# CS/ECE 374 A ✧ Fall 2021

# Midterm 2

**November 8, 2021**

---

## Directions

- ***Don't panic!***
- If you brought anything except your writing implements, your hand-written double-sided 8½" × 11" cheat sheet, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
- **We *strongly* recommend reading the entire exam before trying to solve anything.** If you think a question is unclear or ambiguous, please ask for clarification as soon as possible.
- The exam has five numbered questions, each worth 10 points. (Subproblems are not necessarily worth the same number of points.)
- Write your answers on blank white paper using a dark pen. Please start your solution to each numbered question on a new sheet of paper.
- You have **120 minutes** to write your solutions, after which you have 30 minutes to scan your solutions, convert your scan to a PDF file, and upload your PDF file to Gradescope.
- If you are ready to scan your solutions before 9:00pm, send a private message to the host of your Zoom call ("Ready to scan") and wait for confirmation before leaving the Zoom call.
- Gradescope will only accept PDF submissions. Please do not scan your cheat sheets or scratch paper. Please make sure your solution to each numbered problem starts on a new page of your PDF file. **Low-quality scans will be penalized.**
- Proofs are required for full credit if and only if we explicitly ask for them, using the word ***prove*** in bold italics.
- Finally, if something goes seriously wrong, send email to jeffe@illinois.edu as soon as possible explaining the situation. If you have already finished the exam but cannot submit to Gradescope for some reason, include a complete scan of your exam **as a PDF file** in your email. If you are in the middle of the exam, send Jeff email, continue working until the time limit, and then send a second email with your completed exam **as a PDF file**. Please do not email raw photos.

---

1. Short answers:

   (a) Solve the recurrence $T(n) = 2T(n/3) + O(\sqrt{n})$.

   (b) Solve the recurrence $T(n) = 2T(n/7) + O(\sqrt{n})$.

   (c) Solve the recurrence $T(n) = 2T(n/4) + O(\sqrt{n})$.

   (d) Draw a connected undirected graph $G$ with at most ten vertices, such that every vertex has degree at least 2, and no spanning tree of $G$ is a path.

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

   (f) 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 $Pibby(1, n)$. (Assume all array accesses are legal.)

$$Pibby(i,k) = \begin{cases} 0 & \text{if } i > k \\ A[i] & \text{if } i = k \\ Pibby(i+1,k-1) + A[i] + A[k] & \text{if } A[i] = A[k] \\ \max \left\{ \begin{array}{c} Pibby(i+2,k), \\ Pibby(i+1,k-1), \\ Pibby(i,k-2) \end{array} \right\} & \text{otherwise} \end{cases}$$

2. Your company has two offices, one in San Francisco and the other in New York. Each week you decide whether you want to work in the San Francisco office or in the New York office. Depending on the week, your company makes more money by having you work at one office or the other. You are given a schedule of the profits you can earn at each office for the next $n$ weeks. You'd obviously prefer to work each week in the location with higher profit, but there's a catch: Flying from one city to the other costs \$1000. Your task is to design a travel schedule for the next $n$ weeks that yields the maximum *total* profit, assuming you start in San Francisco.

   For example: suppose you are given the following schedule:

| | | | | | |
|---|---|---|---|---|---|
| SF | \$800 | \$200 | \$500 | \$400 | \$1200 |
| NY | \$300 | \$900 | \$700 | \$2000 | \$200 |

   If you spend the first week in San Francisco, the next three weeks in New York, and the last week in San Francisco, your total profit for those five weeks is $\$800 - \$1000 + \$900 + \$700 + \$2000 - \$1000 + \$1200 = \$3600$.

   (a) ***Prove*** that the obvious greedy strategy (each week, fly to the city with more profit) does not always yield the maximum total profit.

   (b) Describe and analyze an algorithm to compute the maximum total profit you can earn, assuming you start in San Francisco. The input to your algorithm is a pair of arrays $NY[1..n]$ and $SF[1..n]$, containing the profits in each city for each week.

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:

   - 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 $G$ whose remoteness is *smallest*.

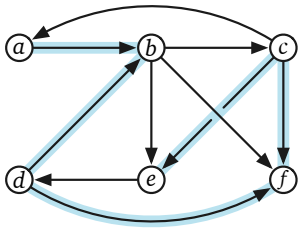
4. Suppose you are given an array $A[1..n]$ of integers such that $A[i] + A[i+1]$ is even for *exactly one* index $i$. In other words, the elements of $A$ alternate between even and odd, except for exactly one adjacent pair that are either both even or both odd.

   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.)

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

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.




   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 $G$.