**Cracking the Coding Interviews Reading Notes**

Chapter 1: **Big O**, Aug 10 – 11

1. Time complexity of recursive function: O(branches ^ depth)

Branches is the number of branches at each call. Depth is the depth of the call stack.

Sometimes, time complexity is the number of times we can divide/reduce a number N to its base case.

1. Questions answered wrong: P57, T9, 11

**Interview Questions:**

Chapter 1: Strings and Arrays, August 11 – 15

1. A close-up of a white paper

   Description automatically generated
2. Answered wrong:

* 1.3 URLify (Edit string in-place, use two pointers)
* 1.5 Oneaway (Use two pointers)
* Sum all digits of an input string/ integer
* **1.7** **Rotate Matrix (wrong second time)**
* **Leetcode 33: Search in Rotated Sorted Array**
* **Be careful when one size\_t subtracts another size\_t. That will result in a huge positive number**

Chapter 2: Linked list, August 15 –

* 2.1 Remove Duplicates, **O(n) time complexity**
* **Leetcode 19 Remove Nth Node from End of Linked List (Similar to 2.2)**
* **2.4 Partition: if head is null, that means it will be the first element to insert. Also, create head and tail nodes are very helpful, but they should be set to nullptrs initially. Also, we should do head = insert instead of head->next = insert. Take a look at the second method on the answer key: pay attention to the way to grow a linked list**

**Chapter 3: Stacks and Queues**

* **3.1 Three in One**
* 3.2 Min Stack (**Don’t maintain a global int minSoFar**)
* 3.4

**Chapter 4: Trees and graphs**

* Full and complete binary tree
* 4.2 Minimal tree
* To find if a path exists between two nodes or the shortest path between two nodes, we use BFS.
* 4.3 List of depths (same as Leetcode level order traversal)