Braden Katzman
COMS W4701 Artificial Intelligence
Professor Ansaf Salleb-Aouissi
HW1 Report

HOW TO RUN THE PROGRAM:
Usage: python <hw1_bmk2137.py> <search alg: bfs, dfs, a*, id_a*>

Note: search algorithm names are case insensitive, but must be typed correctly as above

Lines 714-716 are where to update the board instance. On lines 28-32, I defined a few board configurations (2D lists) and pass these to the main method to solve an instance of the puzzle. Just add new boards to the file anywhere and replace the names and dimension on these lines to reflect:
714 - n = len(puzzle_name)
715 - node = N_Puzzle_Node(n) # this line doesn't need to be changed
716 - node.replace_state(puzzle_name)

4. Summarize and compare the results of the search algorithms: BFS, DFS, A*, Iterative Deepening A*

Table Configuration from HW1.pdf used:
[1,2,5]
[3,4,0]
[6,7,8]

a. Solution:
   i. BFS: ['UP', 'LEFT', 'LEFT']
   ii. DFS: ['UP', 'LEFT', 'LEFT']
   iii. A*: ['UP', 'LEFT', 'LEFT']
   iv. Iterative Deepening A*: ['UP', 'LEFT', 'LEFT']

b. Cost of Path:
   i. BFS: 3
   ii. DFS: 3
   iii. A*: 3
   iv. Iterative Deepening A*: 3

c. Number of Nodes Expanded
   i. BFS: 9

ii. DFS: 3
iii. A*: 42
iv. Iterative Deepening A*: 9

d. Max Depth of Stack/Queue

i. BFS: 4
ii. DFS: 4
iii. A*: 4
iv. Iterative Deepening A*: 4

e. Memory Requirements (exact)

i. BFS: 10 mb
ii. DFS: 10 mb
iii. A*: 10 mb
iv. Iterative Deepening A*: 10 mb

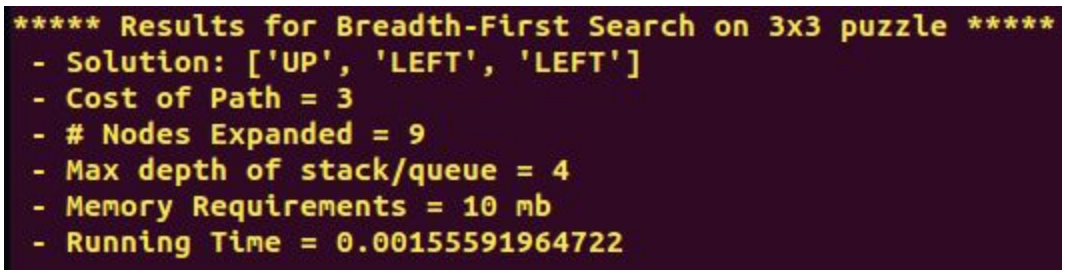
f. Memory Requirements (general)

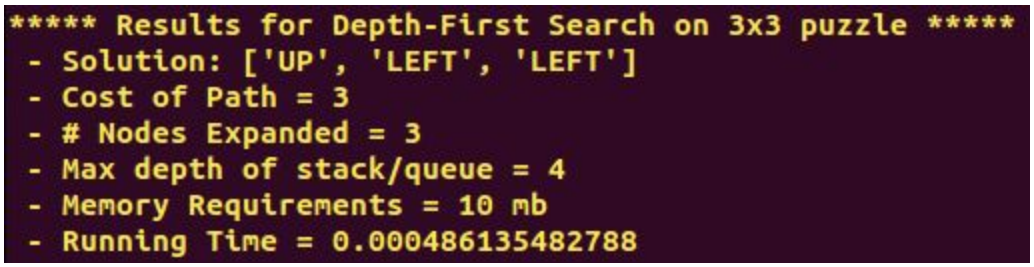
i. BFS
ii. DFS
iii. A*
iv. Iterative Deepening A*

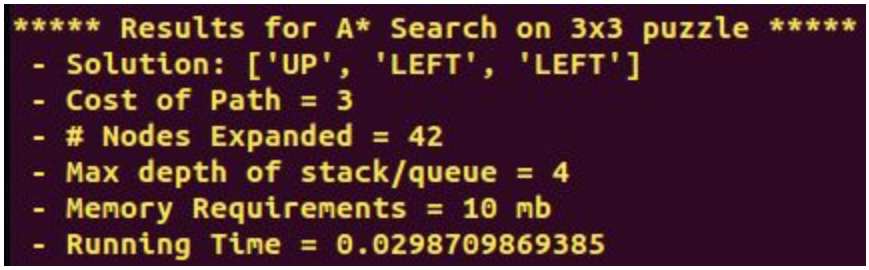
g. Running Time

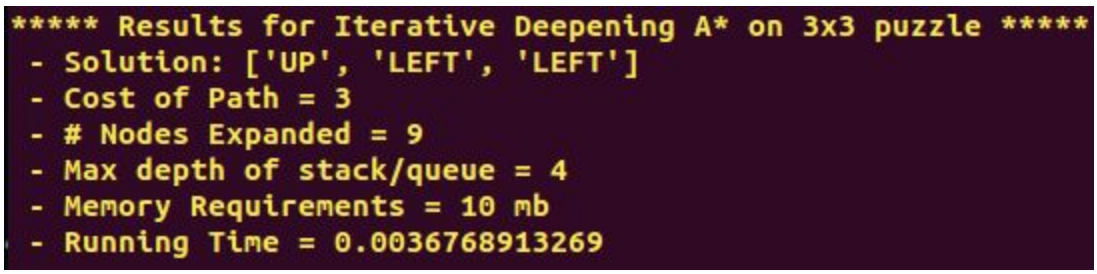
i. BFS: 0.00155591964722 seconds
ii. DFS: 0.000486135482788 seconds
iii. A*: 0.0298709869385 seconds
iv. Iterative Deepening A*: 0.0036768913269 seconds

Here are screenshots of the results for each method:









5. Choice of Heuristic

The Manhattan Distance Heuristic is justified for this puzzle as it finds the distance of each tile from its current location on the board to its goal state position. This distance is summed for each tile in the board and thus the total cost of all of the nodes from their current position to their goal state position serves as a good estimate of “how far” the current state of the board is from the goal state.

6. Knowledge Representation

My node class contains the following components:

- n: the dimension of the board
- parent: the node from which this node was generated
- depth: the depth of the node in the search tree
- heuristic_function_value: h(n) for this configuration (only used by A* and Iterative Deepening A*)
- state: a 2D list which holds the configuration of the board at this node in the tree generated by the search algorithms

Given a starting node, a search agent can perform operations such as checking if the node’s current configuration is a goal configuration, where the class method iterates over its configuration (2D list) and looks for a difference between it and the goal state.

The node class also contains methods for generating successive nodes with swapped blank tiles, by performing the following sequence:

- generate_successors(self) → the node uses its indices_of(self) method to figure out where the blank tile is, and then given these coordinates, identifies legal moves
- generate_copy(self) → next, the node makes as many copies of itself as there are legal moves
- swap_tiles(self, a,b,c,d) → next, the successive node has its tiles swapped in accordance with its legal action, UP, DOWN, LEFT, RIGHT such that a legal successive node with correct configuration is complete

These methods allow the search agent to generate successively deepening nodes in the tree, and with each visited node, check if it is the goal state. At this point, because of parent pointers and depth counter, the search agent has the information necessary to trace the solution back to the root and know the depth at which the solution was found.

Extra Notes

I mentioned this to Antonio in office hours, but some configurations of the board bog down the program to the point where I stopped it from running before finishing because of the time. I couldn't find a pattern among the boards it has a problem with, but included a board - hard_3x3 which BFS and ID_A* weren't able to solve in a reasonable amount of time. Further, some configurations of boards (even medium difficulty ones) gave problems to my DFS.

Thank you for all of the help on Piazza and in OH! Happy grading!