**GFSSOC1 JUNIOR SOLUTION NOTES**

**Problem 1 – Flying Plushies**

Use a counter to keep track of how many plushies hit. A plushie will hit if it's height is greater than or equal to Griffy's flying height.

*Skills needed: If statements, for loops*

*Time complexity: O(N)*

**Problem 2 – Parking Lot**

Keep track of x and y coordinates after each order. North adds B to the y value, South subtracts B from the y value, East adds B to the x value, and west subtracts B from the x value

*Skills needed: Implementation*

*Time complexity: O(N)*

**Problem 3 – Waiting**

Split each time into seconds minutes and hours, and then get the total number of seconds using the formula total = seconds + minutes\*60 + hours\*3600. The difference between the two totals is the answer.

*Skills needed: String manipulation, simple math*

*Time Complexity: O(1)*

**Problem 4 – Tic Tac Moe**

Using brute force, check for 'O' win, then check for 'X' win. Output in accordance. The most difficult part of this problem is actually making the 2-D array.

*Skills needed: If statements, array manipulation*

*Time Complexity: O(1)*

**Problem 5 – Pursuit of Knowledge**

If you tried to run a BFS for each query, it is likely your code would've failed(unless it was in c++ or PYPY2 with good optimization). This is because the time cost is Q\*(N+M), where Q is a ridiculously huge number. Notice that there are many more queries than possible source nodes. Therefore, all we need to do is run BFS from each possible source node and store the values in an answer array. For each query, just grab the value from the answer array. In total, this solution takes N\*(N+M) time, a significant upgrade from the first.

*Skills needed: BFS*

*Time Complexity: O(N\*(N+M)+Q)*

*\*Note:* The difficulty curve from J4 and J5 is quite steep. This question requires an algorithm known as Breadth First Search, which you can search up online or ask any of the execs.

***GFSSOC1 SENIOR SOLUTION NOTES***

**Problem 1 – Friendship is a Number**

Store both ID number and friendship score in a list like structure, sort based on friendship score, and reverse the list. Then print out the first three IDs.

*Skills needed: Array manipulation, for loops*

*Time complexity: about O(NlogN + N\*T), where T is average of all Ts*

**Problem 2 – Speech**

A simple way of approaching this problem is storing the entire string as an array, with the indexes being words. For loop each friendly-unfriendly pair to check the entire string array, and replace all words that match the unfriendly word. Remember to strip out the period and add it back in after the for loop executes.

*Skills needed: Array manipulation, string manipulation, for loops*

*Time complexity: O(N\*M) where M is number of words in the string*

**Problem 3 – Hide n Seek**

Upon examination, we find that the total shortest time is the sum of the time from Griffy to one of the hiders, and hiders to hiders until everyone is found. The easiest way to find the shortest path is just to try all possible permutations! Thus, this turns into a recursive problem (or just use itertools permutations/algorithm next\_permutation) to find the optimal path. First, compute using BFS the distance from each hider to each other hider, and Griffy. Using your method of choice, generate all possible paths and take the best. This solution works due to the fact that T<=5.

*Skills needed: BFS, recursion*

*Time complexity: O(T!+T\*(N+M))*

**Problem 4 – Stalactites**

Straightforward 3-D Binary Indexed Tree. A 3-D Prefix sum array would have TLE'd, as it is too slow in updating queries. The difficulty of this problem is in coming up with the query formula. You should also note that a 32 bit int would NOT suffice to store the sum for all the stalactites.

*Skills needed: Data structures*

*Time Complexity: O(Qlog^3(N))*

*Note\**: If you do not have any prior knowledge on Binary Indexed Trees, this problem is practically impossible. You can find information about them online, and it is recommended you complete the 2-D BIT problem IOI '05 mobile phones before this problem.

**Problem 5 – Stardust snow**

We can turn this problem into a 4D dynamic programming question where we keep track of horizontal position, time, snowflakes left, and temperature left. At each position, consider two cases, one where Griffy takes the snowflake (if a snowflake exists at that position and time), and the other where Griffy does not take the snowflake. TIME STARTS AT 0.

*Skills needed: Dynamic programming*

*Time Complexity: about O(R\*C\*S\*B\*M)*