CSI 3120

Lab 4 Report

Group #6

Carl Li 300235679
Ruoyu Liu 300176134
Roy Rui 300176548

Explanation:

The purpose of this program is to solve a given incomplete 4x4 Sudoku grid using the backtracking algorithm. The input to the program is a 4x4 Sudoku grid, represented as a 2D list, where the number '0' indicates an empty cell. The program will attempt to fill in these empty cells to complete the grid while adhering to the rules of Sudoku. If the grid is solvable, the program outputs the completed grid. Otherwise, it indicates that no solution exists.

Expected Input and Output

- Input**:** A 4x4 Sudoku grid with some cells filled (represented by numbers 1-4) and some cells empty (represented by 0).
- Output**:** If the grid is valid and solvable, the program will output the completed 4x4 Sudoku grid. If the initial grid is invalid or unsolvable, the program will output an appropriate message indicating invalid input or no solution.

1. **Define the Initial Grid:**
   The program starts by defining an incomplete 4x4 Sudoku grid as a list of lists. This grid serves as the input that will be solved by the program. The empty cells are represented by the number '0'.
2. **Check if a Number is Valid:**
   The program includes a function that checks if placing a particular number in a given cell is valid according to Sudoku rules. This function ensures that the number does not already appear in the same row, column, or 2x2 subgrid, which are the core rules of Sudoku.
3. **Verify the Initial Grid:**
   Before attempting to solve the grid, the program verifies that the initial grid is valid. It checks each filled cell to ensure there are no conflicts in the row, column, or subgrid. If the grid is invalid, the program will notify the user and stop further processing.
4. **Implement the Backtracking Algorithm:**
   The backtracking algorithm is used to solve the Sudoku grid. The algorithm finds an empty cell and tries placing numbers from 1 to 4 in that cell. If a number is valid, the algorithm recursively attempts to solve the rest of the grid. If no number works, the algorithm backtracks to the previous cell and tries a different number, repeating this process until the grid is solved or all possibilities are exhausted.
5. **Display the Solved Sudoku Grid:**
   If the grid is successfully solved, the program prints the completed Sudoku grid in a readable format, showing each row of the grid on a separate line.
6. **Handle Edge Cases and Display Results:**
   The main function of the program handles edge cases such as an invalid input grid or an unsolvable grid. It first checks if the initial grid is valid, then attempts

to solve it. Depending on the outcome, it either displays the solved grid or an appropriate message, such as "Invalid input grid" or "No solution exists".

Output:

1.

Sudoku Puzzle 1:

1	0	0	4
0	0	3	0
3	0	0	1
0	2	0	0

```
PS C:\Users\13570\desktop\lab4> opam exec -- dune exec lab4
Initial 4x4 Sudoku Grid:
+---+---+---+---+
| 1  0 | 0  4 |
| 0  0 | 3  0 |
+---+---+---+---+
| 3  0 | 0  1 |
| 0  2 | 0  0 |
+---+---+---+---+

No solution exists for this Sudoku puzzle.
```

2.

Sudoku Puzzle 2:

0	2	0	4
0	0	1	0
0	1	0	0
4	0	0	0

```
PS C:\Users\13570\desktop\lab4> opam exec -- dune exec lab4
Initial 4x4 Sudoku Grid:
+---+---+---+---+
| 0  2 | 0  4 |
| 0  0 | 1  0 |
+---+---+---+---+
| 0  1 | 0  0 |
| 4  0 | 0  0 |
+---+---+---+---+

Sudoku solved successfully:
+---+---+---+---+
| 1  2 | 3  4 |
| 3  4 | 1  2 |
+---+---+---+---+
| 2  1 | 4  3 |
| 4  3 | 2  1 |
+---+---+---+---+
```

3.

Sudoku Puzzle 3:

0	0	0	2
0	3	4	0
0	0	0	0
2	0	0	0

```
PS C:\Users\13570\desktop\lab4> opam exec -- dune exec lab4
Initial 4x4 Sudoku Grid:
+---+---+---+---+
| 0  0 | 0  2 |
| 0  3 | 4  0 |
+---+---+---+---+
| 0  0 | 0  0 |
| 2  0 | 0  0 |
+---+---+---+---+

No solution exists for this Sudoku puzzle.
```

4. one more example to show invalid input

```
PS C:\Users\13570\desktop\lab4> opam exec -- dune exec lab4
Initial 4x4 Sudoku Grid:
+---+---+---+---+
| 2  0 | 0  2 |
| 0  3 | 4  0 |
+---+---+---+---+
| 0  0 | 0  0 |
| 2  0 | 0  0 |
+---+---+---+---+

Invalid input grid
```

Note :
We manually input the numbers that change the Sudoku, so there is only one Sudoku instance in the code section, which is the second test case.

Reference:

1. We know row check to use list.nth, but we don't know how to check column , after assisted by ChatGPT, we use map function to extract the columns first and then use the same logic as for the row check.
2. Then we learned from GPT by using .exists to iterate over all rows, performing checks on each row.
3. The logic of backtracking algorithm is assisted by ChatGPT. find empty cell function finds the first empty cell (value 0) in the grid. It uses recursion to iterate over the grid. If an empty cell is found, it returns Some (row, col).
If there is no empty cell, it returns None. Solve function first look for an empty cell. If there is no empty cell, the solution has been found. fill number function try filling empty cells with numbers (1 to 4). If the number is valid, create a new grid and call solve recursively. If a solution is found, return that solution.
If a solution is not found, try the next number.
4. The display interface for user is improved by ChatGPT since the former version, we only input the 2d array list, then we ask for an optimized intuitive version.