

**“CS/ECE 374 A”: Algorithms & Models of Computation, Spring 2025**  
**Midterm 2 — April 14, 2025**

Name:	
NetID:	

- 
- Please *clearly PRINT* your name and your NetID in the boxes above.
  - This is a closed-book but you are allowed a 1 page (2 sides) hand written cheat sheet that you have to submit along with your exam. If you brought anything except your writing implements, put it away for the duration of the exam. In particular, you may not use *any* electronic devices.
  - **Please read the entire exam before writing anything.** Please ask for clarification if any question is unclear. The exam has 6 problems, each worth 10 points.
  - **You have 150 minutes (2.5 hours) for the exam.**
  - If you run out of space for an answer, continue on the back of the page, or on the blank pages at the end of this booklet, **but please tell us where to look.**
  - **Write everything inside the box around each page.** Anything written outside the box may be cut off by the scanner.
  - **Proofs are required only if we specifically ask for them.** You may state and use (without proof or justification) any results proved in class or in the problem sets unless we explicitly ask you for one.
  - You can do hard things!
  - **Do not cheat.** You know the student code and all that jazz. Grades do matter, but not as much as you may think, and your values are more important.
-

## 1 Sums and Recurrences

(a) Consider the recurrence

$$T(n) = n^{1/4}T(n^{1/4}) + n \quad n \geq 16, \quad T(n) = 1 \quad 1 \leq n < 16$$

Give asymptotically tight bounds for the following, assuming the root of the tree is level one. No work is required; write your final answer in the box.

- (1 pt) Number of children at the second level:

- (1 pt) Work at the second level:

- (1 pt) Depth of the recurrence:

- (1 pt) Value of the recurrence:

(b) Consider the recurrences below and give asymptotically tight bounds for each  $T(n)$ . No work is required; write your final answer in the box.

- (2 pts)  $T(n) = 4T(n/2) + n$  for  $n \geq 2$  and  $T(1) = 1$

- (2 pts)  $T(n) = 4T(n/2) + n^2$  for  $n \geq 2$  and  $T(1) = 1$

- (2 pts)  $T(n) = 4T(n/2) + n^3$  for  $n \geq 2$  and  $T(1) = 1$

## 2 Recursion/Divide and Conquer/Sorting/Selection

$A$  is an array of  $n_1$  numbers that is sorted in *ascending* order.  $B$  is an array of  $n_2$  numbers sorted in *descending* order. You wished to create a sorted array by merging them but by mistake you concatenated  $A$  with  $B$  to create an array  $C$  with  $n = n_1 + n_2$  numbers (assume for simplicity that all the numbers are distinct). You do not have the original arrays anymore nor do you know  $n_1$  and  $n_2$ . For example if  $A = [1, 2, 6, 7]$  and  $B = [8, 4, 3]$  then  $C = [1, 2, 6, 7, 8, 4, 3]$ . Describe an algorithm, as fast as possible, that given  $C$ , its size  $n$ , and a number  $x$  checks whether  $x$  is in  $C$  or not.

### 3 Splitting Strings

Let  $\Sigma$  be a finite alphabet and  $L \subseteq \Sigma^*$  be a language. You have access to a routine  $\text{IsStringinL}(x)$  that on input  $x \in \Sigma^*$  returns whether  $x \in L$  or not. Given  $w \in \Sigma^*$  recall that  $w \in L^*$  if and only if  $w = w_1 w_2 \dots w_h$  for some  $h \geq 1$  such that each  $w_i \in L$ ; in this case we call  $w_1 w_2 \dots w_h$  a valid  $L$ -split. In many applications we are interested in a  $L$ -valid split with some additional properties. Given  $w$  and a split of  $w$  into  $w_1, w_2, \dots, w_h$  we define the  $\text{cost}(w_1, w_2, \dots, w_h)$  to be  $\sum_{i=1}^h |w_i|^2$ .

Describe an algorithm that given  $w \in \Sigma^*$  outputs the minimum cost of any  $L$ -valid split. Your algorithm should output  $\infty$  if there is no  $L$ -valid split of  $w$ . You can assume that  $\text{isStringinL}(x)$  takes  $O(1)$  time in your analysis.

## 4 Collecting Rewards

This is a variant of a problem from Homework 9.

Let  $G = (V, E)$  be a directed graph in which each edge  $e \in E$  has a non-negative reward  $p(e)$  that can be collected by traversing it. Describe an algorithm that given  $G$ , a starting vertex  $s \in V$  and an integer  $k \geq 0$  computes the maximum reward that a walk starting at  $s$  can collect if it is required to only collect reward from at most  $k$  edges. Note the the reward on an edge  $e$  is counted only the first time it is traversed in the walk since it is gone after it is picked up.

(a) (2 pts) How would you solve the problem if  $G$  is strongly connected?

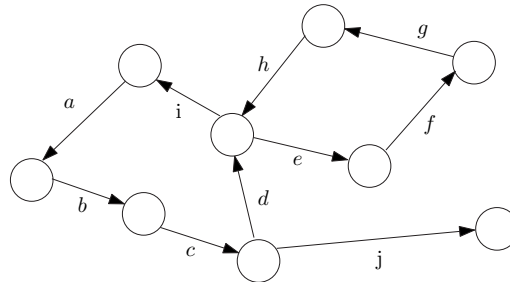
(b) (8 pts) How would you solve the problem if  $G$  is a DAG?

(Exercise for *after* the exam: combine these two parts to get an algorithm that works on an arbitrary directed graph.)

## 5 Graphs

Let  $G = (V, E)$  be a directed graph and let  $e_1 = (a, b)$  and  $e_2 = (x, y)$  be two distinct edges. We wish to know if there exists a *path* that uses both  $e_1$  and  $e_2$ , with  $e_1$  appearing earlier in the path than  $e_2$ . Note that a path does not allow repetitions of vertices or edges while a walk allows both.

(a) Consider the following graph:



- (1 pt) Specify two edges  $e_1$  and  $e_2$  such that there is a path  $P$  in the graph with  $e_1$  before  $e_2$  in  $P$ .

$e_1 =$

$e_2 =$

- (1 pt) Specify two edges  $e_1$  and  $e_2$  such that there is a walk  $W$  in the graph with  $e_1$  before  $e_2$  in  $W$  but there is *no* such path in the graph.

$e_1 =$

$e_2 =$

(Continued on next page)

- (b) (4 pts) Describe an efficient algorithm that given  $G = (V, E)$  and two distinct edges  $e_1, e_2 \in E$  checks if there is a *walk* in  $G$  with  $e_1$  before  $e_2$ .
- (c) (4 pts) Describe an efficient algorithm that given  $G = (V, E)$  and two distinct edges  $e_1, e_2 \in E$  checks if there is a *path* in  $G$  with  $e_1$  before  $e_2$ .





- (c) (4 pts) Let  $\alpha$  be the maximum value that you computed in the previous part. Describe an efficient algorithm to compute the second most valuable vertex your walk can reach subject to visiting the one with value  $\alpha$ . Your walk must still start and end at  $s$  and have total length at most  $L$ .

This page is for extra work.

This page is for extra work.