January JDCC Solutions + Comments

Problem A (Triangle Types):

Solution: Very simple case checking. Using if statements To minimize the number of cases, read the input into an array and sort it.

Time Complexity: O(1)

Space Complexity: O(1)

Problem B (The Number Eight):

Solution: Very simple text output. To minimize coding, create methods for the two types of lines to be used (solid or hole), then call them rather than copy-pasting code.

Time Complexity: $O(N^2)$

Space Complexity: O(1)

Problem C (Letter Swap):

Solution: The key insight for this problem is to realize that Suzie should never swap two zeroes or two ones. Hence, the relative order of the zeroes and ones should be maintained. With this in mind, the solution is to reverse the string, and iterate over the string and it’s reverse, and sum the distances between the N^{th} one in the original string and the N^{th} one in the reverse string.

Time Complexity: $O(N)$

Space Complexity: $O(N)$

Problem D (Robot Dodgeball):

Solution: This problem is asking to find a bipartition of a graph. The key insight is that the distance between two robots on the same team is even and the distance between two robots on different teams is odd. With that in mind, a breadth-first-search (depth-first also works) can be used to traverse the graph and find the number of robots at even and odd distances from the root. Be sure to search in both directions (which robots to shoot at and which robots shot at you).

Time Complexity: O(E), where E is the number of edges
Space Complexity: O(E), where E is the number of edges

Problem E (CPT Elections):

Solution: First, begin by noting that the K initial votes don't matter. Andre starting with a lead of K votes is equivalent with Andre not having a lead and ending with N-K votes.

The small cases of $N <= 1000$ can be solved using dynamic programming. Let $w(A, B)$ be the number of ways the vote could end with Andre receiving A votes and Bertrand receiving B votes. Then:

$w(A, 0) = 1, A >= 0$

$w(A, A+1) = 0, A >= 0$

$w(A, B) = w(A-1, B) + w(A, B-1)$

Solving this with dynamic programming gives an $O(N^2)$ solution.

For the full solution, an exclusion argument by way of Andre's reflection method must be used. Essentially, their are (N + M) choose N ways the vote could go so that Andre has N votes and Bertrand has M votes in the end. Out of these, some votes are "bad", meaning that at some point Bertrand has one more vote than Andre. Using Andre's reflection method, we could "flip" all the votes after that point, so votes for

Andre go to Bertrand and vice versa. Then we would see that Bertrand ends with N+1 votes and Andre with M-1. Moreover, one could undo this operation (flip the votes back), and every vote that ends in (M-1, N+1) can map to a bad vote for (N, M). Hence, the number of good ways the vote can go is simply the number of ways you can get to (N, M) minus the number of ways you can get to (M-1, N+1), or:

Answer = (N + M) choose N – (N+M) choose (N + 1)

Which can be computed in linear time. Note that multiplicative inverses need to be used as we are working modulo a prime.

Time Complexity: O(N*)*

Space Complexity: O(N*)*