The 4 queens problem in Artificial Intelligence is a standard logic challenge to understand the concept of search algorithms in practice and how the process leads to a deterministic set of outputs.

Problem statement – To place 4 “Queens” on a standard 8 x 8 chessboard such that no 2 “Queens” attack each other.

The constraints as justified by the chess language are:

1) Queens move along a column
2) Queens move along a row
3) Queens move along diagonals
4) Maximum number of rows = 8
5) Maximum number of columns = 8

To solve this problem, I’ve used the BFS (Breadth-First Search) Algorithm specifically in this problem, which would help us reduce “ Time Complexity “ by a huge margin compared to other conventional DFS algo and print out results rather quick on an IDE console or a terminal window.

// IDE used – PyCharm CE

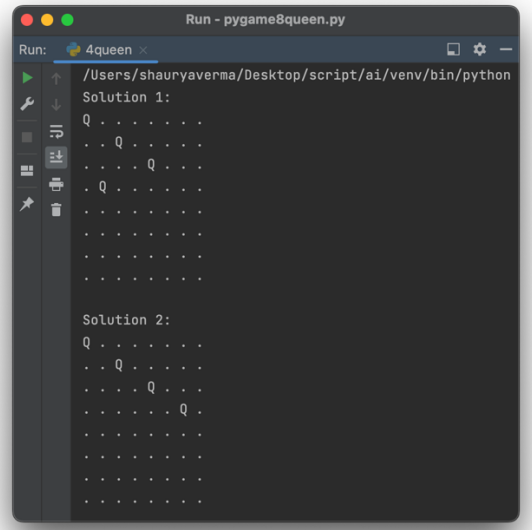
// Checkout the 8puzzle file in the repository to check out the DFS Algorithm

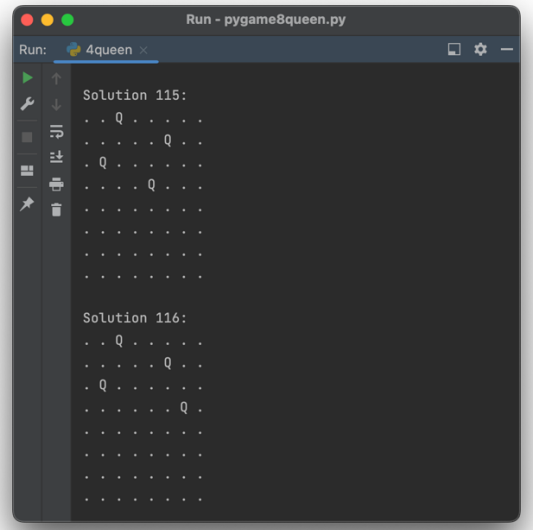
https://github.com/vermashaurya/ai-fundamentals/tree/main/8puzzle

// Checkout the NQueens file in the repository to check out the backtracking method

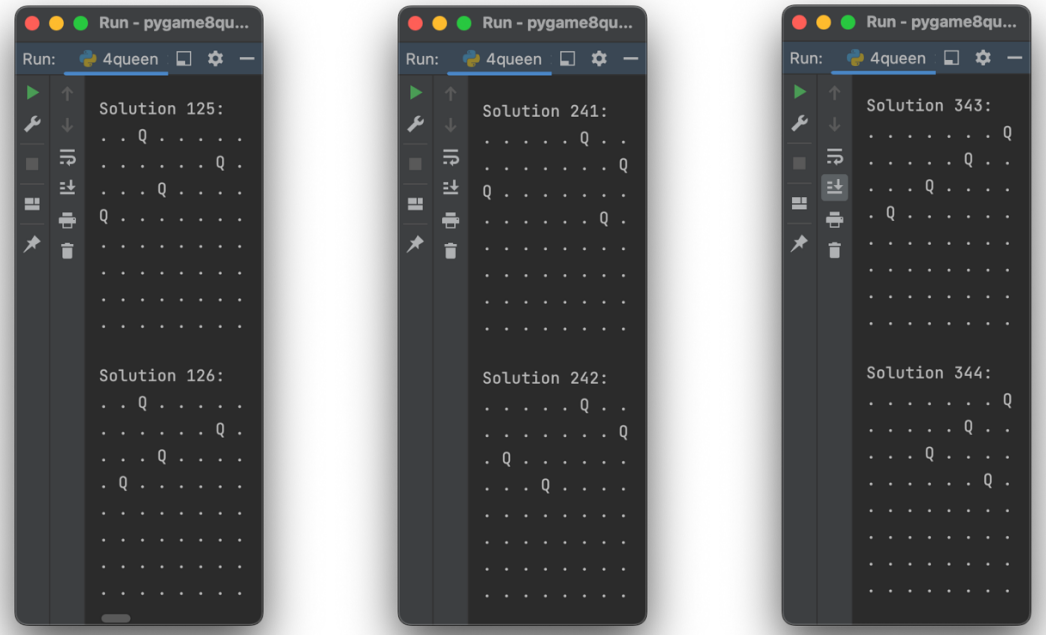
https://github.com/vermashaurya/ai-fundamentals/tree/main/N-Queens

OUTPUT –





“.” represents an empty square on the chessboard
“Q” represents a subsequent Queen occupying the respective square on the chessboard.



Total number of solutions for placing 4Queens on a 8x8 chessboard satisfying the constraints – 344

Checkout this graphical representation of the same, refer to the pygame repository for some more interesting GUI implementations of AI problems using various search algorithms.

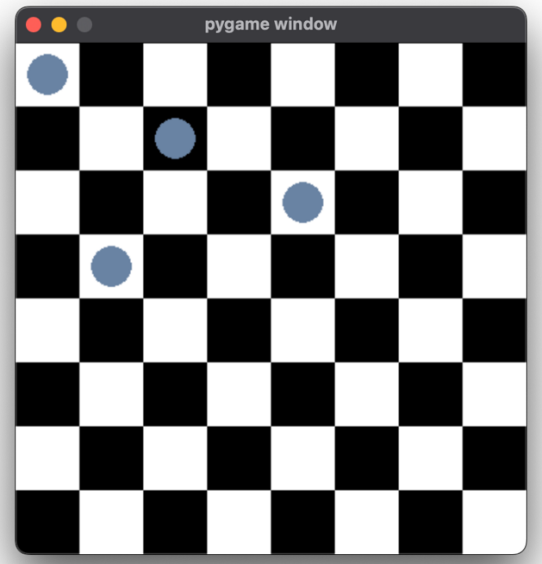


Fig. Solution 1

Python module used – pygame

Happy Coding!