December JDCC Solutions + Comments

Problem A (Simplify):

Solution: To solve this problem, we need to find the Greatest Common Denominator of N and D. This could be done in linear time using a for loop, or it can be done in logarithmic time using the Euclidean Algorithm. Once we have the GCD, we simply divide N and D by it and output the result.

Time Complexity: O(log N)

Space Complexity: O(1)

Problem B (Broken Telephone):

Solution: We iterate over the words, storing the current word and previous word. We first check if the length of the two are the same, then if they differ by two or more characters. If one of the two cases above happen, we need to carefully handle the output : we need to read until the end of the input, and we also need to make sure we only output “Hooligans” once.

Time Complexity: O(N)

Space Complexity: O(M), M is the length of the words

Problem C (Shoe Rental):

Solution: We greedily assign shoes to people in a way that gives us the maximal answer. We iterate over the shoe sizes, and for each shoe of that size, we first check if someone wants size i-1, then size i, then size i+1, and give the shoe to the first available person. Care needs to be taken to first check i-1 when iterating in increasing order, as otherwise we will not get the best answer.

Time Complexity: O($N+K$) where N is the total number of shoes.

Space Complexity: O(K)

Problem D (Pokemon Woes):

Solution: This is the travelling salesman problem, a famous problem notorious for it's difficulty. The brute force solution, which runs in O(N!) time, tries every possible route to see which is the best one.

We can improve on the brute force by using dynamic programming. We memoize for the pokemon we are currently at, and which pokemon we have visited. Then when we revisit this state, we can simply return our previous result. This cuts down the runtime to $O(N^2 2^N)$, since we have $O(N2^N)$ states and O(N) transitions for each state (we can go to any other pokemon from our current location).

Better solutions exist, however they are outside the scope of a contest.

Time Complexity: $O(N^2 2^N)$
Space Complexity: $O(N2^N)$

Problem E (Supermoon Viewing):

Solution: Solving this problem requires some knowledge of calculus. We find the intersection between the two circles, then integrate over that area (we integrate over the function $\text{sqrt}(r^2 - x^2)$).

If you ever encounter problems such as this and do not know the calculus approach, it is good to try Montecarlo approximation. Essentially, randomize a lot of points within a larger area and check how many of them are within the area you want. Then, use the ratio of in/out along with the ratio of the larger area to approximate the target area. In this case however, that solution would not be accurate enough to receive points.

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