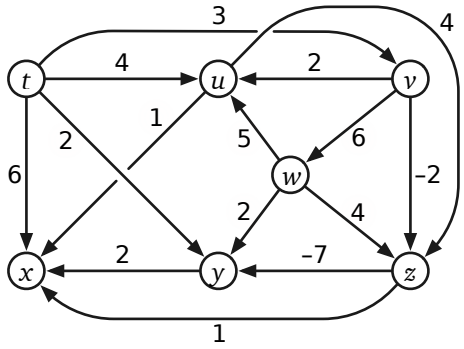
Write your answers in the separate answer booklet.

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

dag?

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:

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

   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***.

DP?

dag?

4. Suppose you are given k arrays $A_1[1..n]$, $A_2[1..n]$, $\ldots$, $A_k[1..n]$, 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.

5. You are planning a hiking trip in Jellystone National Park over winter break. You have a complete map of the park's trails; the map indicates that some trail segments have a high risk of bear encounters. All visitors to the park are required to purchase a canister of bear repellent. You can safely traverse a high-bear-risk trail segment only by *completely* using up a *full* canister of bear repellent. The park rangers have installed refilling stations at several locations around the park, where you can refill empty canisters at no cost. The canisters themselves are expensive and heavy, so you cannot carry more than one. Because the trails are narrow, each trail segment allows traffic in only one direction.

   You have converted the trail map into a directed graph $G = (V, E)$, whose vertices represent trail intersections, and whose edges represent trail segments. A subset $R \subseteq V$ of the vertices indicate the locations of the *R*epellent *R*efilling stations, and a subset $B \subseteq E$ of the edges are marked as having a high risk of *B*ears. Your campsite appears on the map as a particular vertex $s \in V$, and the visitor center is another vertex $t \in V$.

   (a) Describe and analyze an algorithm to decide if you can safely walk from your campsite s to the visitor center t. Assume there is a refill station at your camp site, and another refill station at the visitor center.

   (b) Describe and analyze an algorithm to decide if you can safely walk from any refill station to any other refill station. If there a safe path from u to v for *every* pair of vertices u and v in R, your algorithm should return TRUE; otherwise, it should return FALSE.

CS/ECE 374 A ✧ Fall 2025

ᔕ Midterm 2 Practice 2 ᔓ

November 7, 2025

Name:	Jeff E
NetID:	jeffe

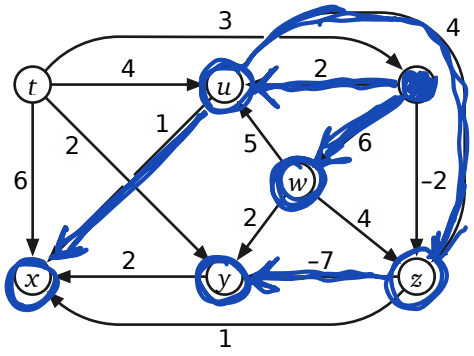
- ***Don't panic!***
- You have 120 minutes to answer five questions. The questions are described in more detail in a separate handout.
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, and your university ID, please put it away for the 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 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.

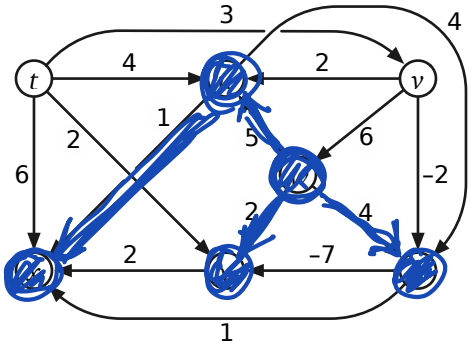
This sentence contains four and three fourths percent a's, five and one half percent c's, three percent d's, eighteen and one fourth percent e's, four and one half percent f's, three fourths percent g's, five percent h's, two and one half percent i's, two percent l's, twelve and one half percent n's, 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 ✧ Fall 2025 Midterm 2 Practice 2 Problem 1	Name: Jeff E

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.

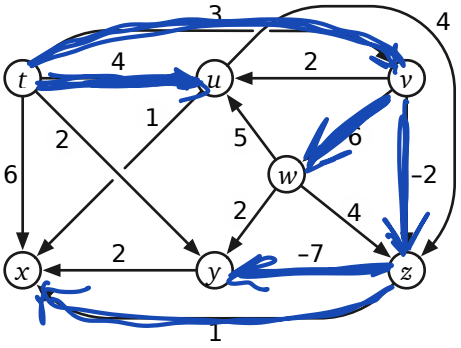


(a) A depth-first search tree rooted at v



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

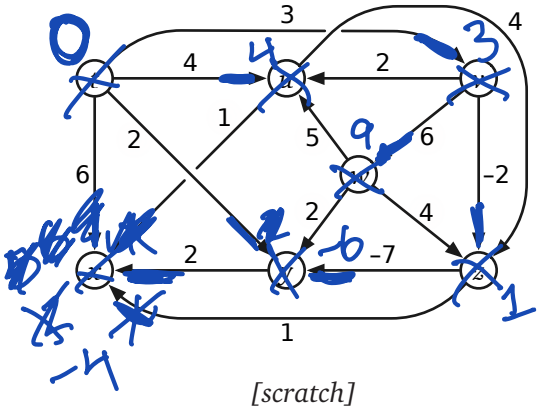
Queue



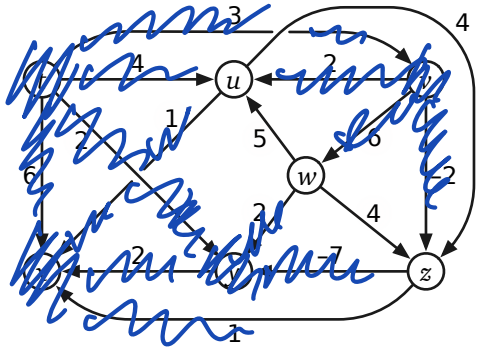
(c) A shortest-path tree rooted at t

t v w u z y x

(d) A list of vertices in topological order



[scratch]



[scratch]

CS/ECE 374 A ✧ Fall 2025 Midterm 2 Practice 2 Problem 2	Name:

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. ↳ NOT DP

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

---

reachable layers

Graph def.

$V' = V \times \{red, green, blue\}$

(v, c) means I'm at v previous edge was color c.

$E' = \{(u,green)\to(v,red), (u,blue)\to(v,red) \mid u\to v \text{ is red}\}$

$\cup \{(u,red)\to(v,green), (u,blue)\to(v,green) \mid u\to v \text{ is green}\}$

$\cup \{(u,red)\to(v,blue), (u,green)\to(v,blue) \mid u\to v \text{ is blue}\}$

(s can reach v in G via rainbow walk $\iff$ (s,c) can reach (v,c') in G' for some colors c and c')

Problem: we need to find all verts in G' reachable from (s,red) or $(s,blue)$ or $(s,green)$

Algorithm: $3 \times$ WFS

$O(V'+E') \times 3$

$= O(V'+E')$

$= O(3V+2E)$

$= \boxed{O(V+E) \text{ time}}$ in terms of orig. input

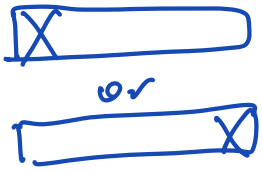
CS/ECE 374 A ✧ Fall 2025 Midterm 2 Practice 2 Problem 3	Name:

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.

includes +1 if X is perfect square

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***.

DP? dag?



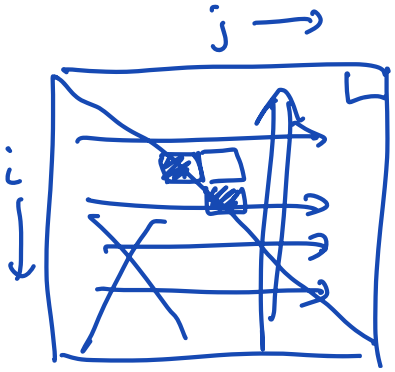
subproblems = intervals

Function spec

Let SD(i,j) = square-depth of X[i..j]

recurrence

$$SD(i,j) = \begin{cases} 0 & \text{if } i > j \\ \cancel{[\text{IsSquare}(X[i])]} \quad \cancel{\text{if } i = j} & \\ [\text{IsSquare}(X[i..j])] & \end{cases} + \max \begin{cases} SD(i+1,j) \\ SD(i,j-1) \end{cases}$$



We want SD(1,n)

iterative details

Each entry takes $O((j-i)^2)$ time $\le O(n^2)$

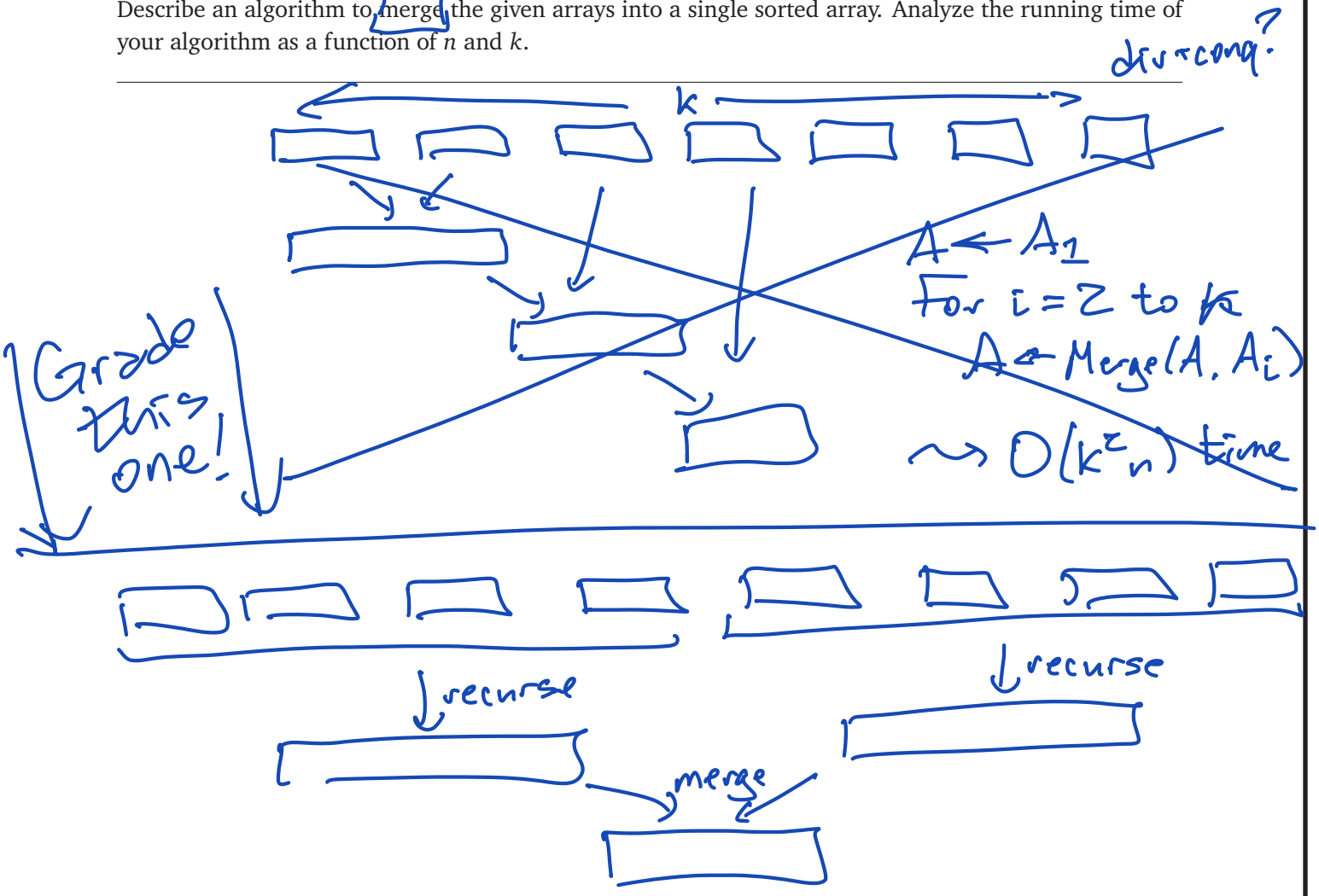
2D array

For i ← n down to 1
  For j ← i to n
    (recurrence)

$\Rightarrow$ $O(n^4)$ time overall

CS/ECE 374 A ✧ Fall 2025 Midterm 2 Practice 2 Problem 4	Name:

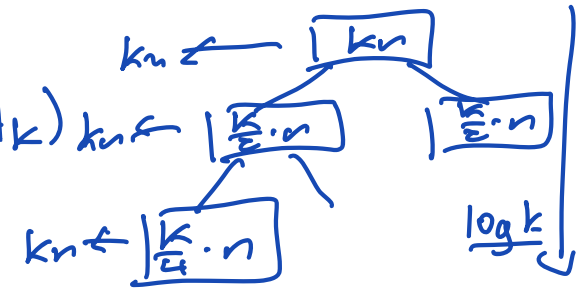
Suppose you are given k sorted arrays $A_1[1..n], A_2[1..n], \ldots, A_k[1..n]$, all with the same length n. Describe an algorithm to merge the given arrays into a single sorted array. Analyze the running time of your algorithm as a function of n and k.



MMerge($A_1 \ldots A_k$):
if $k = 1$
return A_1
else
$A_L \leftarrow$ MMerge($A_1 \ldots A_{k/2}$)
$A_R \leftarrow$ MMerge($A_{k/2+1} \ldots A_k$)
return Merge(A_L, A_R)

$$T(k,n) = \begin{cases} O(1) & \text{if } k = 1 \\ 2T(\frac{k}{2}, n) + O(kn) & \text{o/w} \end{cases}$$

$\neq O(kn \log k)$



CS/ECE 374 A ✧ Fall 2025 Midterm 2 Practice 2 Problem 5	Name:

See the question handout for full description of this problem.

reach layering

(a) Describe and analyze an algorithm to decide if you can safely walk from your campsite s to the visitor center t. Assume there is a refill station at your camp site, and another refill station at the visitor center.

(b) Describe and analyze an algorithm to decide if you can safely walk from any refill station any other refill station. If there a safe path from u to v for *every* pair of vertices u and v in R, your algorithm should return TRUE; otherwise, it should return FALSE.

R = refill vertexes
B = bear edges

At most one Bear edge between Refill vertices

(a) $V' = V \times \{T, F\}$

(v,T) = at v, with bear spray
(v,F) = at v, without bear spray

$$E' = \{(u,F)\to(v,F) \mid u\to v \notin B \text{ and } v \notin R\}$$
$$\cup \{(u,F)\to(v,T) \mid u\to v \notin B \text{ and } v \in R\}$$
$$\cup \{(u,T)\to(v,F) \mid u\to v \in B \text{ and } v \notin R\}$$
$$\cup \{(u,T)\to(v,T) \mid u\to v \notin B \text{ OR } v \in R\}$$

Safe walk in G from s to t $\iff$ walk in G' from (s,T) to (t,T)

check if exists using WFS in $O(V'+E')$ = $O(V+E)$ time

(b) See page 6

5b

(scratch paper)

~~For all $u \in R$~~
~~For all $v \in R$~~
~~run algo from (a)~~
$\rightarrow O(V^3 + V^2E)$ time

If any $\rightarrow$ any
then $s \rightarrow$ any
and any $\rightarrow s$

$s \rightarrow$ any any $\rightarrow s$
$s \rightarrow v$ $u \rightarrow s$
$u \rightarrow s \rightarrow v$
$u \rightarrow v$

GRADE THIS?

use same graph G' from part a

(1) Find all vertices (v,T) with $v \in R$ reachable from (s,T) in G'
via WFS$((s,T))$

(2) Find all vertices (v,T) with $v \in R$ that can reach (s,T) in G'

Reverse all edges in G'
run WFS$((s,T))$ in rev(G')

If all (v,T) with $v \in R$ pass both tests, return TRUE
else FALSE

$O(V+E)$ time

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)