Sudoku Solver Using Backtracking

Heet Shah(2021A7PS0125U) & Arjun Nadar(2021A7PS0062U)

The algorithm used to solve the sudoku puzzle is **Backtracking.** It is basically an add on to the CSP algorithm and uses constraints to solve the problem.

Initialization:

The backtracking algorithm begins with an initial partially filled Sudoku grid.
It aims to find a solution by systematically trying different combinations of numbers in the empty cells.

Recursive Approach:

The algorithm adopts a recursive approach to explore possible solutions.
At each step, it selects an empty cell and tries digits from 1 to 9 to fill that cell.

Constraint Satisfaction:

Before placing a digit in a cell, the algorithm checks if the placement satisfies the Sudoku Rules:

- No duplicate digits in the same row.
- No duplicate digits in the same column.
- No duplicate digits in the same 3x3 subgrid.

If the placement violates any of these constraints, the algorithm backtracks and explores a different digit for that cell.

Backtracking Mechanism:

If the algorithm encounters a dead-end where no valid digit can be placed in an empty cell, it backtracks to the previous step.
Backtracking involves undoing the previous assignment and trying a different digit in the previous cell.
By backtracking, the algorithm explores alternative paths and continues the search for a valid solution.

Exploration of Solution Space:

The algorithm continues this process of recursion and backtracking until it finds a valid solution or exhausts all possibilities.
It systematically explores the solution space, trying different combinations of digits in the empty cells until a valid solution is found.

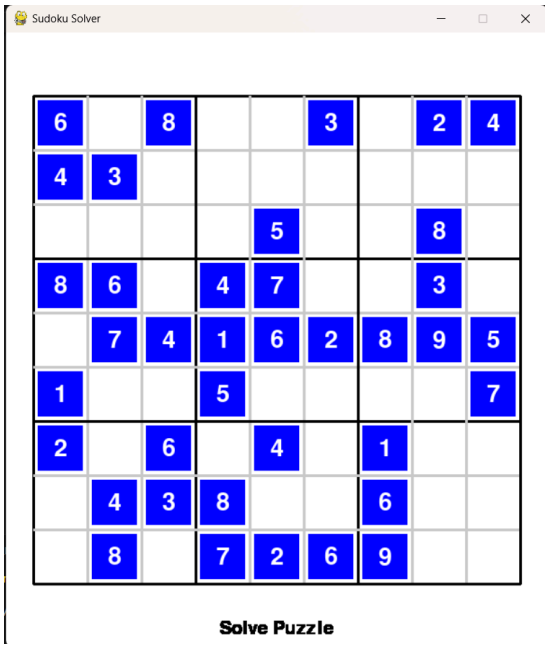
Optimization Techniques:

While the basic backtracking algorithm guarantees finding a solution (if one exists), optimization techniques can be applied to improve efficiency:

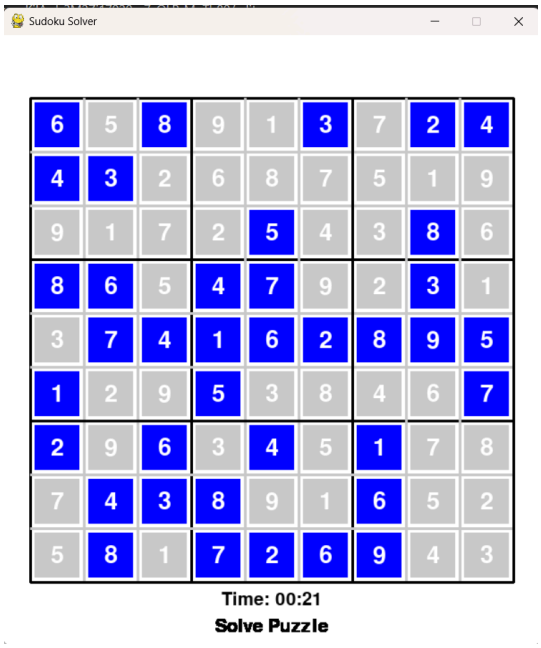
Heuristic selection of the next empty cell to reduce the search space.
Forward checking to eliminate invalid choices early in the process.
Constraint propagation techniques to prune the search tree.

Complexity Analysis:

The time complexity of the backtracking algorithm for Sudoku depends on the number of empty cells in the initial puzzle.
In the worst-case scenario, where the puzzle is mostly empty, the algorithm explores a large solution space, resulting in exponential time complexity.
However, for typical Sudoku puzzles, the algorithm performs efficiently and finds solutions quickly.



Unsolved Puzzle



Solved Puzzle