

**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:

$$\underline{000}111111\underline{10000} = \underline{000} \cdot \underline{111111} \cdot \underline{0000}$$

- (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  $\epsilon$  (because it has no first run).

$\epsilon \notin L_a$

**Prove** that  $L_a$  is not a regular language.

*infinite fooling set*

- (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  $\epsilon$ , but does not contain either 0011100 or 11110.

*0 is even*

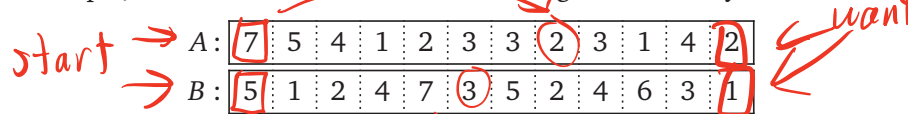
- Describe a DFA or NFA that accepts  $L_b$  **and**
- Give a regular expression that describes  $L_b$ .

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

2. Aladdin and 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. 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 Badroulbador are given the arrays



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

*graph?*

Problems 3–7 appear on the following pages.

*noting gets smaller = mark asid positions... oh*

*dynamic programming?*

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 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.)

good from max ind. set?



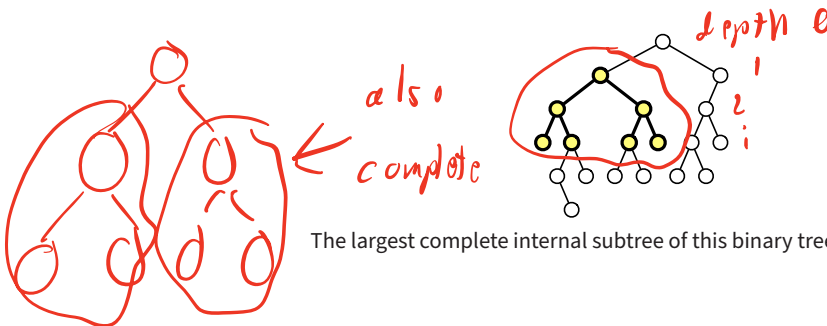
4. Recall that a *palindrome* is any string that is equal to its reversal, like REDIVIDER or POOP.

- (a) Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a palindrome. dynamic programming
- (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. Recall that the *depth* of a vertex  $v$  in a binary tree  $T$  is the length of the unique path in  $T$  from  $v$  to the root of  $T$ . A binary tree is *complete* if every internal node has two children, and every leaf has exactly the same depth. An internal subtree of a binary tree  $T$  is any connected subgraph of  $T$ .

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.



The largest complete internal subtree of this binary tree has depth 2.

Problems 6 and 7 appear on the following pages.

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

☐ No

Suppose  $x = 3$  and  $y = 4$ .

- 3SAT can be solved in polynomial time.

☐ Yes

☒ No

3SAT is NP-hard.

- If  $P = NP$  then Jeff is the Queen of England.

☒ 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:

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

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.

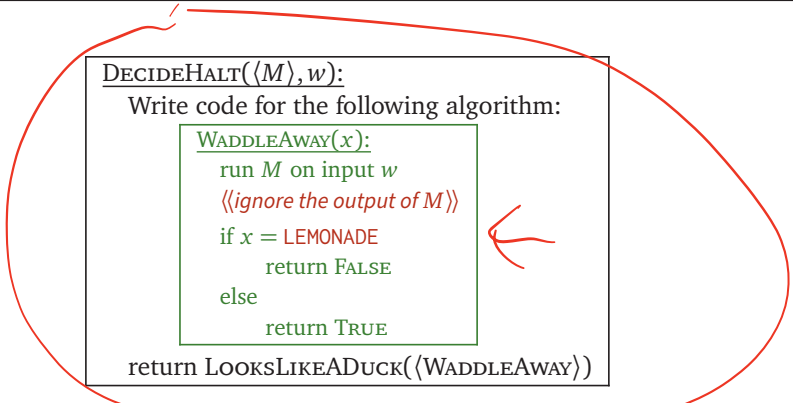
- (b) Suppose we want to prove that the following language is undecidable.

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

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

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

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  $w$ 
       $\langle\langle$ ignore the output of  $M$  $\rangle\rangle$ 
      if  $x = \text{LEMONADE}$ 
        return FALSE
      else
        return TRUE
    return LOOKSLIKEADUCK( $\langle$ 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.
- 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. Rice?

(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  $P \neq NP$ ?

- $A \cap B \neq \emptyset$ .
- 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 ✧ Fall 2025  
☞ Practice Final Exam 2 ☞  
December 9, 2025

Name:	Emily Fox
NetID:	e/cfox

- 
- **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.
-

Final Exam: Friday 12th 8am-11am. - two double  
Conflict Final Exam: Mon 15th 8am-11am sided  
cheat sheets

FLEx evals (at 10%) - close Thur 11th

CA applications open - see Google Form

"full consideration deadline" Mon Dec 29.

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

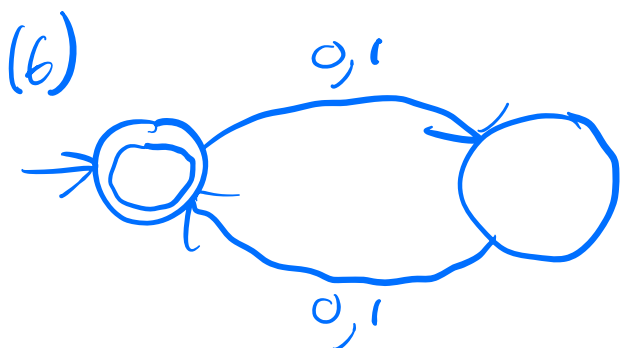
- (a) Let  $L_a$  denote the set of all non-empty strings in  $\{0,1\}^*$  where the length of the first run is equal to the number of runs. **Prove** that  $L_a$  is not a regular language.
- (b) Let  $L_b$  denote the set of all strings in  $\{0,1\}^*$  that contain an even number of odd-length runs. 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.)

track current run?

every new character

slips parity of ~~A~~ odd runs!

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



$(00 + 01 + 10 + 11)^*$

(a) Let  $F = \{0^{2i+1} \mid i \geq 0\}$ .

Let  $x, y \in F$  be distinct.

We  $x = 0^{2j+1}$  &  $y = 0^{2k+1}$  where  $j, k \geq 0$  &  $j \neq k$ .

Let  $z = (10)^j$ .

$xz = 0^{2j+1}(10)^j$  has  $2j+1$  runs, so  $xz \in L_a$ .

$yz = 0^{2k+1}(10)^j$  has  $2j+1 \neq 2k+1$  runs, so

$x, y$  were arbitrary members of  $F$ ,  $yz \notin L_a$ .

so  $F$  is a fooling set of  $L_a$ .  $F$  is infinite, so  $L_a$  is not reg.

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  $B[1..n]$ .

Define a <sup>pos. on A</sup> ~~directed~~ graph  $G = (V, E)$ .  
 $V = \{1..n\} \times \{1..n\}$  <sup>pos. on B</sup>  
 $E = \{(i, j) \rightarrow (i + A[i], j + B[j]) \mid (i, j) \in V \text{ and } (i + A[i], j + B[j]) \in V\}$   
 $\cup \{(i, j) \rightarrow (i - A[i], j - B[j]) \mid (i, j) \in V \text{ and } (i - A[i], j - B[j]) \in V\}$

Can solve puzzle iff  $(1, 1)$  can reach  $(n, n)$ .

Call linear time WFS  $((1, 1))$ .

Return whether  $(n, n)$  gets marked.


Takes  $O(V + E) = O(n^2)$  time.

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 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  $k$ . *max ind set*

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

 (b) Reduction from Max Ind. Set.

Given graph  $G = (V, E)$ .

Build  $H = (V', E')$ .

$$V' = V \times \{1, 2\} \quad E' = \{(u, i)(v, i) \mid uv \in E + i \in \{1, 2\}\} \\ \cup \{(u, 1)(v, 2) \mid u, v \in V\}$$

Return largest  $k$  st.  $H$  has two disjoint ind. sets of size  $k$ .

Takes  $O(V^2 + E)$  which is a polynomial.

Need to prove  $G$  has ind set of size  $k$  iff (b) problem returns  $k$  for  $H$ .

$\Rightarrow$  Suppose  $G$  has ind. set  $S$  of size  $k$ .

$\{(u, 1) \mid u \in S\} + \{(u, 2) \mid u \in S\}$  are two disjoint ind. sets in  $H$ .

$\Leftarrow$  Suppose  $H$  has disjoint ind. sets  $S_1$  and  $S_2$  of size  $k$ .

Emily Fox

- (a) Describe and analyze an algorithm to find the length of the longest subsequence of a given string that is a palindrome.  $A[1..n]$   $B \xrightarrow{\quad} B$  recurse inward..
- (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) Guess where to split A.  $x \xrightarrow{\quad} y$

$LPS(i, j)$ : Length of the longest palindrome subsequence of  $A[i..j]$ .

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

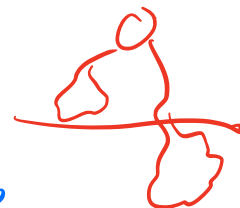
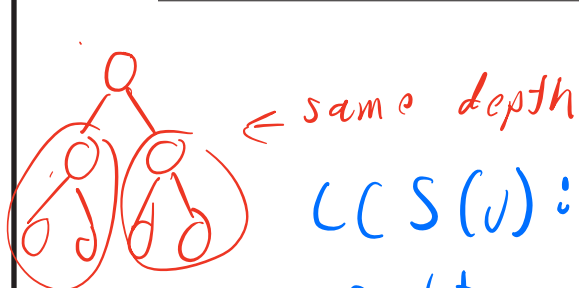
$\max \{LPS(i+1, j), LPS(i, j-1), 2 + LPS(i+1, j-1)\}$  o.w.



Return  $LPS[1, n]$ .  $O(1) \cdot O(n^2) = O(n^2)$  time.

(b) on page 9

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.



$LCS(v)$ : depth of largest complete subtree rooted at  $v$

$$LCS(v) = \begin{cases} 0 & \text{if } v \text{ has } \leq 1 \text{ child} \\ 1 + \min\{LCS(left(v)), LCS(right(v))\} & \text{o.w.} \end{cases}$$

Solve all in postorder + store on nodes,  
Return  $v + LCS(v)$  maximizing  $LCS(v)$ .

$O(n)$  time if  $n$  vertices

( $O(v)$  time)

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

☒ Yes

☐ No

All levels of recursion tree sum to  $n$ .

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

☐ Yes

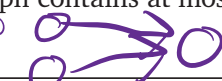
☒ No

Tree gives a decreasing geo series  $\Rightarrow O(n^2)$

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

☐ Yes

☒ No

two sources 

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

☐ Yes

☒ No

Could be an exponential # paths / only explores some.

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

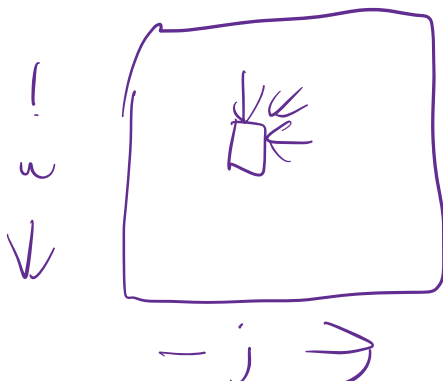
$$Huh(i, j) = \begin{cases} 0 & \text{if } i < 0 \text{ or } j > n \\ \max \left\{ \begin{array}{l} Huh(i, j+1) \\ Huh(i-1, j) \\ A[i] \cdot A[j] + Huh(i-1, j+1) \end{array} \right\} & \text{otherwise} \end{cases}$$

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

Need to eval  $j$  in decreasing order.



Problem 6 continues onto the next page.

(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(WADDLEAWAY)

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

*Get past run  $M$ ,  $x \neq \text{LEMONADE}$ .*

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

☐ Yes

☒ No

*Waddle Away will diverge on  $M$  run.*

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

☒ Yes

☐ No

*Gets past  $M$ , rejects LEMONADE. Accepts <sup>all</sup> else.*

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

☒ Yes

☐ No

*WA diverges  $\Rightarrow$  does not accept GRAPES <sup>= CLAD</sup> rejects*

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

☒ Yes

☐ No

*~~WA still accepts GRAPES~~  
 previous accept lines apply to  
 $M$  halting*

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

$$\emptyset^* = \epsilon \cdot \{0,1\}^* = \epsilon$$

$$\{0,1\}^* = \epsilon^* = \epsilon$$

•  $L^* = (L^*)^*$

☒ Yes ☐ No

Concat of concats of  $L$  is concat of  $L + \epsilon$ .

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

☒ Yes ☐ No

Try all subdivisions of input, running  $L$  algo for each piece.

•  $L$  is either regular or NP-hard.

☐ Yes ☒ No

$\{0^n 1^n \mid n \geq 0\}$  can be decided in poly time.

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

☒ Yes ☐ No

No infinite fooling set  $\Rightarrow$  regular  $\Rightarrow$  an algo (check what DFA does)

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

☐ Yes ☒ No

Let  $L = \text{Halt}$ . Can always reject.

Reduce? May not be a y machine?

(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

$B = \Sigma^* \setminus A$ .

• 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

"Inline" the  $B$  program into an  $A$  program.

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

☐ Yes ☒ No

Reduction in wrong direction.

(Let  $A \in P \subseteq NP$ )

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

☒ Yes ☐ No

Algo for  $B$  used to decide  $A$ .

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

☐ Yes ☒ No

Suppose  $M$  decides  $B + A = \Sigma^* \setminus B$ .

3(b) cont.

(overflow / scratch paper)

All  $(u,1) + (v,2)$  share an edge, so

$S_1$  &  $S_2$  each live in one copy of  $G$

(so all  $(u,1)$  in  $S_1$  or all  $(u,2)$ ).

So  $S_1$  is an independent set in  $G$

after removing second component.

+  $|S_1| = k$ .

(overflow / scratch paper)

7(6) Return  $\max_{1 \leq k \leq n-1} \{LPS(1, k) + LPS(k+1, n)\}$

↑  
final  
character  
of first pal

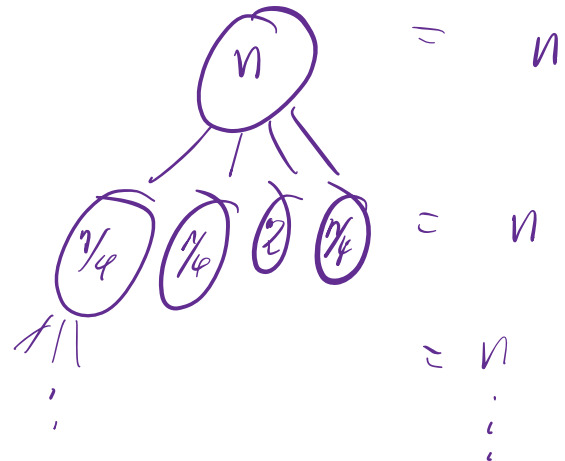
Fill LPS table & then evaluate  
the max above.

$O(n^2) + O(n) \approx O(n^2)$  time

↑                      ↑  
fill LPS[]          find best  
                                 k

(overflow / scratch paper)

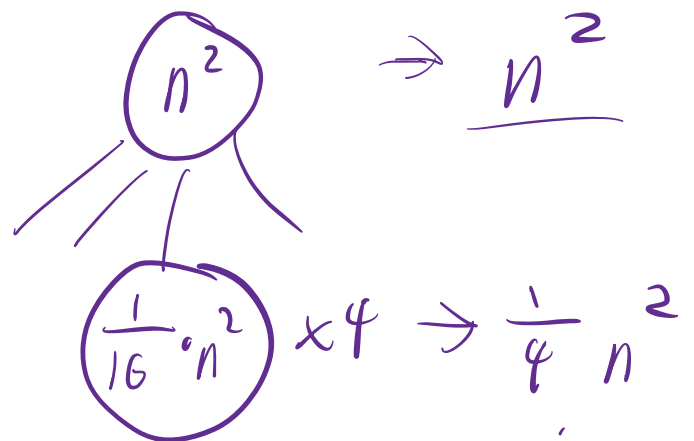
$$T(n) = 4 T(\underline{n/4}) + \underline{O(n)}$$



all same

$$\text{no. of levels} \approx O(\log n)$$

$$T(n) = 4 T(\underline{n/4}) + \underline{O(n^2)}$$



$$\text{level } i: \left(\frac{1}{4}\right)^i n^2$$

decreasing geometric

$$O(n^2)$$

# DO NOT GRADE

3(a) Input  $G$



Reduction from Max Ind Set.

Given  $G = (V, E)$ .

Build  $H = (V', E')$

✓ extra vertices

$$V' = V \sqcup \{1, \dots, |V|\}$$

disjoint union

$$E' = E \cup \{ij \mid i, j \in \{1, \dots, |V|\}\}$$

Find largest mostly ind. sets in  $H$  & return its vertices with no neighbors in  $S'$ .

$O(V^2)$  time which is polynomial.

Want to prove  $G$  has ind. set of size  $k$  iff  $H$  has mostly ind. set of size  $2k-1$ . Note largest in  $H$  has odd size,  $\Rightarrow$  Let  $S$  of size  $k$  be ind. in  $G$ . can add another from complete graph on  $\{1, \dots, |V|\}$

Let  $S' = S \cup \{1, \dots, k-1\}$ .  $S'$  is mostly ind. in  $H$ .

$\Leftarrow$  Let  $S'$  of size  $2k-1$  be mostly ind. in  $H$ .

There are  $k$  vertices  $S$  in  $S'$  without neighbors in  $S'$ . At most one lives outside  $G$ . If so take  $S \cap V$  & one other member of  $S'$  in  $V$  to get ind. set of size  $k$  in  $G$ , o.w.  $S$  is ind. set in  $G$ .

**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?