Write your answers in the separate answer booklet.

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

1. 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* ≠ *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 for ***every*** language $L \subseteq \{0, 1\}^*$?

   - $(L^*)^*$ is infinite.
   - If L is decidable then its complement $\overline{L}$ is undecidable.
   - $\{\langle M \rangle \mid M \text{ accepts } L\}$ is undecidable.
   - Either L is finite or L is NP-hard.
   - Either L has an infinite fooling set or $L \in P$.

   (b) Consider the following pair of languages:

   - Tree = $\{G \mid G$ is a connected undirected graph with no cycles$\}$
   - HamPath = $\{G \mid G$ is an undirected graph that contains a Hamiltonian path$\}$

   (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 ≠ NP?

   - Tree is NP-hard.
   - Tree ∩ HamPath is NP-hard.
   - Tree ∪ HamPath is NP-hard.
   - HamPath is undecidable.
   - A reduction from Tree to HamPath would imply P = NP.

Problems 2–7 appear on the next three pages.

2. **More of the same:** Follow exactly the same instructions as problem 1.

(a) Which of the following statements are true?

- The recurrence $T(n) = 3T(n/3) + O(n^2)$ implies $T(n) = O(n^2 \log n)$.
- The recurrence $T(n) = T(n/2) + T(n/3) + T(n/6) + O(n)$ implies $T(n) = O(n)$.
- There is a forest with 374 vertices and 225 edges. (Recall that a *forest* is an undirected graph with no cycles.)
- Given any directed graph G whose edges have positive weights, we can compute shortest paths from one vertex s to every other vertex of G in $O(VE)$ time using Bellman-Ford.
- Suppose $A[1..n]$ is an array of integers. Consider the following recursive function:

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

  We can compute $Ohio(n)$ by memoizing this function into a ~~two-dimensional~~ one array $Ohio[1..n]$, which we fill by increasing i in $O(n)$ time.

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

$$\textsc{Chalmers} := \left\{\langle M\rangle \;\middle|\; M \text{ accepts both } \texttt{STEAMED} \text{ and } \texttt{HAMS}\right\}$$

Professor Skinner suggests a reduction from the standard halting language

$$\textsc{Halt} := \left\{(\langle M\rangle, w) \;\middle|\; M \text{ halts on input}\cancel{\text{s}}\ w\right\}.$$

Specifically, Professor Skinner claims that if there is a Turing machine SuperNintendo that decides the language Chalmers, then the following algorithm decides Halt.

```
DecideHalt($(\langle M\rangle, w)$):
    Encode the following Turing machine:
        AuroraBorealis($x$):
            if $x$ = STEAMED or $x$ = HAMS or $x$ = UTICA
                run $M$ on input $w$
                return False
            else
                return True
    return SuperNintendo($\langle$AuroraBorealis$\rangle$)
```

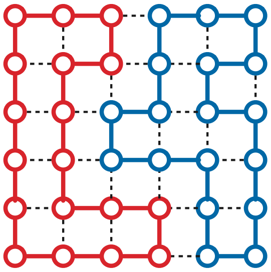
Which of the following statements ~~is~~ are true for all inputs $(\langle M\rangle, w)$?

- If M hangs on w, then AuroraBorealis accepts ALBANY.
- If M accepts w, then SuperNintendo accepts $\langle$AuroraBorealis$\rangle$.
- If M hangs on w, then DecideHalt rejects $(\langle M\rangle, w)$.
- DecideHalt decides the language Halt. (That is, Professor Skinner's reduction is correct.)
- We could have proved that Chalmers is undecidable using Rice's theorem instead of this reduction.

Problems 3–7 appear on the next two pages.

3. Submit a solution to ***exactly one*** of the following problems.

   (a) A *Hamiltonian bicycle* in a graph G is a pair of simple cycles in G, with identical lengths, such that every vertex of G lies on exactly one of the two cycles.



A Hamiltonian bicycle in the 6×6 grid graph.

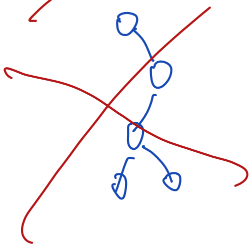
   Prove that it is NP-hard to determine whether a given graph G has a Hamiltonian bicycle.

   (b) A *clique-partition* of a graph $G = (V, E)$ is a partition of the vertices V into disjoint subsets $V_1 \cup V_2 \cup \cdots \cup V_k$ such that for each index i, every pair of vertices in subset V_i is connected by an edge in G. The *size* of a clique partition is the number of subsets V_i. $= k$

   Prove that it is NP-hard to compute the minimum-size clique partition of a given undirected graph G.

In fact, both of these problems are NP-hard, but we only want a proof for one of them. Don't forget to tell us which problem you've chosen!

recursion



4. let T be a *full* binary tree, meaning that every node has either two children or no children.

   - Recall that the *height* of a vertex v in T is the length of the longest path in T from v down to a leaf. In particular, every leaf of T has height zero.
   - A vertex v is *AVL-balanced* if v is a leaf, or if the heights of v's children differ by at most 1. (You might recall from CS 225 that an ***AVL-tree*** is a binary search tree in which *every* vertex is AVL-balanced.)

   Describe and analyze an algorithm to compute the number of AVL-balanced vertices in T.

5. Suppose we are given a directed graph $G = (V, E)$, where every edge $e \in E$ has a *positive* weight $w(e)$, along with two vertices s and t.

   (a) Suppose each *vertex* of G is colored either orange, green, or purple. Describe and analyze an algorithm to find the shortest walk from s to t in G that never visits two consecutive *vertices* with the same color.

Dijkstra

   (b) Now suppose each *edge* of G is colored either orange, green, or purple. Describe and analyze an algorithm to find the shortest walk from s to t in G that never traverses two consecutive *edges* with the same color.

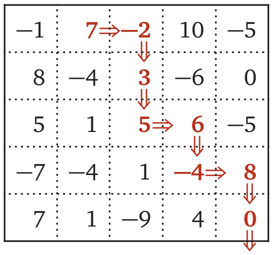
Problems 5 and 6 appear on the next page.

Kevin Milans

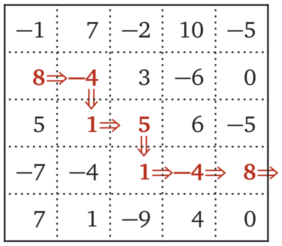
6. *Vankin's Mile* is a solitaire game played on an $n \times n$ square grid. The player starts by placing a token on any square of the grid. Then on each turn, the player moves the token either one square to the right or one square down. The game ends when player moves the token off the edge of the board. Each square of the grid has a numerical value, which could be positive, negative, or zero. The player starts with a score of zero; whenever the token lands on a square, the player adds its value to his score. The object of the game is to score as many points as possible.

   For example, given the grid below, we can score $7-2+3+5+6-4+8+0=23$ points by following the path on the left, or we can score $8-4+1+5+1-4+8=15$ points by following the path on the right.

DP/ ~~graph~~ dag



−1	**7**⇒	**−2** ⇓	10	−5
8	−4	**3** ⇓	−6	0
5	1	**5**⇒	**6** ⇓	−5
−7	−4	1	**−4**⇒	**8** ⇓
7	1	−9	4	**0** ⇓



−1	7	−2	10	−5
8⇒	**−4** ⇓	3	−6	0
5	**1**⇒	**5** ⇓	6	−5
−7	−4	**1**⇒	**−4**⇒	**8**⇒
7	1	−9	4	0

   (a) Describe and analyze an efficient algorithm to compute the maximum possible score for a game of Vankin's Mile, given the $n \times n$ array of values as input.

   (b) A variant called *Vankin's Niknav* adds an additional constraint to Vankin's Mile: *The sequence of values that the token touches must be a* ***palindrome***. Thus, the example path on the right is valid, but the example path on the left is not. Describe and analyze an efficient algorithm to compute the maximum possible score for an instance of Vankin's Niknav, given the $n \times n$ array of values as input.

7. (a) Let L_a denote the set of all strings $w \in \{0,1,2\}^*$ such that $\#(1,w) + 2 \cdot \#(2,w)$ is divisible by 3. For example, L_a contains the strings 0012 and 20210202 and the empty string ε, but L_a does not include the strings 121 or 0122210.

   Σ digits!

   Describe a DFA ~~of~~ or NFA that accepts L_a. (You do not need to prove that your answer is correct.)

   (b) Let L_b denote the set of all strings $w \in \{0,1,2\}^*$ such that no two symbols appear the same number of times, or in other words, the integers $\#(0,w)$ and $\#(1,w)$ and $\#(2,w)$ are all different. For example, L_b contains the strings 110212 and 20220, but L_b does not include the strings 01212 or 2120210 or the empty string ε.

   Prove that L_b is not a regular language.

Fooling set?

CS/ECE 374 A ✧ Fall 2025

Practice Final Exam 3

December 10, 2025

Name:	Jeff E
NetID:	jeffe

- ***Don't panic!***

- You have 180 minutes to answer seven questions. The questions are described in more detail in a separate handout.

- If you brought anything except your writing implements, your two **hand-written** double-sided 8½" × 11" cheat sheets, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.

- Please clearly print your name and your NetID in the boxes above.

- Please also print your name at the top of every page of the answer booklet, except this cover page. We want to make sure that if a staple falls out, we can reassemble your answer booklet. (It doesn't happen often, but it does happen.)

- Greedy algorithms require formal proofs of correctness to receive any credit, even if they are correct. Otherwise, proofs or other justifications 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.

CS/ECE 374 A ✧ Fall 2025 **Practice Final 3 Problem 1**	Name:

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

- $(L^*)^*$ is infinite.

  ☐ Yes ☒ No $L = \emptyset \rightarrow L^* = \{\varepsilon\} \longrightarrow (L^*)^* = \{\varepsilon\}^* = \{\varepsilon\}$

- If L is decidable then its complement $\overline{L}$ is undecidable.

  ☐ Yes ☒ No Decide$\overline{L}$(w): return not(Decide$L$(w))

- $\{\langle M \rangle \mid M \text{ accepts } L\}$ is undecidable. Rice's Theorem technicality:

  ☐ Yes ☒ No L could be unacceptable $\Rightarrow \{\langle M \rangle \mid M \text{accepts} L\} = \emptyset$

- Either L is finite or L is NP-hard.

  ☐ Yes ☒ No $L = 0^*$

- Either L has an infinite fooling set or $L \in P$.

  ☒ Yes ☐ No No infinite fooling set $\rightleftarrows$ regular $\longrightarrow P$

Problem 1 continues onto the next page.

(b) Consider the following pair of languages:

- TREE = $\{G \mid G$ is a connected undirected graph with no cycles$\}$
- HAMPATH = $\{G \mid G$ is an undirected graph that contains a Hamiltonian path$\}$

(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 ≠ NP?

- TREE is NP-hard.

  ☐ Yes ☒ No We ean use WFS to decide if G is a tree. in O(V+E) time

- TREE ∩ HAMPATH is NP-hard.

  ☐ Yes ☒ No L = PATH!

- TREE ∪ HAMPATH is NP-hard. Follows from previous question

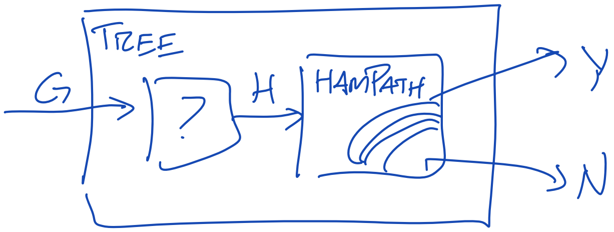
  ☒ Yes ☐ No reduction proving HamPath hard outputs nontrees.

- HAMPATH is undecidable.

  ☐ Yes ☒ No try all n! permutations

- A reduction from TREE to HAMPATH would imply P = NP.

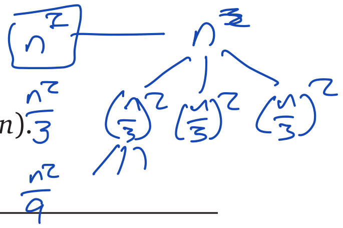
  ☐ Yes ☒ No wrong way / you didn't say poly-time



CS/ECE 374 A ✧ Fall 2025 **Practice Final 3 Problem 2**	Name:

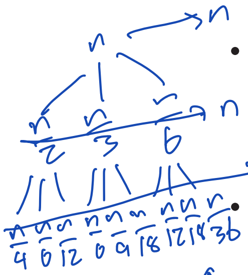
For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume *P* ≠ *NP*.** If there is any other ambiguity or uncertainty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

(a) Which of the following statements are true?



- The solution to the recurrence $T(n) = 3T(n/3) + O(n^2)$ is $T(n) = O(n^2 \log n)$.

  □ Yes ☒ No T(n)=O(n²)



n
n
$\frac{n}{2}$ $\frac{n}{3}$ $\frac{n}{6}$
n
$\frac{n}{4}$ $\frac{n}{6}$ $\frac{n}{12}$ $\frac{n}{6}$ $\frac{n}{9}$ $\frac{n}{18}$ $\frac{n}{12}$ $\frac{n}{18}$ $\frac{n}{36}$
n

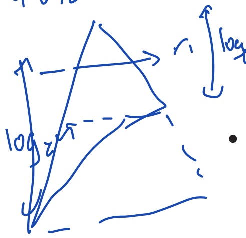
- The solution to the recurrence $T(n) = T(n/2) + T(n/3) + T(n/6) + O(n)$ is $T(n) = O(n)$.

  □ Yes ☒ No T(n)=O(nlogn)

- There is a forest with 374 vertices and 225 edges. (Recall that a *forest* is an undirected graph with no cycles.)

  tree with n verts has n-1 edges.

  ☒ Yes □ No Tree with 226 vertices plus 374-226 isolated verts.



- Given any directed graph G whose edges have positive weights, we can compute shortest paths from one vertex s to every other vertex of G in $O(VE)$ time using Bellman-Ford.

  □ Yes ☒ No G might be disconnected

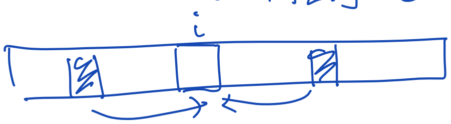
- Suppose $A[1..n]$ is an array of integers. Consider the following recursive function:

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

  We can compute $Ohio(n)$ by ~~memoizing~~ this function into a ~~two~~ one-dimensional array $Ohio[1..n]$, which we fill by increasing i in ~~$O(n)$~~ time.

  □ Yes ☒ No ∞ loops if A[i]=1 for all i

  or A[i]=0 for all i



Problem 2 continues onto the next page.

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

$$\text{CHALMERS} := \{\langle M\rangle \mid M \text{ accepts both STEAMED and HAMS}\}$$

Professor Skinner suggests a reduction from the standard halting language

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

Specifically, Professor Skinner claims that if there is a Turing machine SUPERNINTENDO that decides the language CHALMERS, then the following algorithm decides HALT.

```
DECIDEHALT(⟨M⟩, w):
  Encode the following Turing machine:
      AURORABOREALIS(x):
        if x = STEAMED or x = HAMS or x = UTICA
            ~~run M on input w~~ Halts
            return FALSE
        else
            return TRUE
  return SUPERNINTENDO(⟨AURORABOREALIS⟩)
```

Always rejects

Which of the following statements is true for all inputs (⟨M⟩, w)?

- If M hangs on w, then AURORABOREALIS accepts ALBANY.

  ☒ Yes □ No ALBANY is not STEAMED or HAMS or UTICA

  AURORABOREALIS accepts STEAMED and HAMS

- If M accepts w, then SUPERNINTENDO accepts ⟨AURORABOREALIS⟩.

  □ Yes ☒ No AuroraB actually rejects STEAMED

- If M hangs on w, then DECIDEHALT rejects (⟨M⟩, w). ⇔ SN rejects ⟨AB⟩ ⇔ AB does not accept both STEAMED and HAMS.

  ☒ Yes □ No AuroraB loops on STEAMED

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

  □ Yes ☒ No M halts on w ⇒ AB rejects STEAMED ⇒ DecideHalt rejects

  True ⇔ False are reversed

- We could have proved that CHALMERS is undecidable using Rice's theorem instead of this reduction.

  ☒ Yes □ No $\mathcal{L}$ = { all languages containing STEAMED + HAMS }

  $Y = M_{\text{ACCEPT}}$ $N = M_{\text{REJECT}}$

CS/ECE 374 A ✧ Fall 2025	Name:
Practice Final 3 Problem ~~2~~ 3	

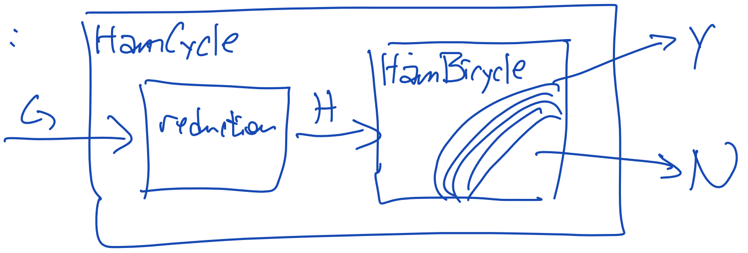
Submit a solution to ***exactly one*** of the following problems.

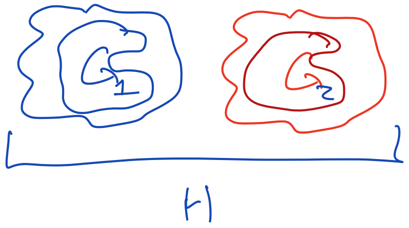
(a) A *Hamiltonian bicycle* in a graph G is a pair of simple cycles in G, with identical lengths, such that every vertex of G lies on exactly one of the two cycles. ***Prove*** that it is NP-hard to determine whether a given graph G has a Hamiltonian bicycle.

(b) A *clique-partition* of a graph $G = (V, E)$ is a partition of V into disjoint subsets $V_1 \cup V_2 \cup \cdots \cup V_k$, such that for each index i, every pair of vertices in subset V_i is connected by an edge in G. The *size* of a clique partition is the number of subsets V_i. ***Prove*** that it is NP-hard to compute the minimum-size clique partition of a given undirected graph G.

In fact, both of these problems are NP-hard, but we only want a proof for one of them. Don't forget to tell us which problem you've chosen!

(a) Reduce from HamCycle:
Given a graph $G=(V,E)$
Build a new graph H
$=$ two disjoint copies of G, $G_1 \cup G_2$





($\Longrightarrow$) Suppose G has a Ham. cycle C.
Then G_1 has Ham cycle C_1
G_2 has Ham cycle C_2
$\Longrightarrow$ $C_1 \cup C_2$ is a Ham bicycle in H.

($\Longleftarrow$) Suppose H has a Ham bicycle $C \cup C'$ ← two cycles of length V
WLOG C contains a vertex of G_1
$\Longrightarrow$ C is entirely in G_1 (no edges between G_1 and G_2)
$\Longrightarrow$ C is Ham cycle in G_1!

Easy to build H in poly time ✓

See page 8

CS/ECE 374 A ✧ Fall 2025	Name:
Practice Final 3 Problem ~~3~~ 4	

Let T be a *full* binary tree, meaning that every node has either two children or no children.

recursion?

- Recall that the *height* of a vertex v in T is the length of the longest path in T from v down to a leaf. In particular, every leaf of T has height zero.
- A vertex v is *AVL-balanced* if v is a leaf, or if the heights of v's children differ by at most 1. (You might recall from CS 225 that an ***AVL-tree*** is a binary search tree in which *every* vertex is AVL-balanced.)

Describe and analyze an algorithm to compute the number of AVL-balanced vertices in T.

---

① compute height of every vertex in T

$$\text{height}(v) = \begin{cases} 0 & \text{if } v \text{ is leaf} \\ 1 + \max(\text{height}(v.\text{left}), \\ \quad \text{height}(v.\text{right})) \end{cases}$$

memoize into v.height

eval in postorder in $O(n)$ time

②

```
count ← 0
for all vertices v
    if (v. is ~~not~~ a leaf) or ~~and~~ (|v.left.height − v.right.height| < 2)
        count ← count+1
return count
```

$\boxed{O(n) \text{ time}}$

CS/ECE 374 A ✧ Fall 2025 **Practice Final 3 Problem ~~4~~** 5	Name:

Suppose we are given a directed graph $G = (V, E)$, where every edge $e \in E$ has a *positive* weight $w(e)$, along with two vertices s and t.

Dijstra?

(a) Suppose each *vertex* of G is colored either orange, green, or purple. Describe and analyze an algorithm to find the shortest walk from s to t in G that never visits two consecutive *vertices* with the same color.

(b) Now suppose each *edge* of G is colored either orange, green, or purple. Describe and analyze an algorithm to find the shortest walk from s to t in G that never traverses two consecutive *edges* with the same color.

ⓐ Build subgraph G' of G by deleting every edge $u \to v$ where $u.\text{color} = v.\text{color}$

Run Dijkstra to find shortest path in G' from s to t

$\boxed{O(E \log V) \text{ time}}$

ⓑ Build new graph $G' = (V', E')$ as follows:

3V verts → $V' = V \times \{\text{orange}, \text{green}, \text{purple}\}$

2E edges → $E' = \{(u,o) \to (v,g),\ (u,p) \to (v,g) \mid \text{green } u \to v \in E\}\ \cup$

$\{(u,g) \to (v,p),\ (u,o) \to (v,p) \mid \text{purple } u \to v \in E\}\ \cup$

$\{(u,g) \to (v,o),\ (u,p) \to (v,o) \mid \text{orange } u \to v \in E\}$

weight of $(u,c) \to (v,c')$ is $w(u \to v)$ in G

We want shortest path in G' from $(s,o), (s,g),$ or (s,p) to $(t,o), (t,g),$ or (t,p)

Dijstra ×3 in $O(E' \log V') = \boxed{O(E \log V) \text{ time}}$

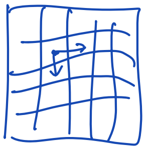
CS/ECE 374 A ✧ Fall 2025 Practice Final 3 Problem ~~5~~	Name: 6

DP? ? dag..

(a) Describe and analyze an efficient algorithm to compute the maximum possible score for a game of Vankin's Mile, given the $n \times n$ array of values as input. *(See the question handout for a detailed description of Vankin's Mile.)*

(b) A variant called *Vankin's Niknav* adds an additional constraint: *The sequence of values that the token touches must be a **palindrome**.* Describe and analyze an efficient algorithm to compute the maximum possible score for an instance of Vankin's Niknav, given the $n \times n$ array of values as input.

(a)



- start anywhere
- end by falling off edge ←

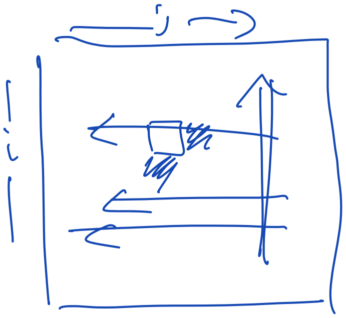
input array is $A[1..n, 1..n]$

$\text{Vankin}(i,j)$ = max score possible starting at row i, col j.

$$\text{Vankin}(i,j) = \begin{cases} 0 & \text{if } i>n \text{ or } j>n \\ A(i,j) + \max\left\{ \begin{array}{l} \text{Vankin}(i,j+1) \\ \text{Vankin}(i+1,j) \end{array} \right\} & \text{o/w} \end{cases}$$

memoize into 2d array

$\boxed{O(n^2) \text{ time}}$



(b)

→ palindromes ←

$$P \to \underbrace{0P0 \mid 1P1}_{\text{recursive}} \mid \underbrace{0 \mid 1 \mid \varepsilon}_{\text{base}}$$

$\text{Niknav}(i,j,i',j')$ = max score along any path from (i,j) to (i',j')

we need $\max\left\{ \text{Niknav}(i,j,i',j') \;\middle|\; \begin{array}{ll} 1 \le i \le n & 1 \le j \le n \\ 1 \le i' \le n & 1 \le j' \le n \\ \multicolumn{2}{c}{(i=n \text{ or } j=n)} \\ \multicolumn{2}{c}{A(i,j) = A(i',j')} \end{array} \right\}$

$\boxed{\text{see page 7}}$

CS/ECE 374 A ✧ Fall 2025 **Practice Final 3 Problem ~~6~~**	Name: 7

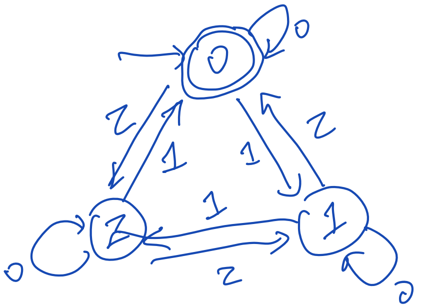
Σ digits

(a) Let L_a denote the set of all strings $w \in \{0,1,2\}^*$ such that $\boxed{\#(1,w) + 2 \cdot \#(2,w)}$ is divisible by 3. Describe a DFA or NFA that accepts L_a. (You do not need to prove that your answer is correct.)

(b) Let L_b denote the set of all strings $w \in \{0,1,2\}^*$ such that no two symbols appear the same number of times, or in other words, the integers $\#(0,w)$ and $\#(1,w)$ and $\#(2,w)$ are all different. ***Prove*** that L_b is not a regular language.

Fooling set?

ⓐ



state = (Σ digits so far) mod 3

ⓑ Aim for $L_b \cap \cancel{0^*} 1^* 2^*$

$0^a \underbrace{1^b 2^c}$ where $a = 0$, $b, c \neq 0$

Let $F = 1^+ = \{1^b \mid b > 0\}$

Pick any two strings $x, y \in F$

$\Rightarrow x = 1^i$ and $y = 1^j$ for some $i \neq j$, $i > 0$ $j > 0$

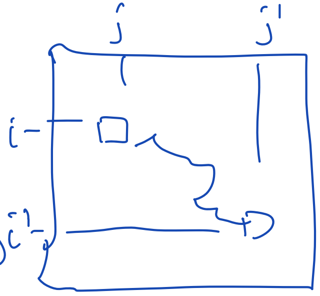
Let $z = 2^i$

Then $xz = 1^i 2^i \notin L_b$ because $\#1 = \#2$

$yz = 1^j 2^i \in L_b$ because $i \neq j$

So F is ∞ fooling set for L_b. □

Niknaw continued



$$\text{Niknaw}(i,j,i',j') = \begin{cases} A[i,j] & \text{if } (i,j)=(i',j') \\ A[i,j]+A[i',j'] & \text{if } (i',j')=(i+1,j) \text{ or } (i,j+1) \\ -\infty & \text{if } i>i' \text{ or } j>j' \\ -\infty & \text{if } A[i,j] \neq A[i',j'] \\ A[i,j]+A[i',j'] + \max \left\{ \begin{array}{l} \text{Niknaw}(i+1,j,i'-1,j') \\ \text{Niknaw}(i+1,j,i',j'-1) \\ \text{Niknaw}(i,j+1,i'-1,j') \\ \text{Niknaw}(i,j+1,i',j'-1) \end{array} \right\} & \end{cases}$$

memoize into 4D array

Niknaw[1..n, 1..n, 1...n, 1...n]

```
eval: for i ← 1 to n
        for j ← 1 to n
          for i' ← n down to 1
            for j' ← n down to 1
              // recurrence
```

$$\boxed{O(n^4) \text{ time}}$$

clique-partition = split vertices into as few categories as possible
s.t. every category is clique
= all pairs connected

chromatic # = split V into as few categories
as possible s.t. each category
is an independent set
= no pairs connected.

Given $G=(V,E)$ define $\overline{G} = (V, \binom{V}{2} \setminus E)$
$(uv \in E \iff uv \notin \overline{E})$ edge-complement

☆ every clique in G = ind set in $\overline{G}$ and vice versa!

(⇒) Suppose G can be colored with k colors
then call color classes $V_1, V_2, \dots V_k$
partition of V into indep sets in G
= partition of V into k cliques in $\overline{G}$
so $\overline{G}$ has clique partition of size k.

(⇐) Suppose $\overline{G}$ has clique partition of size k
$V = V_1 \cup V_2 \cup \dots V_k$
each V_i is clique in $\overline{G}$
⇒ each V_i is indep set in G
⇒ G has a proper k-coloring

So Chrom#(G) = CliquePart($\overline{G}$)

poly time ✓

(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 $S_1, S_2, \ldots, S_m$ 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?

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 $c \in Z^d$, 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?