

Constraint Satisfaction

Unit No 5

Dr. Praveen Barapatre

Constraint Satisfaction

- A Constraint Satisfaction Problem (CSP) is a mathematical problem where we need to find a solution that satisfies a given set of constraints. It's a common paradigm in artificial intelligence and operations research

Key Components of a CSP

Variables:

- These are the unknowns that need to be assigned values.
- Domains: Each variable has a domain, which is the set of possible values it can take.
- Constraints: These are the restrictions on the values that can be assigned to the variables. Constraints can be unary (involving a single variable), binary (involving two variables), or higher-order (involving more than two variables).

Various techniques are used to solve CSPs

Backtracking Search:

- A systematic search algorithm that explores the search space by making assignments to variables one at a time.
- If a partial assignment violates a constraint, the algorithm backtracks to the previous variable and tries a different value.

Constraint Propagation:

- A technique that reduces the domains of variables by removing values that cannot be part of any solution.
- Common techniques include arc consistency, forward checking, and AC-3.

Local Search:

- A heuristic-based approach that starts with an initial assignment and iteratively improves it by making small changes.
- Techniques like hill climbing, simulated annealing, and genetic algorithms are often used.

Real-World Applications of CSPs

- Scheduling: Scheduling tasks, appointments, or resources.
- Resource Allocation: Allocating resources to tasks or projects.
- Robotics: Planning robot movements and actions.
- Computer Vision: Image analysis and object recognition.
- Natural Language Processing: Parsing and semantic analysis.

N-Queens Problem

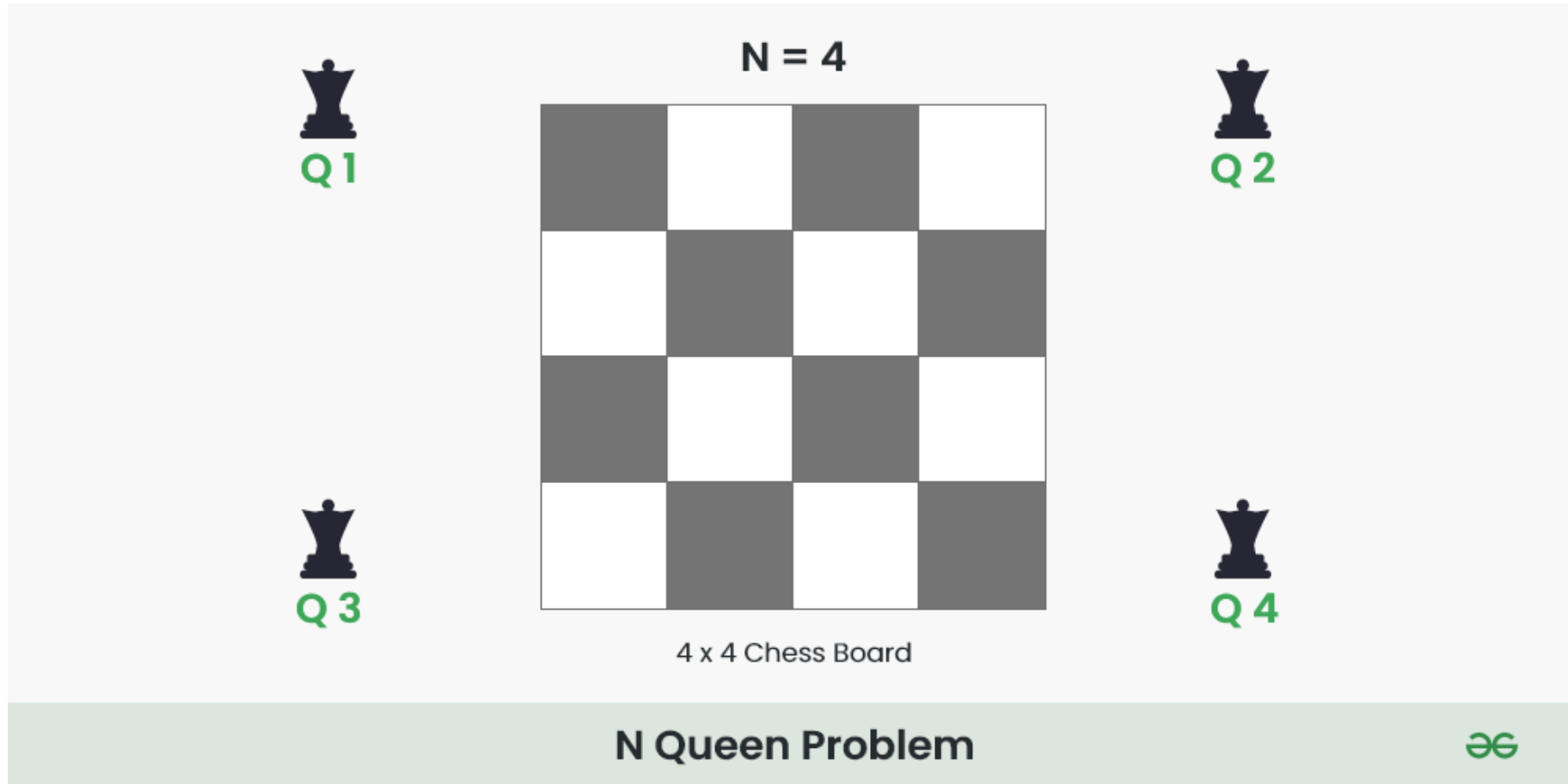
How to solve it

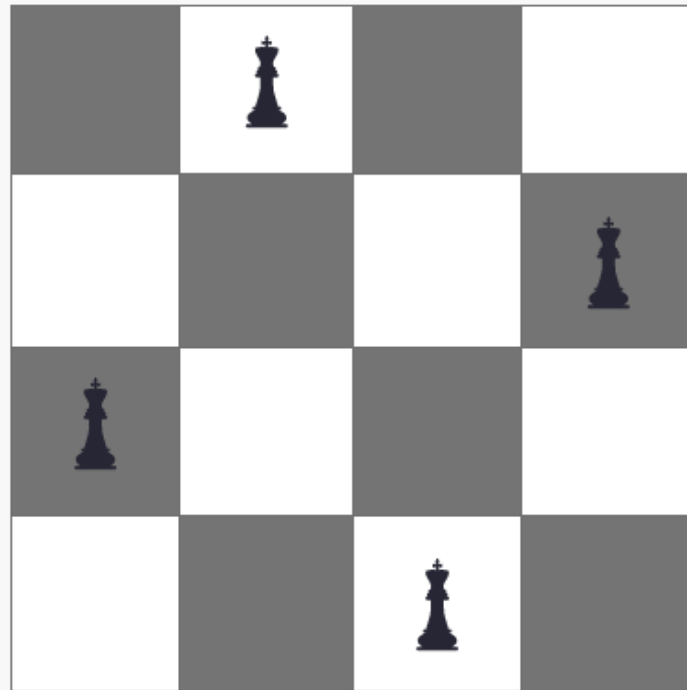
There are several approaches to solve this problem, but a common one involves backtracking:

Backtracking Algorithm:

- Initialize: Create an empty $N \times N$ chessboard.
- Place Queen: Start with the first row and try to place a queen in each column of that row.
- Check for Conflicts: For each placement, check if it conflicts with any previously placed queen.
- Backtrack: If a conflict is found, backtrack to the previous row and try a different column.
- Recurse: If no conflict is found, recursively place a queen in the next row.
- Solution: If all N queens are placed without conflicts, you have found a solution.

The **N** Queen is the problem of placing **N** chess queens on an **N×N** chessboard so that no two queens attack each other.





Solution Of 4 Queen Problem



- The expected output is in the form of a matrix that has 'Q's for the blocks where queens are placed and the empty spaces are represented by '.' . For example, the following is the output matrix for the above 4-Queen solution.

. Q . .

. . . Q

Q . . .

. . Q .

N Queen Problem using Backtracking

- The idea is to place queens one by one in different columns, starting from the leftmost column.
- When we place a queen in a column, we check for clashes with already placed queens.
- In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution.
- If we do not find such a row due to clashes, then we backtrack and return false.



Follow the steps mentioned below to implement the idea:

1. Start in the leftmost column
2. If all queens are placed return true
3. Try all rows in the current column. Do the following for every row.

4. If the queen can be placed safely in this row
 1. Then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
 2. If placing the queen in [row, column] leads to a solution then return true.
 3. If placing queen doesn't lead to a solution then unmark this [row, column] then backtrack and try other rows.
5. If all rows have been tried and valid solution is not found return false to trigger backtracking.