Muneel Haider i210640
AI – CS(G)
Assignment 2

Assignment 2 - Report

Q1: Time Complexity: O(n^2) – Space Complexity: O(n^2)

Pseudocode:

Start:
Initialize an empty Sudoku board.
Fill the main diagonal of the board with random numbers from 1 to 9.
Solve the Sudoku puzzle.
Remove some numbers from the board to create a puzzle.
Display the puzzle.

Solve the Sudoku puzzle:
Find an empty cell in the Sudoku board.
If there are no empty cells:
The puzzle is solved.
Return true.
For each number from 1 to 9:
If the number is valid to be placed in the empty cell:
Place the number in the cell.
If solving the puzzle with this number leads to a solution:
Return true.
Remove the number from the cell (backtrack).
Return false (no solution found).

Remove some numbers from the board to create a puzzle:
Randomly select cells in the board.
If the selected cell is not empty:
Remove the number from the cell.
Decrease the count of numbers to remove.
Repeat until the desired number of numbers is removed.

Display the puzzle:
Print the Sudoku board with some numbers hidden.

Output:

```
One solution found:
[[6 7 8 2 9 1 3 5 4]
 [5 2 1 3 8 4 6 9 7]
 [9 4 3 5 6 7 1 2 8]
 [2 3 7 1 5 9 4 8 6]
 [8 1 5 6 4 2 7 3 9]
 [4 9 6 7 3 8 5 1 2]
 [3 8 9 4 7 5 2 6 1]
 [7 6 2 9 1 3 8 4 5]
 [1 5 4 8 2 6 9 7 3]]
```

```
Multiple solutions found:
[[1 9 2 3 5 4 6 7 8]
 [8 6 4 7 9 2 3 5 1]
 [7 5 3 1 6 8 4 2 9]
 [2 1 5 4 8 3 7 9 6]
 [9 7 6 2 1 5 8 3 4]
 [3 4 8 6 7 9 5 1 2]
 [4 3 1 5 2 6 9 8 7]
 [5 2 9 8 4 7 1 6 3]
 [6 8 7 9 3 1 2 4 5]]

[[1 9 2 3 5 4 6 7 8]
 [8 6 4 7 9 2 3 5 1]
 [7 5 3 1 6 8 2 4 9]
 [2 1 5 4 8 3 7 9 6]
 [9 7 6 2 1 5 8 3 4]
 [3 4 8 6 7 9 5 1 2]
 [4 3 1 5 2 6 9 8 7]
 [5 2 9 8 4 7 1 6 3]
 [6 8 7 9 3 1 4 2 5]]
```

Muneel Haider i210640
AI – CS(G)
Assignment 2

Q2: Time Complexity: O(n) – Space Complexity: O(n)

Pseudocode:

Start:

1. Set the size of the population to 50, number of generations to 1000, and mutation rate to 0.3.
2. Generate an initial population of matrices filled with random numbers from 1 to 9.
3. Display the initial population.

Evolution loop:

For each generation:

a. Sort the population based on how close each matrix is to a magic square.
b. If a perfect magic square is found, return it.
c. Create a new generation by selecting the top half of the current population.
d. Breed and mutate to fill the new generation.
e. Replace the old generation with the new one.

Check if a given matrix is a magic square:

Check if the sums of all rows, columns, and diagonals equal 15.

Calculate fitness of a matrix:

Calculate how much each row, column, and diagonal deviates from a magic square.

Generate a population:

Create a population of matrices with numbers 1 to 9 in random order.

Select parents:

Randomly choose two matrices from the population to be parents for breeding.

Crossover:

Mix parts of two parent matrices to create children.

Mutate:

Randomly swap two numbers in a child matrix to maintain uniqueness.

Display population:

Show all matrices in the population.

Main function to run the genetic algorithm:

Run the genetic algorithm and display the solution if found.

End.

Output: