February JDCC Solutions + Comments

Problem A (Vibberish or Cibberish?):

Solution: The solution is fairly simple. For each word, count the number of vowels in the word and see if it's more than the number of consontants. Coding this could be sped up if you use a data structure with a .contains() method.

Time Complexity: O(N)

Space Complexity: O(26)

Problem B (Phi Circuits):

Solution: Simply remember what the last two numbers were and update them as you go along according to the problem rules. Note that this is very similar to the Fibonacci recursion, indeed every number in the sequence will be a Fibonacci number. The problem hints at this property, as "Phi" usually denotes the golden ratio, which is present in the closed form of the Fibonacci numbers.

Time Complexity: O(N)

Space Complexity: O(1)

Problem C (Big Integer Factorization):

Solution: The key insight here is that if N is divisible by P, then N^N is divisible by P^N. With this in mind, simply factor N and multiply each prime exponent by N. Be sure to use a long to not have overflow.

Time Complexity: O(*sqrt(N)*)

Space Complexity: O(*1*)

Problem D (Gas N Go):

Solution: The key insight for this problem is that it could be simpified with the following strategy: Take the shortest path to some gas station, then fill up at that gas station. With this in mind, you would never travel to a more expensive gas station to fill up, so you would only need to buy enough gas to travel to your next filling stop.

Using this information, we can begin by using Floyd-Warshall to find the shortest distance between every pair of stops, then update the distances as:

cost(i, j) = distance(i, j) * price(i)

Then perform Floyd-Warshall again to get the answer. This is not the fastest solution as it does needless computations, but it is asymptotically the same as a Dijkstra's-type solution and is much faster to code.

Time Complexity: $O(N^3)$

Space Complexity: $O(N^2)$

Problem E (Mind Boggling):

Solution: The key insight for this problem is the Linearity of Expectations, which states the expected value of multiple events is the sum of the expected values of those events. In this case, we are finding the expected number of words over all possible paths of the board. We can simplify this as the sum over all words and paths of the expected value of a given word over a given path. But this expected value is just 1/26^(length of the word). Hence, for each K, the answer is just:

1/26^K * number of words with length K * number of paths of length K

The last can be computed using a naive depth-first-search in $O(N^2K^K)$ time. This could be slightly improved using symmetry.

Time Complexity: $O(NK^K + KM)$, where K is the length of the longest word.

Space Complexity: $O(N^2 + KM)$