CS451 Report – Assignment 1

Barış Özbaş – S014669

1. Introduction

This report has information and details of implementation of algorithms such as

- BFS - Breadth-First Search
- DFS - Depth-First Search
- UCS - Uniform-Cost Search
- A* - A star

2. Algorithms

- **Breadth-First Search**

  Breath- Search is an algorithm used to navigate and scan data structures, node, and graph. It begins from the root and examines all the neighboring nodes at the current point, instead of going on to the child nodes of the neighbors at the next level.

- **Depth-First Search**

  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.

- **Uniform-Cost Search**

  Uniform-Cost Search is a variant of Dijkstra's algorithm. Here, in preference to placing all vertices into a priority queue, we insert simplest source, then separately insert when needed. In every step, we test if the item is already in priority queue (the usage of visited array). If yes, we perform decrease key, else we insert it.

- **A* Algorithm**

  A* is a graph traversal and path search algorithm, which is often used in computer science due to its completeness, optimality, and optimal efficiency. One major practical drawback is its space complexity, as it stores all generated nodes in memory. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A* is still the best solution in many cases.

3. Implementation

- **Breadth-First Search**
  - Initialize a queue with a start node. Pick any node, visit the adjacent unvisited vertex, mark it as visited and insert it in a queue. If there are no remaining adjacent vertices left, remove the first vertex from the queue. Repeat previous 2 steps until the queue is empty or the desired node is found.

- **Depth-First Search**
  - Initialize a stack with a start node. Pop an item from the stack and mark it as visited. Check if the current item is our target state if it is return it as solution if not expand it and append unvisited children to stack. Repeat until the stack is empty or the desired node is found.

- **Uniform-Cost Search**
  - Initialize a priority queue with a start node which has 0 cost. Pop the least costly item from the queue and mark it as visited. Expand the item and append the unvisited children having cost of parent's cost + 1, check if it is the solution if yes, return it, if not, continue until the queue is empty.

- **A-star Search**
  - Initialize the priority queue with a start node which has cost of Manhattan Distance (MD) to the goal state. Pop the least costly item from the queue and mark it as visited. Expand the item and append the unvisited children having cost of parent's cost + MD + 1. Check if it is the solution if yes, return it, if not, continue until the queue is empty.

4. Performance

Initial State

1	2	3
8		4
7	6	5

Goal State

2	8	1
	4	3
7	6	5

Algorithm	Number of visited nodes	Depth	Time elapsed (s)
BFS	59	6	0.002
DFS	1807	1320	0.053
UCS	68	6	0.003
A-star	23	6	0.002

5.Conclusion

Four different algorithms are implemented to solve the N-puzzle problem. Their performance shows that A-star algorithm is the best, in case of time complexity. Manhattan distance calculation has improved the performance of UCS and created the A-star algorithm.