Max Hoac
CS 4200.E01

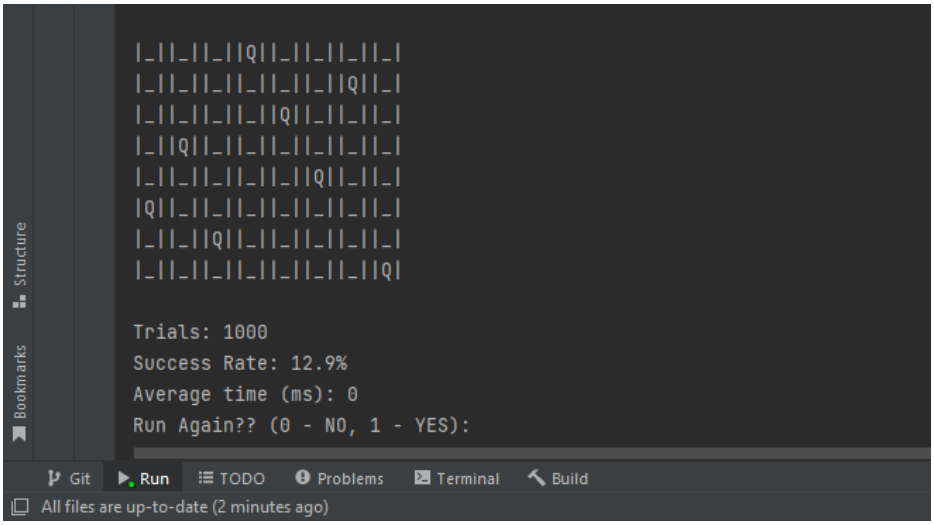
# CS 4200 Project 2 Report

**Goal:** The primary goal of this implementation of the N-Queen problem is to utilize two different search algorithms to accomplish the goal of having no pair of queens directly or indirectly one another. The first algorithm being used is Steepest Hill Climbing, whereas the second algorithm being used is Minimum Conflicts (CSP).

**Approach:** For my implementation, I planned on having my code be capable of generating a designated number of randomized puzzles as well as having the option for the user to generate their own puzzle through a user input scanner. In order to accomplish this, I created separate class files for each differing puzzle type as well as a separate class file that depicts the format of how the separate numerical nodes will connect with one another. The A* algorithm will be implemented together in a single file known where the class will possess two separate functions regarding the different heuristic functions. Altogether these components will be utilized in a working file where the user is able to choose to generate a randomized 8-puzzle or create their own 8-puzzle. When the randomized option is chosen, the code will prompt the user for the number of puzzles they want to create. Once the input is taken, both algorithms (h1 and h2) will go through the generated puzzles until all are completed. Upon completion, the depth, search cost, and time statistics will be printed out for both heuristic functions. Additionally, when asked to solve several puzzles, a table depicting the same values will be generated. If the user input option is chosen, the code will operate similarly, however it will only solve the inputted puzzle. Once again upon completion, the depth, search cost, and time statistics for the puzzle will printed out.

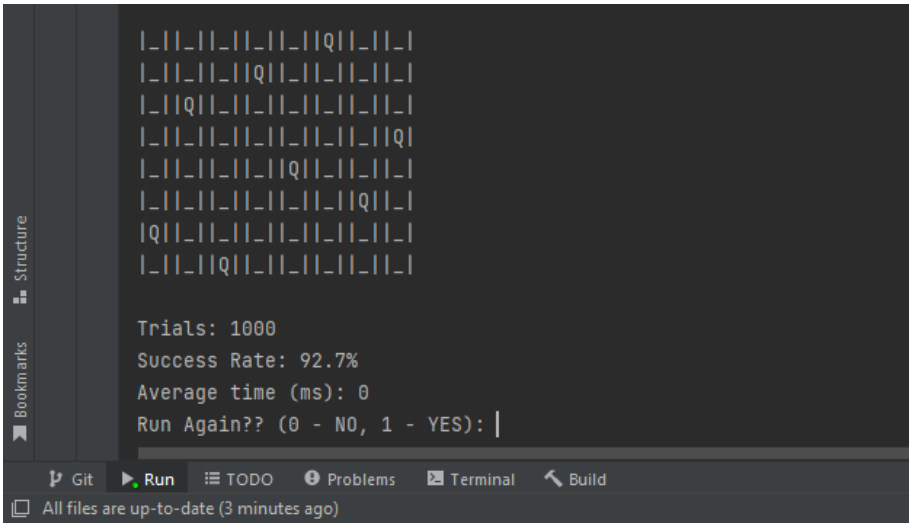
**Analysis:**

**Default Steepest Hill Climb Results**

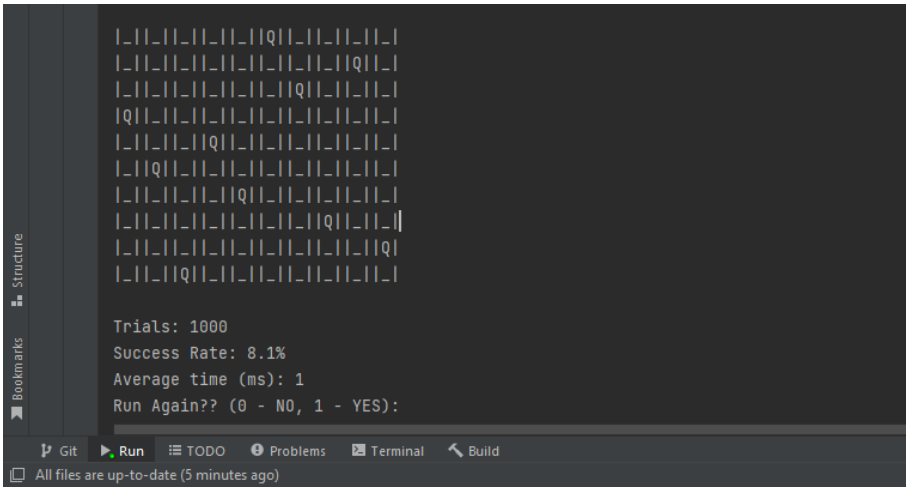


Max Hoac
CS 4200.E01

## Default Minimum Conflicts (CSP) Results




## Customized (10x10 Board) Steepest Hill Climbing Results




## Customized (10x10 Board) Minimum Conflicts Results

Based on these results, it is clear that the Steepest Hill Climbing search algorithm experienced a significantly more difficult time in having success in solving the N-Queen problem when being compared to the Minimum Conflicts Algorithm. The reasoning behind the Steepest Hill Climbing algorithm's struggles is most likely the result of the algorithm trying to reach a solution too quickly. As a result of resembling the greedy algorithm, the Steepest Hill Climbing constantly failed as it would try to progress in large chunks causing it to frequently get stuck in local maxima and local minima points. As for the minimum conflicts algorithm, the success rate was far higher regardless of a change in the board size. In this algorithm, instead of aimlessly making the best available move, my implementation randomly selected a queen that was on the board and essentially move the piece to the spot that would have the lowest number of conflicts with other queen pieces.

**Findings**

Overall, the Steepest Hill Climbing algorithm proved to show that it would have a rather low success rate in solving the N-Queen problem since it essentially mirrors the greedy algorithm causing it to get stuck or fail after completing only a few steps. Comparatively the Minimum Conflicts algorithm which qualifies as being a Constraint Satisfaction Problem of CSP, experienced a significantly higher rate of success when trying to solve the problem, since the steps were less one-minded. As a result of randomly selecting a queen and moving it to the least conflicted tile on the chess board, this algorithm was able to accomplish more successes since it would not fail within the first couple of steps.