Breadth-First Search (BFS)

This algorithm implements Breadth-First Search (BFS) to solve a puzzle represented as a matrix. It starts by initializing a frontier with the initial state of the puzzle and an empty path. It iteratively explores possible states by replacing zero values with adjacent numbers, adding these states to the frontier along with the corresponding paths. The process continues until a solved state is found, at which point it prints the solution path, final state, number of explored nodes, time taken, and solution length.

Depth-First Search (DFS)

This function implements a depth-first search algorithm to solve a puzzle specified in a file. It initializes a frontier stack with the initial state and explores possible moves until finding a solution or reaching a depth limit. If a solution is found, it prints the path taken, the final state, and relevant statistics; otherwise, it reports that no solution was found within the depth limit.

Iterative Deepening Search (IDS)

This algorithm implements Iterative Deepening Depth-First Search (IDS) for solving a sliding puzzle problem. It repeatedly performs depth-limited searches, incrementing the maximum depth limit with each iteration until a solution is found or a maximum depth is reached. If no solution is found within a predefined maximum depth limit, the algorithm terminates and reports failure.

A* Search

This algorithm is an implementation of A* search with a Manhattan distance heuristic for solving a sliding puzzle problem. It maintains a frontier of states sorted by their total cost (path cost plus heuristic cost) and explores states in order of increasing total cost. The algorithm terminates when it finds a solution state or exhausts the frontier, returning the solution path if found, along with relevant statistics such as nodes explored and solution length.