Assignment 2

Implement Constraint Satisfaction Problem (CSP)

Problem Statement:

The goal of this assignment is to solve a constraint satisfaction problem (CSP) like Sudoku using backtracking. You will learn to represent the problem, define its constraints, and implement an algorithm to find a valid solution.

Objectives:

- Learn to represent and solve CSPs.
- Implement backtracking to find solutions effectively.

Theory

What is a CSP?

A CSP consists of:

- **Variables**: Elements that need to be assigned values (e.g., cells in Sudoku).
- **Domains**: The set of possible values each variable can take (e.g., numbers 1-9 for Sudoku).
- **Constraints**: Rules that dictate how variables can be assigned values (e.g., no two cells in the same row, column, or sub-grid can have the same value in Sudoku).

Methodology

1. **Define Variables, Domains, and Constraints**:
    - For a Sudoku puzzle, each cell is a variable.
    - The domain for each variable is the numbers 1 to 9.
    - Constraints ensure that no two variables in the same row, column, or 3x3 sub-grid can have the same value.
2. **Start with an Empty Assignment**:
    - Initialize an empty grid for Sudoku where you will assign values to variables.
3. **Use Backtracking**:
    - Assign a value to a variable and check if the assignment satisfies all constraints.
    - If it does, move to the next variable.
    - If it does not, backtrack and try a different value for the previous variable.
    - This process continues until a valid solution is found or all possibilities are exhausted.
4. **Continue Until a Valid Solution is Found**:
    - If a valid configuration of the entire grid is reached, output the solution.

- If no solution is possible, indicate that as well.

Working Principle / Algorithm

Here's a basic outline of the backtracking algorithm to solve Sudoku:

1. **Find the Next Empty Cell**:
   - Scan the grid for the next cell that needs a value.
2. **Try Each Possible Value**:
   - For each number from 1 to 9:
     - Assign the number to the empty cell.
     - Check if the assignment is valid (i.e., it does not violate any Sudoku constraints).
     - If valid, recursively call the function to assign values to the next cell.
3. **Backtrack If Necessary**:
   - If the recursive call leads to a solution, propagate that solution upwards.
   - If not, reset the cell and try the next number.
4. **End Condition**:
   - If all cells are filled correctly, the puzzle is solved.

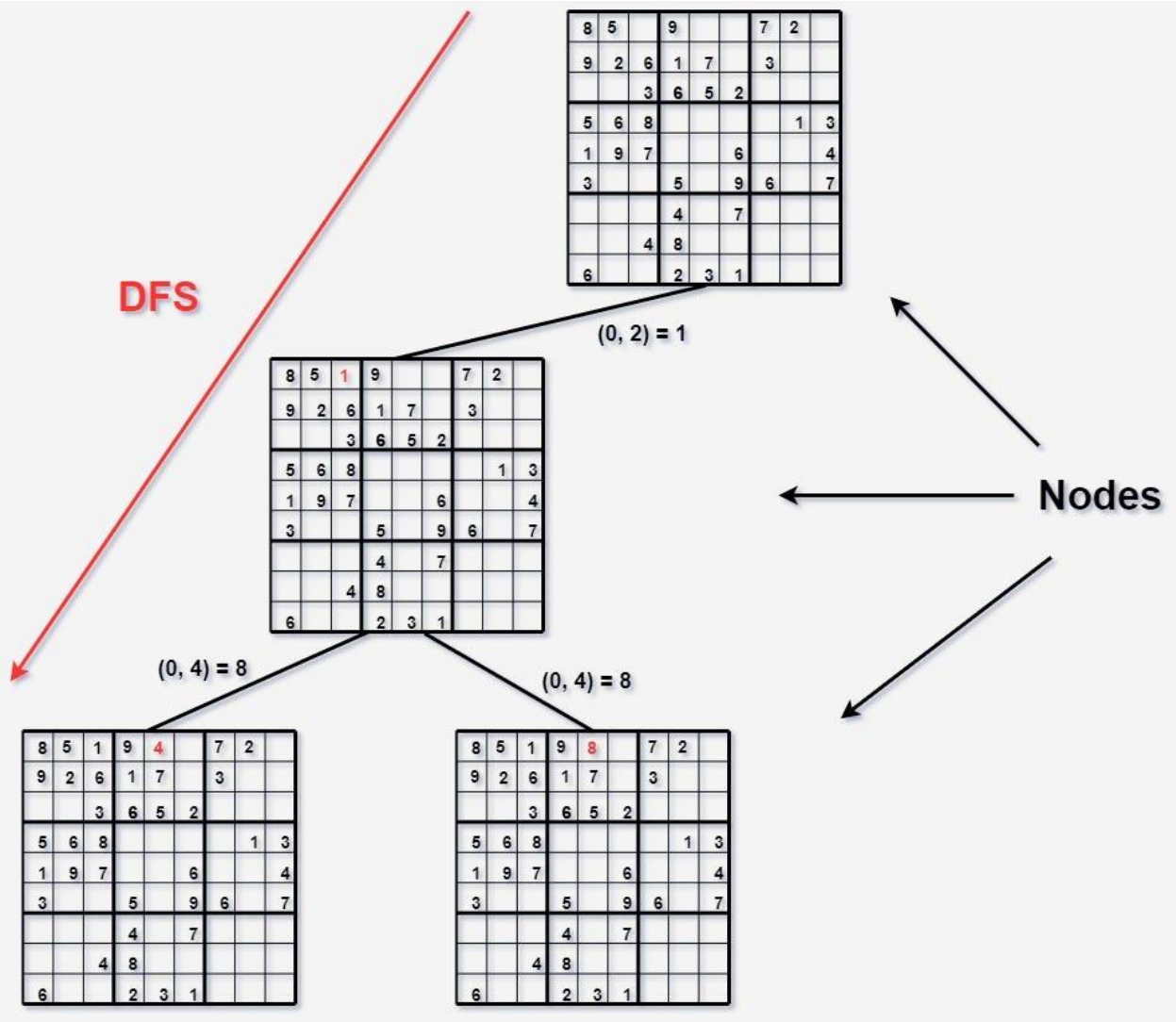
Advantages

- **Structured Approach**: CSPs provide a clear framework to structure problems using variables, domains, and constraints.
- **Effective for Many Variables**: The backtracking algorithm is particularly effective for problems with numerous variables and complex constraints.

Disadvantages / Limitations

- **Performance**: Backtracking can be computationally expensive and slow for large CSPs, particularly as the number of variables and constraints increases.
- **Exponential Time Complexity**: The worst-case time complexity can grow exponentially with the size of the problem.

Diagram



Conclusion

CSPs offer a systematic way to approach complex problems by framing them in terms of constraints. The backtracking technique allows us to explore possible assignments efficiently, making it a valuable tool for solving problems like Sudoku and many others.