nfl writeup

WO1 Clayton E. Williams

September 2023

0.1 Project Summary

nfl is a program that loads a player/team database of NFL players and calculates the degrees of separation between any two players, as well as provide statistics for any given player through the use of command lien options.

0.2 Challenges

I think I did a good job of initial design and sticking to it, so my biggest challenge was simply writing the code for more complex features, such as oracle, and trying to figure out how to access and modify players or teams and maintain persistence for the next iteration. As I mention in Lessons Learned, doing a bit of a deeper dive into that function would have helped me identify the issue with updated players, which are pointers, and planning a possible solution before beginning.

The next biggest challenge for me was self-induced and resulted in me getting sucked into oracle and the optimization rabbit hole for about three days. The first time it ran successfully, it took about 7 minutes, and I quickly realized why. I fixed that issue and got down to 4 minutes. At that point, it worked, and I should have moved on to documentation or cleaning up my test suites, but each time a name was added to the board, or a time was updated with a quicker time, my competitive nature took over, and I got pulled in, trying to optimize to beat one of the times on the board. This is where I should have leaned on my design plan and milestone timeline and stayed focused on the task at hand to get to MVP, and then refactor if there is time.

0.3 Successes

The biggest success of this project by far was the initial design and planning phase. To start with, I felt I had a much better grasp on data structures and graph theory, which allowed me to take the first day and a half to just plan and design without actually writing any code. I was able to easily see and conceptualize the data structures needed, and more importantly, it made sense why to chose those structures. Having filled both sides of the white board with data structures, and pseudo code for functionality that would be required meant that once I started actually typing code, adhereing to the initial design plan, I was able to tackle functional pieces one after another.

This is also the first time that I truly set milestones for myself with realistic, attainable timelines. I utilized the milestones feature in GIT, and part of my first day design was looking at the functionality required, and doing an estimation on how long it would take to complete each feature. Doing this allowed me to track the progress of my work overall, because it was easy to see what milestones were left, as well as whether or not I was on target to achieve MVP by a certain date, and a project worth submitting with appropriate documentation and testing by the due date.

0.4 Lessons Learned

Although I think I did will with my initial design, I realized I should have gone another level deeper on it. I compartmentalized each function to write pseudo code and understand how it should work, which was good, but once that was done, I should have taken a more hollistic view at the project and all its functions. This would have helped identify the behavior and interactions between functions, which would have better directed what, if anything, each returns, which would have helped me avoid re-writing tests as I attempted to approach this project using TDD.

Along with a taking a closer look at the behavior of functions, in the future, I need to do a more accurate assessment on completion time for each feature. I felt was fairly conservative, allotting 1 day for each of the options, stats, player, team, etc., to get to functional code by the end of week one. In reality, after I had data parsed and in appropriate data structures, I was able to knock out 2 or 3 features a day. This overestimation left me thinking I was well ahead of my timeline, only to find myself falling behind when I tackled oracle that I had somewhat underestimated.