Assignment No: - 6

Name :- Isha Ghorpade

Enrollnment :- 22420020

Roll :- 381066

Problem Statement

Implement basic search strategies for solving the 8-Queens Problem.

Objectives

- To understand the concept of state-space search in Artificial Intelligence.
- To apply search strategies (DFS, BFS, Backtracking, Heuristic Search) to solve the 8-Queens Problem.
- To study how AI agents solve constraint-based problems.
- To visualize different possible solutions for placing 8 queens on a chessboard.

Theory

Introduction to 8-Queens Problem

The 8-Queens Problem is a classical constraint satisfaction problem (CSP) in Artificial Intelligence. The task is to place 8 queens on an 8×8 chessboard such that no two queens attack each other.

Constraints:

1. No two queens can be in the same row.
2. No two queens can be in the same column.
3. No two queens can be in the same diagonal.

Representation

- State space → Each state represents a configuration of queens on the board.
- Initial state → Empty chessboard.
- Goal state → 8 queens placed without violating any constraint.
- Operators → Place a queen in the next row, ensuring no conflicts.

Basic Search Strategies

1. Brute Force Search
    - Generate all possible placements of 8 queens.
    - Total = (648)64 \choose 8 ≈ 4.4 billion possibilities.
    - Very inefficient.
2. Depth First Search (DFS)
    - Place queens row by row.
    - Backtrack when a conflict occurs.
    - Explores one solution path fully before trying another.
3. Breadth First Search (BFS)
    - Explore all possible placements level by level.
    - Guarantees the shortest solution but consumes high memory.
4. Backtracking (Most Common Method)
    - Place a queen in one row, check if it is safe.
    - If conflict arises, backtrack to previous row and try next column.
    - Efficient for solving CSPs like N-Queens.
5. Heuristic / Informed Search
    - Use hill climbing or min-conflicts heuristic.
    - Choose the placement with minimum conflicts at each step.
    - Faster but may get stuck in local minima.

Example (Backtracking Approach)

1. Start with an empty chessboard.
2. Place a queen in the first row and first column.
3. Move to the next row and try a safe column.
4. If no safe position → backtrack to the previous row.
5. Repeat until all 8 queens are placed safely.

Advantages

- Demonstrates problem-solving using search strategies.
- Provides a foundation for constraint satisfaction problems.
- Multiple strategies can be compared in terms of time and space complexity.

Limitations

- Brute-force is computationally expensive.
- BFS requires huge memory.
- DFS may go deep into wrong paths.
- Backtracking is efficient but still exponential in worst case.

Applications

- CSP solving in AI (Sudoku, Scheduling, Timetabling).
- Optimization problems in robotics and resource allocation.
- Testing and benchmarking AI search algorithms.

Algorithm (Backtracking Pseudocode)

```
function solveNQueens(board, row):
   if row >= N:
      print(board)   # Solution found
      return True

   for col in range(0, N):
      if isSafe(board, row, col):
         placeQueen(board, row, col)
         solveNQueens(board, row+1)
         removeQueen(board, row, col)  # Backtrack

   return False
```

Conclusion

The 8-Queens Problem is a benchmark for testing AI search strategies. It demonstrates how brute-force, BFS, DFS, and backtracking approaches differ in efficiency. Among them, backtracking and heuristic search methods are most effective in practice. The problem provides an excellent foundation for solving complex CSPs in real-world AI applications.