Name: Aziz Sayyad
Roll No: 381069
Batch: A3

Practical 6

Implement Basic Search Strategies – 8-Queens Problem

Problem Statement

The goal of this assignment is to solve the 8-queens problem using basic search strategies, specifically through the application of backtracking. The challenge is to place 8 queens on an 8x8 chessboard in such a way that no two queens can attack each other.

Objectives

- Learn to apply search strategies to constraint problems.
- Solve the 8-queens problem using a backtracking approach.

Theory

What is the 8-Queens Problem?

The 8-queens problem is a classic constraint satisfaction problem where the objective is to place eight queens on a chessboard such that no two queens threaten each other. Queens can attack in the same row, column, or diagonal, making it necessary to ensure that all queens are placed in distinct rows, columns, and diagonals.

Methodology

1. **Start with an Empty Board**:
    - Begin with an empty 8x8 chessboard where no queens are placed.
2. **Place Queens One by One**:
    - Attempt to place a queen in a column of the current row and check if it is a valid position (i.e., it does not threaten any previously placed queens).
3. **Use Backtracking**:
    - If placing a queen leads to a conflict (another queen can attack it), backtrack by removing the last placed queen and trying the next available position.
    - Continue this process until all 8 queens are successfully placed or all possibilities are exhausted.
4. **Continue Until a Valid Configuration is Found**:
    - The algorithm will explore all possible configurations of queen placements until it finds one where all queens are safe.

Working Principle / Algorithm

Here's a simple outline of the backtracking algorithm for the 8-queens problem:

1. **Initialize the Chessboard**:
    - Create an 8x8 array initialized to zero, indicating empty squares.
2. **Define a Function to Place Queens**:
    - Use a recursive function that takes the current row as a parameter.
    - If all queens are placed (row equals 8), return true (solution found).
3. **For Each Column in the Current Row**:
    - Check if placing a queen in the current column is valid (no other queens threaten this position).
    - If valid, place the queen (set the board position to 1) and recursively attempt to place the next queen in the next row.
4. **Backtrack If Necessary**:
    - If placing a queen in the current column does not lead to a solution, remove the queen (set the board position back to 0) and try the next column.
5. **Return the Result**:
    - If a valid configuration is found, print or return the chessboard configuration.

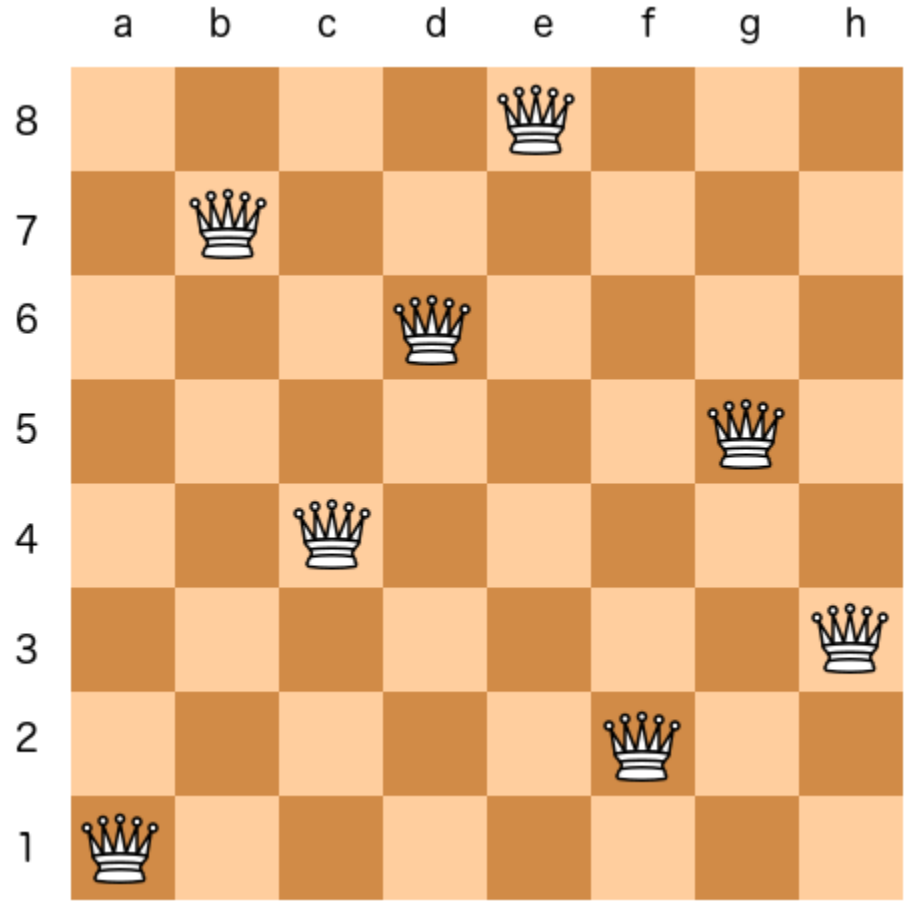
Advantages

- **Simplicity**: The backtracking algorithm is straightforward and easy to implement for problems with clear constraints.
- **Effectiveness**: It can efficiently explore possible configurations to find solutions for constraint problems.

Disadvantages / Limitations

- **Exponential Growth**: The search space grows exponentially with the increase in board size, making it less feasible for larger N-Queens problems (e.g., 16-queens or 20-queens).
- **Performance**: The time complexity can be high if many backtracking steps are needed to find a solution.

Diagram



Conclusion

Basic search strategies like backtracking can effectively solve constraint problems such as the 8-queens problem. This approach not only provides a valid solution but also enhances the understanding of how search algorithms work in problem-solving contexts.