Documentation for N-Queens Solver

Introduction

The N-Queens problem is a classic puzzle in the field of combinatorial optimization and computer science. It involves placing N queens on an N x N chessboard such that no two queens threaten each other. A queen can attack horizontally, vertically, and diagonally. This documentation provides an overview of the solution approach and implementation for solving the N-Queens problem.

Problem Statement

Given an integer N, the task is to find all distinct solutions to the N-Queens puzzle. Each solution should contain a distinct board configuration of the N-Queens' placement, where 'Q' represents a queen and '.' represents an empty space.

Approach

The solution employs backtracking, a popular algorithmic technique for finding all solutions to a problem by systematically building candidates for a solution, and abandoning a candidate as soon as it is determined to be invalid.

Solution Overview

The solution is implemented in Python using a class named `Solution`. It contains two main functions:

1. **is_safe(board, row, col):** This function checks whether it is safe to place a queen at the specified position `(row, col)` on the board. It verifies if there are any queens in the same column or in the diagonals that could attack the position.

2. **backtrack(board, row):** This function recursively explores all possible placements of queens on the board while maintaining the constraints of the problem. It updates the board configuration and proceeds to the next row if a safe placement is found.

Implementation Details

- The `solveNQueens` method takes an integer `n` as input and returns a list of lists representing all distinct solutions to the N-Queens puzzle.
- The `backtrack` function is responsible for exploring all valid configurations of queens on the board.
- The solution utilizes helper functions to check the safety of queen placements in columns and diagonals.
- The algorithm employs a depth-first search approach to explore the search space efficiently.

Example

For example, given `n = 4`, the output would be `[[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]`, representing two distinct solutions to the 4-Queens puzzle.

Constraints

- The input integer `n` is constrained to be between 1 and 9, inclusive.

Conclusion

The provided solution efficiently solves the N-Queens problem using backtracking, producing all distinct solutions for a given board size. It demonstrates an elegant approach to solving combinatorial problems by systematically exploring the solution space while maintaining constraints.