CS4750-HW2

Team member's names: Song Vu Nguyen, Caleb Fagan, Charlermpon Thongmotai

References: For *depth_first_graph_search.py, iterative_deepening_tree_search.py, node.py, problem.py*, we implemented using the code from https://github.com/aimacode/aima-python/blob/master/search.py and modified it a bit to fit with the assignment requirement. For the A* tree search using Manhattan distance heuristic (*a-star-python.py*), we implemented our code based on the pseudo code from https://en.wikipedia.org/wiki/A*_search_algorithm.

Description: Iterative deepening tree search and depth first graph search are implemented using the Node and Problem class. The node class handles keeping track of parent, child and expanding itself, basically it handles the tree/graph structure so our search doesn't have to worry about that. The problem class handles goal testing and managing the 4x4 square, again so that our search algorithm doesn't have these details to worry about. Iterative deepening tree search uses a counter and depth first limited search. This is so that depth first search acts more like breadth first search without the memory issues. Depth first graph search is just depth first search, that doesn't expand the same node twice. These were both developed on a windows laptop (8GB of 1600MHz LPDDR3 RAM and 2.2GHz dual-core Intel Core i5) using VS Code, python extension, to run and debug. For A* algorithm, all the nodes are arranged in a tree where links between nodes represent valid steps in solving the problem. However, we need a heuristic function to help us cut down on this huge search problem. What we need is to use our Manhattan distance heuristic at each node to make an estimate of how far we are from the goal. In pathfinding we know exactly how far we are, because we know how far we can move each step,

and we can calculate the exact distance to the goal. For 15-puzzle solving, we use array to represent the tree. We set the test case input as a start state (i.e. root node), and then for each node we expand all the nodes beneath it in the tree by applying all the possible moves that can be made at each point. We give each node a score, that calculates from Manhattan distance heuristic function, on how good we think it is. This score should be thought of as the cost of getting from the node to the goal plus the cost of getting to where we are. Traditionally this has been represented by the letters f, g and h.

- **'g'** is the sum of all the costs it took to get here
- **'h'** is our heuristic function, the estimate of what it will take to get to the goal.
- **'f'** is the sum of these two, we also store **'f'** in each node.

Using the f, g and h values the A* algorithm will be directed, subject to conditions we will look at further on, towards the goal and will find it in the shortest route possible. This was developed on a MacBook Air laptop (8GB of 1600MHz LPDDR3 RAM and 1.8GHz dual-core Intel Core i5) using Jupyter notebook to run and debug.

Result:

Test case 1:

1. Iterative deepening tree search (IDS)

    a. First 5 search nodes that expanded (order left to right):

1	2	7	3
5	6	11	4
9	10	15	8
13	14	12	

1	2	7	3
5	6	11	4
9	10	15	8
13	14	12	

1	2	7	3
5	6	11	4
9	10	15	
13	14	12	8

1	2	7	3
5	6	11	4
9	10	15	8
13	14		12

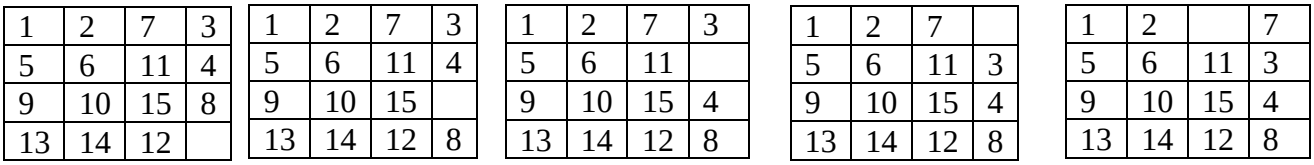
1	2	7	3
5	6	11	4
9	10	15	8
13	14	12	

    b. Solution Sequence
        i. 8 Steps
        ii. ['LEFT', 'UP', 'UP', 'UP', 'RIGHT', 'DOWN', 'DOWN', 'DOWN']

c. Nodes Expanded
    i. 2705 Nodes
d. Time Taken
    i. 37.142 ms

2. Depth-first graph search (DFGS)

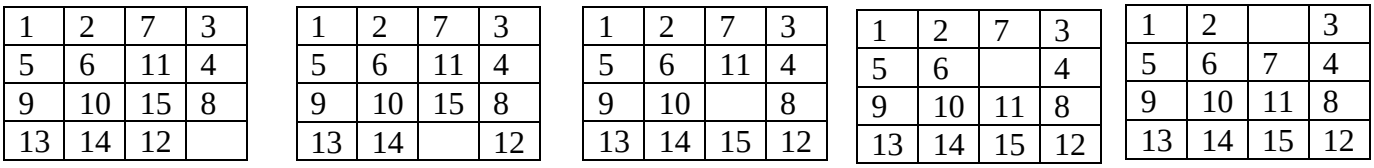
a. First 5 search nodes that expanded (order left to right):



b. Solution Sequence
    i. No solution found
c. Nodes Expanded
    i. 1,000,000 Nodes
d. Time Taken
    i. 18571.336 ms

3. A* tree search

a. First 5 search nodes that expanded (order left to right):

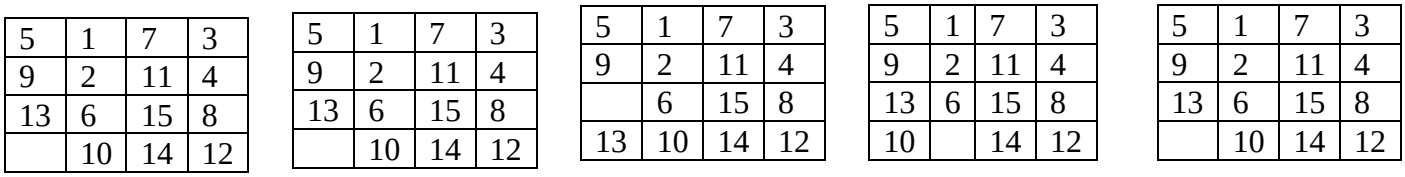


b. Solution and total number of moves to reach the solution:
    i. 8 Steps
    ii. [LEFT, UP, UP, UP, RIGHT, DOWN, DOWN, DOWN]
c. Number of nodes expanded:
    i. 9 nodes
d. CPU execution time:
    i. 5.204 ms

Test case 2:

1. Iterative deepening tree search (IDS)

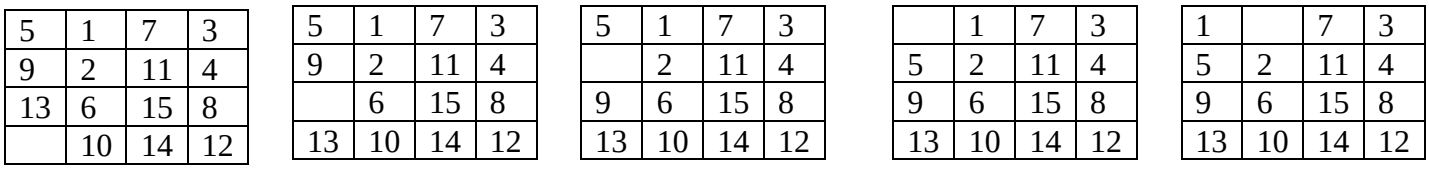
    a. First 5 search nodes that expanded (order left to right):



    b. Solution Sequence
        i. No solution found
    c. Nodes Expanded
        i. 1,000,000 Nodes
    d. Time Taken
        i. 12948.824 ms

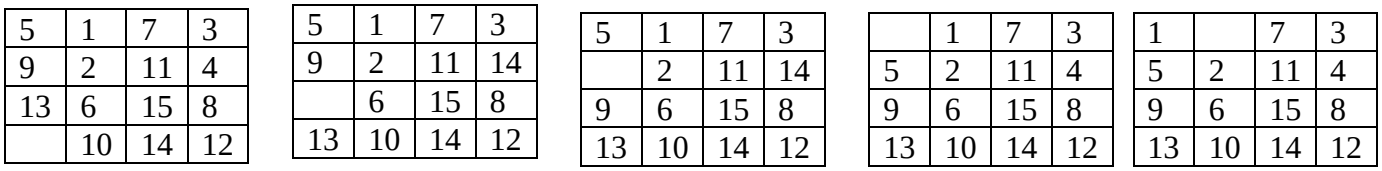
2. Depth-first graph search (DFGS)

    a. First 5 search nodes that expanded (order left to right):



    b. Solution Sequence
        i. No solution found
    c. Nodes Expanded
        i. 1,000,000 Nodes
    d. Time Taken
        i. 18714.244 ms

3. A* tree search
    a. First 5 search nodes that expanded (order left to right):



b. Solution and total number of moves to reach the solution:
    i. 15 Steps
    ii. [UP, UP, UP, RIGHT, DOWN, DOWN, DOWN, RIGHT, UP, UP, UP, RIGHT, DOWN, DOWN, DOWN]
c. Number of node expanded:
    i. 16 Nodes
d. CPU execution time:
    i. 9.639 ms