1. Short answers:

(a) Solve the recurrence $T(n) = 3T(n/2) + O(n^2)$.

Solution: $T(n) = O(n^2)$.

The ith level of the recursion tree sums to $3^i(n/2^i)^2 = n^2 \cdot (3/4)^i$. So the level sums form a descending geometric series; only value of the root matters. ■

Rubric: 1 point.

(b) Solve the recurrence $T(n) = 7T(n/2) + O(n^2)$.

Solution: $T(n) = O(n^{\log_2 7}) = O(n^{2.80735})$.

The ith level of the recursion tree sums to $7^i(n/2^i)^2 = n^2 \cdot (7/4)^i$. So the level sums form a ascending geometric series; only the number of leaves matters. The recursion tree has depth $\log_2 n$ and each level ℓ has 7^ℓ nodes, so the number of leaves is $7^{\log_2 n} = n^{\log_2 7}$. ■

Rubric: 1 point.

(c) Solve the recurrence $T(n) = 4T(n/2) + O(n^2)$.

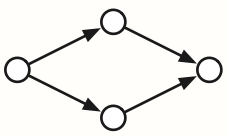
Solution: $T(n) = O(n^2 \log n)$.

Every level of the recursion tree sums to n^2 and there are $\log_2 n$ levels. ■

Rubric: 1 point.

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

Solution: Here is the simplest example:

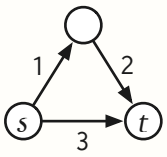


■

Rubric: 2 points. This is not the only correct solution.

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

Solution: Here is a simple example:



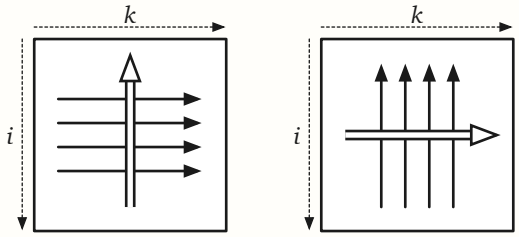
■

Rubric: 2 points. This is not the only correct solution.

(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 $Huh(1, n)$.

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

Solution: $O(n^2)$ time. Here are two valid evaluation orders:



■

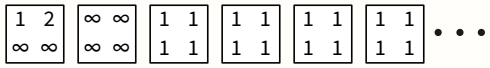
Solution: We can memoize Huh into a two-dimensional array. We can fill this array in $O(n^2)$ time using two nested loops, one decreasing i and the other increasing k. (The nesting order doesn't matter.) ■

Rubric: 3 points = 1 for structure + 1 for evaluation order + 1 for time bound. These are not the only correct solutions.

2. *Quadhopper* is a solitaire game played on a row of n squares. Each square contains four positive integers. The player begins by placing a token on the leftmost square. On each move, the player chooses one of the numbers on the token's current square, and then moves the token that number of squares to the right. The game ends when the token moves past the rightmost square. The object of the game is to make as many moves as possible before the game ends.

   (a) ***Prove*** that the obvious greedy strategy (always choose the smallest number) does not give the largest possible number of moves for every Quadhopper puzzle.

Solution: Consider the following Quadhopper puzzle. Every square except the first two contains four 1s. (If you don't like ∞s, replace then with any integer larger than n.)



The greedy algorithm makes only two moves: first 1 and then ∞.
But if we choose 2 in the first move, we can actually make $n-1$ moves. ■

Rubric: 2 points. This is not the only correct solution, but every correct solution requires an explicit counterexample.

(b) Describe and analyze an efficient algorithm to find the largest possible number of legal moves for a given Quadhopper puzzle.

Solution (dynamic programming): Suppose the given puzzle is represented by an $n \times 4$ array $Q[1..n, 1..4]$, where $Q[i, j]$ is the jth number in the ith square.

Let $MaxMoves(i)$ denote the maximum number of moves we can make starting from square i. We need to compute $MaxMoves(1)$. This function obeys the following recurrence:

$$MaxMoves(i) = \begin{cases} 0 & \text{if } i > n \\ \max\left\{1 + MaxMoves(i + Q[i, j]) \;\middle|\; 1 \le j \le 4\right\} & \text{otherwise} \end{cases}$$

We can memoize this function into a one-dimensional array $MaxMoves[1..n]$, which we can fill in reverse order in $\boldsymbol{O(n)}$ ***time***. ■

Rubric: 10 points, standard dynamic programming rubric. This is not the only correct solution.

Solution (dag traversal): Suppose the given puzzle is represented by an $n \times 4$ array $Q[1..n, 1..4]$, where $Q[i, j]$ is the jth number in the ith square.

We construct a directed acyclic graph $G = (V, E)$ as follows:

- $V = \{0, 1, 2, \ldots, n\}$; each vertex corresponds to a square.
- $E = \{i \to (i + Q[i, j]) \mid 1 \le j \le 4 \text{ and } i + Q[i, j] \le n\}$

Altogether G has n vertices and at most $4n - 4$ edges. We need to compute the longest path in G starting from node 1. We can compute this longest path in $O(V + E) = \boldsymbol{O(n)}$ ***time*** using the dynamic programming / postorder traversal algorithm described in class. ■

Rubric: 8 points, standard graph reduction rubric (scaled). This is not the only correct graph-based solution.

3. Suppose you are given a directed graph $G = (V, E)$, each of whose vertices is either red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag.

   Describe and analyze an algorithm to find a shortest path in G that contains at least one vertex of each color. (In particular, your algorithm must choose the best start and end vertices for the path.)

Solution: We construct a new graph $G' = (V', E')$ as follows:

- $V' = V \times \{0,1\} \times \{0,1\} \times \{0,1\} \cup \{s\}$.
  - Vertex s is an artificial start vertex.
  - Each vertex of the form (v, r, b, g) means that we are at vertex v in G, we have visited a red vertex iff $r = 1$, we have visited a blue vertex iff $b = 1$, and we have visited a green vertex iff $g = 1$.
- E' has six types of edges:
  - $s \to (v, 1, 0, 0)$ for every red vertex $v \in V$,
  - $s \to (v, 0, 1, 0)$ for every blue vertex $v \in V$,
  - $s \to (v, 0, 0, 1)$ for every green vertex $v \in V$,
  - $(v, r, b, g) \to (w, 1, b, g)$ for every edge $v \to w \in E$ such that w is red,
  - $(v, r, b, g) \to (w, b, 1, g)$ for every edge $v \to w \in E$ such that w is blue,
  - $(v, r, b, g) \to (w, b, b, 1)$ for every edge $v \to w \in E$ such that w is green.

Altogether, G' has $8V + 1$ vertices and $8E + V$ edges.

We need to find the shortest path from s to any vertex of the form $(v, 1, 1, 1)$. We can find this shortest path in $O(V' + E') = \boldsymbol{O(V + E)}$ ***time*** using breadth-first search. ■

Solution: Let's call a path that visits at least one vertex of each color a *rainbow path*. We will exploit two properties of the shortest rainbow path P.

- The first vertex of P is the only vertex in P with its color. (Otherwise, discarding the first vertex of P would give us a shorter rainbow path.)
- The last vertex of P is the only vertex in P with its color. (Otherwise, discarding the last vertex of P would give us a shorter rainbow path.)

We can find a shortest path in G that starts with a red vertex, passes through one or more blue vertices, and then ends with a green vertex. First we construct a graph $G' = (V', E')$ as follows:

- $V' = V \cup \{r\}$, where r is a new vertex.
- E' contains four types of edges:
  - A new edge $r \to w$ for every red vertex w.
  - Every edge $w \to x \in E$ such that w is red and x is blue.
  - Every edge $x \to y \in E$ such that x and y are both blue.
  - Every edge $y \to z \in E$ such that y is blue and z is green.

Then we find a shortest path in G' from r to a green vertex using breadth-first search, and remove the initial edge $r \to w$.

Similar algorithms find shortest rainbow paths with the other five color permutations. At the end, we return the shortest of those six paths. The entire algorithm runs in $O(V+E)$ *time*. ■

Rubric: 10 points: standard graph reduction rubric. These are not the only correct solutions.

4. Your grandmother dies and leaves you her treasured collection of n radioactive Beanie Babies. Her will reveals that one of the Beanie Babies is a rare specimen worth 374 million dollars, but all the others are worthless. All of the Beanie Babies are equally radioactive, except for the valuable Beanie Baby, which is is either slightly more or slightly less radioactive, but you don't know which. Otherwise, as far as you can tell, the Beanie Babies are all identical.

   You have access to a state-of-the-art Radiation Comparator at your job. The Comparator has two chambers. You can place any two disjoint sets of Beanie Babies in Comparator's two chambers; the Comparator will then indicate which subset emits more radiation, or that the two subsets are equally radioactive. (Two subsets are equally radioactive if and only if they contain the same number of Beanie Babies, and they are all worthless.) The Comparator is slow and consumes a *lot* of power, and you really aren't supposed to use it for personal projects, so you *really* want to use it as few times as possible.

   Describe an efficient algorithm to identify the valuable Beanie Baby. How many times does your algorithm use the Comparator in the worst case, as a function of n?

Solution: The problem is trivial when $n = 1$ and impossible to solve when $n = 2$, so assume $n \geq 3$.

If $n < 10$ (or whatever), we can find the valuable BB by brute force. We test each pair of BBs using the Comparator, and keep the only one that is differently radioactive from all the others. This case requires at most $\binom{n}{2} \leq 45 = O(1)$ tests.

Otherwise, we compare any two disjoint subsets of $\lfloor n/4 \rfloor$ BBs. This test has two possible outcomes.

- Suppose the Comparator reports that those two subsets are equally radioactive. Then the valuable BB *cannot* be inside the Comparator. So we recursively search through the $n - 2\lfloor n/4 \rfloor \approx n/2$ BBs outside the Comparator.
- Suppose the Comparator reports that those two subsets are *not* equally radioactive. Then the valuable BB *must* be inside the Comparator. So we recursively search through the $2\lfloor n/4 \rfloor \approx n/2$ BBs inside the Comparator.

Ignoring floors and ceilings, the number of tests satisfies the recurrence $T(n) = 1 + T(n/2)$. We conclude that the algorithm uses $\log_2 n + O(1) = \boldsymbol{O(\log n)}$ ***tests***. ■

Solution: To keep the description simple, assume that n is a power of 3.

Label each BB with a unique string of length $\log_3 n$ over the alphabet $\{0, 1, 2\}$; for example, label the ith BB with the base-3 representation of the integer i. Then for every index i between 1 and $\log_3 n$, we do the following:

- Place all BBs whose ith digit is 0 into the left chamber and all BBs whose ith digit is 2 into the right chamber, and run the Comparator.
- If the Comparator says both chambers are equally radioactive, discard all BBs in both chambers, and then erase the ith digit of the surviving BBs.
- If the Comparator says the left chamber is more radioactive, change the ith digit of every BB in the left chamber to 2, and change the ith digit of every BB in the

right chamber to `0`.

- If the Comparator says the right chamber is more radioactive, do nothing.

At the end of this procedure, every surviving BB still has a unique label. Moreover, if the valuable BB is heavy, its label will consists entirely of `2`s, and if the valuable BB is light, its label will consists entirely of `0`s.

- If there is only one BB whose label is all `0`s or all `2`s, that's the target.
- If there is an all-`0`s BB and an all-`2`s BB, put the all-`0`s BB into the Comparator, opposite any other BB. If the Comparator reports that the all-`0`s BB is less radioactive, that's the target; otherwise, the Comparator must say that the two BBs are equally radioactive, and the all-`2`s BB is the target.

In the worst case, this algorithm uses $\log_3 n + 1 = \boldsymbol{O(\log n)}$ ***tests***. ■

Rubric: 10 points = 2 for binary search pattern + 2 for explicit base case (What does "brute force" actually mean?) + 4 for recursive case + 2 for time analysis. This is not the only correct solution. The second solution is not precisely optimal for all n, but it is within one test of optimal.

- No penalty for assuming $n > 2$
- No penalty for assuming that n is a power of 4, or otherwise ignoring floors and ceilings.
- Max 5 points for an $O(n)$-time algorithm
- Max 3 points for a brute-force $O(n^2)$-time algorithm

This problem is usually posed with coins and a pan balance. Although other kinds of pan-balance problems date back to at least the 16th century, counterfeit coin problems like this one were first published in 1945. Legend has it that during World War II, the Allies dropped leaflets describing these problems behind enemy lines, to distract the Germans.[a]

[a] Wenn ist das Nunstück git und Slotermeyer? Ja! Beiherhund das Oder die Flipperwaldt gersput!

5. Ronnie and Hyde are a professional robber duo who plan to rob a house in the Leverwood neighborhood of Sham-Poobanana. They have managed to obtain a map of the neighborhood in the form of a directed graph G, whose vertices represent houses, whose edges represent one-way streets.

   - One vertex s represents the house that Ronnie and Hyde plan to rob.
   - A set X of special vertices designate eXits from the neighborhood.
   - Each directed edge $u \to v$ has a non-negative weight $w(u \to v)$, indicating the time required to drive directly from house u to house v.
   - Thanks to Leverwood's extensive network of traffic cameras, speeding or driving backwards along any one-way street would mean certain capture.

   Describe and analyze an algorithm to compute the shortest time needed to exit the neighborhood, starting at house s. The input to your algorithm is the directed graph $G = (V, E)$, with clearly marked subset of exit vertices $X \subseteq V$, and non-negative weights $w(u \to v)$ for every edge $u \to v$.

   Solution: We do not need to modify the input graph. First we compute shortest paths from s to every other vertex in the input graph G using Dijkstra's algorithm. Then, using a brute-force scan, we find the vertex $x \in X$ such that $dist(v)$ is minimized. The algorithm runs in $\boldsymbol{O(E \log V)}$ ***time***. ■

   Rubric: 10 points: Standard graph reduction rubric.