CS/ECE 374 A ✧ Fall 2021 Fake Midterm 1 Problem 1	Name:

For each statement below, check "Yes" if the statement is ***ALWAYS*** true and "No" otherwise, and give a ***brief*** explanation of your answer.

(a) Every integer in the empty set is prime.

Yes No

(b) The language $\{0^m 1^n \mid m + n \leq 374\}$ is regular.

Yes No

(c) The language $\{0^m 1^n \mid m - n \leq 374\}$ is regular.

Yes No

(d) For all languages L, the language L^* is regular.

Yes No

(e) For all languages L, the language L^* is infinite.

Yes No

(f) For all languages $L \subset \Sigma^*$, if L can be represented by a regular expression, then $\Sigma^* \setminus L$ is recognized by a DFA.

Yes No

(g) For all languages L and L', if $L \cap L' = \emptyset$ and L' is not regular, then L is regular.

Yes No

(h) Every regular language is recognized by a DFA with exactly one accepting state.

Yes No

(i) Every regular language is recognized by an NFA with exactly one accepting state.

Yes No

(j) Every language is either regular or context-free.

Yes No

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 1 Problem 2	Name:

For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either ***prove*** that the language is regular or ***prove*** that the language is not regular. ***Exactly one of these two languages is regular.*** Both of these languages contain the string 00110100000110100.

1. $\left\{0^n w 0^n \;\middle|\; w \in \Sigma^+ \text{ and } n > 0\right\}$

2. $\left\{w 0^n w \;\middle|\; w \in \Sigma^+ \text{ and } n > 0\right\}$

CS/ECE 374 A ✧ Fall 2021 **Fake Midterm 1 Problem 3**	Name:

The *parity* of a bit-string w is `0` if w has an even number of `1`s, and `1` if w has an odd number of `1`s. For example:

$$\mathit{parity}(\varepsilon) = \texttt{0} \qquad \mathit{parity}(\texttt{0010100}) = \texttt{0} \qquad \mathit{parity}(\texttt{00101110100}) = \texttt{1}$$

(a) Give a *self-contained*, formal, recursive definition of the *parity* function. (In particular, do ***not*** refer to # or other functions defined in class.)

(b) Let L be an arbitrary regular language. Prove that the language $\mathit{OddParity}(L) := \{w \in L \mid \mathit{parity}(w) = \texttt{1}\}$ is also regular.

(c) Let L be an arbitrary regular language. Prove that the language $\mathit{AddParity}(L) := \{\mathit{parity}(w) \cdot w \mid w \in L\}$ is also regular.

[Hint: Yes, you have enough room.]

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 1 Problem 4	Name:

For each of the following languages L, give a regular expression that represents L ***and*** describe a DFA that recognizes L. You do ***not*** need to prove that your answers are correct.

(a) All strings in $(0 + 1)^*$ that do not contain the substring 0110.

(b) All strings in 0^*10^* whose length is a multiple of 3.

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 1 Problem 5	Name:

For any string $w \in \{0,1\}^*$, let $obliviate(w)$ denote the string obtained from w by removing every 1. For example:

$$obliviate(\varepsilon) = \varepsilon$$
$$obliviate(000000) = 000000$$
$$obliviate(111111) = \varepsilon$$
$$obliviate(010001101) = 00000$$

Let L be an arbitrary regular language.

1. ***Prove*** that the language $\textsc{Obliviate}(L) = \{obliviate(w) \mid w \in L\}$ is regular.

2. ***Prove*** that the language $\textsc{Unobliviate}(L) = \{w \in \{0,1\}^* \mid obliviate(w) \in L\}$ is regular.

CS/ECE 374 A ✧ Fall 2021

ᔕ Midterm 1 ᔓ

September 27, 2021

ᔕ Directions ᔓ

- ***Don't panic!***
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- The exam has five numbered questions.
- Write your answers on blank white paper. Please start your solution to each numbered question on a new sheet of paper.
- You have 150 minutes to write, scan, and submit your solutions. The exam is designed to take at most 120 minutes to complete. We are providing 30 minutes of slack to scan and submit in case of unforeseen technology issues.
- If you are ready to scan your solutions before 9:15pm, send a private message to the host ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Please scan ***all*** paper that you used during the exam — first your solutions, in the correct order, then your cheat sheet (if any), and finally any scratch paper.
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics. In particular, if we ask you to show that a language is regular, you can provide a regular expression, DFA, NFA, or boolean combination *without justification*. Similarly, if we ask you to give a DFA or NFA, you to *not* have to name or describe the states.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam in your email. If you are in the middle of the exam, send Jeff email, finish the exam (if you can) within the time limit, and then send a second email with your completed exam.

---

1. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either prove that the language is regular (by constructing an appropriate DFA, NFA, or regular expression) or prove that the language is not regular (by constructing an infinite fooling set and proving that the set you construct is indeed a fooling set for that language).

   (a) $\{0^p 1^q 0^r \mid r = p + q\}$

   (b) $\{0^p 1^q 0^r \mid r = p + q \bmod 2\}$

   [Hint: First think about the language $\{0^p 1^q \mid q = p \bmod 2\}$*]*

2. Let L be any regular language over the alphabet $\Sigma = \{0, 1\}$.

   Let take2skip2(w) be a function takes an input string w and returns the subsequence of symbols at positions $1, 2, 5, 6, 9, 10, \ldots 4i+1, 4i+2, \ldots$ in w. In other words, take2skip2(w) takes the first two symbols of w, skip the next two, takes the next two, skips the next two, and so on. For example:

   $$\text{take2skip2}(\underline{1}) = 1$$
   $$\text{take2skip2}(\underline{01}0) = 01$$
   $$\text{take2skip2}(\underline{01}\underline{00}\underline{11}\underline{00}\underline{1}1) = 0111001$$

   Choose exactly one of the following languages, and prove that your chosen language is regular. (In fact, *both* languages are regular, but we only want a proof for one of them.) Don't forget to tell us which language you've chosen!

   (a) $L_1 = \{w \in \Sigma^* \mid \text{take2skip2}(w) \in L\}$.

   (b) $L_2 = \{\text{take2skip2}(w) \mid w \in L\}$.

3. Prove that the following languages are not regular by building an infinite fooling set for each of them. For each language, prove that the set you constructed is indeed a fooling set.

   (a) $\{0^p 1^q 0^r \mid r > 0 \quad \text{and} \quad q \bmod r = 0 \quad \text{and} \quad p \bmod r = 0\}$

   (b) $\{0^p 1^q \mid q > 0 \quad \text{and} \quad p = q^q\}$

4. Consider the following recursive function:

   $$\text{MINGLE}(w, z) := \begin{cases} z & \text{if } w = \varepsilon \\ \text{MINGLE}(x, azа) & \text{if } w = a \cdot x \text{ for some symbol } a \text{ and string } x \end{cases}$$

   For example, MINGLE(01, 10) = MINGLE(1, 0100) = MINGLE(ε, 101001) = 101001.

   (a) Prove that $|\text{MINGLE}(w, z)| = 2|w| + |z|$ for all strings w and z.

   (b) Prove that $\text{MINGLE}(w, z \bullet z^R) = (\text{MINGLE}(w, z \bullet z^R))^R$ for all strings w and z.

(There's one more question on the next page)

5. For each statement below, write "Yes" if the statement is always true and write "No" otherwise, and give a brief (one short sentence) explanation of your answer. Read these statements very carefully—small details matter!

   (a) If L is a regular language over the alphabet $\{0, 1\}$, then $\{w1w \mid w \in L\}$ is also regular.

   (b) If L is a regular language over the alphabet $\{0, 1\}$, then $\{x1y \mid x, y \in L\}$ is also regular.

   (c) The context-free grammar $S \to 0S1 \mid 1S0 \mid SS \mid 01 \mid 10$ generates the language $(0+1)^+$

   (d) Every regular expression that does not contain a Kleene star (or Kleene plus) represents a finite language.

   (e) Let L_1 be a finite language and L_2 be an arbitrary language. Then $L_1 \cap L_2$ is regular.

   (f) Let L_1 be a finite language and L_2 be an arbitrary language. Then $L_1 \cup L_2$ is regular.

   (g) The regular expression $(00 + 01 + 10 + 11)^*$ represents the language of all strings over $\{0, 1\}$ of even length.

   (h) The ε-reach of any state in an NFA contains the state itself.

   (i) The language $L = 0^*$ over the alphabet $\Sigma = \{0, 1\}$ has a fooling set of size 2.

   (j) Suppose we define an ε-DFA to be a DFA that can additionally make ε-transitions. Any language that can be recognized by an ε-DFA can also be recognized by a DFA that does not make any ε-transitions.

CS/ECE 374 A ✧ Fall 2021

☙ Conflict Midterm 1 ❧

September 28, 2021

☙ Directions ❧

- ***Don't panic!***
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- The exam has five numbered questions.
- Write your answers on blank white paper. Please start your solution to each numbered question on a new sheet of paper.
- You have 150 minutes to write, scan, and submit your solutions. The exam is designed to take at most 120 minutes to complete. We are providing 30 minutes of slack to scan and submit in case of unforeseen technology issues.
- If you are ready to scan your solutions before 9:15pm, send a private message to the host ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Please scan ***all*** paper that you used during the exam — first your solutions, in the correct order, then your cheat sheet (if any), and finally any scratch paper.
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics. In particular, if we ask you to show that a language is regular, you can provide a regular expression, DFA, NFA, or boolean combination *without justification*. Similarly, if we ask you to give a DFA or NFA, you to *not* have to name or describe the states.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam in your email. If you are in the middle of the exam, send Jeff email, finish the exam (if you can) within the time limit, and then send a second email with your completed exam.

---

1. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either prove that the language is regular (by constructing an appropriate DFA, NFA, or regular expression) or prove that the language is not regular (by constructing an infinite fooling set and proving that the set you construct is indeed a fooling set for that language).

   (a) $\{0^p 1^q 0^r \mid p = (q + r) \bmod 2\}$

   (b) $\{0^p 1^q 0^r \mid p = q + r\}$

2. Let L be any regular language over the alphabet $\Sigma = \{0, 1\}$.

   Let $\mathsf{compress}(w)$ be a function that takes a string w as input, and returns the string formed by compressing every run of 0s in w by half. Specifically, every run of $2n$ 0s is compressed to length n, and every run of $2n + 1$ 0s is compressed to length $n + 1$. For example:

   $$\mathsf{compress}(\underline{00000}11\underline{0001}) = \underline{000}11\underline{001}$$
   $$\mathsf{compress}(11\underline{000010}) = \underline{11001}\underline{0}$$
   $$\mathsf{compress}(11111) = 11111$$

   Choose exactly one of the following languages, and prove that your chosen language is regular. (In fact, *both* languages are regular, but we only want a proof for one of them.) Don't forget to tell us which language you've chosen!

   (a) $\left\{w \in \Sigma^* \;\middle|\; \mathsf{compress}(w) \in L\right\}$

   (b) $\left\{\mathsf{compress}(w) \;\middle|\; w \in L\right\}$

3. Recall that the *greatest common divisor* of two positive integers p and q, written $\gcd(p, q)$, is the largest positive integer r that divides both p and q. For example, $\gcd(21, 15) = 3$ and $\gcd(3, 74) = 1$.

   Prove that the following languages are not regular by building an infinite fooling set for each of them. For each language, prove that the set you constructed is indeed a fooling set.

   (a) $\{0^p 1^q 0^r \mid p > 0 \text{ and } q > 0 \text{ and } r = \gcd(p, q)\}$.

   (b) $\{0^p 1^{pq} \mid p > 0 \text{ and } q > 0\}$

4. Consider the following recursive function, RO (short for remove-ones) that operates on any string $w \in \Sigma^*$, where $\Sigma = \{0, 1\}$:

   $$\mathsf{RO}(w) := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ 0 \cdot \mathsf{RO}(x) & \text{if } w = 0 \cdot x \text{ for some string } x \\ \mathsf{RO}(x) & \text{if } w = 1 \cdot x \text{ for some string } x \end{cases}$$

   (a) Prove that $|\mathsf{RO}(w)| \le |w|$ for all strings w.

   (b) Prove that $\mathsf{RO}(\mathsf{RO}(w)) = \mathsf{RO}(w)$ for all strings w.

(There's one more question on the next page)

5. For each statement below, write "Yes" if the statement is always true and write "No" otherwise, and give a brief (one short sentence) explanation of your answer. Read these statements very carefully—small details matter!

   (a) $\{0^n1 \mid n > 0\}$ is the only infinite fooling set for the language $\{0^n10^n \mid n > 0\}$.

   (b) $\{0^n10^n \mid n > 0\}$ is a context-free language.

   (c) The context-free grammar $S \to 00S \mid S11 \mid 01$ generates the language 0^n1^n.

   (d) Any language that can be decided by an NFA with ε-transitions can also be decided by an NFA without ε-transitions.

   (e) For any string $w \in (0+1)^*$, let w^C denote the string obtained by flipping every 0 in w to 1, and every 1 in w to 0.
   If L is a regular language over the alphabet $\{0,1\}$, then $\{ww^C \mid w \in L\}$ is also regular.

   (f) For any string $w \in (0+1)^*$, let w^C denote the string obtained by flipping every 0 in w to 1, and every 1 in w to 0.
   If L is a regular language over the alphabet $\{0,1\}$, then $\{xy^C \mid x, y \in L\}$ is also regular.

   (g) The ε-reach of any state in an NFA contains the state itself.

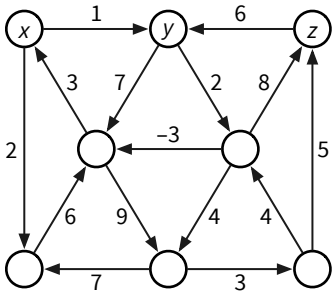
   (h) Let L_1, L_2 be two regular languages. The language $(L_1 + L_2)^*$ is also regular.

   (i) The regular expression $(00+11)^*$ represents the language of all strings over $\{0,1\}$ of even length.

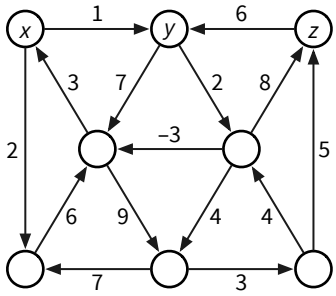
   (j) The language $\{0^{2p} \mid p \text{ is prime}\}$ is regular.

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 2 Problem 1	Name:

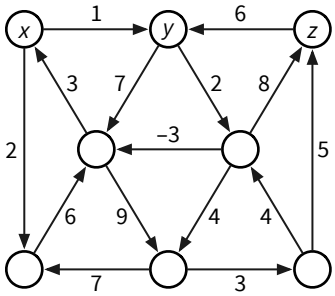
Clearly indicate the following structures in the directed graph below, or write NONE if the indicated structure does not exist. Don't be subtle; to indicate a collection of edges, draw a heavy black line along the entire length of each edge.



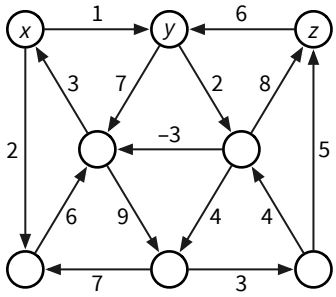
(a) A depth-first tree rooted at x.



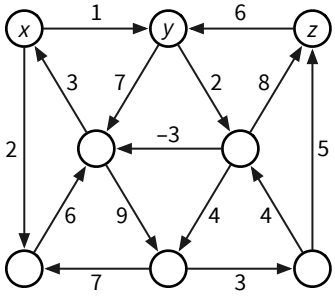
(b) A breadth-first tree rooted at y.



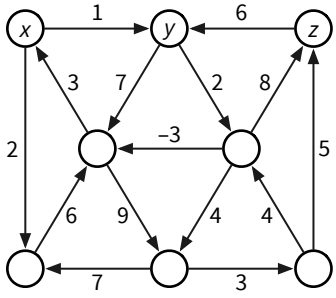
(c) A shortest-path tree rooted at z.



(d) The shortest directed cycle.



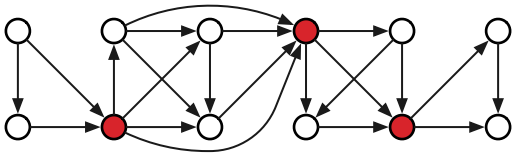
[scratch]



[scratch]

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 2 Problem 2	Name:

A vertex v in a (weakly) connected graph G is called a ***cut vertex*** if the subgraph $G - v$ is disconnected. For example, the following graph has three cut vertices, which are shaded in the figure.



Suppose you are given a (weakly) connected *dag* G with one source and one sink. Describe and analyze an algorithm that returns TRUE if G has a cut vertex and FALSE otherwise.

CS/ECE 374 A ✧ Fall 2021 **Fake Midterm 2 Problem 3**	Name:

You decide to take your next hiking trip in Jellystone National Park. You have a map of the park's trails that lists all the scenic views in the park, but also warns that certain trail segments have a high risk of bear encounters. To make the hike worthwhile, you want to see at least three scenic views. You also don't want to get eaten by a bear, so you are willing to hike along at most one high-bear-risk segment. Because the trails are narrow, each trail segment allows traffic in only one direction.

Your friend has converted the map into a directed graph $G = (V, E)$, where V is the set of intersections and E is the set of trail segments. A subset S of the edges are marked as *Scenic*; another subset B of the edges are marked as *high-Bear-risk*. You may assume that $S \cap B = \emptyset$. Each segment $e \in E$ is also labeled with a positive length $\ell(e)$ in miles. Your campsite appears on the map as a particular vertex $s \in V$, and the visitor center is another vertex $t \in V$.

Describe and analyze an algorithm to compute the shortest hike from your campsite s to the visitor center t that includes *at least* three scenic trail segments and *at most* one high-bear-risk trail segment. You may assume such a hike exists.

CS/ECE 374 A ✧ Fall 2021 **Fake Midterm 2 Problem 4**	Name:

During a family reunion over Thanksgiving break, your ultra-competitive thirteen-year-old nephew Elmo challenges you to a card game. At the beginning of the game, Elmo deals a long row of cards. Each card shows a number of points, which could be positive, negative, or zero. After the cards are dealt, you and Elmo alternate taking either the leftmost card or the rightmost card from the row, until all the cards are gone. The player that collects the most points is the winner.

For example, is the initial card values are $[4, 6, 1, 2]$, the game might proceed as follows:

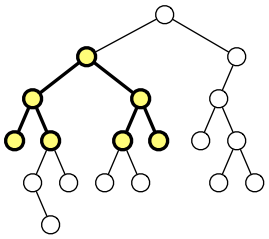
- You take the 4 on the left, leaving $[6, 1, 2]$.
- Elmo takes the 6 on the left, leaving $[1, 2]$.
- You take the 2 on the right, leaving $[1]$.
- Elmo takes the last 1, ending the game.
- You took $4 + 2 = 6$ points, and Elmo took $6 + 1 = 7$ points, so Elmo wins!

Describe and analyze an algorithm to determine, given the initial sequence of cards, the maximum number of points that you can collect playing against a *perfect* opponent. Assume that Elmo generously lets you move first.

CS/ECE 374 A ✧ Fall 2021 Fake Midterm 2 Problem 5	Name:

For this problem, a *subtree* of a binary tree means any connected subgraph. A binary tree is *complete* if every internal node has two children, and every leaf has exactly the same depth.

Describe and analyze a recursive algorithm to compute the ***largest complete subtree*** of a given binary tree. Your algorithm should return both the root and the depth of this subtree. For example, given the following tree T as input, your algorithm should return the left child of the root of T and the integer 2.



CS/ECE 374 A ✧ Fall 2021

Midterm 2

November 8, 2021

Directions

- ***Don't panic!***
- If you brought anything except your writing implements, your hand-written double-sided 8½" × 11" cheat sheet, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- **We *strongly* recommend reading the entire exam before trying to solve anything.** If you think a question is unclear or ambiguous, please ask for clarification as soon as possible.
- The exam has five numbered questions, each worth 10 points. (Subproblems are not necessarily worth the same number of points.)
- Write your answers on blank white paper using a dark pen. Please start your solution to each numbered question on a new sheet of paper.
- You have **120 minutes** to write your solutions, after which you have 30 minutes to scan your solutions, convert your scan to a PDF file, and upload your PDF file to Gradescope.
- If you are ready to scan your solutions before 9:00pm, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Gradescope will only accept PDF submissions. Please do not scan your cheat sheets or scratch paper. Please make sure your solution to each numbered problem starts on a new page of your PDF file. **Low-quality scans will be penalized.**
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam **as a PDF file** in your email. If you are in the middle of the exam, send Jeff email, continue working until the time limit, and then send a second email with your completed exam **as a PDF file**. Please do not email raw photos.

---

1. Short answers:

   (a) Solve the recurrence $T(n) = 2T(n/3) + O(\sqrt{n})$.

   (b) Solve the recurrence $T(n) = 2T(n/7) + O(\sqrt{n})$.

   (c) Solve the recurrence $T(n) = 2T(n/4) + O(\sqrt{n})$.

   (d) Draw a connected undirected graph G with at most ten vertices, such that every vertex has degree at least 2, and no spanning tree of G is a path.

   (e) Draw a directed acyclic graph with at most ten vertices, exactly one source, exactly one sink, and more than one topological order.

   (f) Describe an appropriate memoization structure and evaluation order for the following (meaningless) recurrence, and give the running time of the resulting iterative algorithm to compute $Pibby(1, n)$. (Assume all array accesses are legal.)

$$Pibby(i,k) = \begin{cases} 0 & \text{if } i > k \\ A[i] & \text{if } i = k \\ Pibby(i+1, k-1) + A[i] + A[k] & \text{if } A[i] = A[k] \\ \max \left\{ \begin{array}{c} Pibby(i+2,k), \\ Pibby(i+1,k-1), \\ Pibby(i,k-2) \end{array} \right\} & \text{otherwise} \end{cases}$$

2. Your company has two offices, one in San Francisco and the other in New York. Each week you decide whether you want to work in the San Francisco office or in the New York office. Depending on the week, your company makes more money by having you work at one office or the other. You are given a schedule of the profits you can earn at each office for the next n weeks. You'd obviously prefer to work each week in the location with higher profit, but there's a catch: Flying from one city to the other costs \$1000. Your task is to design a travel schedule for the next n weeks that yields the maximum *total* profit, assuming you start in San Francisco.

   For example: suppose you are given the following schedule:

SF	\$800	\$200	\$500	\$400	\$1200
NY	\$300	\$900	\$700	\$2000	\$200

   If you spend the first week in San Francisco, the next three weeks in New York, and the last week in San Francisco, your total profit for those five weeks is $\$800 - \$1000 + \$900 + \$700 + \$2000 - \$1000 + \$1200 = \$3600$.

   (a) ***Prove*** that the obvious greedy strategy (each week, fly to the city with more profit) does not always yield the maximum total profit.

   (b) Describe and analyze an algorithm to compute the maximum total profit you can earn, assuming you start in San Francisco. The input to your algorithm is a pair of arrays $NY[1..n]$ and $SF[1..n]$, containing the profits in each city for each week.

3. Suppose you are given a directed graph $G = (V, E)$, whose vertices are either red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. The ***remoteness*** of a vertex v is the maximum of three shortest-path lengths:

   - The length of a shortest path to v from the closest red vertex
   - The length of a shortest path to v from the closest blue vertex
   - The length of a shortest path to v from the closest green vertex

   In particular, if v is not reachable from vertices of all three colors, then v is infinitely remote.

   Describe and analyze an algorithm to find a vertex of G whose remoteness is *smallest*.

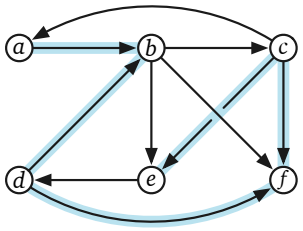
4. Suppose you are given an array $A[1..n]$ of integers such that $A[i] + A[i+1]$ is even for *exactly one* index i. In other words, the elements of A alternate between even and odd, except for exactly one adjacent pair that are either both even or both odd.

   Describe and analyze an efficient algorithm to find the unique index i such that $A[i] + A[i+1]$ is even. For example, given the following array as input, your algorithm should return the integer 6, because $A[6] + A[7] = 88 + 62$ is even. (Cells containing even integers are shaded blue.)

17	40	23	72	39	88	62	13	40	53	92	21	10	73	68

5. A *zigzag walk* in in a directed graph G is a sequence of vertices connected by edges in G, but the edges alternately point forward and backward along the sequence. Specifically, the first edge points forward, the second edge points backward, and so on. The *length* of a zigzag walk is the sum of the weights of its edges, both forward and backward.

   For example, the following graph contains the zigzag walk $a \to b \leftarrow d \to f \leftarrow c \to e$. Assuming every edge in the graph has weight 1, this zigzag walk has length 5.



   Suppose you are given a directed graph G with non-negatively weighted edges, along with two vertices s and t. Describe and analyze an algorithm to find the shortest zigzag walk from s to t in G.

CS/ECE 374 A ✧ Fall 2021

Conflict Midterm 2

November 9, 2021

Directions

- ***Don't panic!***
- If you brought anything except your writing implements, your hand-written double-sided 8½" × 11" cheat sheet, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- **We *strongly* recommend reading the entire exam before trying to solve anything.** If you think a question is unclear or ambiguous, please ask for clarification as soon as possible.
- The exam has five numbered questions, each worth 10 points. (Subproblems are not necessarily worth the same number of points.)
- Write your answers on blank white paper using a dark pen. Please start your solution to each numbered question on a new sheet of paper.
- You have **120 minutes** to write your solutions, after which you have 30 minutes to scan your solutions, convert your scan to a PDF file, and upload your PDF file to Gradescope.
- If you are ready to scan your solutions before 9:00pm, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Gradescope will only accept PDF submissions. Please do not scan your cheat sheets or scratch paper. Please make sure your solution to each numbered problem starts on a new page of your PDF file. **Low-quality scans will be penalized.**
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam **as a PDF file** in your email. If you are in the middle of the exam, send Jeff email, continue working until the time limit, and then send a second email with your completed exam **as a PDF file**. Please do not email raw photos.

---

1. Short answers:

   (a) Solve the recurrence $T(n) = 3T(n/2) + O(n^2)$.

   (b) Solve the recurrence $T(n) = 7T(n/2) + O(n^2)$.

   (c) Solve the recurrence $T(n) = 4T(n/2) + O(n^2)$.

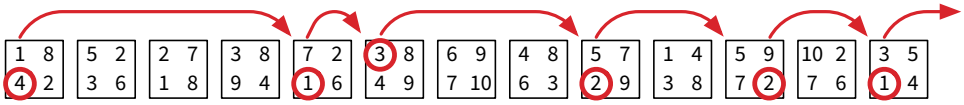
   (d) Draw a directed acyclic graph with at most ten vertices, exactly one source, exactly one sink, and more than one topological order.

   (e) Draw a directed graph with at most ten vertices, with distinct edge weights, that has more than one shortest path from some vertex s to some other vertex t.

   (f) Describe an appropriate memoization structure and evaluation order for the following (meaningless) recurrence, and give the running time of the resulting iterative algorithm to compute $Huh(1, n)$.

$$Huh(i,k) = \begin{cases} 0 & \text{if } i > n \text{ or } k < 0 \\ \min\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} + A[i,k] & \text{if } A[i,k] \text{ is even} \\ \max\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} - A[i,k] & \text{if } A[i,k] \text{ is odd} \end{cases}$$

2. *Quadhopper* is a solitaire game played on a row of n squares. Each square contains four positive integers. The player begins by placing a token on the leftmost square. On each move, the player chooses one of the numbers on the token's current square, and then moves the token that number of squares to the right. The game ends when the token moves past the rightmost square. The object of the game is to make as many moves as possible before the game ends.



A quadhopper puzzle that allows six moves. (This is ***not*** the longest legal sequence of moves.)

   (a) ***Prove*** that the obvious greedy strategy (always choose the smallest number) does not give the largest possible number of moves for every quadhopper puzzle.

   (b) Describe and analyze an efficient algorithm to find the largest possible number of legal moves for a given quadhopper puzzle.

3. Suppose you are given a directed graph $G = (V, E)$, each of whose vertices is either red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag.

   Describe and analyze an algorithm to find a shortest path in G that contains at least one vertex of each color. (In particular, your algorithm must choose the best start and end vertices for the path.)

4. Your grandmother dies and leaves you her treasured collection of n radioactive Beanie Babies. Her will reveals that one of the Beanie Babies is a rare specimen worth 374 million dollars, but all the others are worthless. All of the Beanie Babies are equally radioactive, except for the valuable Beanie Baby, which is is either slightly more or slightly less radioactive, but you don't know which. Otherwise, as far as you can tell, the Beanie Babies are all identical.

   You have access to a state-of-the-art Radiation Comparator at your job. The Comparator has two chambers. You can place any two disjoint sets of Beanie Babies in Comparator's two chambers; the Comparator will then indicate which subset emits more radiation, or that the two subsets are equally radioactive. (Two subsets are equally radioactive if and only if they contain the same number of Beanie Babies, and they are all worthless.) The Comparator is slow and consumes a *lot* of power, and you really aren't supposed to use it for personal projects, so you *really* want to use it as few times as possible.

   Describe an efficient algorithm to identify the valuable Beanie Baby. How many times does your algorithm use the Comparator in the worst case, as a function of n?

5. Ronnie and Hyde are a professional robber duo who plan to rob a house in the Leverwood neighborhood of Sham-Poobanana. They have managed to obtain a map of the neighborhood in the form of a directed graph G, whose vertices represent houses, whose edges represent one-way streets.

   - One vertex s represents the house that Ronnie and Hyde plan to rob.
   - A set X of special vertices designate eXits from the neighborhood.
   - Each directed edge $u \to v$ has a non-negative weight $w(u \to v)$, indicating the time required to drive directly from house u to house v.
   - Thanks to Leverwood's extensive network of traffic cameras, speeding or driving backwards along any one-way street would mean certain capture.

   Describe and analyze an algorithm to compute the shortest time needed to exit the neighborhood, starting at house s. The input to your algorithm is the directed graph $G = (V, E)$, with clearly marked subset of exit vertices $X \subseteq V$, and non-negative weights $w(u \to v)$ for every edge $u \to v$.

CS/ECE 374 A ✧ Fall 2021

☙ Final Exam ❧

December 15, 2021

☙ Directions ❧

- ***Don't panic!***
- If you brought anything except your writing implements, your two hand-written double-sided 8½" × 11" cheat sheets, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- **We *strongly* recommend reading the entire exam before trying to solve anything.** If you think a question is unclear or ambiguous, please ask for clarification as soon as possible.
- The exam has six numbered questions, each worth 10 points. (Subproblems are not necessarily worth the same number of points.)
- You have **150 minutes** to write your solutions, after which you have 30 minutes to scan your solutions, convert your scan to a PDF file, and upload your PDF file to Gradescope. (Both of these times are extended if you have time accommodations through DRES.)
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Write your answers on blank white paper using a dark pen. Please start your solution to each numbered question on a new sheet of paper.
- If you are ready to scan your solutions and there are more than 15 minutes of writing time remaining, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Gradescope will only accept PDF submissions. Please do not scan your cheat sheets or scratch paper. Please make sure your solution to each numbered problem starts on a new page of your PDF file.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam **as a PDF file** in your email. If you are in the middle of the exam, send Jeff email, continue working until the time limit, and then send a second email with your completed exam **as a PDF file**. Please do not email raw photos.

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 $S_1, S_2, \ldots, S_m$ of a set S, what is the size of the smallest subcollection whose union is S?

MinHittingSet: Given a collection of subsets $S_1, S_2, \ldots, S_m$ 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?

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 $c \in Z^d$, 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?

SteamedHams: Aurora borealis? At this time of year, at this time of day, in this part of the country, localized entirely within your kitchen? May I see it?

1. For each statement below, write "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, write "NO". For example:

   - $x + y = 5$
     NO — Suppose $x = 3$ and $y = 4$.
   - 3SAT can be solved in polynomial time.
     NO — 3SAT is NP-hard.
   - If P = NP then Jeff is the Queen of England.
     YES — The hypothesis is false, so the implication is true.

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

Which of the following statements are true?

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

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

(c) Every directed acyclic graph contains at least one sink.

(d) Given *any* undirected graph G, we can compute a spanning tree of G in $O(V + E)$ time using whatever-first search.

(e) Suppose we want to iteratively evaluate the following recurrence:

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

We can fill the array $What[0..n, 0..n]$ in $O(n^2)$ time, by decreasing i in the outer loop and decreasing j in the inner loop.

Which of the following statements are true for ***at least one*** language $L \subseteq \{0, 1\}^*$?

(f) $L^* = (L^*)^*$

(g) L is decidable, but L^* is undecidable.

(h) L is neither regular nor NP-hard.

(i) L is in P, and L has an infinite fooling set.

(j) The language $\{\langle M \rangle \mid M \text{ accepts } L\}$ is undecidable.

---

2. For each statement below, write "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, write "NO".

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

   (Please remember to start your answers to this problem on a new page. Yes, this is really just a continuation of problem 1; we split it into two problems to make grading easier.)

Consider the following pair of languages:

- ACYCLIC := $\{\text{undirected graph } G \mid G \text{ contains no cycles}\}$
- HALFIND := $\{\text{undirected graph } G = (V, E) \mid G \text{ has an independent set of size } |V|/2\}$

(For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.) The language HALFIND is actually NP-hard; **you do *not* need to prove that fact**.

Which of the following statements are true, assuming P $\neq$ NP?

(a) ACYCLIC is NP-hard.

(b) HALFIND $\setminus$ ACYCLIC $\in$ P
(Recall that $X \setminus Y$ is the subset of elements of X that are not in Y.)

(c) HALFIND is decidable.

(d) A polynomial-time reduction from HALFIND to ACYCLIC would imply P=NP.

(e) A polynomial-time reduction from ACYCLIC to HALFIND would imply P=NP.

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

(f) A is a subset of B.

(g) If $B \in P$, then $A \in P$.

(h) If B is NP-hard, then A is NP-hard.

(i) If B is decidable, then A is decidable.

(j) If B is regular, then A is decidable.

---

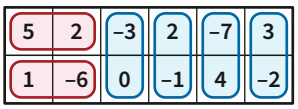
3. Suppose you are asked to tile a $2 \times n$ grid of squares with dominos (1×2 rectangles). Each domino must cover exactly two grid squares, either horizontally or vertically, and each grid square must be covered by exactly one domino.

   Each grid square is worth some number of points, which could be positive, negative, or zero. The ***value*** of a domino tiling is the sum of the points in squares covered by vertical dominos, *minus* the sum of the points in squares covered by horizontal dominos.

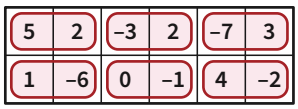
   Describe and analyze an efficient algorithm to compute the largest possible value of a domino tiling of a given $2 \times n$ grid. Your input is an array $Points[1..2, 1..n]$ of point values.

   As an example, here are three domino tilings of the same 2×6 grid, along with their values. The third tiling is optimal; no other tiling of this grid has larger value. Thus, given this 2×6 grid as input, your algorithm should return the integer 16.

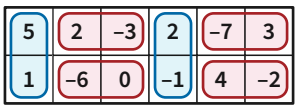
5	2	−3	2	−7	3
1	−6	0	−1	4	−2



value = −6



value = 2



value = 16

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

   (a) Let Φ be a boolean formula in conjunctive normal form, with exactly three literals per clause (or in other words, an instance of 3SAT). ***Prove*** that it is NP-hard to decide whether Φ has a satisfying assignment in which *exactly half* of the variables are TRUE.

   (b) Let $G = (V, E)$ be an arbitrary undirected graph. Recall that a *proper 3-coloring* of G assigns each vertex of G one of three colors—red, blue, or green—so that every edge in G has endpoints with different colors. ***Prove*** that it is NP-hard to decide whether G has a proper 3-coloring in which *exactly half* of the vertices are red.

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

5. Suppose you are given a height map of a mountain, in the form of an $n \times n$ grid of evenly spaced points, each labeled with an elevation value. You can safely hike directly from any point to any neighbor immediately north, south, east, or west, but only if the elevations of those two points differ by at most Δ. (The value of Δ depends on your hiking experience and your physical condition.)

   Describe and analyze an algorithm to determine the longest hike from some point s to some other point t, where the hike consists of an uphill climb (where elevations must increase at each step) followed by a downhill climb (where elevations must decrease at each step). Your input consists of an array $Elevation[1..n, 1..n]$ of elevation values, the starting point s, the target point t, and the parameter Δ.

6. 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:

$$\texttt{00011111110000} = \texttt{000} \bullet \texttt{1111111} \bullet \texttt{0000}$$

   (a) Let L_a denote the set of all strings in $\{0,1\}^*$ where every `0` is followed immediately by at least one `1`.

   For example, L_a contains the strings `010111` and `1111` and the empty string ε, but does not contain either `001100` or `1111110`.

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

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

   (b) Let L_b denote the set of all strings in $\{0,1\}^*$ whose run lengths are increasing; that is, every run except the last is followed immediately by a *longer* run.

   For example, L_b contains the strings `0110001111` and `1100000` and `000` and the empty string ε, but does not contain either `000111` or `100011`.

   Prove that L_b is not a regular language.

CS/ECE 374 A ✧ Fall 2021
ᔕ Conflict Final Exam ᔓ

ᔕ Directions ᔓ

- ***Don't panic!***
- If you brought anything except your writing implements, your two hand-written double-sided 8½" × 11" cheat sheets, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- **We *strongly* recommend reading the entire exam before trying to solve anything.** If you think a question is unclear or ambiguous, please ask for clarification as soon as possible.
- The exam has six numbered questions, each worth 10 points. (Subproblems are not necessarily worth the same number of points.)
- You have **150 minutes** to write your solutions, after which you have 30 minutes to scan your solutions, convert your scan to a PDF file, and upload your PDF file to Gradescope. (Both of these times are extended if you have time accommodations through DRES.)
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Write your answers on blank white paper using a dark pen. Please start your solution to each numbered question on a new sheet of paper.
- If you are ready to scan your solutions and there are more than 15 minutes of writing time, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Gradescope will only accept PDF submissions. Please do not scan your cheat sheets or scratch paper. Please make sure your solution to each numbered problem starts on a new page of your PDF file.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam **as a PDF file** in your email. If you are in the middle of the exam, send Jeff email, continue working until the time limit, and then send a second email with your completed exam **as a PDF file**. Please do not email raw photos.

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 $S_1, S_2, \ldots, S_m$ of a set S, what is the size of the smallest subcollection whose union is S?

MinHittingSet: Given a collection of subsets $S_1, S_2, \ldots, S_m$ 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?

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 $c \in Z^d$, 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?

SteamedHams: Aurora borealis? At this time of year, at this time of day, in this part of the country, localized entirely within your kitchen? May I see it?

1. For each statement below, write "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, write "NO". For example:

   - $x + y = 5$
     NO — Suppose $x = 3$ and $y = 4$.
   - 3SAT can be solved in polynomial time.
     NO — 3SAT is NP-hard.
   - If P = NP then Jeff is the Queen of England.
     YES — The hypothesis is false, so the implication is true.

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

Which of the following statements are true?

(a) The solution to the recurrence $T(n) = \mathbf{2}T(n/\mathbf{4}) + O(n^2)$ is $T(n) = O(n^2)$.

(b) The solution to the recurrence $T(n) = \mathbf{4}T(n/\mathbf{2}) + O(n^2)$ is $T(n) = O(n^2)$.

(c) For every directed graph G, if G has at least one source, then G has at least one sink.

(d) Given *any* undirected graph G, we can compute a spanning tree of G in $O(V + E)$ time using whatever-first search.

(e) Suppose we want to iteratively evaluate the following recurrence:

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

We can fill the array $What[0..n, 0..n]$ in $O(n^2)$ time, by decreasing i in the outer loop and decreasing j in the inner loop.

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

(f) $L^* = (L^*)^*$

(g) If L is decidable, then L^* is decidable.

(h) L is either regular or NP-hard.

(i) If L is undecidable, then L has an infinite fooling set.

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

---

2. For each statement below, write "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, write "NO".

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

   (Please remember to start your answers to this problem on a new page. Yes, this is really just a continuation of problem 1; we split it into two problems to make grading easier.)

Consider the following pair of languages:

- DIRHAMPATH $:= \{G \mid G \text{ is a directed graph with a Hamiltonian path}\}$
- ACYCLIC $:= \{G \mid G \text{ is a directed acyclic graph}\}$

(For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.) Which of the following statements are true, assuming P $\neq$ NP?

(a) ACYCLIC $\in$ NP

(b) ACYCLIC $\cap$ DIRHAMPATH $\in$ P

(c) DIRHAMPATH is decidable.

(d) A polynomial-time reduction from DIRHAMPATH to ACYCLIC would imply P=NP.

(e) A polynomial-time reduction from ACYCLIC to DIRHAMPATH would imply P=NP.

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?

(f) $A \subseteq B$.

(g) 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.

(h) If A is NP-hard then B is NP-hard.

(i) If A is decidable then B is decidable.

(j) If a Turing machine M accepts B, the same Turing machine M also accepts A.

---

3. 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 to the rightmost squares.

   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 $B[1..n]$.

4. 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 less than half the vertices of S have a neighbor that is also in S. ***Prove*** that finding the largest mostly independent set in G is NP-hard.

   (b) Let $G = (V, E)$ be an arbitrary directed graph with colored edges. A *rainbow Hamiltonian cycle* in G is a cycle that visits every vertex of G exactly one, in which no pair of consecutive edges have the same color. ***Prove*** that it is NP-hard to decide whether G has a rainbow Hamiltonian cycle.

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

5. Suppose we are given an n-digit integer X. Repeatedly remove one digit from either end of X (your choice) until no digits are left. The *square-depth* of X is the maximum number of perfect squares that you can see during this process. For example, the number 32492 has square-depth 3, by the following sequence of removals:

$$32492 \to 3249 \overset{57^2}{\to} 324 \to 24 \overset{18^2}{\to} 4 \to \varepsilon \overset{2^2}{}.$$

Describe and analyze an algorithm to compute the square-depth of a given integer X, represented as an array $X[1..n]$ of n decimal digits. Assume you have access to a subroutine IsSquare that determines whether a given k-digit number (represented by an array of digits) is a perfect square ***in $O(k^2)$ time***.

6. 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:

$$\texttt{00011111110000} = \texttt{000} \bullet \texttt{1111111} \bullet \texttt{0000}$$

(a) Let L_a denote the set of all strings in $\{0,1\}^*$ in which every run of 1s has even length and every run of 0s has odd length.

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

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

(b) Let L_b denote the set of all strings in $\{0,1\}^*$ in which every run of 0s is immediately followed by a *longer* run of 1s. ***Prove*** that L_b is *not* a regular language.

Both of these languages contain the strings `0111100011` and `110001111` and `111111` and the empty string ε, but neither language contains `000111` or `100011` or `0000`.