

CS/ECE 374 A ✧ Fall 2025  
☞ Practice Final Exam 2 ☞  
December 9, 2025

Name:	
NetID:	

- 
- ***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  $8\frac{1}{2}'' \times 11''$  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 are required for full credit if and only if we explicitly ask for them, using the word ***prove*** or ***justify*** in bold italics.

- 
- **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.
  - Breathe in. Breathe out. You've got this.
-



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.
- (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.)
-

Aladdin and Badroulbador 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 Badroulbador can solve their puzzle, given the input arrays  $A[1..n]$  and  $B[1..n]$ .

---

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 finding the largest mostly independent set in  $G$  is NP-hard.
- (b) **Prove** that the following problem is NP-hard: Given an undirected graph  $G$ , find the largest integer  $k$  such that  $G$  contains *two disjoint* independent sets of size  $k$ .

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

---

- (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).]
-

Describe and analyze a recursive algorithm to compute the *largest complete internal subtree* of a given binary tree. Your algorithm should return both the root and the depth of this internal subtree.

---

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  $T(n) = O(n \log n)$ .

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

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

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

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	No
-----	----

\_\_\_\_\_



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

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

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

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

Specifically, suppose there is a Turing machine LOOKSLIKEADUCK that decides DUCK. Professor Canard claims that the following algorithm decides HALT.

DECIDEHALT( $\langle M \rangle, w$ ):  
 Write code for the following algorithm:

WADDLEAWAY( $x$ ):  
 run  $M$  on input  $w$   
*⟨ignore the output of  $M$ ⟩*  
 if  $x = \text{LEMONADE}$   
     return FALSE  
 else  
     return TRUE

return LOOKSLIKEADUCK( $\langle \text{WADDLEAWAY} \rangle$ )

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

- If  $M$  accepts  $w$ , then WADDLEAWAY accepts GRAPES.

Yes

No

- If  $M$  diverges on  $w$ , then WADDLEAWAY rejects GRAPES.

Yes

No

- If  $M$  accepts  $w$ , then LOOKSLIKEADUCK accepts  $\langle \text{WADDLEAWAY} \rangle$ .

Yes

No

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

Yes

No

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

Yes

No

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  $L \subseteq \{0, 1\}^*$ ?

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

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

- $L$  is either regular or NP-hard.

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

(b) Suppose there is a *polynomial-time* reduction from some language  $A \subseteq \{0, 1\}^*$  reduces 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	No
-----	----

\_\_\_\_\_

- 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.

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

- If  $B$  is decidable, then  $A$  is decidable.

Yes	No
-----	----

\_\_\_\_\_

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

Yes	No
-----	----

\_\_\_\_\_

(overflow / scratch paper)

(overflow / scratch paper)

(overflow / scratch paper)

(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.

**CIRCUITSAT:** Given a boolean circuit, are there any input values that make the circuit output TRUE?

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

**MAXINDEPENDENTSET:** Given an undirected graph  $G$ , what is the size of the largest subset of vertices in  $G$  that have no edges among them?

**MAXCLIQUE:** Given an undirected graph  $G$ , what is the size of the largest complete subgraph of  $G$ ?

**MINVERTEXCOVER:** Given an undirected graph  $G$ , what is the size of the smallest subset of vertices that touch every edge in  $G$ ?

**MINSETCOVER:** Given a collection of subsets  $S_1, S_2, \dots, S_m$  of a set  $S$ , what is the size of the smallest subcollection whose union is  $S$ ?

**MINHITTINGSET:** Given a collection of subsets  $S_1, S_2, \dots, S_m$  of a set  $S$ , what is the size of the smallest subset of  $S$  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?

**CHROMATICNUMBER:** Given an undirected graph  $G$ , what is the minimum number of colors required to color its vertices, 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, what is the minimum total weight of any Hamiltonian path/cycle in  $G$ ?

**LONGESTPATH:** Given a graph  $G$  (either directed or undirected, possibly with weighted edges), what is the length of the longest simple path in  $G$ ?

**STEINERTREE:** Given an undirected graph  $G$  with some of the vertices marked, what is the minimum number of edges in a subtree of  $G$  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?

**INTEGERLINEARPROGRAMMING:** Given a matrix  $A \in \mathbb{Z}^{n \times d}$  and two vectors  $b \in \mathbb{Z}^n$  and  $c \in \mathbb{Z}^d$ , compute  $\max\{c \cdot x \mid Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$ .

**FEASIBLEILP:** Given a matrix  $A \in \mathbb{Z}^{n \times d}$  and a vector  $b \in \mathbb{Z}^n$ , determine whether the set of feasible integer points  $\max\{x \in \mathbb{Z}^d \mid Ax \leq b, x \geq 0\}$  is empty.

**DRAUGHTS:** Given an  $n \times n$  international draughts configuration, what is the largest number of pieces that can (and therefore must) be captured in a single move?

**SUPERMARIOBROTHERS:** Given an  $n \times n$  Super Mario Brothers level, can Mario reach the castle?