**For the past week, I’ve been working on two main things:**

1. **Finding a modified quickest path from one career to another that considers the number of transitions needed (where less transitions is better than more).**
   1. I implemented Dijkstra’s shortest path algorithm to minimize:
      1. The “distance travelled” (this is the path I went over last week)
      2. The number of jobs (transitions to one of ten jobs are now all the same)
      3. Both distance and number of jobs (rank distance + weight for each node)
         * The choice of weight I used was 5.5 and 1. I used 5.5 since it was the average rank distance for each job and 1 as a standard.
         * The choice of weight is arbitrary but more weight means more cost for transitioning from career to career
   2. Overall Findings (Full data findings in *Career Changer Paths.pdf)*:
      1. Found paths for Nannies to Aerospace Engineers and vice versa as well as Computer Programmers to Chief Executives and vice versa.
      2. Using both distance and number of jobs with weight 5.5 yield the best results for the most part.
      3. I also tried to do an intersection between the original distance travelled path and the modified path which worked for some careers and not others in terms of transition improvements.
   3. Questions:
      1. Are there any other transitions/career pairs you guys think might be useful to look into in order to figure out the best path-finding method?
      2. Which out of the current paths look the most compelling to you guys?
      3. What other weights of each node should I try next?
2. **Replicating the Career Changer algorithm in the ONET report.**
   1. I was able to fully implement the algorithm mentioned in the report. However, the Career Changer rankings I found ended up not matching what was on the Career Changer Matrix. I will be going over my code/the report to see where I can see where they differ.
   2. Questions:
      1. Any suggestions on finding where the elements may differ? (Assuming there were no bugs in the code)
      2. Assuming I implemented everything correctly, what other factors should I consider in the similarity ratings next?