Implementation of Different Algorithms on 8-Puzzle

This article tries to explain and show the differences between different techniques in tree search while solving 8-Puzzle. In this project, techniques are implemented and tested in Python.

There are 4 different algorithms: BFS(Breadth-First Search), DFS(Depth-First Search), UCS(Unit Cost Search), A\* search. BFS is searching for the shallowest node while DFS is looking for the deepest node by expanding most-left node always. UCS is a bit more evolved, looking for minimum cost node. A\* is the newer technique for tree search that measures the manhattan-distance, which is total of cost and the estimated distance between result and current node.

All of the algorithms are implemented to a one-dimension array. To avoid the infinity tree, a dictionary(list of visited nodes) is hold and all of the nodes are checked if visited or not. Visited nodes have not been extended. To simulate BFS and DFS algorithms, just an array is implemented and managed add and delete operations based on FIFO(queue)(for BFS) and LIFO(stack)(for DFS). In UCS and A\*, priority queue is hold. In each step, the priority queue is pulling the node with minimum value of cost(for UCS) and manhattan-distance(for A\*) from array and evaluating the node.

Test is made by only checking the thought worst-case situation. Which is:

0 1 2 8 7 6

3 4 5 to 5 4 3

6 7 8 2 1 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Result: | BFS | DFS | UCS | A\* |
| Num. Of visited Nodes | 178224 | 89463 | 180457 | 657 |
| Depth of Graph | 28 | 84098 | 28 | 28 |
| Runtime(sec) | 18.24 | 88.63 | 20.31 | 0.08 |

Hereby, you can see that DFS is not the worst case by num. of visited nodes, but is the worst case by runtime, because DFS is dealing with too many redundancies and losing time with dictionary checks in 8-Puzzle example. BFS and UCS’s values are close to each other. UCS is working like BFS in this case, because the cost of the nodes are step values of these nodes. And we can see A\* search is highly effective based on other algorithms because it doesn’t check all nodes until it finds a solution, just trying to take the shortest way to the solution by checking estimated distance from result.