CS/ECE 374 A ✧ Spring 2026

Midterm 2 Practice 1

April 9, 2026

Name:	Ruta Mehta
NetID:	rutamehf

- ***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.**

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. Short answers:

   (a) Solve the following recurrences:
   - $A(n) = 3A(n/2) + O(n^2)$
   - $B(n) = 7B(n/2) + O(n^2)$
   - $C(n) = 4C(n/2) + O(n^2)$

   (b) Draw a directed acyclic graph with at most ten vertices, exactly one source, exactly one sink, and more than one topological order.

   (c) Draw a directed graph with at most ten vertices, with distinct positive edge weights, that has more than one shortest path from some vertex s to some other vertex t.

   (d) Describe an appropriate memoization structure and evaluation order for the following (meaningless) recurrence, and give the running time of the resulting iterative algorithm to compute $Huh(1, n)$.

$$Huh(i,k) = \begin{cases} 0 & \text{if } i > n \text{ or } k < 0 \\ \min\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} + A[i,k] & \text{if } A[i,k] \text{ is even} \\ \max\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} - A[i,k] & \text{if } A[i,k] \text{ is odd} \end{cases}$$

2. You and your eight-year-old nephew Elmo decide to play a simple card game. At the beginning of the game, the cards are dealt face up in a long row. Each card is worth a different number of points, which could be positive, negative, or zero. After all the cards are dealt, you and Elmo take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, you can decide which of the two cards to take. The winner of the game is the player that has collected the most points when the game ends.

   Having never taken an algorithms class, Elmo follows the obvious greedy strategy—when it's his turn, Elmo *always* takes the card with the higher point value. Your task is to find a strategy that will beat Elmo whenever possible. (It might seem mean to beat up on a little kid like this, but Elmo absolutely *hates* it when grown-ups let him win.)

   (a) ***Prove*** that you should not also use the greedy strategy. That is, show that there is a game that you can win, but only if you do *not* follow the same greedy strategy as Elmo. Assume Elmo plays first.

   (b) Describe and analyze an algorithm to determine, given the initial sequence of cards, the maximum number of points that you can collect playing against Elmo.

Recursion ↓ DP?

[0] [10] - - - [-1]

3. Suppose you are given a directed graph $G = (V, E)$, whose vertices are either red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. The ***remoteness*** of a vertex v is the *maximum* of three shortest-path lengths:

Graph Reduction?

   - The length of a shortest path to v from the closest red vertex
   - The length of a shortest path to v from the closest blue vertex
   - The length of a shortest path to v from the closest green vertex

   In particular, if v is not reachable from vertices of all three colors, then v is infinitely remote. Describe and analyze an algorithm to find a vertex of G with *minimum* remoteness.

4. Suppose you are given an array $A[1..n]$ of integers such that $A[i] + A[i+1]$ is even for *exactly one* index i. In other words, the elements of A alternate between even and odd, except for exactly one adjacent pair that are either both even or both odd.

   Describe and analyze an efficient algorithm to find the unique index i such that $A[i] + A[i+1]$ is even. For example, given the following array as input, your algorithm should return the integer 6, because $A[6] + A[7] = 88 + 62$ is even. (Cells containing even integers are shaded blue.)

Binary search

A[i]:

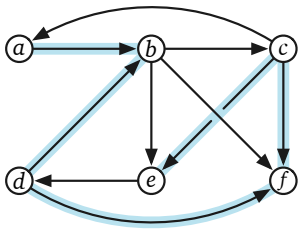
17	40	23	72	39	88	62	13	40	53	92	21	10	73	68

i: 1 2 3 4 ...

5. A *zigzag walk* in in a directed graph G is a sequence of vertices connected by edges in G, but the edges alternately point forward and backward along the sequence. Specifically, the first edge points forward, the second edge points backward, and so on. The *length* of a zigzag walk is the sum of the weights of its edges, both forward and backward.

   For example, the following graph contains the zigzag walk $a \to b \leftarrow d \to f \leftarrow c \to e$. Assuming every edge in the graph has weight 1, this zigzag walk has length 5.

Graph Reduction.



   Suppose you are given a directed graph G with non-negatively weighted edges, along with two vertices s and t. Describe and analyze an algorithm to find the shortest zigzag walk from s to t in G.

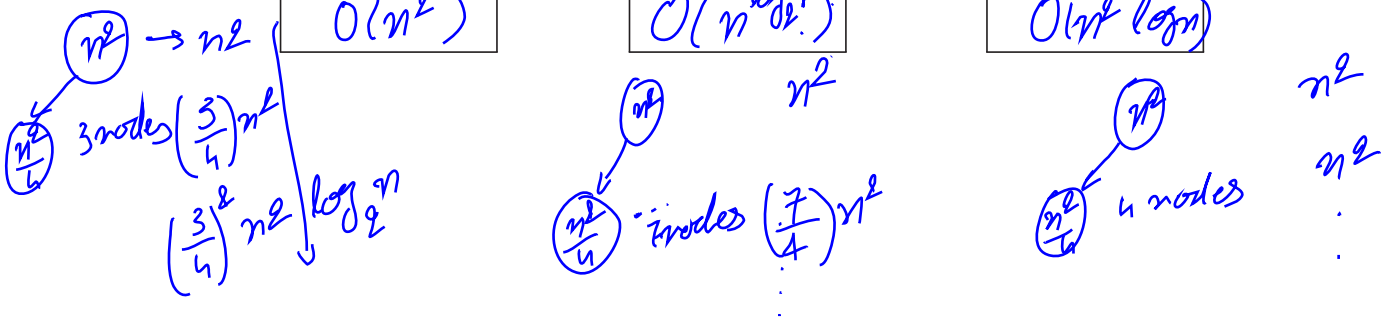
This sentence contains four and three fourths percent a's, five and one half percent c's, three percent d's, eighteen and one fourth percent e's, four and one half percent f's, three fourths percent g's, five percent h's, two and one half percent i's, two percent l's, twelve and one half percent n's, 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 1 Problem 1	Name: RUTA MEHTA

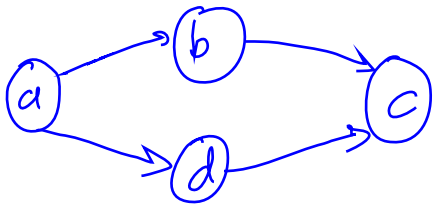
(a) Write the solution to each of the following recurrences in the box immediately below it. (Use the space below the boxes for scratch work.)

$A(n) = 3A(n/2) + O(n^2)$ $B(n) = 7B(n/2) + O(n^2)$ $C(n) = 4C(n/2) + O(n^2)$

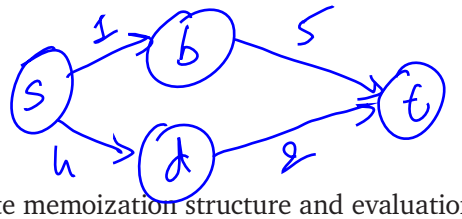
$O(n^2)$ $O(n^{\log_2 7})$ $O(n^2 \log n)$



(b) Draw a directed acyclic graph with at most ten vertices, exactly one source, exactly one sink, and more than one topological order.



(c) Draw a directed graph with at most ten vertices, with distinct positive edge weights, that has more than one shortest path from some vertex s to some other vertex t.

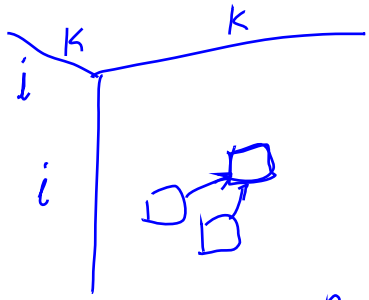


(d) Describe an appropriate memoization structure and evaluation order for the following (meaningless) recurrence, and give the running time of the resulting iterative algorithm to compute $Huh(1, n)$.

$$Huh(i,k) = \begin{cases} 0 & \text{if } i > n \text{ or } k < 0 \\ \min\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} + A[i,k] & \text{if } A[i,k] \text{ is even} \\ \max\begin{Bmatrix} Huh(i+1,k-2) \\ Huh(i+2,k-1) \end{Bmatrix} - A[i,k] & \text{if } A[i,k] \text{ is odd} \end{cases}$$

O(1)

Huh[i, k]
$1 \le i \le (n+2)$
$-2 \le k \le n$



Eval Order:
Decreasing in i
OR
Increasing in k.

Runtime: $O(n^2)$

CS/ECE 374 A ✧ Spring 2026 Midterm 2 Practice 1 Problem 2	Name: RUTA MEHTA

See the question sheet for a detailed description of your game with Elmo.

(a) ***Prove*** that you should not also use the greedy strategy. That is, show that there is a game that you can win, but only if you do *not* follow the same greedy strategy as Elmo. Assume Elmo plays first.

(b) Describe and analyze an algorithm to determine, given the initial sequence of cards, the maximum number of points that you can collect playing against Elmo.

---

a) 2, 1, 100, 10, -1

★ I play Greedy
Elmo: 2, 100, -1 → 101
Me: 1, 10 → 11
Elmo wins

★ Better:
Elmo: 2 + 10 + 1 = 13
Me: -1 + 100 = 99
I win!

b) Dynamic Programming:

Subproblems: MaxP(i,j) = Max points I can collect when the seq. of cards is A[i] .... A[j] at my turn.
$1 \le i, j \le n$

Base case: MaxP(i,j) = 0 if $i > j$
= A[i] if $i = j$

Recursion:
$$MaxP(i,j) = \max \begin{cases} A[i] + \begin{cases} MaxP(i+2, j) & \text{if } A[i+1] > A[j] \\ MaxP(i+1, j-1) & \text{o.w.} \end{cases} \\ A[j] + \begin{cases} MaxP(i+1, j-1) & \text{if } A[i] > A[j-1] \\ MaxP(i, j-2) & \text{o.w.} \end{cases} \end{cases}$$

Ans: MaxP(1, n)

Evaluation Order: MaxP[i,j] = MaxP(i,j)
Decreasing in i from n to 1
& within that, Increasing in j from 1 to n

Runtime: $O(n^2)$

CS/ECE 374 A ✧ Spring 2026 Midterm 2 Practice 1 Problem 3	Name: RUTA MEHTA

Suppose you are given a directed graph $G = (V, E)$, whose vertices are either red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. The ***remoteness*** of a vertex v is the *maximum* of three shortest-path lengths:

- The length of a shortest path to v from the closest red vertex
- The length of a shortest path to v from the closest blue vertex
- The length of a shortest path to v from the closest green vertex

In particular, if v is not reachable from vertices of all three colors, then v is infinitely remote. Describe and analyze an algorithm to find a vertex of G with *minimum* remoteness.

$G' = (V', E')$; $V' = V \cup \{s_R, s_B, s_G\}$

$m = |E|$, $n = |V|$

$E' = E \cup \{(s_R, v) \mid v \in V \text{ w/ color Red}\}$
$\cup \{(s_B, v) \mid v \in V \text{ w/ color Blue}\}$
$\cup \{(s_G, v) \mid v \in V \text{ w/ color Green}\}$

Alg:

$O(m+n)$ ① construct G' as above

$O(m+n)$ ② Run BFS(s_R), BFS(s_B), BFS(s_G) to compute $dist(s_R, v)$, $dist(s_B, v)$, $dist(s_G, v)$ $\forall v \in V$.

$O(n)$ ③ For each $v \in V$ do{

④ }

$Remoteness(v) = \max\{dist(s_R, v), dist(s_B, v), dist(s_G, v)\} - 1$

$O(n)$ ⑤ Return vertex v with minimum $Remoteness(v)$.

Runtime: $O(m+n) \equiv O(E+V)$

CS/ECE 374 A ✧ Spring 2026 Midterm 2 Practice 1 Problem 4	Name: RUTA MEHTA

Suppose you are given an array $A[1..n]$ of integers such that $A[i]+A[i+1]$ is even for *exactly one* index i. In other words, the elements of A alternate between even and odd, except for exactly one adjacent pair that are either both even or both odd. Describe and analyze an efficient algorithm to find the unique index i such that $A[i]+A[i+1]$ is even.

Observation: $(A[1]+1) \bmod 2 = (A[k]+k) \bmod 2$ $\forall k \le i$

We will do binary search based on the above observation.

Al'g:

```
① high=n; low = 1
② while (low < high+1) do{
③       mid = ⌈(low+high)/2⌉
④       If (mid + A[mid]) mod 2 = (1 + A[1]) mod 2
⑤        then    low ← mid
⑥        else    high ← mid
⑦ }
⑧ Return low;
```

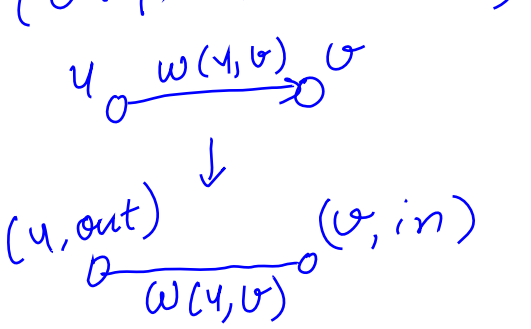
Runtime: $O(\log n)$

CS/ECE 374 A ✧ Spring 2026 Midterm 2 Practice 1 Problem 5	Name: RUTA MEHTA

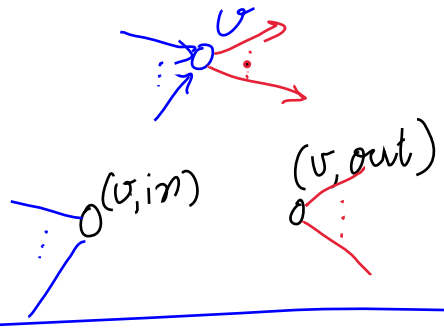
A *zigzag walk* in in a directed graph G is a sequence of vertices connected by edges in G, but the edges alternately point forward and backward along the sequence. Specifically, the first edge points forward, the second edge points backward, and so on. The *length* of a zigzag walk is the sum of the weights of its edges, both forward and backward.

Suppose you are given a directed graph G with non-negatively weighted edges, along with two vertices s and t. Describe and analyze an algorithm to find the shortest zigzag walk from s to t in G.

(Graph Reduction)



Observation:



① Construct undirected graph w/ non-neg edge weights

$G' = (V', E')$; $V' = \{(v, in), (v, out) \mid v \in V\}$

$E' = \{(u, out) - (v, in) \mid u \to v \in E\}$

$w((u, out) - (v, in)) = w(u, v)$

$O(V+E)$

② Run Dijkstra with start = (s, out) } $O(E \log V)$

③ Return $\min\{dist((t, in)), dist((t, out))\}$

Runtime: $O(V+E) + O(E \log V) = O(E \log V)$

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)

(scratch paper)