A Comparative Analysis of Breadth-First Search (BFS), Depth-First Search (DFS), and A* Algorithms for Solving the 8-Puzzle Game

Geraldine Marie M. Viray

I. BFS, DFS, and A* Algorithms for the 8-Puzzle Game

A. Definition and Comparison of the three algorithms

Breadth-First Search (BFS) is an uninformed search algorithm that begins at the initial state of the 8-puzzle game and then traverses the puzzle through levels. It prioritizes the path that reaches the goal state by exploring the nodes at the earliest levels first. After visiting a node, its adjacent tile will be visited next and then stored in the queue until frontier list is empty. **Depth-First Search (DFS)** is an uninformed search algorithm that also begins at the initial state of the puzzle, then traverses the puzzle depth-wise until the maximum level of that branch is already reached. If the branch explored did not produce a path, it backtracks and searches for a path on the next branch. This will continue until a solution is found. Unlike BFS and DFS, **A* Algorithm** is an informed search algorithm that solves for the heuristic function to estimate the cost of the path that will take to reach the goal state. This algorithm prioritizes the path with the lowest estimated cost. The commonly used heuristic for the 8-puzzle is the Manhattan Distance, which is calculated by getting the sum of the distances of each tile to their goal position.

In terms of data structures used, both BFS and A* Algorithm use a queue, employing the First In First Out (FIFO) method to keep track of the states to explore or their frontiers. On the other hand, DFS uses a stack data structure, following the Last In First Out (LIFO).

B. Test cases: time spent and space occupied

The search algorithms are compared with each other through test cases in terms of time spent and space occupied. Time is measured using the built-in time module in Python, while space occupied is measured by the explored states of each search algorithm.

TEST CASE 1: 1 2 3 4 8 0 7 6 5

search algorithm	path cost	time	explored states
BFS	5	0.0338	45
DFS	57	0.0361	59
A*	5	0.0095	6

TEST CASE 1: 2 3 0 1 5 6 4 7 8

search algorithm	path cost	time	explored states
BFS	6	0.0477	90
DFS	6	0.0110	7
A*	6	0.0117	7

Based on the tables presented above, it can be seen that BFS and DFS have higher numbers of explored states, even though the path cost is the same or their values are near. However, A* algorithm consistently maintains low numbers of explored states and path cost. Additionally, in terms of the time it took for algorithms to find the solution, A* also achieved the lowest average time for both test cases.

C. Most appropriate approach for solving 8-puzzle game

Based on the data gathered on path cost, time spent, and explored states of each search algorithm, it can be concluded that **A* algorithm** is the most appropriate approach for solving an 8-puzzle game. In terms of finding the optimal solution, A* provides the shortest path from the initial state to the goal state with the least space and time consumption. This is achieved because A* prioritizes the path that the node should take toward the goal state using its heuristic function.

D. Least appropriate approach for solving 8-puzzle game

The least appropriate approach for solving an 8-puzzle game is the **Depth-First Search (DFS)**. The primary reason is the extended time it takes for DFS to find the solution since it backtracks after exploring a single path. DFS prioritizes reaching the end of a branch before proceeding to the next, resulting in a higher number of explored states compared to BFS and A* algorithms. Additionally, DFS exhibits inefficiency due to its lack of heuristics, which are utilized by the A* algorithm. Moreover, it does not guarantee an optimal search unless the solution is shallow.

II. Informed and Uninformed Search Strategies

A. Distinguishing informed and uninformed search strategies

Uninformed search algorithms explore problem states without relying on any specific knowledge or heuristics concerning the game and its objective. They make decisions and progress exclusively based on the structure of the search space, without factoring in the quality or the potential cost of the path leading to the goal state. Examples of such algorithms include Breadth-First Search and Depth-First Search.

Informed search is a type of search algorithm that possesses information about the goal state for each tile, which significantly enhances the efficiency of the search process. This information is acquired through the utilization of a heuristic function that estimates the proximity of the current state to

the goal state. Moreover, informed search takes path cost into account when determining which states to prioritize during exploration. An example of such a search algorithm is the A* algorithm.

B. Comparison of informed and uninformed search behavior and performance

Informed and uninformed search exhibit different behaviors and performance when seeking a path or solution. Informed search utilizes a heuristic function to estimate the distance or cost of a path leading from the current state to the goal state. This algorithm follows a directed path, prioritizing traversal that brings it closer to the goal state. Informed search is particularly valuable for solving puzzles and finding optimal paths.

On the other hand, uninformed search lacks additional information about the goal state and relies on systematic exploration of the search space. It is a blind approach as it lacks a mechanism for seeking paths closer to the goal state. Consequently, this results in a slower search process compared to informed search. Uninformed search is useful when the objective is to explore all possible states without a specific goal state in mind.

Overall, the choice between informed and uninformed search depends on the problem's nature and the availability of heuristic information. Informed search is efficient when heuristic knowledge is accessible, while uninformed search is suitable for exhaustive exploration without specific guidance towards a goal state.