Name: WU BOWEN Student ID: 20257982 Date: 14/10/2025

A One-Page Report on N-Queens and N-Sliding Puzzle

1. Solving the N-Queens Problem

At first, I tried to solve the N-Queens problem in Python. After researching online, I realized that many implementations used a different approach from what I initially imagined. Instead of using a 2D array to represent the chessboard, many use a 1D list like **position = [-1] * n**, where the index represents the row and the value represents the column of the queen. This method is much more memory-efficient.

During the coding process, I encountered many issues and recorded my solutions in the comments. When testing different values of *n*, I found that the program became extremely slow when *n = 13*. After further research, I learned that for the N-Queens problem, **DFS (Depth-First Search)** is generally more efficient than **IDDFS (Iterative Deepening DFS)**—typically 2–5 times faster—because the solution depth is known and all solutions must be found. IDDFS is more useful when the solution depth is unknown or when memory is limited. Both algorithms can find all valid configurations, but DFS tends to have better time complexity for the N-Queens problem.

2. Implementing the N-Sliding Puzzle

Next, I attempted to solve the N-Sliding Puzzle problem. This task was significantly more challenging and required much more code. Some of the main difficulties included how to represent and store states, how to define legal moves, and how to determine whether a puzzle configuration is solvable.

After implementing DFS and IDDFS, and with AI's guidance, I also explored the **A*** and **IDA*** algorithms. I learned that these two heuristic search algorithms perform far better for sliding puzzles, where heuristic functions like the *Manhattan distance* can guide the search efficiently. However, I also noticed that IDDFS tends to be about 2–3 times slower than DFS in this context.

3. Reflections

Although it took me about three days to complete these implementations, the process helped me review Python fundamentals and learn valuable programming and algorithmic problem-solving techniques. Overall, this project deepened my understanding of search algorithms and their practical trade-offs between time, space, and completeness.
You may check my codes at https://github.com/whatsup114514/AI_Courseworks