CS/ECE 374 A ✧ Fall 2023

2nd Practice Midterm 2

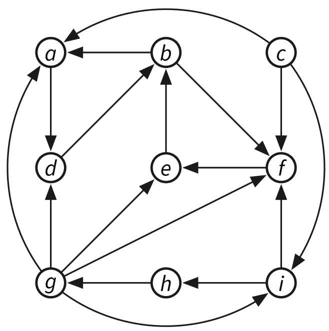
November 4, 2023

Name:	
NetID:	

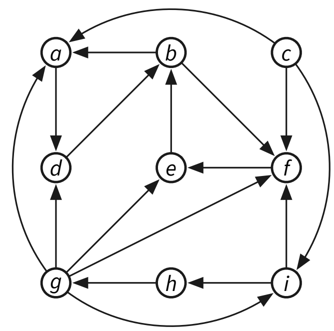
- ***Don't panic!***
- You have 120 minutes to answer five questions. The questions are described in more detail in a separate handout.
- If you brought anything except your writing implements, your **hand-written** double-sided 8½" × 11" cheat sheet, and your university ID, please put it away for the 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.)
- **Do not write outside the black boxes on each page**. These indicate the area of the page that our scanner can actually see. Anything you write outside the boxes will be erased before we start grading.
- If you run out of space for an answer, please use the scratch pages at the back of the answer booklet, but **please clearly indicate where we should look**. Please ask for more scratch paper if you need it.
- 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 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.**

CS/ECE 374 A ✧ Fall 2023 Practice Midterm 2 Problem 1	Name:

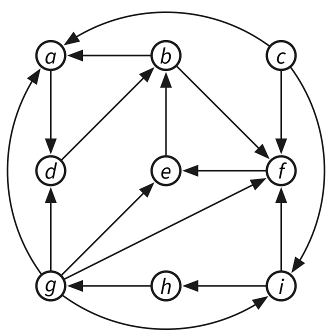
Clearly indicate the following structures in the directed graph G drawn below, or write `NONE` if the indicated structure does not exist. Don't be subtle; for example, to indicate a collection of edges, draw a **HEAVY BLACK LINE** along the entire length of each edge.



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

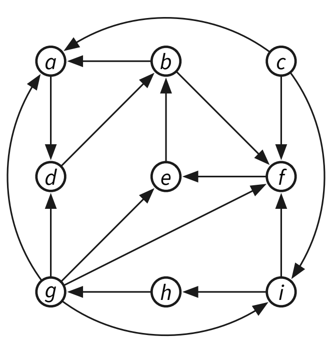


(b) A breadth-first search tree rooted at c.

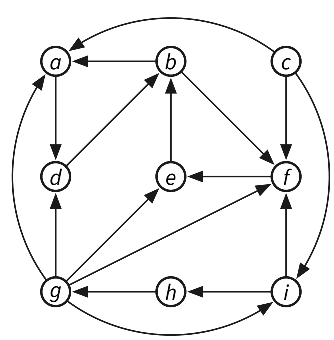


(c) Circle each strong component.

(d) Draw the strong-component graph of the example graph G.



[scratch]



[scratch]

CS/ECE 374 A ✧ Fall 2023 Practice Midterm 2 Problem 2	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.

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*** $\boldsymbol{O(k^2)}$ ***time***.

CS/ECE 374 A ✧ Fall 2023 Practice Midterm 2 Problem 3	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.

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

CS/ECE 374 A ✧ Fall 2023 Practice Midterm 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.

CS/ECE 374 A ✧ Fall 2023 Practice Midterm 2 Problem 5	Name:

After moving to a new city, you decide to walk from your home to your new office. To get a good daily workout, you want to reach the highest possible altitude during your walk (to maximize exercise), while keeping the total length of your walk below some threshold (to get to your office on time). Describe and analyze an algorithm to compute the best possible walking route.

Your input consists of an undirected graph G, where each vertex v has a height $h(v)$ and each edge e has a positive length $\ell(e)$, along with a start vertex s, a target vertex t, and a maximum length L. Your algorithm should return the maximum height reachable by a walk from s to t in G, whose total length is at most L.

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)