BACKTRACKING

- Backtracking is a problem-solving technique used for finding solutions by exploring all possible options, but systematically pruning (discarding) those that cannot lead to a valid solution.

It works by:

- Building a solution step by step.
- Checking constraints at each step.
- If the current step leads to a dead end, the algorithm unwraps (backtracks) the last step and tries another option.
- Works on the principle of **Depth-First Search (DFS)**.
- Reduces the search space by eliminating invalid possibilities early.

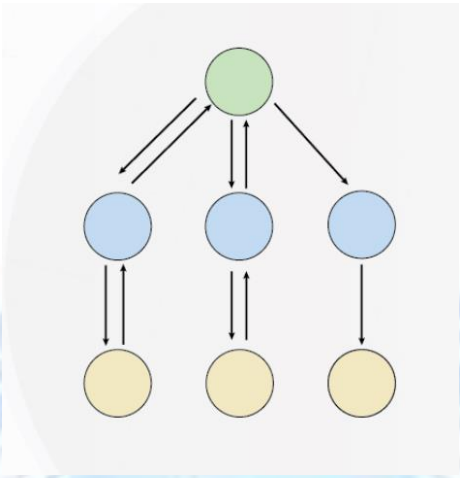
Commonly applied in:

  - **N-Queens Problem**
  - **Sudoku Solver**
  - **Maze Solving**
  - **Graph Coloring**
  - **Subset/Permutation generation**
- It is commonly used in situations where we need to explore multiple possibilities to solve a problem, like searching for a path in a maze or solving puzzles like Sudoku.
- When a dead end is reached, the algorithm backtracks to the previous decision point and explores a different path until a solution is found or all possibilities have been exhausted.

We use backtracking when we need to explore all possible paths (or permutations). Instead of going through every path, we backtrack to avoid

- unnecessary work whenever we are sure that no path from here would lead to a valid solution.

Back tracking



Types of Backtracking Problems

Problems associated with backtracking can be categorized into 3 categories:

- Decision Problems: Here, we search for a feasible solution.
- Optimization Problems: For this type, we search for the best solution.
- Enumeration Problems: We find set of all possible feasible solutions to the problems of this type**.**

N-Queens Problem

- The N-Queens problem is the challenge of placing N queens on an N × N chessboard in such a way that no two queens attack each other.
- A queen can move any number of squares horizontally, vertically, or diagonally.
- Therefore, the conditions are:
    - No two queens in the same row
    - No two queens in the same column
    - No two queens in the same diagonal
- The goal is to find one or all valid arrangements of queens on the chessboard.

Time Complexity

- O(n!) For generating all permutations.

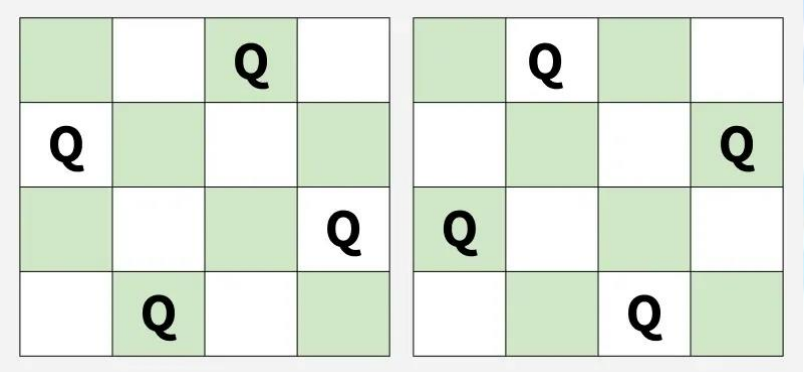
Example

- 4 Queens Problem

Algorithm

1. Start with an empty chessboard.
2. Place a queen in the first row (try each column one by one).
3. For the next row:
    1. Place a queen in a column such that it is safe (no conflict with already placed queens).
    2. If safe → place queen and move to the next row.
    3. If not safe → try the next column.
    4. If no column is valid in the current row → backtrack (remove the queen from the previous row and try another position).
    5. Repeat until:
        1. All queens are placed → solution found.
        2. All possibilities are exhausted → no solution exists.

Solution



Step 1: Start with empty 4×4 board.

Step 2: Place queen in row 0 → column 1.