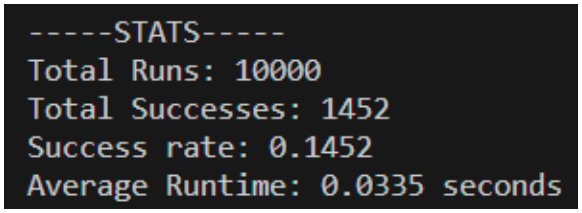
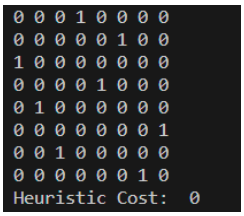
Alec Urbany
CS 4200
Project 2
7/31/23

This goal of this project was to create a solver for the n-queen problem. Two algorithms were used. Steepest-Ascent Hill Climb, per request of the assignment, and a Genetic Algorithm as per my discretion.

Steepest-Ascent Hill Climb Algorithm:

For this algorithm implementation, I kept things very standard, and opted to not implement side to side movement. Out of a batch of 1,000 random boards, my algorithm recieved a success rate of about 16.10% With 10,000 runs, it was 14.52%. I've gotten percent variances of about +-2% over the many times I've run it. The 1,000 run test in particular also gave me the fastest running time per problem I've seen at only .0282 seconds. Usually the running time is in the .033 second range.

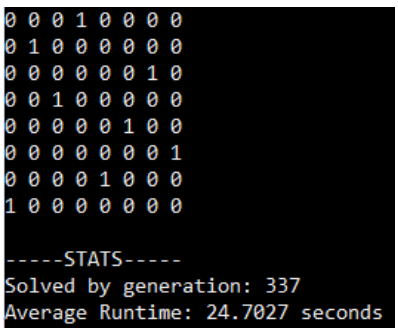


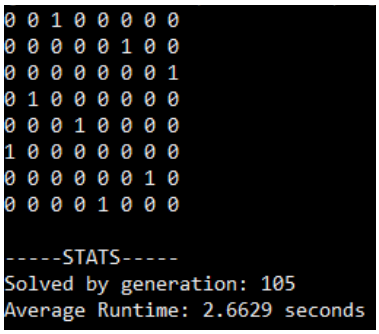


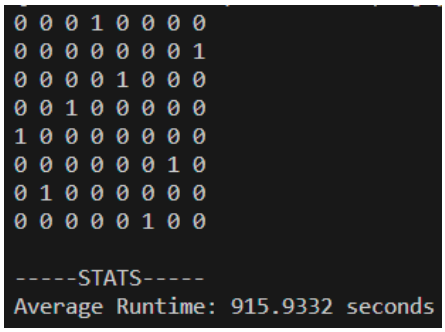
Genetic Algorithm:

This algorithm was much more chaotic than S-A Hill Climb. As far as implementation goes, we start off with a board, we populate it with random genes. We then evolve and mutate each gene until we get a solution with fitness = 28 (in the case of the 8 queen problem). Where the chaos comes in is that my implementation of this algorithm is largely random. So, individually, it shoots out a lot of guesses incredibly fast - faster than S-A Hill Climb. However, around the range of 25-28, I notice it struggles just getting past the finish line. Running the 8 queen problem did give me a proper answer, however, sometimes it took minutes, with generations in the thousands. Othertimes I managed to get an answer in less than 3 seconds. While not the most elegant solution, it did at least work. It's very brute force, and lacks a certain amount of finess. This algo was largely inconsistent.

The two to the left were done via a linux VM in Powershell. The right was VSCode







^Got cut off in the screenshot but it finished by generation 25,416

To run the project:
python3 4200Project2.py