**Özyeğin University**

**CS551 – Introduction to Artificial Intelligence**

**Assignment-1**

**Solving N-Puzzle Problem with Search Algorithms**

Furkan Cantürk

03.30.2021

1. **PROBLEM**

**n-puzzle game** consists of a board holding  tiles numbered from 1 to and one blank space which is represented as 0 in this assignment. Given an initial state and goal state of the board, the combinatorial search problem is to find a sequence of moves that transitions this state to the goal state. Actions that can be taken to solve the problem is swapping the blank space in one of the four directions . Moving from a state to its one of successor state has a unit cost. Thus, the total cost of path from the initial state to the goal state is the number of moves.

1. **ALGORITHMS**
   1. **Breadth-First Search (BFS)**

Breadth-first search always expands the shallowest unexpanded node. a First-in First-out (FIFO) queue is used for the frontier. The newly generated nodes always go to the back of the queue, while the older nodes get expanded first. Goal test is applied for each successor node before inserting into the frontier.

BFS is complete. BFS is optimal if the path cost is a non-decreasing function of depth.

Its both time and space complexity are where is the branching factor and is tree depth which goal state is reached at.

* 1. **Uniform Cost Search (UCS)**

UCS expands the node with least path cost. A Priority Queue based on cost is used for the frontier. Goal test on the node is performed when it is selected for expansion. This is because the first node generated could be a sub-optimal path.

Completeness is guaranteed only if cost of every step is positive.

Uniform-cost search is optimal. At every step the path with the least cost is chosen, and paths never gets shorter as nodes are added, ensuring that the search expands nodes in the order of their optimal path cost.

Its both time and space complexity are where is optimal path cost and is least cost at a step.

* 1. **Depth-First Search (DFS)**

Depth -first search always expands the deepest unexpanded node. a Last-in First-out (LIFO) queue (i.e. a stack) is used for the frontier. Goal test is applied for each successor node before inserting into the frontier.

DFS is not complete and non-optimal in nature.

Its both time and space complexity are where is the maximum depth of the search tree.

* 1. **A\* Search**

A\* search is an informed search algorithm. When a search algorithm is guided by an information obtained during search, which is generally heuristic, it becomes an informed search. It combines behaves of Dijkstra’s algorithm prioritizing states that are close to the starting point and Best-First-Search prioritizing states that are closer to the goal.

Let g(n) represent the cost of the path from the initial state to state n, and h(n) represent an admissible heuristic estimated cost of the path from n to the goal. A heuristic information is admissible if it does not overestimate cost to reach goal. A\* builds a priority queue of nodes ordered on f(n) = g(n) + h(n) which is the total estimated cost through n. Goal test is applied for a node when it is expanded. In this assignment, Manhattan Distance is used as the heuristic for A\*.

A\* search has a time complexity of and keeps all generated nodes in memory.

1. **RESULTS**

In this assignment, the algorithms benchmarked for 8-puzzle problem. I created 50 random initial states and got goal states with some random number (between 10 and 50) of random moves starting from given initial states. All tests are done with Python 3.8.3. The following table presents number of expanded nodes, reached tree depth (number of moves from initial state to goal state), and solve time in milliseconds by each algorithm for each problem instance. The last row of the table is the average values of each column.

DFS can find a solution within 1181 ms at average but its solution cost is too away from optimal cost. Since it is not guided based on any cost information, it follows a random path from initial state to goal state.

A\* can find the optimal solution in 2 ms at average and it outperforms BFS and UCS by a substantial difference. A\* proceeds an informed search using a heuristic so that it does not make exhaustive search and goes on one direction during the search until f value of current path increases. In the comparison of BFS and UCS, BFS expands less nodes and so can find the solution earlier than UCS since BFS applies the goal test before inserting nodes into the frontier.



**Table 1. Expanded nodes, search depth, and solution time in ms by each search algorithm**