**Implement basic search strategies – 8-Queens Problem**

**1. Introduction to the 8-Queens Problem**

The 8-Queens Problem is a classic problem in computer science and artificial intelligence, where the objective is to place 8 queens on an 8x8 chessboard such that no two queens threaten each other. In chess, a queen can attack any other piece in the same row, column, or diagonal.

The problem demonstrates the application of constraint satisfaction problems (CSPs) and various search algorithms to solve it.

**2. Problem Statement**

The goal is to position 8 queens on a chessboard in such a way that:

* No two queens are in the same row.
* No two queens are in the same column.
* No two queens share a diagonal.

**3. Basic Search Strategies for Solving the Problem**

There are various search techniques that can be used to solve the 8-Queens problem. Some of the key strategies include:

* **Depth-First Search (DFS)**: Explores each possible arrangement of queens by going deep into one path before backtracking.
* **Breadth-First Search (BFS)**: Explores all arrangements at a given level before moving on to deeper levels.
* **Backtracking**: A more efficient version of DFS, where solutions that violate constraints are abandoned early.
* **Heuristic Search**: Techniques like Hill-Climbing and A\* Algorithm are used to explore the search space based on a heuristic or cost function.

**4. Depth-First Search (DFS) Approach**

**Depth-First Search (DFS)** explores the search tree by going as deep as possible along each branch before backtracking. In the case of the 8-Queens problem, DFS would try placing queens row by row, and backtrack if it finds any conflicts.

**DFS Steps:**

1. Start with an empty chessboard.
2. Place a queen in the first column of the first row.
3. Move to the next row and try placing a queen in a column.
4. If placing a queen leads to a conflict (another queen in the same column or diagonal), backtrack to the previous row and try the next column.
5. Continue until 8 queens are placed or all configurations are exhausted.

While DFS can solve the problem, it is not efficient because it explores all possible configurations, even if they violate the constraints early on.

**5. Breadth-First Search (BFS) Approach**

**Breadth-First Search (BFS)** explores all possible positions for the queens level by level. For the 8-Queens problem:

1. In the first level, the queen is placed in the first row.
2. In the second level, all possible placements of the second queen are considered (in different columns of the second row).
3. The search continues for each subsequent row, exploring all possibilities at each level until a valid solution is found.

BFS ensures that the shallowest solutions are found first, but it is memory-intensive because it stores all possible configurations at each level.

**6. Backtracking Search**

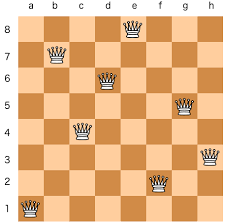
**Backtracking** is a more optimized version of DFS, where invalid states (those that violate constraints) are eliminated as soon as they are encountered, rather than exploring them to the end.

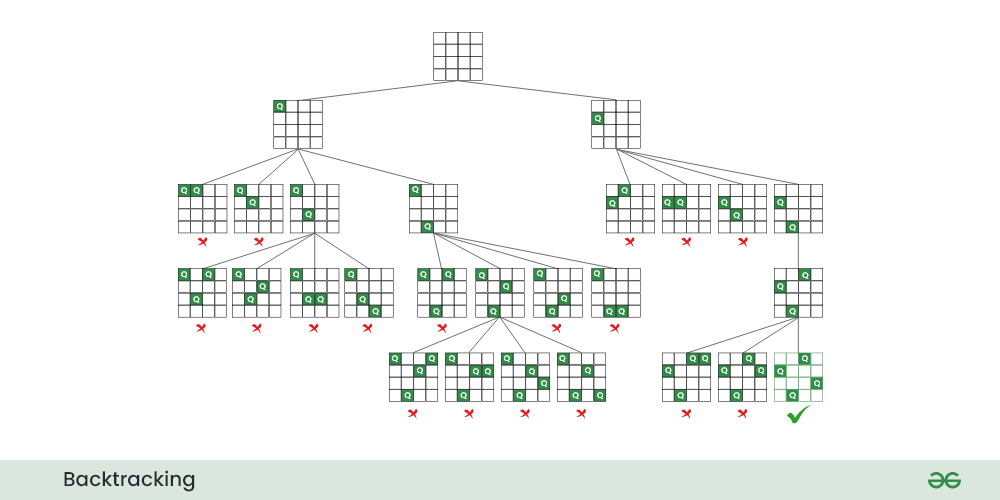
**Steps for Backtracking:**

1. Place a queen in the first row.
2. Move to the next row and try placing a queen in any column that does not conflict with previously placed queens.
3. If no column is valid, backtrack to the previous row and move the queen to the next available column.
4. Continue the process until a solution is found or all possible configurations have been tried.

Backtracking reduces the number of configurations that need to be explored, as it cuts off invalid paths early.

**Diagram**

****

****

**7. Example of 8-Queens Solution Using Backtracking**

Let’s consider an example where queens are placed row by row using backtracking. Each step of the algorithm will involve checking if placing a queen in a particular column of the current row causes any conflicts with queens placed in previous rows.

**Example:**

* Place the first queen in the first row, first column.
* For the second row, try placing a queen in each column:
  + If placing a queen in a column leads to no conflicts, proceed to the next row.
  + If a conflict occurs, backtrack to the previous row and move the queen to a new column.

This method efficiently narrows down the search space by eliminating invalid configurations early.

**8. Heuristic Approaches: Hill-Climbing**

**Hill-Climbing** is a heuristic search algorithm that starts with an initial configuration and attempts to improve it step by step. In the 8-Queens problem, Hill-Climbing aims to reduce the number of conflicts between queens.

**Steps:**

1. Start with a random configuration of queens on the board.
2. Evaluate the number of conflicts (queens that can attack each other).
3. Move one queen to a new position if it reduces the number of conflicts.
4. Repeat until no conflicts remain.

While Hill-Climbing is faster than DFS and backtracking, it can get stuck in **local minima** where no further improvement is possible, even though the solution is not optimal.

**9. Conclusion**

The 8-Queens Problem is a fundamental example of constraint satisfaction problems and demonstrates how various search strategies can be applied to find solutions. Among the approaches:

* Backtracking offers a more efficient solution compared to brute-force methods like DFS and BFS, by pruning invalid configurations early.
* Heuristic methods, such as Hill-Climbing, provide faster solutions but may not always find the optimal one.