**SUMMARY:**

**Program Overview**

Our program started simple. We based most of our knowledge on a research paper, added our own notes to make sense of it all, and supplemented the process with Reddit and Stack Overflow resources. We initially wrote a version of the program that could solve a 9x9 Sudoku puzzle. However, we quickly wanted to get started with an improvement such as the 3D Sudoku and the dynamic grid.

**Decision to Implement Dynamic Grid**

Wafae noticed that the 3D Sudoku would be difficult to implement since it would fill up memory faster and take exponentially longer to solve, so we decided to implement the dynamic grid instead. While writing the dynamic grid, we found our program to be somewhat quick on time but lacking in space.

**Multiple Solutions and Time Tracking**

Riley also got it to figure out if there were multiple solutions to a puzzle. If multiple solutions were found, the program would display them and log the times taken for each solution.

**BFS vs. DLS Performance**

We noticed that BFS and DLS performed differently depending on the puzzle size:

* **For BFS**: It’s generally faster for smaller grids due to exploring level by level and rarely backtracks, but we noticed an increase in its memory usage with larger grids. It eventually failed for puzzles larger than 16x16 due to heap memory issues.
* **For DLS**: It’s slower for most grids in comparison to BFS because it explores deeper paths first and frequently backtracks, DLS consumed less memory which allowed it to solve larger puzzles like 25x25. However, DLS became inefficient for larger grids, often running for a long time without finding a solution.

**Issues with Larger Puzzle Sizes**

We were able to get the program to solve a 16x16 puzzle but when we kept going up in size, specifically a 25x25, we hit a brick wall. It first started giving us errors – “not enough heap memory”- and tried to increase our memory in the IDE but no luck. Next, we tried only solving the code using DLS (since BFS was the one failing) to try isolating the issue except this time DLS was not “failing” instead it would keep running even after 30 minutes of trying to solve the puzzle, so it was back to the drawing board.

**Optimization Challenges**

We made attempts to optimize our code but kept running into the same memory issues, which caused new errors. Due to time constraints and the persistence of these issues, we decided to leave the program as it was, knowing that it still worked for solving 9x9 and 16x16 puzzles. We speculated that we might need better hardware to solve larger puzzles.

**Puzzle Failures and Debugging**

The program could solve most puzzles. However, there were a few cases where it failed to find a solution to puzzles that were solvable by hand. For example, we tested a puzzle from a Sudoku website, and although it was solvable manually, the program failed during debugging around the halfway point.

**Future Improvements**

If we had more time, we would get the 25x25 to work by trying to save space, since we believe the issue had to be that the board states were not clear during BFS and with DLS it looked to be that it just didn’t have a good number to tell itself to stop and backtrack. Additionally, we would like to research on why some puzzles were just failing altogether because we were mostly focusing on getting the program to read and solve a dynamic grid size.