CS/ECE 374 A ✧ Fall 2026
Conflict Midterm 2 Problem 1 Solution

(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.)

$A(n) = 5A(n/3) + O(n)$ $B(n) = 9B(n/3) + O(n^2)$ $C(n) = C(n/2) + C(n/3) + O(n)$

$O(n^{\log_3 5})$	$O(n^2 \log n)$	$O(n)$

Solution: Level ℓ of the recursion tree for $A(n)$ has 5^ℓ nodes, each with value $n/3^\ell$, so the total value at level ℓ is $(5/3)^\ell n$. The level sums form an increasing geometric series, so only the number of leaves matters. The depth of the tree is $\log_3 n$, so $A(n) = O(5^{\log_3 n}) = O(n^{\log_3 5})$.

Level ℓ of the recursion tree for $B(n)$ has 9^ℓ nodes, each with value $n^2/9^\ell$, so the total value is n^2. Because every level has total value at most n^2, and the depth of the tree is $O(\log n)$, we have $B(n) = O(n^2 \log n)$.

The recursion tree for $C(n)$ is a ternary tree with unbalanced problem sizes. The root has value n. The root has two children, one with value $n/2$ and one with value $n/3$, so the total value of the children is $5n/6$. More generally, level ℓ of the recursion tree for $C(n)$ has total value at most $(5/6)^\ell n$. The level sums form a decreasing geometric series, so only the root value n matters. (Fans of the Master Theorem should notice that it cannot be used to solve this recurrence. If you've never heard of the Master Theorem, you're not missing much.) ■

Rubric:

2.5 points each for $A(n)$ and $C(n)$.

2 points for $B(n)$.

Explanations and/or recursion tree drawings are not required for full credit, only the final answers in the boxes.

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

$$Bar(i,j,k) = \begin{cases} 0 & \text{if } i > n,\ j < 1, \text{ or } k < 1 \\ \max \begin{Bmatrix} A[i] \cdot B[j] + Bar(i+1, j-1, k) \\ A[i] \cdot C[k] + Bar(i+1, j, k-1) \\ B[j] \cdot C[k] + Bar(i, j-1, k-1) \end{Bmatrix} & \text{otherwise} \end{cases}$$

Solution: Three-dimensional array, filled in *decreasing* order of i and *increasing* order of j and k with the loops nested in any order, in $O(n^3)$ time. ■

Rubric: 3 points = 1 for correct structure + 1 for correct evaluation order + 1 for running time.

CS/ECE 374 A ✧ Fall 2026
Conflict Midterm 2 Problem 2 Solution

Describe an algorithm that, given a sequence of n pairs of integers $P[1..n]$, computes the maximum shininess of any semi-increasing subsequence of P. You can use $P[i].a$ to denote the first component of pair $P[i]$ and $P[i].b$ to denote the second component of $P[i]$.

Solution: Let $SSIS(i)$ be the maximum shininess of any semi-increasing subsequence of $P[i..n]$ *that begins with* $P[i]$. We need to compute $\max\{SSIS(i) \mid 1 \le i \le n\}$. The function obeys the recurrence

$$SSIS(i) = \max\left\{\begin{array}{c} 1, \\ 1 + \max\{SSIS(j) \mid i < j \le n \text{ and } (P[j].a > P[i].a \textbf{ xor } P[j].b > P[i].b)\}, \\ 2 + \max\{SSIS(j) \mid i < j \le n \text{ and } (P[j].a > P[i].a \textbf{ and } P[j].b > P[i].b)\} \end{array}\right\}$$

where $\max\varnothing := -\infty$.

We can memoize this recurrence using an array $SSIS[1..n]$. We fill it in decreasing order of i in $O(n^2)$ time. (Evaluating each entry $SSIS[i]$ requires a for loop over j.) ■

Rubric: 10 points: standard dynamic programming rubric.

CS/ECE 374 A ✧ Fall 2026
Conflict Midterm 2 Problem 3 Solution

Describe an algorithm that, given an integer k, a dag $G = (V, E)$, and the *Damage* function as described in the questions handout, computes the minimum number of times the PC must pop open a Sprite Bottle while following some path from s to b.

Solution (dynamic programming): We'll return ∞ if there is no path from s to b.

Let $MinPops(u, \ell)$ be the minimum number of times the PC must pop open a Sprite Bottle while following a path from u to b assuming they start with ℓ health (not counting any damage caused at u itself). We need to compute $MinPops(s, k)$. The function obeys the recurrence

$$MinPops(u,\ell) = \begin{cases} 0 & \text{if } u = b \\ \min\limits_{u \to v} \left(\begin{cases} MinPops(v, \ell - Damage(v)) & \text{if } Damage(v) < \ell \\ 1 + MinPops(v, k/2) & \text{otherwise} \end{cases} \right) & \text{otherwise} \end{cases}$$

where $\max \varnothing := \infty$.

We can memoize this function by giving each vertex u an array $u.MinPops[1..k]$. We can evaluate the subproblems using an outer loop over ℓ increasing and postorder over the vertices. The running time is $O(k(|V| + |E|))$.

■

Rubric: 10 points: standard dynamic programming rubric (scaled).

Solution (graph reduction): We reduce the problem to shortest paths in a dag. We build a directed graph $G' = (V', E')$ with edge weights $w' : E \to \mathbb{R}$ where

- $V' := V \times \{1, \ldots, k\}$. Vertex (v, ℓ) represents reaching v from s and finishing at v with ℓ health remaining.
- $E' := \{(u,\ell) \to (v, \ell - Damage(v)) \mid u \to v \in E, Damage(v) < \ell\}$
  $\cup \{(u,\ell) \to (v, k/2) \mid u \to v \in E, Damage(v) \geq \ell\}$
- $w'((u,\ell) \to (v, \ell - Damage(v))) := 0$ when $u \to v \in E$ and $Damage(v) < \ell$.
- $w'((u,\ell) \to (v, k/2) := 1$ when $u \to v \in E$ and $Damage(v) \geq \ell$.

The minimum number of bottle pops going from s to b is the distance from (s, k) to any (b, ℓ). Graph G' is a dag, so we run DagSSSP$((s, k))$ and return $\min\{(b, \ell).dist \mid 1 \leq \ell \leq k\}$. The running time is $O(|V'| + |E'|) = O(k(|V| + |E|))$. ■

Rubric: 10 points: standard graph reduction rubric (scaled).

CS/ECE 374 A ✧ Fall 2026
Conflict Midterm 2 Problem 4 Solution

A collection of n RAM slots have IDs given in an array $R[1..n]$. Exactly **two** RAM slots are broken. You have access to a subroutine $\textsc{Test}(X[1..k])$ that takes an array of k RAM slot IDs and returns $\textsc{True}$ if X contains the ID of *at least one* broken slot and $\textsc{False}$ otherwise. Describe an algorithm to find the IDs of the broken slots. **Your running time should be the number of times you call $\textsc{Test}$ (boot the server).**

Solution: The procedure $\textsc{PrintBroken}(R[1..n])$ takes an array of RAM slot IDs. It "prints" the IDs in R for broken slots. We need to run $\textsc{PrintBroken}(R[1..n])$.

```
PrintBroken(R[1..n]):
  if n = 1 and Test(R[1..1])
      print R[1]
      return
  m ← ⌈n/2⌉
  if Test(R[1..m])
      PrintBroken(R[1..m])
  if Test(R[m+1..n])
      PrintBroken(R[m+1..n])
```

With our case of two broken RAM slots, the recursion tree looks like a path down to a single node with two children, and the subtrees of those children are themselves paths. The problem sizes half between each parent and child, so all three paths have length $O(\log n)$. There are a constant number of calls to $\textsc{Test}$ for each subproblem, so the running time is $O(\log n)$. ■

Rubric: 10 points: standard recursion rubric.

CS/ECE 374 A ✧ Fall 2026
Conflict Midterm 2 Problem 5 Solution

You have a map of your village given as an undirected graph $G = (V, E)$ where V is the set of islands and E is the set of bridges. The map also describes a function $c : E \to \mathbb{R}^+$ where $c(e)$ is the *cost* of crossing bridge e. Describe an algorithm to compute the minimum cost for traveling from your home on island $s \in V$ to the market on island $t \in V$.

Solution: We build a *directed* graph $G' = (V', E')$ with non-negative edge weights $w' : E \to \mathbb{R}^+$ as follows:

- $V := V \times \{0, 1, 2\}$. Vertex (v, x) means you're on island v and must take the next x bridge crossings for free.
- $E := \{(u,0)\to(v,2) \mid uv \in E\}$
  $\cup \{(u,1)\to(v,0) \mid uv \in E\}$
  $\cup \{(u,2)\to(v,1) \mid uv \in E\}$
- $w'((u,0)\to(v,2)) := c(uv)$ for all $uv \in E$.
- $w'((u,1)\to(v,0)) := 0$ for all $uv \in E$.
- $w'((u,2)\to(v,1)) := 0$ for all $uv \in E$.

The minimum cost for traveling from island s to island t is the minimum distance from $(s, 0)$ to any (t, x) in G'. Therefore, we run DIJKSTRA$((s, 0))$ and return $\min_x(t, x).dist$. The running time is $O(|E'| \log |V'|) = O(|E| \log |V|)$. ■

Rubric: 10 points: standard graph reduction rubric.