Ferris Hussein

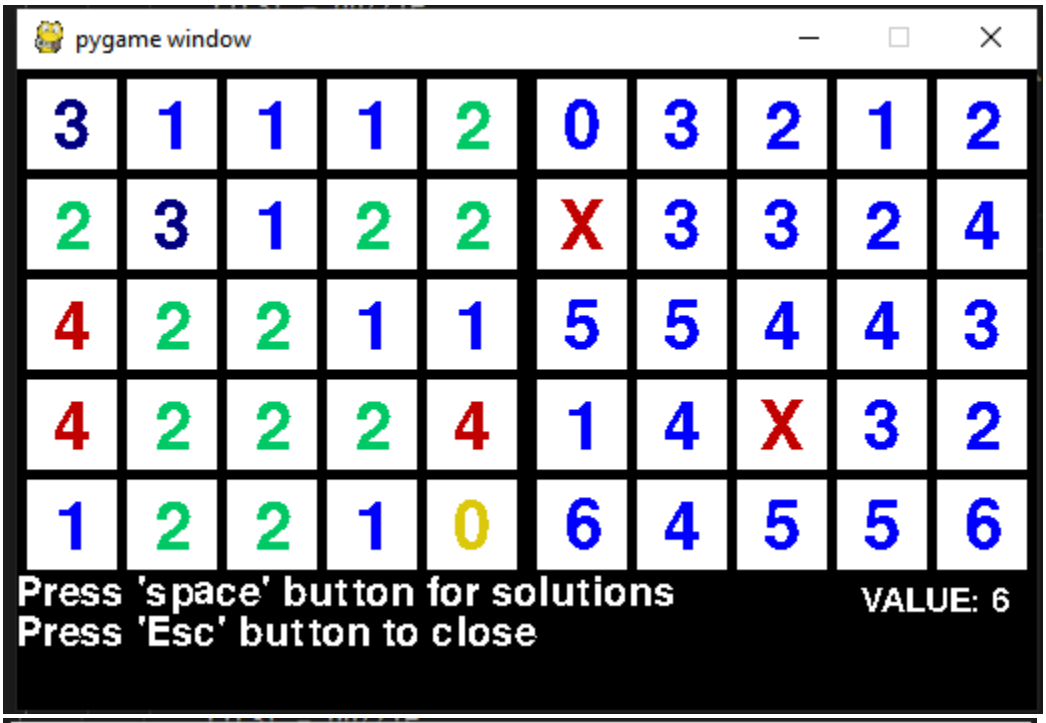
Report

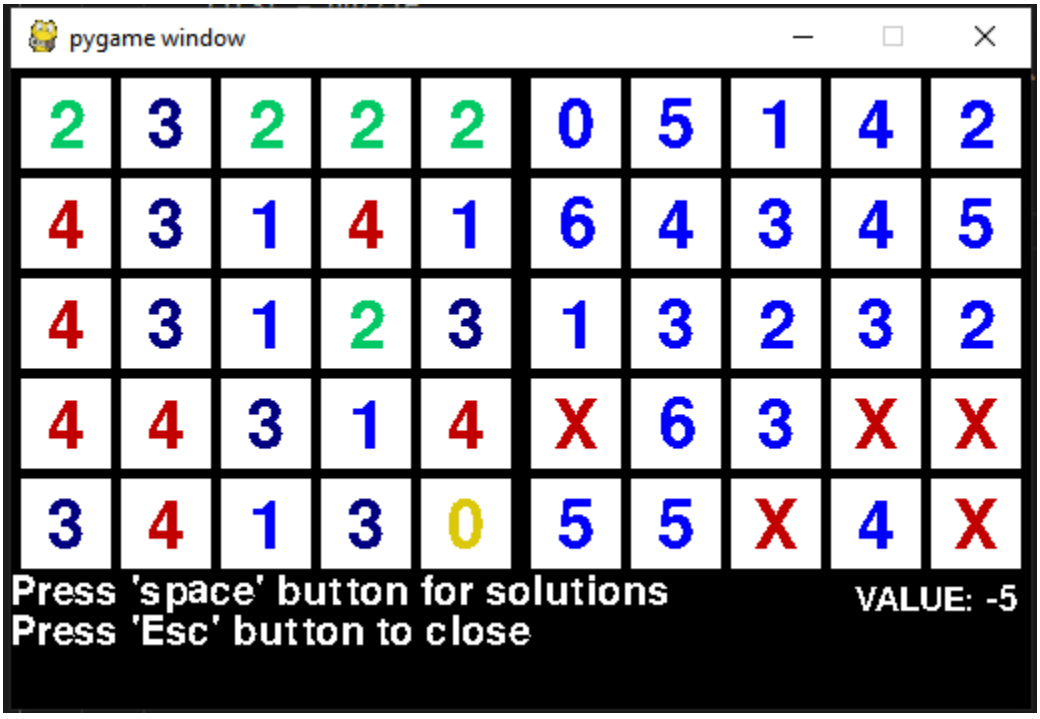
Task 1: Problem Formation

Environment – The environment for this project is the grid (sizes 5x5, 7x7, 9x9, 11x11) where all the weighted cells contained in.
State Space – The state space for this project depends on the grid size. It ranges anywhere from a 5x5 array, to an 11x11 array.
Actions – The actions for this project are any legal moves that we're able to make. We can move anywhere in the grid as long as the number of spaces we are moving does not go out of bounds of the grid. For example, if we are in a 5x5 grid, at space (5,2), we can only move 3 spaces down, 1 space up, and 4 spaces across. Any moves greater than these are not legal actions.
Perceptions/Observations – In this project, the perceptions/observations are the grids we originally start with, with randomized weights in each cell. We then solve the grid, giving us a solution grid of the proper path (if applicable) or a "negative k value" which means that there is no legal path from the starting point to the ending point in the grid.
Transition Function – In this project, the transition function is the given algorithm we are using in order to solve the grid. Whether it be A*, BFS, Hill Climbing, or Shortest Path First, these are all potential transition functions we use in order to evaluate the grid and return a solution for the most optimal path from start to finish.
Evaluation Metric – In this project, we have an evaluation metric of "K". K will be a positive integer if there is a legal path to solve a given grid, based on the number of moves it takes to solve the grid in the most optimal solution. For example, if the best path to solve a grid has 5 moves, K will have a value of 5. K will be a negative integer if there is no legal path to solve a given grid. The value of the "-K" will be based on the number of cells in the grid that are not reachable, including the ending point. For example, in a 5x5 grid, if there is no legal path to solve a grid, and there is no way to reach the point (5,2) in the grid, K will have a value of -2.

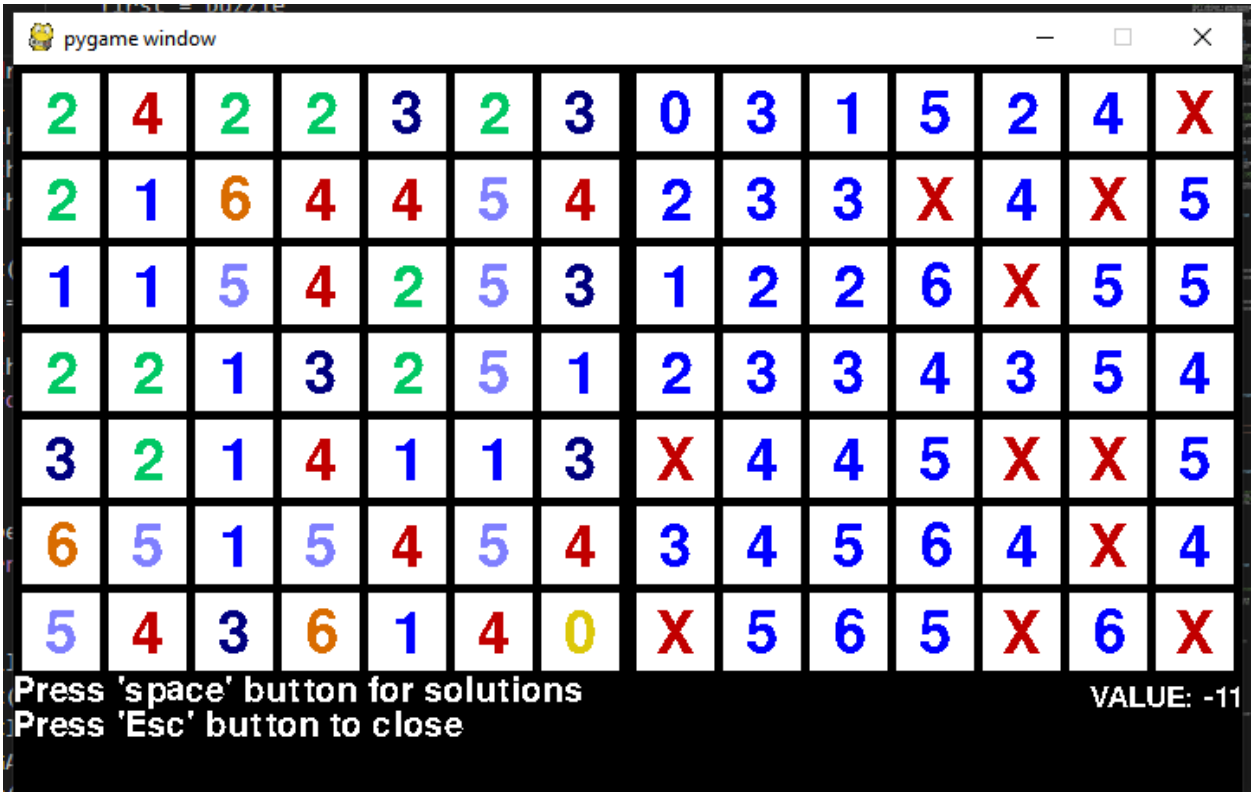
Task 3: Puzzle Evaluation
(Puzzles with and without solutions)

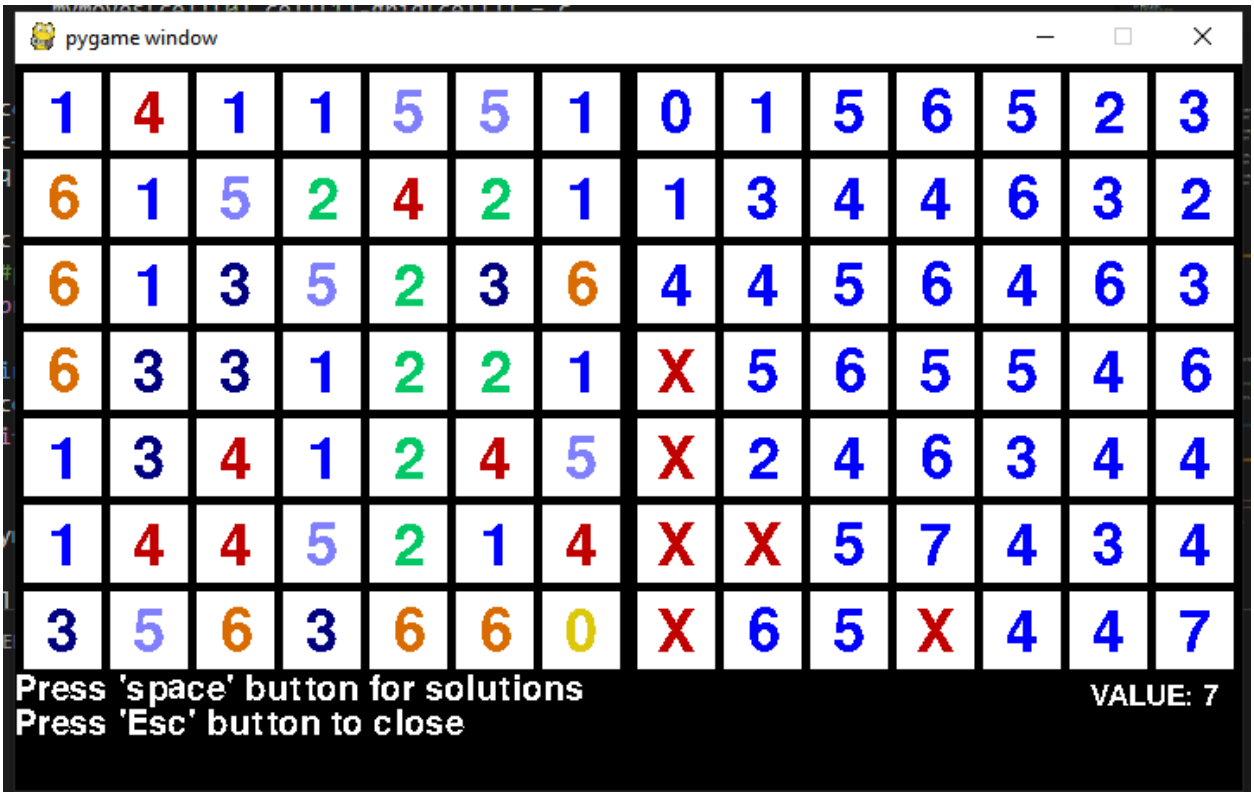
5x5 Grids



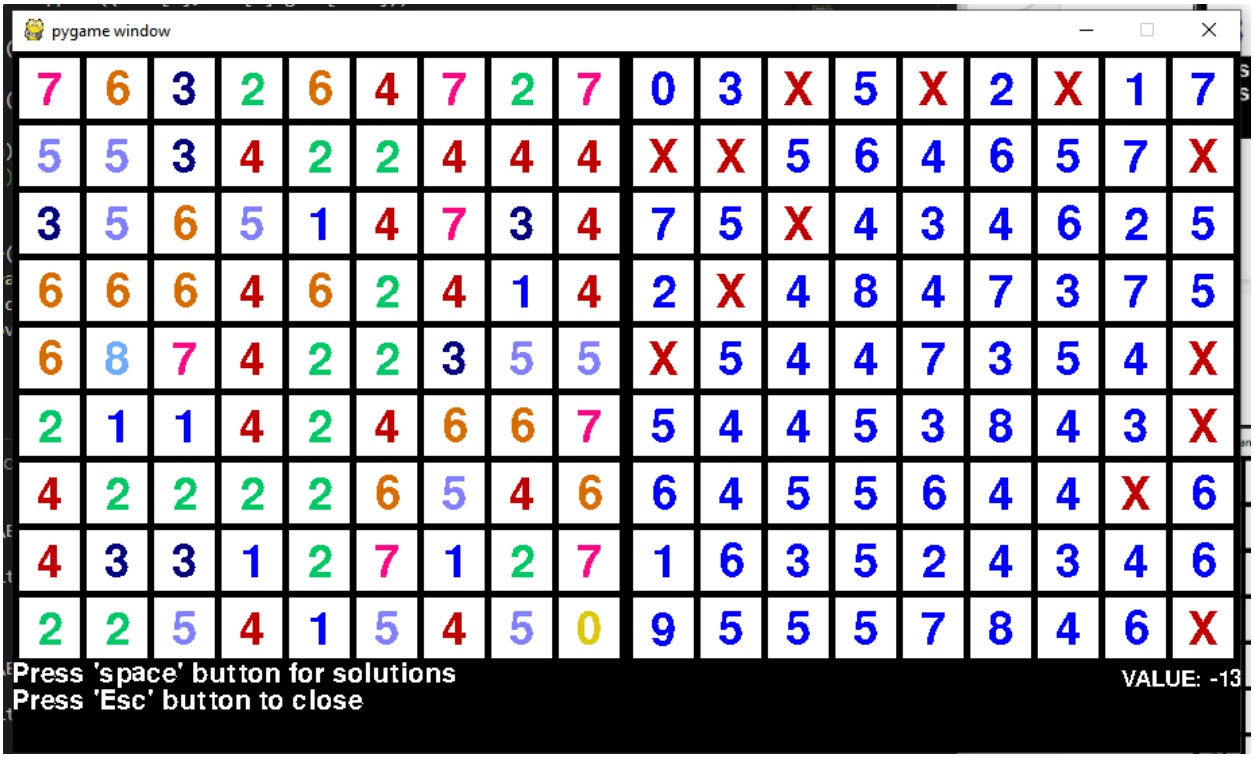


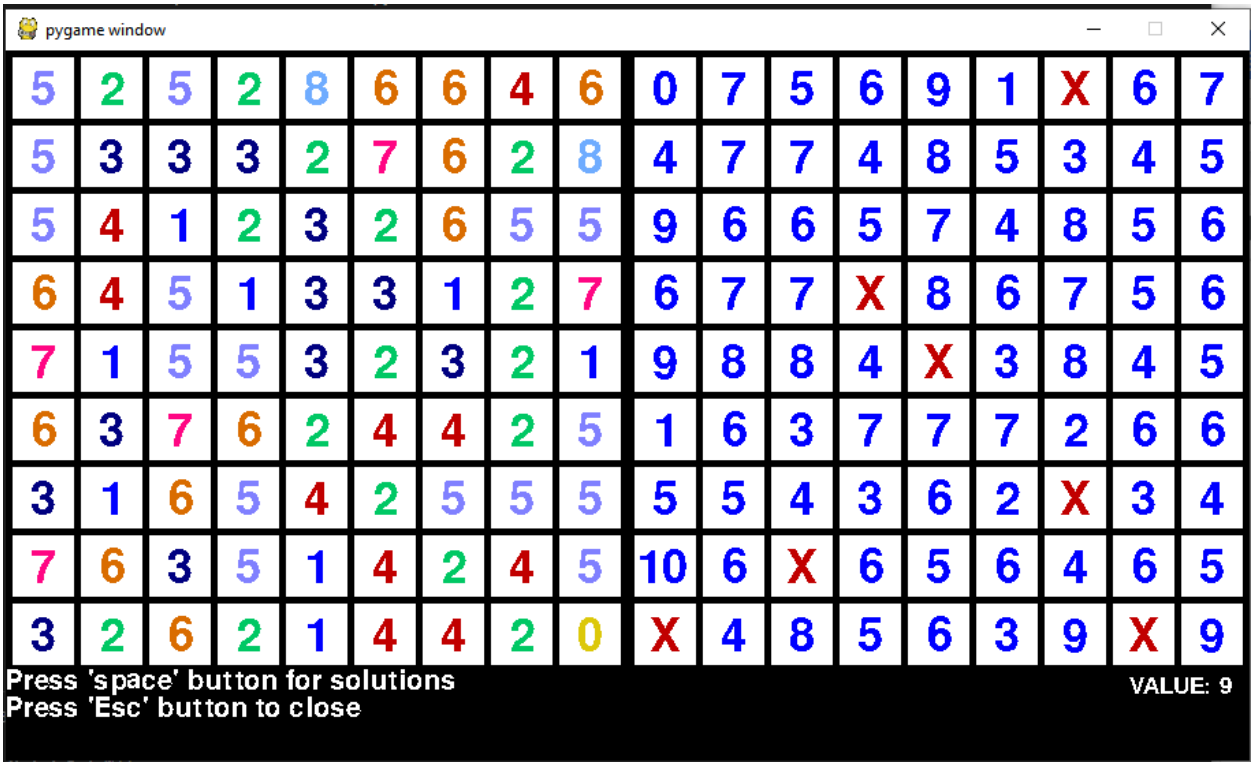
7x7 Grids


pygame window
2 4 2 2 3 2 3 0 3 1 5 2 4 X
2 1 6 4 4 5 4 2 3 3 X 4 X 5
1 1 5 4 2 5 3 1 2 2 6 X 5 5
2 2 1 3 2 5 1 2 3 3 4 3 5 4
3 2 1 4 1 1 3 X 4 4 5 X X 5
6 5 1 5 4 5 4 3 4 5 6 4 X 4
5 4 3 6 1 4 0 X 5 6 5 X 6 X
Press 'space' button for solutions
Press 'Esc' button to close
VALUE: -11

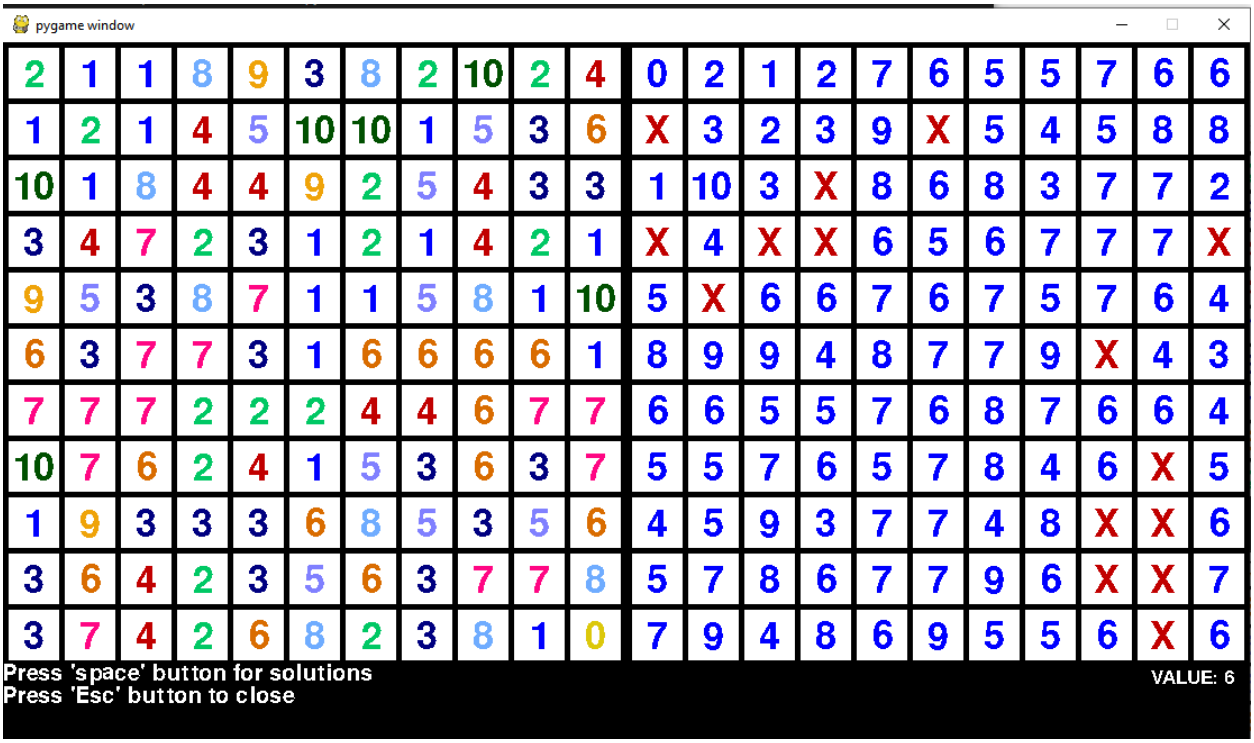

pygame window
1 4 1 1 5 5 1 0 1 5 6 5 2 3
6 1 5 2 4 2 1 1 3 4 4 6 3 2
6 1 3 5 2 3 6 4 4 5 6 4 6 3
6 3 3 1 2 2 1 X 5 6 5 5 4 6
1 3 4 1 2 4 5 X 2 4 6 3 4 4
1 4 4 5 2 1 4 X X 5 7 4 3 4
3 5 6 3 6 6 0 X 6 5 X 4 4 7
Press 'space' button for solutions
Press 'Esc' button to close
VALUE: 7

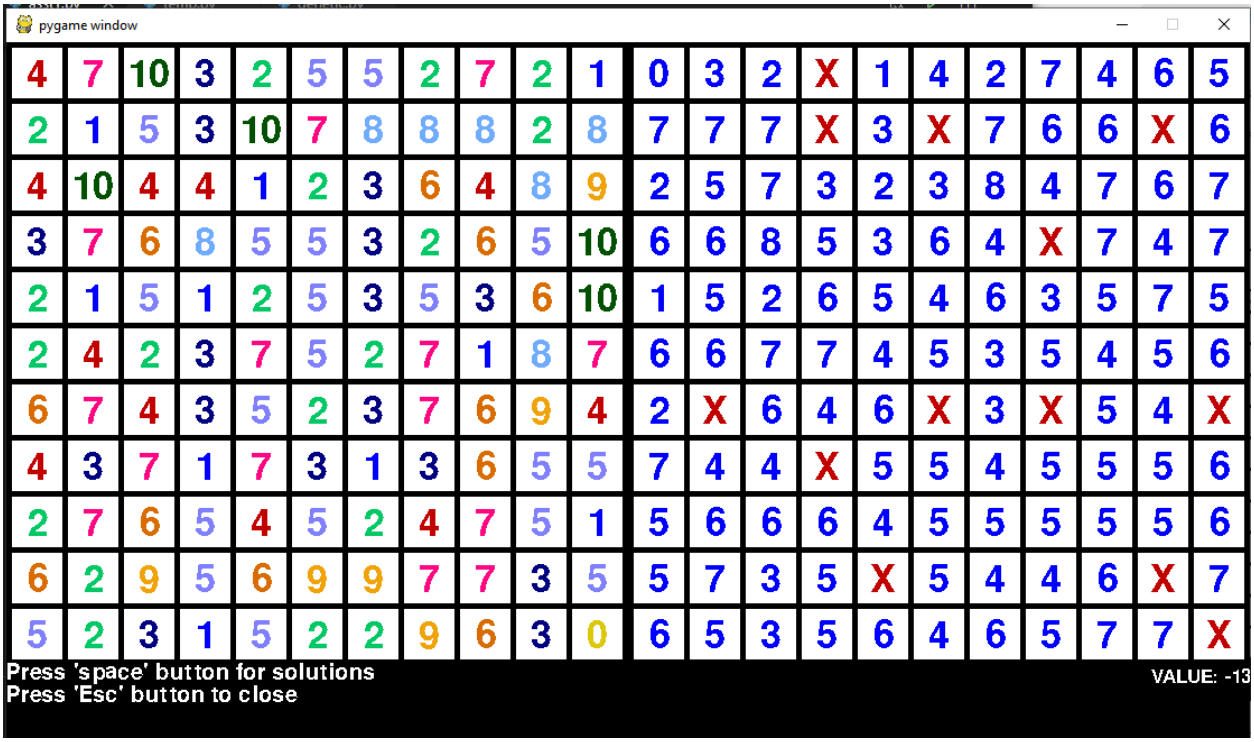
9x9 Grids:


pygame window
7 6 3 2 6 4 7 2 7 0 3 X 5 X 2 X 1 7
5 5 3 4 2 2 4 4 4 X X 5 6 4 6 5 7 X
3 5 6 5 1 4 7 3 4 7 5 X 4 3 4 6 2 5
6 6 6 4 6 2 4 1 4 2 X 4 8 4 7 3 7 5
6 8 7 4 2 2 3 5 5 X 5 4 4 7 3 5 4 X
2 1 1 4 2 4 6 6 7 5 4 4 5 3 8 4 3 X
4 2 2 2 2 6 5 4 6 6 4 5 5 6 4 4 X 6
4 3 3 1 2 7 1 2 7 1 6 3 5 2 4 3 4 6
2 2 5 4 1 5 4 5 0 9 5 5 5 7 8 4 6 X
Press 'space' button for solutions
Press 'Esc' button to close
VALUE: -13

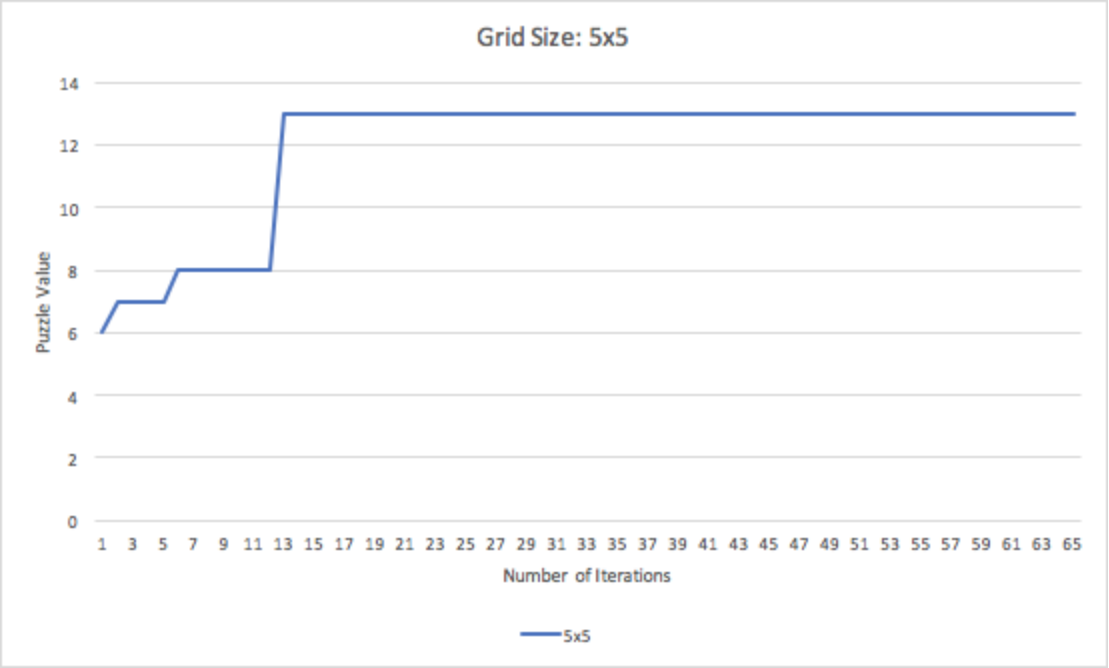

pygame window
5 2 5 2 8 6 6 4 6 0 7 5 6 9 1 X 6 7
5 3 3 3 2 7 6 2 8 4 7 7 4 8 5 3 4 5
5 4 1 2 3 2 6 5 5 9 6 6 5 7 4 8 5 6
6 4 5 1 3 3 1 2 7 6 7 7 X 8 6 7 5 6
7 1 5 5 3 2 3 2 1 9 8 8 4 X 3 8 4 5
6 3 7 6 2 4 4 2 5 1 6 3 7 7 7 2 6 6
3 1 6 5 4 2 5 5 5 5 5 4 3 6 2 X 3 4
7 6 3 5 1 4 2 4 5 10 6 X 6 5 6 4 6 5
3 2 6 2 1 4 4 2 0 X 4 8 5 6 3 9 X 9
Press 'space' button for solutions
Press 'Esc' button to close
VALUE: 9

11x11 Grids:

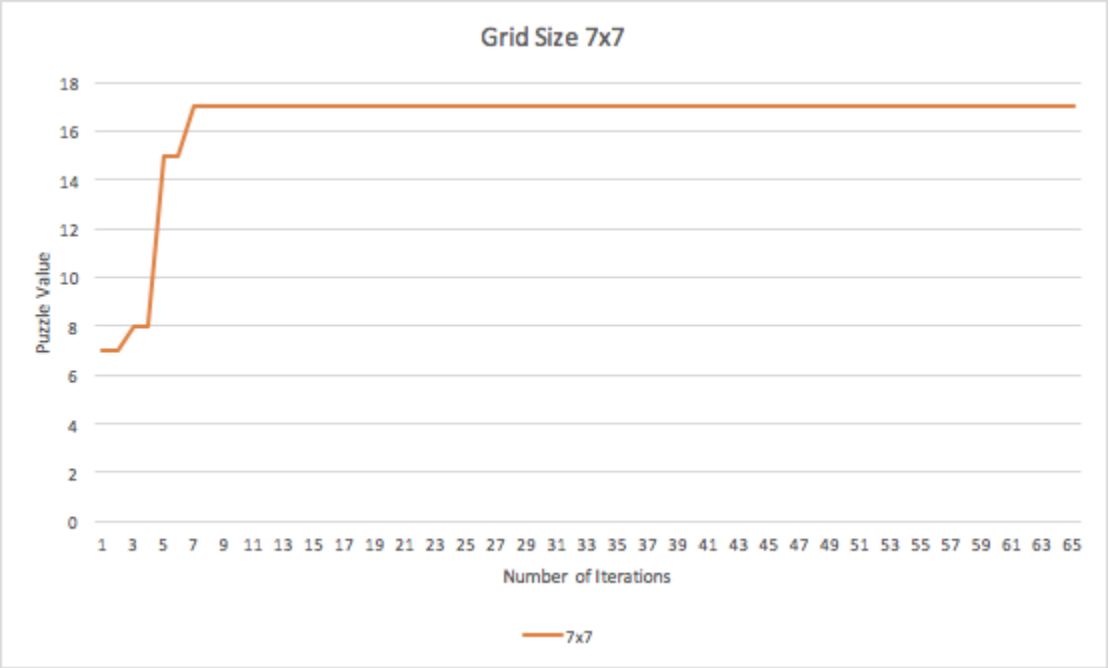




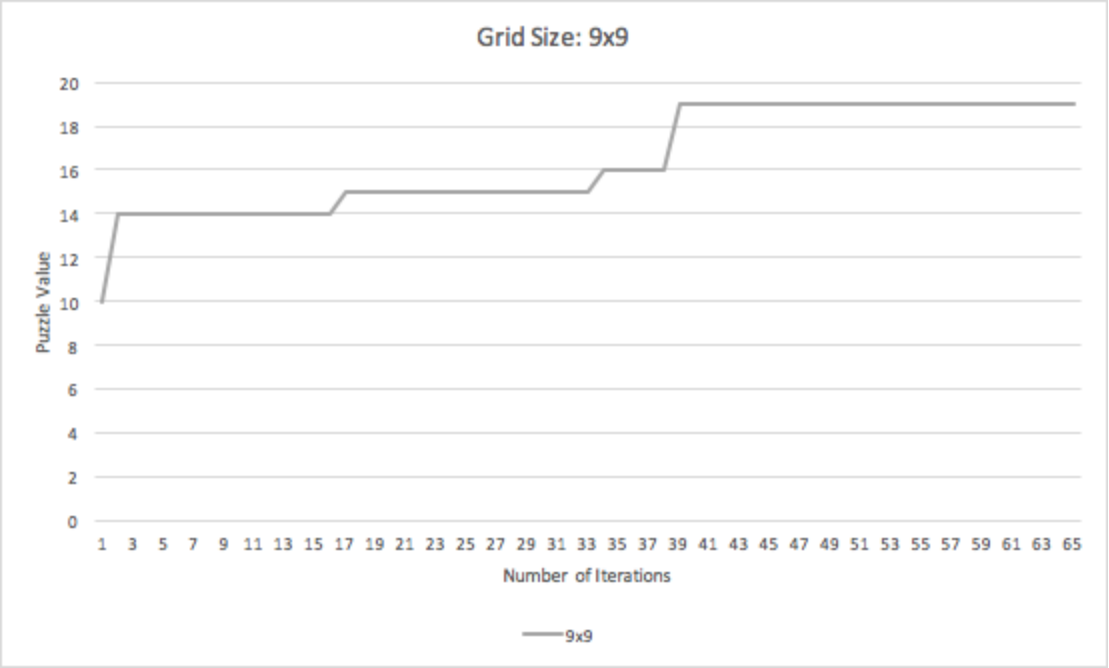
Task 4: Hill Climbing



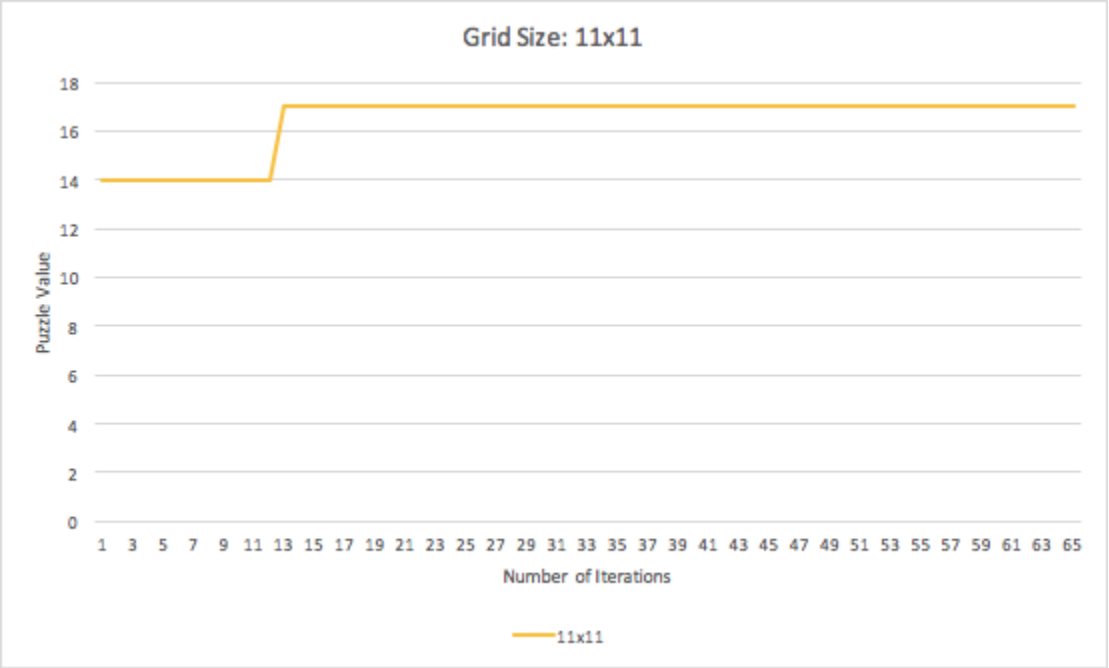
Time to run: 0.71875 seconds



Time to run: 4.859375 seconds



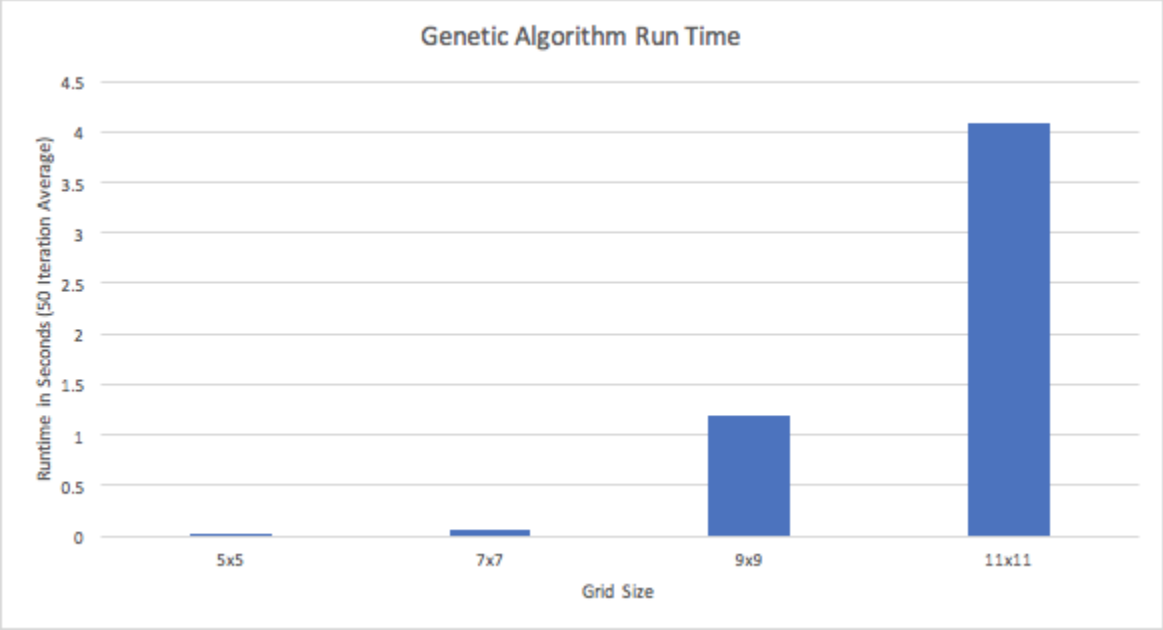
Time to run: 16.0 seconds



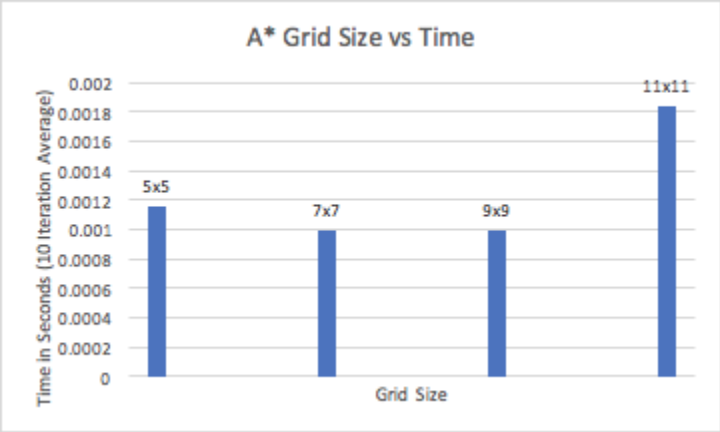
Time to run: 45.171875 seconds

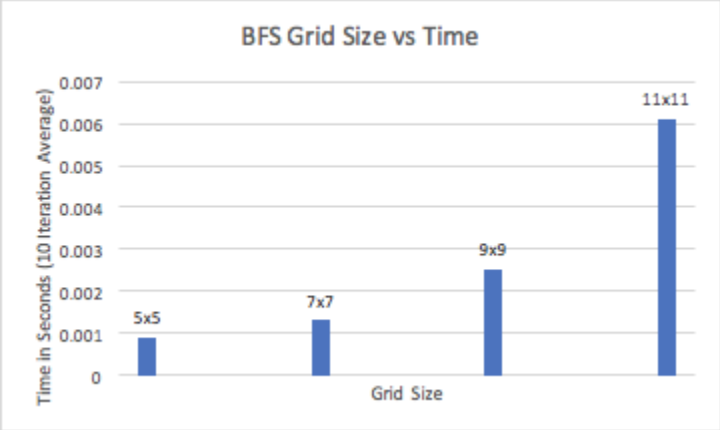
Task 7: Population-Based Approach

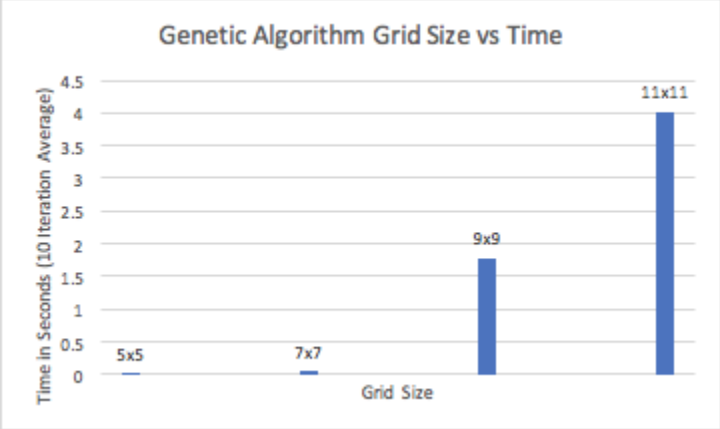
The method started out with 10 parent puzzles, each of the same size (either 5x5, 7x7, 9x9, 11x11) and ran A* or BFS algorithms on each one to determine which two "parents" had the best puzzle values. I then used these two parents, and crossovered them in order to obtain 10 more "children", each consisting of values from the original two parents. Once I got the children, I ran mutations (10% chance) on each cell of each grid in order to shorten the number of moves it would take to solve a puzzle. I held our mutations at a 10% rate to try and mimic real life genetics (though evolution takes significantly longer) so that we can truly understand the meaning of a genetic algorithm. After the mutations, I again ran either A* or BFS algorithms to determine the two new best parents for future "generations". I continuously ran this loop until I obtained a puzzle with the best possible configuration, meaning the shortest amount of moves. In my case, this would mean that the puzzle could be solved in exactly 2 moves.



Task 8: Evaluation







Advantages and Disadvantages of the Algorithms:

The advantage of A* is that it runs the quickest out of all the algorithms, though if your heuristic is bad, it can over/under estimate the distance of the path which can lead to inaccurate results. The advantage of BFS is that you are guaranteed the shortest path in a given grid/maze, though

the disadvantage is that it computes the shortest path by going through every possible path, making it take longer than A*.
The advantage of GA (genetic algorithm) is that after running it, you are guaranteed (based on our fitness test) to have the shortest/lowest number of moves (only 2) to complete a grid. The disadvantage of A* is that since it takes so many iterations and mutations, it takes a few seconds to complete this objective.

Task 9: Extra Credit

For the extra credit portion of the assignment I decided to implement our Hill Climbing method over 25 iterations on a 31x31 array. The computation time took about 44.1513397 minutes to create a puzzle with a puzzle value of 15. In order to increase our puzzle value the number of iterations must also be increased, which will also increase computation time. Below is the final puzzle configuration using Breadth First Search.