**Assignment 6: Implement Basic Search Strategies – 8-Queens Problem**

**Problem Statement**

To implement basic search strategies to solve the 8-Queens problem, where 8 queens are placed on a 8x8 chessboard without any two attacking each other.

**Objectives**

* To implement the 8-Queens problem using basic search strategies like backtracking.
* To explore and test solutions for placing queens without conflicts on the chessboard.
* To ensure constraint satisfaction by avoiding queen placements that result in threats (rows, columns, diagonals).

**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. **Define the Chessboard and Queens' Positions:**
   * Use an 8x8 matrix to represent the board.
   * Each queen is placed on a unique row and column.
2. **Backtracking Search Strategy:**
   * Begin by placing a queen in the first row and proceed to the next row.
   * Check if placing a queen in a column satisfies the non-attacking constraints (no conflicts on the same column or diagonals).
   * If a conflict is found, backtrack to the previous row and attempt a different placement.
3. **Continue Until a Valid Solution is Found:**
   * Recursively place queens row by row while backtracking whenever a conflict arises.
   * Stop when all 8 queens are successfully placed on the board.

**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.

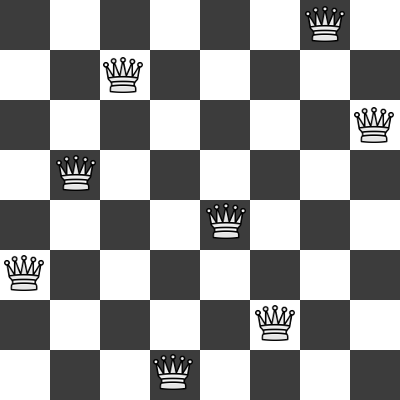
**Advantages**

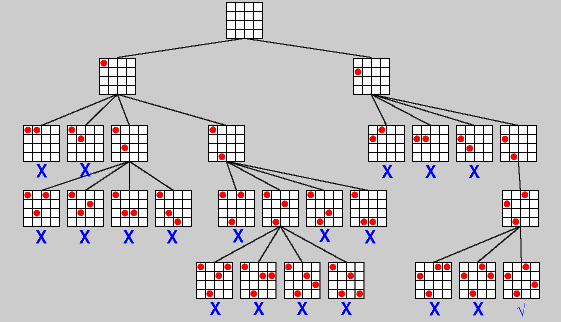
* **Simple Implementation:** The backtracking approach is conceptually simple and easy to code.
* **Constraint Satisfaction:** Directly addresses the constraints of non-attacking queens.
* **Guaranteed Solution:** Ensures that a valid solution, if exists, will be found by exploring all possibilities.

**Disadvantages / Limitations**

* **Exponential Time Complexity**: Backtracking explores all possible solutions, leading to high time complexity for larger problems.
* **Inefficiency for Larger Boards**: As the size of the board increases, the number of configurations to explore grows exponentially, making the algorithm slower.

**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.