# December JDCC Solutions + Comments

## **Problem A** (Countdown):

Solution: Use a loop (or recursion!) to print all the numbers from N to 0.

Time Complexity: O(N)
Space Complexity: O(1)

## **Problem B** (Splitting Presents):

Solution: Notice that if Emily reaches N-(M+1), then no matter Matthew's move she'd be able to take the last present. Likewise, if Matthew reaches N-(M+1), then Matthew wins. Hence, the solution to (N, M) is the same as (N-(M+1), M), which is the same as (N-2(M+1), M), which is the same as (N – 3(M+1), M) and so on. Eventually, the first number will be less than M. If it's not equal to 0, then Emily can take that amount of presents and guarantee victory. If it is 0, then Matthew wins.
This means that to figure out who wins, we need to check whether N = 0 mod (M+1).

Time Complexity: O(1)
Space Complexity: O(1)

## **Problem C** (CCC):

Solution: The naïve solution is to take all the last words, reverse them, and store them in an array. Then, we loop through all pairs of points and sum their rhyme scores which gives us the answer.

Time Complexity: $O(N^2L)$ where $L$ is the average length of a word.
Space Complexity: $O(NL)$

Better Solution: It is possible to improve this runtime by using a trie. We take the last words, reverse them, and store them in the trie. Then, we traverse the trie and if a certain letter is shared by N words, then there are N choose 2 = N(N-1)/2 pairs of lines that contain that letter in their longest common suffix. By summing this value for each node, we obtain the correct answer.

Time Complexity: $O(NL)$ where $L$ is the average length of a word.
Space Complexity: $O(NL)$

## **Problem D** (Skiing):

Solution: This is a fairly simple dynamic programming problem and is solved in a similar way to how Pascal's triangle is constructed. If a skier is at some point (X, Y), then they must have come either from (X-1, Y-1), (X-1, Y), or (X-1, Y+1), so the number of ways to get to (X, Y) is equal to the sum of the number of ways to get to (X-1, Y-1), (X-1, Y), and (X-1, Y+1). We loop through the map, ignoring walls and updating the answer for passable terrain as we go. Then, we sum the last row of the map to get the final answer.

Time Complexity: $O(N^2)$
Space Complexity: $O(N^2)$

## **Problem E** (Fireworks):

Solution: This problem can be solved with difference arrays or coordinate compression.

Difference Arrays are a way to add or subtract a value to every element in a range in O(1) time. Simply put, difference arrays store the difference between consecutive values of an array. By incrementing the difference array at the time the firework goes off and decrementing it at the time it fades, we update the brightness for the entire range. Once all the fireworks are processed, we transform the difference array into a regular array and use the regular array to find the brightest point.

Time Complexity: $O(N+M)$
Space Complexity: $O(N)$

Coordinate compression is a method to reduce the number of coordinates being used. It works by taking out the "gaps" between points, that is if we have a firework light up the sky between t = 2 and t = 5 and another between t = 10 and t = 1000, the only "important" coordinates we have are {2, 5, 10, 1000}. The brightness will be the same at every point between 2 and 5 or 10 and 1000, so there's no need to waste hundreds of array indices storing them. Hence, we can compress the array to make it much smaller. To implement coordinate compression, we store all the times the fireworks go off or fade, then sort them into an array and remove the duplicates. Then, we process all the fireworks using a binary search to find their times inside the compressed array.
If we were to just use coordinate compression and manually increment every element in a range, the complexity would be:

Time Complexity: $O(M^2 \log M)$
Space Complexity: $O(M)$

If we were to use both methods, the complexity would be:

Time Complexity: $O(M \log M)$
Space Complexity: $O(M)$