Syed Shayaan Hasnain Ahmad

20I-0647

Section A

Artificial Intelligence – Assignment 1 2.0

Q1. Sudoku Puzzle

Explanation of functions used:

The generate_puzzle () function, generates a Sudoku puzzle.

The function print_puzzle(puzzle) prints the puzzle in a nicely formatted way.

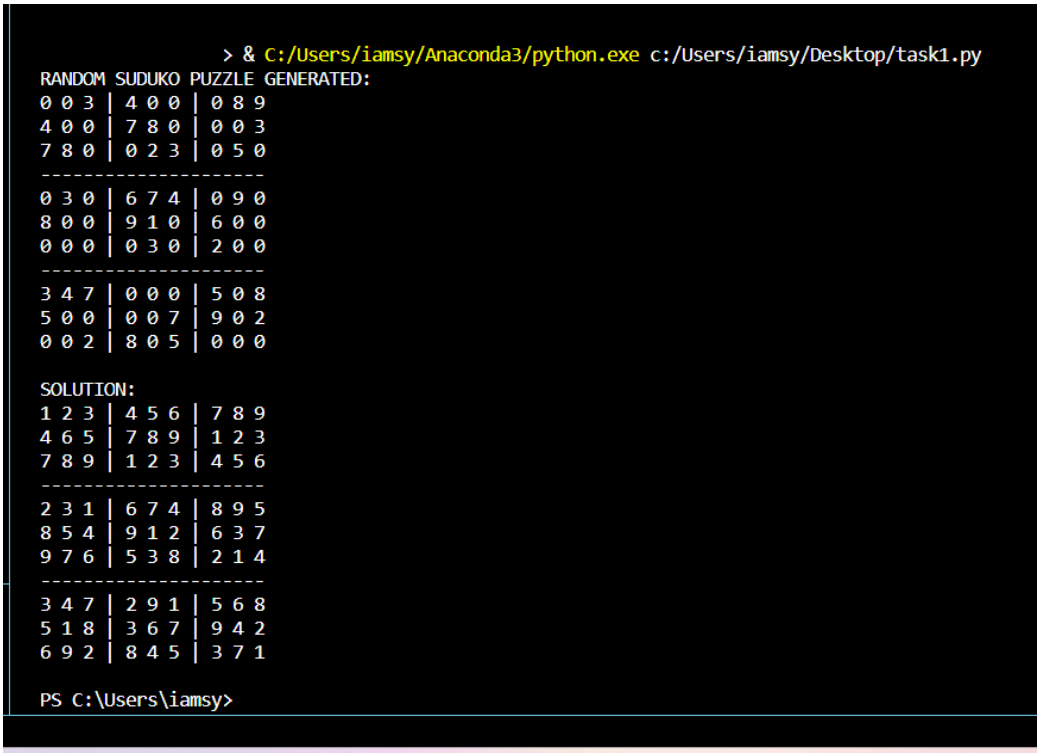
The solve_puzzle (puzzle) function first finds the empty cell with the fewest number of possible values by using the "find minimal_cell(puzzle) function. This function iterates over all empty cells in the puzzle, gets the set of possible values for each cell using the ' get_possible_values (puzzle, row, col) function, and returns the cell with the smallest set of possible values.

The get_possible values (puzzle, row, col) function returns the set of values that are not present in the same row, column, or 3x3 square as the specified cell. These values are sorted by the number of cells in the puzzle that they can fill, in ascending order, to give the A* algorithm a better chance of choosing the right value first.

The solve_puzzle(puzzle) function then tries all possible values for the chosen cell, in order of least to most common, until a solution is found.

- solve_puzzle(puzzle) - this function solves the Sudoku puzzle using heuristic and A* algorithms.
- find_minimal_cell(puzzle) - this function finds the next empty cell with the fewest number of possible values.
- get_possible_values(puzzle, row, col) - this function returns a list of possible values for a cell.
- find_empty_cell(puzzle) - this function finds the next empty cell in the puzzle.
- is_valid(puzzle, row, col, num) - this function checks if a number is valid in a given cell.

Overall, the time and space complexity of the code is reasonable for solving Sudoku puzzles of moderate size, but may become prohibitively large for very large puzzles or for solving a large number of puzzles in a short amount of time.



Q2. Magic Square

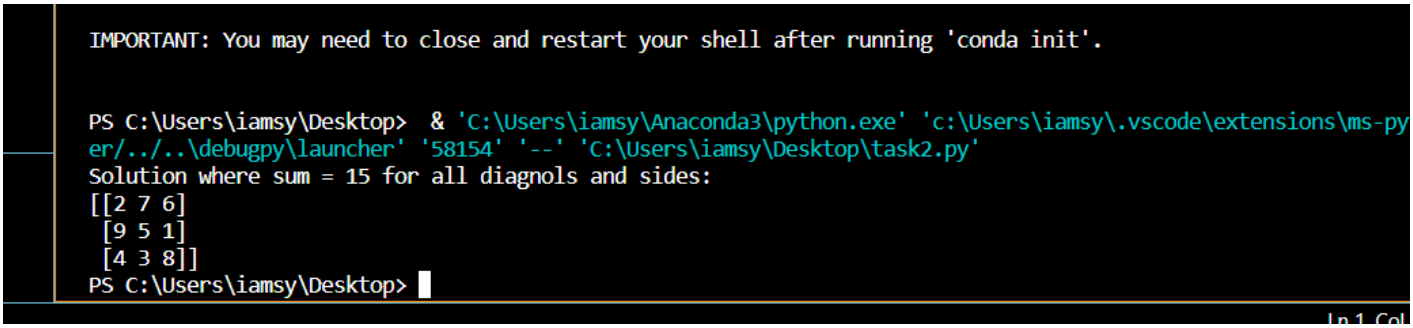
The isSolved() function checks whether a given board is a valid solution for the magic square problem. The function first computes the sum of the diagonal elements of the board and checks whether they are equal to the constant K, which is the desired sum for the magic square.

The isVisited() function checks whether a given state of the board has already been visited.

The solveMagicsquare() uses heuristic approach by assigning a heuristic score to each state in the search space. The heuristic function estimates the distance between the current state and the goal state. This heuristic calculates the difference between the sum of each row in the current state and the target sum that is 15. We use priority queue to store the states to be explored. Each state is assigned a priority based on its heuristic score. We calculate the heuristic score for a state using the rowSumsHeuristic function, which calculates the difference between the sum of each row in the current state and the target sum.

This reduces the number of states that need to be explored, leading to faster solution.

Overall, the time and space complexity of the code is reasonable, but may become prohibitively large for very complex patterns.


IMPORTANT: You may need to close and restart your shell after running 'conda init'.
PS C:\Users\iamsy\Desktop> & 'C:\Users\iamsy\Anaconda3\python.exe' 'c:\Users\iamsy\.vscode\extensions\ms-py
er/../..\debugpy\launcher' '58154' '--' 'C:\Users\iamsy\Desktop\task2.py'
Solution where sum = 15 for all diagnols and sides:
[[2 7 6]
[9 5 1]
[4 3 8]]
PS C:\Users\iamsy\Desktop>
Ln 1 Col