

**CS/ECE 374 A ✦ Spring 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.)

$$X(n) = 9X(n/3) + O(n^2) \quad Y(n) = 4Y(n/3) + O(n) \quad Z(n) = Z(n/3) + 3Z(n/6) + O(n)$$

$$O(n^2 \log(n))$$

$$O(n^{\log_3 4})$$

$$O(n)$$

**Solution:** Level  $\ell$  of the recursion tree for  $X(n)$  has  $9^\ell$  nodes, each with value  $n^2/3^{2\ell}$ , so the total value at level  $\ell$  is  $n^2$ . The depth of the tree is  $\log_3 n$ , so  $X(n) = O(n^2 \log_3 n) = O(n^2 \log(n))$ .

Level  $\ell$  of the recursion tree for  $Y(n)$  has  $4^\ell$  nodes, each with value  $n/3^\ell$ . So the total value at level  $\ell$  is  $(4/3)^\ell n$ . This forms an increasing geometric sequence, and hence its sum is dominated by the value at the leaf level (highest level). The depth of the tree is  $\log_3 n$ , so  $Y(n) = O((4/3)^{\log_3 n} n) = O(n^{\log_3 4/3} n) = O(n^{\log_3 4})$ .

The recursion tree for  $Z(n)$  is a quadtree with unbalanced problem sizes. The root has value  $n$ , and it has four children, one with value  $n/3$  and three with value  $n/6$ , so the total value of the children is  $5n/6$ . More generally, level  $\ell$  of the recursion tree for  $Z(n)$  has total value  $(5/6)^\ell n$ . 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 points for  $X(n)$ .

2.5 points each for  $Y(n)$  and  $Z(n)$ .

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 computing  $BestBobble(1, n)$  using the following recurrence, and state the running time of the resulting iterative algorithm.

$$BestBobble(i, k) = \begin{cases} 0 & \text{if } i \geq k \\ \min \left\{ \begin{array}{l} BestBobble(i, j-1) \\ + i \cdot j \cdot A[j] \\ + BestBobble(j, k) \end{array} \middle| i < j \leq k \right\} & \text{otherwise} \end{cases}$$

**Solution:** Two-dimensional array, filled in *decreasing* order of  $i$  and *increasing* order of  $k$  with the loops nested either way, in  $O(n^3)$  time. (Evaluating each entry  $BestBobble[i, k]$  requires a for loops over  $j$ .) ■

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

**CS/ECE 374 A ✦ Spring 2026**  
**Conflict Midterm 2 Problem 2 Solution**

Suppose you are given a sorted array of  $n$  distinct numbers that has been rotated *forward*  $k$  steps for some **unknown** integer  $k$  between 1 and  $n - 1$ . In other words, you are given an array  $A[1..n]$  such that some prefix  $A[1..k]$  is sorted in increasing order, the complementary suffix  $A[k + 1..n]$  is sorted in increasing order, and  $A[k + 1] < A[1]$ .

Describe an algorithm to determine the unknown number  $k$ . (See the questions handout for an example of the input.)

**Solution:** Note that, for each  $i \leq k$ ,  $A[i] > A[n]$ . We will use this fact to do a binary search to find the index  $1 \leq k < n$ .

```
ROTINDEX(A[1..n]):  
  if n = 2  
    return 1  
  low ← 1, high ← n.  
  while low < (high - 1)  
    m ← ⌊(low + high)/2⌋  
    if A[m] > A[m + 1] ⟨⟨Found k⟩⟩  
      return m;  
    if A[m] > A[n] ⟨⟨1 < m ≤ k⟩⟩  
      low ← m  
    else  
      high ← m  
  return low
```

Since the prefix and suffix of  $A$ , before and after the  $k$ th element respectively, are sorted in increasing order, only at the  $k$ th index the value decreases in the array. That is,  $A[k] > A[k + 1]$ .

Pivots  $low$  and  $high$  are initialized to 1 and  $n$ , and  $low \leq k < high$  is always maintained. The middle element between  $low$  and  $high$  is  $m$ , and the algorithm correctly returns  $k = m$  if  $A[m] > A[m + 1]$ .

Otherwise, using the fact that for all  $i \leq k$ ,  $A[i] > A[n]$ , the algorithm moves the  $low$  or the  $high$  pivot. That is, for the middle element  $A[m]$ , if  $A[m] > A[n]$  then  $k$  must be in the right-half, and hence  $low$  pivot is moved to  $m$ . Else  $k$  must be in the left-half, and the  $high$  pivot is moved to  $m$ .

The  $(high - low)$ , starting with  $(n - 1)$ , is reduced by half in every iteration of the while loop. And it is lower bounded by 2. Hence the running time of the algorithm is  $O(\log n)$ . ■

**Rubric:** 10 points: standard recursive algorithm rubric. The proofs in gray are not required for full credit. An equivalent recursive algorithm is worth full credit. A correct  $O(n)$ -time algorithm is worth 3 points.

**CS/ECE 374 A ♦ Spring 2026**  
**Conflict Midterm 2 Problem 3 Solution**

Describe an algorithm that given a list of positive integers, computes the minimum number of turns that can be taken before completing the game described in the questions handout. You may assume the input to your algorithm is an array  $A[1..n]$  where  $A[i]$  is the  $i$ th integer from the left in the list.

**Solution (dynamic programming):**

1. **Subproblem Definition.** For all  $1 \leq i, j \leq n$ , define

$MinTurns(i, j) :=$  minimum number of turns required for the input subarray  $A[i..j]$ , where  $i > j$  implies the input is empty.

2. **Recursive Formula.**

$$MinTurns(i, j) = \begin{cases} 0 & \text{if } i > j \\ 1 + \min\{MinTurns(i + A[j], j), MinTurns(i, j - A[i])\} & \text{otherwise} \end{cases}$$

3. **Evaluation Order.** Define 2D array  $MT[1..n][1..n]$ , where  $MT[i, j]$  store the value of  $MinTurns(i, j)$ . Evaluate the array in increasing order of  $j$  in the outer loop and decreasing order of  $i$  in the inner loop. Alternatively, decreasing order of  $i$  in the outer loop and increasing order of  $j$  in the inner loop also works.

The resulting iterative algorithm takes  $O(n^2)$  time total.

**Final answer:** Output  $MT[1, n]$ . ■

**Rubric:** 10 points: standard dynamic programming rubric.

**CS/ECE 374 A ✦ Spring 2026**  
**Conflict Midterm 2 Problem 4 Solution**

Describe an algorithm to compute the minimum cost to get from your friend's house at vertex  $s$  to the Transit Authority at vertex  $t$ .

(See the questions handout for a full description of the problem.)

**Solution:** We reduce to shortest paths in a two new directed graphs  $G' = (V', E')$  and  $G'' = (V'', E'')$  with positive edge lengths  $\ell' : E' \rightarrow \mathbb{R}^+$  and  $\ell'' : E'' \rightarrow \mathbb{R}^+$  as follows.

- $V' := V \times \{X, Y\}$  and  $V'' := V \times \{X, Y\}$ . Vertex  $(v, a)$  means we're at station  $v$  while appearing to be a member of System  $a$ .
- $E' := \{((u, a), (v, a)) \mid (u, v) \in E, a \in \{X, Y\}\} \cup \{((v, X), (v, Y)) \mid v \in V\}$ , and  
 $E'' := \{((u, a), (v, a)) \mid (u, v) \in E, a \in \{X, Y\}\} \cup \{((v, Y), (v, X)) \mid v \in V\}$
- $\ell'(((u, a), (v, a))) := \$(u, v)/2$  if  $system((u, v)) = a$ , and  $\ell''(((u, a), (v, a))) := \$(u, v)/2$  if  $system((u, v)) = a$ .
- $\ell'(((u, a), (v, a))) := \$(u, v)$  if  $system((u, v)) \neq a$ , and  $\ell''(((u, a), (v, a))) := \$(u, v)$  if  $system((u, v)) \neq a$ .
- $\ell'(((v, X), (v, Y))) = 0$  and  $\ell''(((v, Y), (v, X))) = 0$  for all edges  $v \in V$ .

The minimum cost to get from  $s$  to  $t$  in  $G$  is equal to the minimum of two distances: (i) distance from  $(s, X)$  to  $(t, Y)$  in  $G'$ , and (ii) distance from  $(s, Y)$  to  $(t, X)$  in  $G''$ . Therefore, we run Dijkstra's algorithm twice: once in  $G'$  from  $(s, X)$  to compute  $d_{G'}((t, Y))$ , and next in  $G''$  from  $(s, Y)$  to compute  $d_{G''}((t, X))$ . Finally, we return  $\min\{d_{G'}((t, Y)), d_{G''}((t, X))\}$ .

Our algorithm runs in  $O(|E'| \log |V'|) + O(|E''| \log |V''|) = O(m \log n)$  time. ■

**Rubric:** 10 points: standard graph reduction rubric.

**CS/ECE 374 A ✦ Spring 2026**  
**Conflict Midterm 2 Problem 5 Solution**

Suppose we are given an  $n \times n$  grid containing the points  $\{(i, j) \mid 1 \leq i \leq n, 1 \leq j \leq n\}$ , where some grid nodes are marked as dangerous.  $\text{DANGEROUS}(i, j)$  returns TRUE if grid point  $(i, j)$  is dangerous, and it returns FALSE otherwise.

From every grid point  $(i, j)$ , there is an edge to the right and above the grid point, namely to  $(i + 1, j)$  and  $(i, j + 1)$ . A path is called *safe* if it contains no dangerous nodes.

Describe an algorithm to find the number of safe paths from node  $(1, 1)$  to node  $(n, n)$ .

(See the questions handout for examples of the input.)

**Solution (dynamic programming):**

Pre-processing step: If  $\text{DANGEROUS}(1, 1)$  or  $\text{DANGEROUS}(n, n)$  returns TRUE then no safe paths from node  $(1, 1)$  to  $(n, n)$  and hence output zero. Else do the following DP.

Although, the algorithm works without this pre-processing step as well.

- Subproblem Definition.** For all  $1 \leq i, j \leq n$ , define

$\text{MaxPaths}(i, j) :=$  maximum number of paths from  $(i, j)$  to  $(n, n)$ .

- Recursive Formula.**

$$\text{MaxPaths}(i, j) = \begin{cases} 0 & \text{if } i = j = n \\ 0 & \text{if } \text{DANGEROUS}(i, j) = \text{TRUE} \\ \text{MaxPaths}(i + 1, j) + \text{MaxPaths}(i, j + 1) & \text{if } 1 \leq i, j < n \\ \text{MaxPaths}(i + 1, j) & \text{if } j = n \\ \text{MaxPaths}(i, j + 1) & \text{otherwise} \end{cases}$$

- Evaluation Order.** Define 2D array  $MP[1..n][1..n]$ , where  $MP[i, j]$  store the value of  $\text{MaxPaths}(i, j)$ . First evaluate the base cases by going over all the  $(i, j)$  pairs, and setting  $MP(i, j)$  to zero if  $\text{DANGEROUS}(i, j) = \text{TRUE}$ . And set  $MP(n, n) = 0$ .

Evaluate the array in decreasing order of  $i$  in the outer loop and decreasing order of  $j$  in the inner loop. Alternatively, decreasing order of  $j$  in the outer loop and decreasing order of  $i$  in the inner loop also works.

The resulting iterative algorithm takes  $O(n^2)$  time total.

**Final answer:** Output  $MP[1, 1]$ . ■

**Rubric:** 10 points: standard dynamic programming rubric.