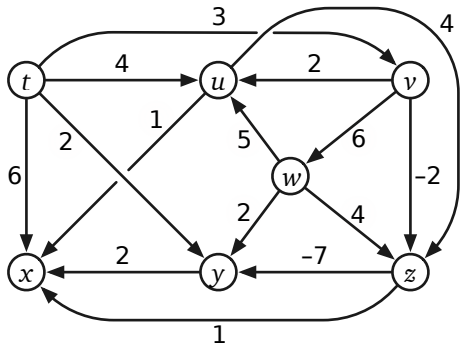
Write your answers in the separate answer booklet.

You have 120 minutes (after you get the answer booklet) to answer five questions.

Please return this question sheet and your cheat sheet with your answers.

1. ***Clearly*** indicate the following structures in the directed graph below. Don't be subtle! To indicate a subset of edges, draw a **HEAVY BLACK LINE** along the entire length of each edge. If the requested structure does not exist, write the word NONE. (The answer booklet contains several copies of this graph.)



   (a) A depth-first search tree rooted at v

   (b) A breadth-first search tree rooted at w

   (c) A shortest-path tree rooted at t

   (d) A list of vertices in topological order

graph reduction

2. Suppose you are given a directed graph $G = (V, E)$, each of whose edges are colored red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. A *rainbow walk* is a walk in G that does *not* contain two consecutive edges with the same color.

   Describe and analyze an algorithm to find all vertices in G that are reachable from a given vertex s through a rainbow walk.

3. Suppose we are given an n-digit integer X. Repeatedly delete 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 integer 32492 has square-depth 3, by the following sequence of digit deletions:

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

   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***.

variable

backtracking

dynamic programming

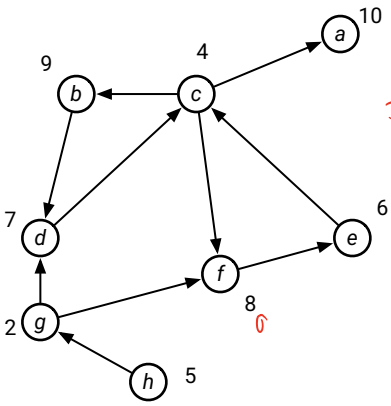
4. Suppose you are given k arrays $A_1[1..n]$, $A_2[1..n]$, ..., $A_k[1..n]$, each with the same length n, and each sorted in increasing order. Describe an algorithm to merge the given arrays into a single sorted array. Analyze the running time of your algorithm as a function of n and k.

Copy all to new B[1..nk] + sort : $O(nk \log nk)$

divide-and-conquer?

$\Omega(nk)$ just to read

5. Let $G = (V, E)$ be a directed graph with vertex weights $w : V \to \mathbb{R}$. Given a vertex u, let $MinReach(u) := \min\{w(v) \mid u \text{ can reach } v\}$ be the minimum weight over all vertices v such that u can reach v. For example, if G is the graph shown below, then $MinReach(f) = 4$ and $MinReach(h) = 2$.



u → v

f can reach c

h can reach g

propagate back?

(a) Suppose G is a directed acyclic graph (dag). Describe an algorithm to compute $MinReach(u)$ for *every* vertex $u \in V$.

DP?

(b) Extend your algorithm to the case of a general directed graph. You may use the algorithm from part (a) as a black-box whether or not your solution for part (a) is correct.

↑ build SCC meta-graph

$O(|V| + |E|)$ time

CS/ECE 374 A ✧ Spring 2026

ᔕ Midterm 2 Practice 2 ᔓ

April 10, 2026

Name:	Emily Fox
NetID:	ekfox

- ***Don't panic!***
- You have 120 minutes to answer five questions. The questions are described in more detail in a separate handout.
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- Please clearly print your name and your NetID in the boxes above.
- Please also print your name at the top of every page of the answer booklet, except this cover page. We want to make sure that if a staple falls out, we can reassemble your answer booklet. (It doesn't happen often, but it does happen.)
- Greedy algorithms require formal proofs of correctness to receive any credit, even if they are correct. Otherwise, proofs or other justifications beyond items listed in the standard algorithms rubrics are required for full credit if and only if we explicitly ask for them, using the word ***prove*** or ***justify*** in bold italics.

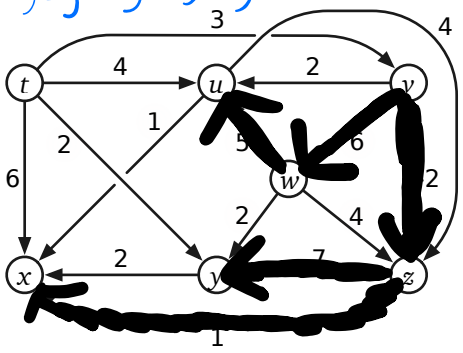
- **Please do not write outside the black boxes on each page.** These indicate the area of the page that our scanners can actually scan. If the scanner can't see your work, we can't grade it.
- If you run out of space for an answer, please use the overflow/scratch pages at the back of the answer booklet, 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.
- Please return ***all*** paper with your answer booklet: your question sheet, your cheat sheet, and all scratch paper. **Please put all loose paper *inside* your answer booklet.**

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

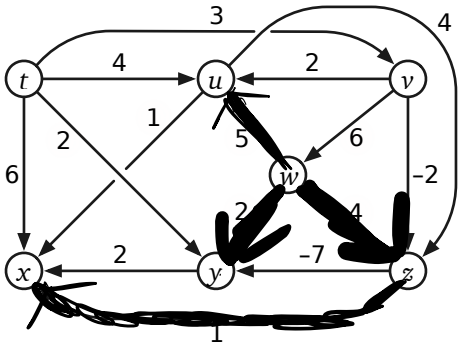
CS/ECE 374 A ✧ Spring 2026 Midterm 2 Practice 2 Problem 1	Name: Emily Fox

Clearly indicate the following structures in the directed graph below. Don't be subtle! To indicate a subset of edges, draw a **HEAVY BLACK LINE** along the entire length of each edge. If the requested structure does not exist, write the word NONE.

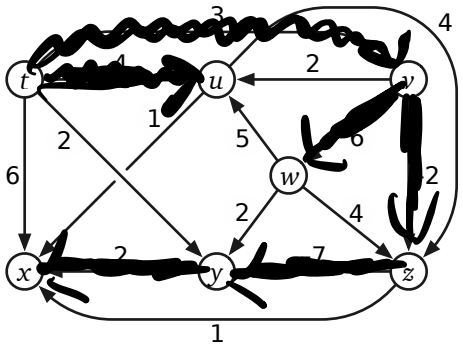
po: x, y, z, u, w, v, t



(a) A depth-first search tree rooted at v



(b) A breadth-first search tree rooted at w

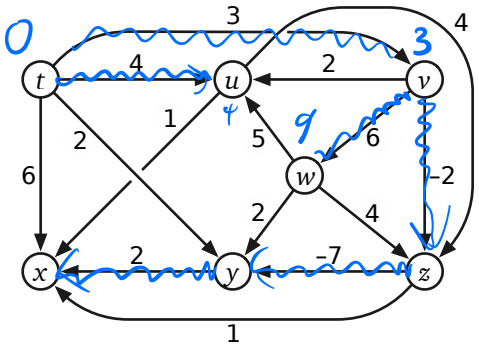


(c) A shortest-path tree rooted at t

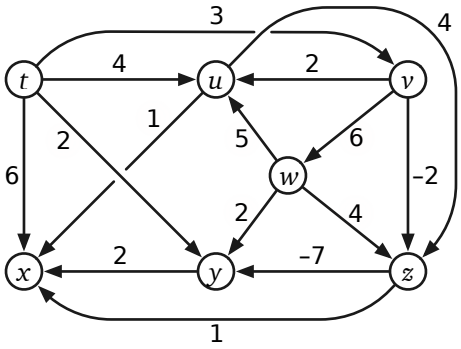
t, v, w, u, z, y, x

(d) A list of vertices in topological order

t: 0
v: 3
w: 9
u: 4
z: 1
y: -6
x: -4



[scratch]



[scratch]

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

Suppose you are given a directed graph $G = (V, E)$, each of whose edges are colored red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. A *rainbow walk* is a walk in G that does *not* contain two consecutive edges with the same color.

Describe and analyze an algorithm to find all vertices in G that are reachable from a given vertex s through a rainbow walk.

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

$V' := V \times \{R, G, B\}$ (vertex, color of prev. edge)

$E' := \{((u, c_1), (v, c_2)) \mid (u,v) \in E,$ color of (u,v) is c_2, and $c_1 \neq c_2\}$

Want to find all vertices v s.t. (s,R), (s,G), or (s,B) can reach (v,R), (v,G), or (v,B).

So call a linear time search like BFS from each of (s,R), (s,G), & (s,B) & return all v where any of the three copies is reached by at least one search.

Time: $O(|V'| + |E'|) = O(|V| + |E|)$

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

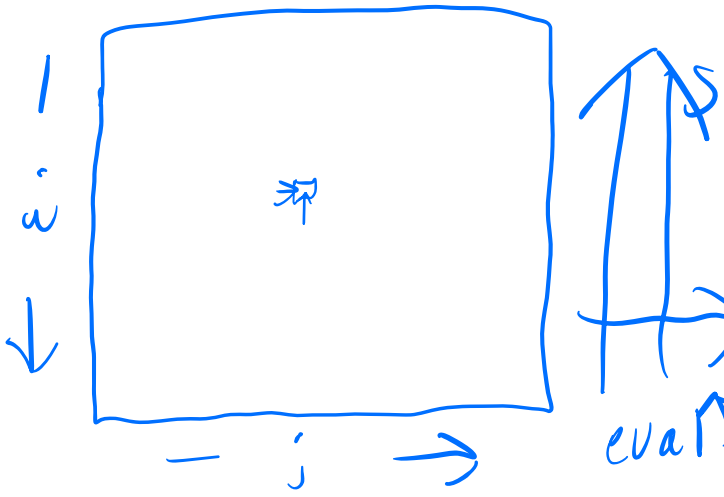
Suppose we are given an n-digit integer X. Repeatedly remove one digit from either end of X (your choice) until no digits are left. The *square-depth* of X is the maximum number of perfect squares that you can see during this process.

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***.

SqDepth(i,j): Square-depth of $X[i..j]$.

$$SqDepth(i,j) = \begin{cases} 0 & i > j \\ [IsSquare(X[i..j])] + \max\{SqDepth(i+1,j), SqDepth(i,j-1)\} & \text{o.w.} \end{cases}$$

Need SqDepth(1,n).



Store in a 2D array SqDepth[1..n+1, 0..n].

-or-

for i ← n+1 down to 1
for j ← 0 up to n

Time: $O(n^2) \cdot O(n^2) = O(n^4)$

↑ IsSquare ↑ # subproblems

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

Suppose you are given k sorted arrays $A_1[1..n], A_2[1..n], \ldots, A_k[1..n]$, all with the same length n. Describe an algorithm to merge the given arrays into a single sorted array. Analyze the running time of your algorithm as a function of n and k.

If $k=1$, return $A_1[1..n]$.

(If $k=0$, return empty array.)

$m \leftarrow \lceil k/2 \rceil$

$A_{\leftarrow} \leftarrow$ Recursively merge A_1 through A_m.

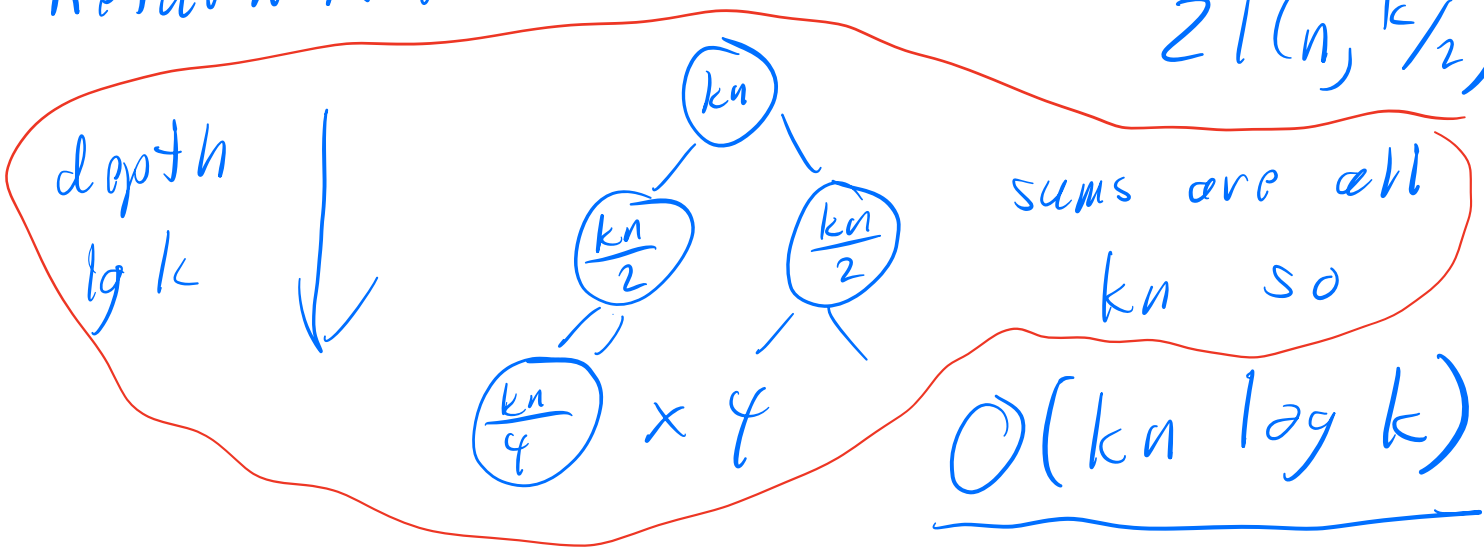
$A_{\rightarrow} \leftarrow$ Recursively merge A_{m+1} through A_k.

Merge $A_{\leftarrow}$ and $A_{\rightarrow}$ using algorithm from lectures/book.

to make A^*.

Return A^*.

Time: $T(n,k) = kn + 2T(n, k/2)$



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

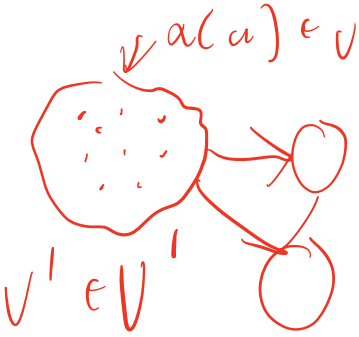
Let $G = (V, E)$ be a directed graph with vertex weights $w : V \to \mathbb{R}$. Given a vertex u, let $MinReach(u) := \min\{w(v) \mid u \text{ can reach } v\}$ be the minimum weight over all vertices v such that u can reach v.

(See the question handout for an example graph.)

(a) ~~1.~~ Suppose G is a directed acyclic graph (dag). Describe an algorithm to compute $MinReach(u)$ for *every* vertex $u \in V$.

(b) ~~2.~~ Extend your algorithm to the case of a general directed graph. You may use the algorithm from part (a) as a black-box whether or not your solution for part (a) is correct.

(b) Build SCC meta-graph ⇐ a DAG $G' = (V', E')$ using algorithm from lecture in $O(|V| + |E|)$ time.



$v \in a(u) \in v' \in \min w(x)\ x \in v'$

Give each subset $u' \in V'$ a weight $w'(u') := \min\{w(u) \mid u \in u'\}$.

MinReach(u) for $u \in V$ in G MinReach(u') where $u \in u' \in V'$ in G'.

Use algo from part (a) on G'. (a) on page 6.

Time: $O(|V| + |E|)$

(scratch paper)

5. (a) $MinReach(u) =$

$$\min\{w(u), \min\{MinReach(v) \mid (u,v) \in E\}\}$$

Store on each vertex as $u.MinReach$.

Eval in postorder.

Time: $O(|V| + |E|)$

i.e. for each vertex u in reverse topological order

$MinReach(u) \leftarrow \min\{w(u), \min\{MinReach(v) \mid (u,v) \in E\}\}$

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)