CS451/551 – Introduction to Artificial Intelligence

Assignment-1

Solving N-Puzzle Problem with Search Algorithms

Author: Ertuğrul Özvardar / S012366

Introduction

In this assignment, 4 different well-known search algorithms were implemented in order to solve N-Puzzle Problem where N is 8. To be more specific, breadth-first search (BFS), depth-first search (DFS), uniform cost search (UCS) and A* search (A*) algorithms were implemented respectively. During the process, some differences occurred among algorithms in terms of implementation and obtained results.

Algorithms

1) Breadth-first search (BFS) Algorithm

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

2) Depth-first search (DFS) Algorithm

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

3) Uniform cost search (UCS) Algorithm

Uniform-Cost Search is a variant of Dijkstra's algorithm. Here, instead of inserting all vertices into a priority queue, we insert only source, then one by one insert when needed. In every step, we check if the item is already in priority queue. If yes, we perform decrease key, else we insert it.

4) A* Search (A*) Algorithm

A* is a modification of Dijkstra's Algorithm that is optimized for a single destination. Dijkstra's Algorithm can find paths to all locations; A* finds paths to one location, or the closest of several locations. It prioritizes paths that seem to be leading closer to a goal. In this algorithm, priority queue is used. Also, a function which tries to estimate the cost to reach the target-node is applied in order to optimize the algorithm and get better results. This function is commonly called Heuristic function.

Results

As it is mentioned in the introduction part, different result sets were obtained for the running processes of the algorithms. As an example,

In one scenario, our initial and goal state sets are the following,

Initial state is entered as:

0	***2***	***3***
1	***4***	***5***
8	***7***	***6***

And the goal state that we are trying to achieve is entered as:

1	***2***	***3***
8	***0***	***4***
7	***6***	***5***

I. Results obtained from the first case (BFS Algorithm):

Num. of visited nodes: 59
Depth of graph: 6
Elapsed time: 0.010804891586303711 secs.

II. **Results obtained from the second case (DFS Algorithm):**
Num. of visited nodes: 1335
Depth of graph: 1320
Elapsed time: 0.04289054870605469 secs.

III. **Results obtained from the third case (UCS Algorithm):**
Num. of visited nodes: 68
Depth of graph: 6
Elapsed time: 0.003000020980834961 secs.

IV. **Results obtained from the fourth case (A* Algorithm):**
Num. of visited nodes: 15
Depth of graph: 6
Elapsed time: 0.001995563507080078 secs.

Implementation Details

Although implementation of these algorithms differed from each other, they were comprised of one coding structure. This structure is basically the coding template of breadth-first search algorithm. Firstly, I implemented BFS Algorithm. It consists queue structure for implementation. After that, I only made small modifications for the remaining algorithms (DFS, UCS, A*). Since DFS Algorithm uses stack structure, I updated the methods which can be used for stack rather than queue. Similarly, for the UCS, I changed the methods that can be applicable for priority queue structure. Yet, the most challenging one was the A* Algorithm. Because replacing the methods for the priority queue was not enough. The implementation of finding the depth of the graph was also changed. Also, Manhattan distance which is an admissible heuristic function differentiated the A* algorithm from the rest of them.

Summary

As a conclusion, 4 different well-known search algorithms such as breadth-first search (BFS), depth-first search (DFS), uniform cost search (UCS) and A* search (A*) were implemented to solve N-Puzzle Problem with the given initial and goal state sets that are mentioned in the results part. Consequently, we end up three types of conclusion such as the number of visited nodes, depth of graph and elapsed time.

i. **Among those 4 algorithms; the number of the visited nodes can be denoted as:**

number_nodes (DFS) > number_Nodes (UCS) > number_Nodes(BFS) > number_Nodes(A*)

ii. **Also, depth of the graphs has the relation as follows.**

depth (DFS) > depth (UCS) = depth (BFS) = depth (A*)

iii. **Below, the elapsed times of the algorithms are listed.**
total_time (DFS) > total_time (BFS) > total_time (UCS) > total_time (A*)