement *very* carefully; some of these are deliberately subtle!

(a) Which of the following statements are true?

- The solution to the recurrence $T(n) = 4T(n/4) + O(n)$ is $T(n) = O(n \log n)$.

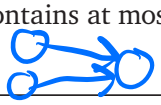
  [X] Yes [ ] No every level of recursion tree sums to n



- The solution to the recurrence $T(n) = 4T(n/4) + O(n^2)$ is $T(n) = O(n^2 \log n)$.

  [ ] Yes [X] No decreasing geometric level sums / $T(n) = O(n^2)$

- Every directed acyclic graph contains at most one source and at most one sink.

  [ ] Yes [X] No 

- Depth-first search explores every path from the source vertex s to every other vertex in the input graph.

  [ ] Yes [X] No Does not explore all of the many many paths.

- Suppose $A[1..n]$ is an array of integers. Consider the following recursive function:

$$Huh(i,j) = \begin{cases} 0 & \text{if } i < 0 \text{ or } j > n \\ \max \begin{Bmatrix} Huh(i, j+1) \\ Huh(i-1, j) \\ A[i] \cdot A[j] + Huh(i-1, j+1) \end{Bmatrix} & \text{otherwise} \end{cases}$$

  We can compute $Huh(n, 0)$ by memoizing this function into an array $Huh[0..n, 0..n]$ in $O(n^2)$ time, increasing i in the outer loop and increasing j in the inner loop.

  [ ] Yes [X] No Need decreasing j.

Problem 6 continues onto the next page.

(b) Suppose we want to prove that the following language is undecidable.

$$\textsc{Duck} := \{\langle M \rangle \mid M \text{ accepts } \texttt{GRAPES} \text{ but rejects } \texttt{LEMONADE}\}$$

Professor Canard, your wetlands-ornithology instructor, suggests a reduction from the standard halting language

$$\textsc{Halt} := \{(\langle M \rangle, w) \mid M \text{ halts on input } w\}.$$

Specifically, suppose there is a Turing machine LooksLikeADuck that decides Duck. Professor Canard claims that the following algorithm decides Halt.

```
DecideHalt(⟨M⟩, w):
  Write code for the following algorithm:
    WaddleAway(x):
      run M on input w
      ⟨⟨ignore the output of M⟩⟩
      if x = LEMONADE
          return False
      else
          return True
  return LooksLikeADuck(⟨WaddleAway⟩)
```

Which of the following statements ***must be*** true ***for all*** inputs $(\langle M \rangle, w)$?

- If M accepts w, then WaddleAway accepts GRAPES.

  [X] Yes [ ] No — M halts on w, & if check is True

- If M diverges on w, then WaddleAway rejects GRAPES.

  [ ] Yes [X] No — WaddleAway diverges on every x.

- If M accepts w, then LooksLikeADuck accepts ⟨WaddleAway⟩.

  [X] Yes [ ] No — Gets to if ⇒ rejects LEMONADE & accepts all else

- If M diverges on w, then DecideHalt rejects $(\langle M \rangle, w)$.

  [X] Yes [ ] No — WaddleAway always diverges ⇒ LLAD rejects

- DecideHalt decides the language Halt. (That is, Professor Canard's reduction is correct.)

  [X] Yes [ ] No — See last two (same behavior for M rejecting w)

CS/ECE 374 A ✧ Spring 2026 Practice Final 2 Problem 7	Name: Emily Fox

For each statement below, check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No". Read each statement *very* carefully; some of these are deliberately subtle!

(a) Which of the following statements are true for ***all*** languages $L \subseteq \{0,1\}^*$?

- $L^* = (L^*)^*$

  [X] Yes [ ] No — Concats of concats are concats.

  If $L = \emptyset$, $L^* = (L^*)^* = \{\varepsilon\}$. False.

- If L is decidable, then L^* is decidable.

  [X] Yes [ ] No — Given w, guess a split $w_1, w_2 \ldots w_k = w$ & check if all $w_i \in L$.

- L is either regular or NP-hard.

  [ ] Yes [X] No — $L = \{0^n 1^n \mid n \geq 0\}$ is in P.

- If L is undecidable, then L has an infinite fooling set.

  [X] Yes [ ] No — undecidable $\Rightarrow$ not regular

- The language $\{\langle M \rangle \mid M \text{ decides } L\}$ is undecidable.

  [ ] Yes [X] No — L = Halt

(b) Suppose there is a ***polynomial-time*** many-one reduction from some language $A \subseteq \{0,1\}^*$ to some other language $B \subseteq \{0,1\}^*$. Which of the following statements are true, assuming P $\neq$ NP?

- $A \cap B \neq \emptyset$.

  [ ] Yes [X] No — $A = (00)^*$, $B = 0(00)^*$.

- There is an algorithm to transform any Python program that solves B in polynomial time into a Python program that solves A in polynomial time.

  [X] Yes [ ] No — Do reduction & solve using B program.

- If B is NP-hard, then A is NP-hard.

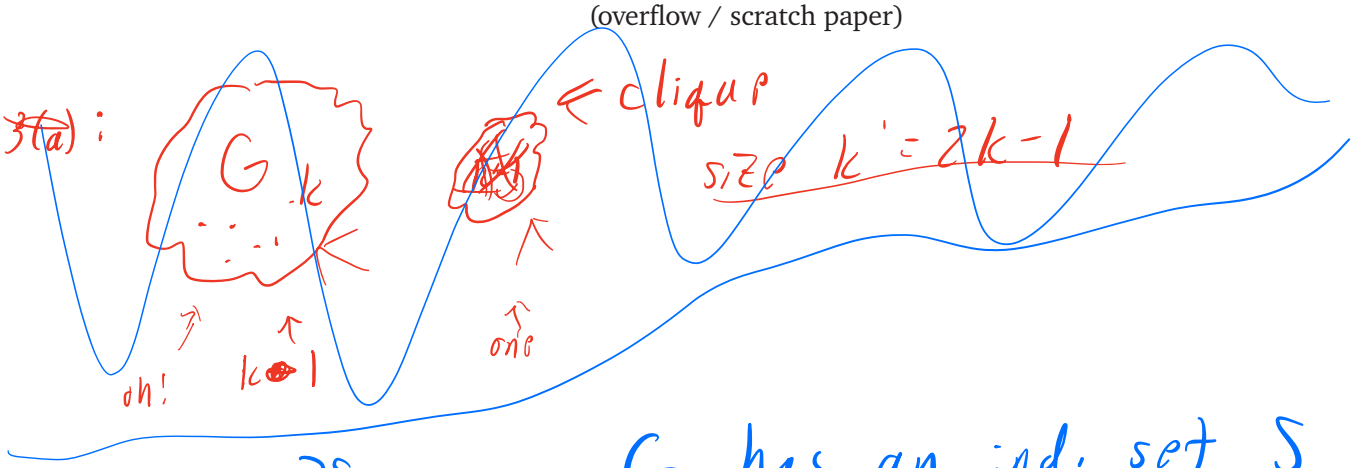
  [X] Yes [ ] No — backwards / A = Acyclic & B = 3SAT

- If B is decidable, then A is decidable.

  [X] Yes [ ] No — Do reduction. Run machine for B.

- If a Turing machine M accepts every string in B, the *same* Turing machine M also accepts every string in A.

  [ ] Yes [X] No — M decides $B = 0(00)^*$ & $A = (00)^*$.



3(a) cont. ⇒ Suppose G has an ind. set S of size k. Let S' be S ∪ all members of the new clique.
$|S'| = 2k-1$. Also no vertex in S has an edge within S', so S' is mostly ind.

⇐ Suppose G' has a mostly ind. set S' of size $k' = 2k-1$. There are ≥ k vertices s" in S' with no edges to other members of S'.
At most one vertex of s" is in the new clique.
If none in new clique, then s" contains an ind. set of size k in G.
Otherwise, if k=1, any {v} with $v \in V$ is an ind. set in G of size k.
Finally, if k>1, s" has ≥ k-1 vertices in G.
If exactly k-1, then S'\S" has at least one vertex v in G. S" ∪ {v} is ind. in G & has size k. o.w, grab k members of s".

7(b) cont.

Compute table for Palin as described in (a).

return $\max\{Palin(1,k) + Palin(k+1,n) \mid 1 \le k \le n\}$.

loop takes $O(1)$ additional time

Time: $O(n^2) + O(n) = \underline{O(n^2)}$

3(b) DO NOT GRADE!

In NP: Certificate is the ind. sets.

NP-hard: Reduction from ind. set.

Given undirected $G=(V,E)$ & integer k.

Build graph $G'=(V',E')$ with integer k'.

$V' = V \times \{0,1\}$

$E' = \{(u,i)-(v,i) \mid u-v \in E \text{ and } i \in \{0,1\}\} \cup \{(u,0)-(v,1) \mid u,v \in V\}$

$k' = k$.

Takes polynomial $O(|V^2|)$ time.

Need to prove G has ind. set of size k iff G' has two disjoint ind. sets of size $k'=k$.

$\Rightarrow$ Suppose G has an ind. set S of size k. $S' \leftarrow$ the copies of S in the copies of G in G'. S' is two disjoint ind. sets of size k.

$\Leftarrow$ Let $\{S_1, S_2\}$ be disjoint ind. sets of size k in G'. Any two vertices in S_1 come from same copy of G. o.w. they would share an edge. So S_1 in ind. in G.

(overflow / scratch paper)

Some useful NP-hard problems. You are welcome to use any of these in your own NP-hardness proofs, except of course for the specific problem you are trying to prove NP-hard.

3SAT: Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment?

INDEPENDENTSET: Given an undirected graph G and an integer k, is there a subset of k vertices in G that have no edges among them?

CLIQUE: Given an undirected graph G and an integer k, is there a subset of k vertices in G where there is an edge between every pair of them?

VERTEXCOVER: Given an undirected graph G and an integer k, is there a subset of k vertices that touch every edge in G?

DOMINATINGSET: Given an undirected graph G and an integer k, is there a subset of k vertices such that every vertex of G is either in the set or adjacent to a member of the set?

SETCOVER: Given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set S and an integer k, is there a subcollection of k of these subsets whose union is S?

HITTINGSET: Given a collection of subsets $S_1, S_2, \ldots, S_m$ of a set S and an integer k, is there a subset of S of size k that intersects every subset S_i?

3COLOR: Given an undirected graph G, can its vertices be colored with three colors so that every edge touches vertices with two different colors?

HAMILTONIANPATH: Given graph G (either directed or undirected), is there a path in G that visits every vertex exactly once?

HAMILTONIANCYCLE: Given a graph G (either directed or undirected), is there a cycle in G that visits every vertex exactly once?

TRAVELINGSALESMAN: Given a graph G (either directed or undirected) with weighted edges and a number L, is there a Hamiltonian path/cycle of weight at most L in G?

LONGESTPATH: Given a graph G (either directed or undirected, possibly with weighted edges) and a number L, is there a simple path of length at least L in G?

STEINERTREE: Given an undirected graph G with some of the vertices marked and an integer k, is there a subtree of G with at most k edges that contains every marked vertex?

SUBSETSUM: Given a set X of positive integers and an integer k, does X have a subset whose elements sum to k?

PARTITION: Given a set X of positive integers, can X be partitioned into two subsets with the same sum?

3PARTITION: Given a set X of $3n$ positive integers, can X be partitioned into n three-element subsets, all with the same sum?

DRAUGHTS: Given an $n \times n$ international draughts configuration and an integer k, is there a move that can (and therefore must) capture at least k pieces in a single move?