Rene Sanchez // A11866286
Chris Weller // A10031853
Divyansh H Vaishnav //

CSE 150

Assignment 1, Problem 6

Description of the problem and the algorithms used to solve problems 2 - 5.

The problem was solving an mxn sized puzzle using a variety of different search algorithms, where the goal was always to reach a “solved” state defined as having the squares of the puzzle in increasing order.
Algorithms used:
p2) Bread-First Search
p3) (non-recursive) depth-limited search traversal algorithm
p4) iterative deepening depth-first traversal
p5) A* search with a Manhattan-distance heuristic

Data structure used in each algorithm

p2) Bread-First Search: A **queue** that contained moves that created a never before seen board.
p3) (non-recursive) depth-limited search: A **stack** that contained moves that created a never before seen board
p4) iterative deepening depth-first traversal: Same as p3 with an additional for loop.
p5) A* search with a Manhattan-distance heuristic: I used a **Priority queue** that held duples with the form (priority, moves). Where priority = (length of moves) + (number of squares that are not in their “solved” position in the board generated from said moves)

Some analysis on the efficacy of the different algorithms with different puzzles. For instance, you can vary the sizes of the puzzles or the number of random moves used to generate the puzzle. You can then measure the number of nodes visited / maximum size of the queue / time it took to run the code, etc., for each cases. How do they compare against what you would expect from the big-O analysis?

The following stats were obtained from using the “time” linux terminal command to calculate the time efficiency of each algorithm.

We are using “gutty” python module in conjunction with the h.heap() method to look at the memory efficiency of each algorithm.

We are using simple print statements to determine the number of nodes visited and the max size of the queue for each algorithm.

The boards that we are performing tests on are the following:

NOTE: All boards listed here are solvable in 5 moves in less, this is so that we could check the performance of Depth Limited Search with a limit of 5 to the other algorithms.

test 1:
1,4,2
3,0,5

test 2:
0,2,1
3,4,5

test 3:
6,0,2,3,4
1,5,7,8,9
10,11,12,13,14
15,16,17,18,19
20,21,22,23,24

test 4:
1,4,2
3,5,8
6,0,7

test 5:
3,1,2
6,8,0
4,7,5

test 6:
5,1,2,3,4
10,6,7,8,9,
11,16,12,13,14

15,17,0,18,19
20,21,22,23,24

test 7:
1,2,5
3,4,8
6,7,11
9,10,0
12,13,14
15,16,17
18,19,20
21,22,23
24,25,26
Result table for running time (seconds)

	Test 1	test 2	Test 3	Test 4	Test 5	Test 6	Test 7
BFS	0.211	0.285	0.237	0.227	19.175	0.245	0.226
Limited Depth Search	0.258	0.235	0.257	0.235	0.235	0.245	0.246
Iterative DPS	0.244	0.269	0.267	0.255	0.457	0.254	0.261
A*	0.343	0.415	0.372	0.371	0.451	0.372	0.398

Result table for memory usage

	Test 1	test 2	Test 3	Test 4	Test 5	Test 6	Test 7
BFS	3.9 MB	4.07 MB	4.01 MB	3.95 MB	10.69 MB	4.07 MB	4.03 MB
Limited Depth Search	4.26 MB	4.25 MB	4.26 MB	4.26 MB	4.25 MB	4.26 MB	4.26 MB
Iterative DPS	4.26 MB	4.25 MB	4.26 MB	4.26 MB	4.26 MB	4.26 MB	4.26 MB
A*	5.80 MB	5.96 MB	5.80 MB	5.80 MB	6.09 MB	5.81 MB	5.82 MB

Result table for queue insertions

	Test 1	test 2	Test 3	Test 4	Test 5	Test 6	Test 7
BFS	4	360	114	55	11762	184	73
Limited Depth Search	12	17	30	10	33	50	72
Iterative DPS	4	133	30	10	969	50	72
A*	4	360	10	11	483	13	15

Result table for maximum queue size

	Test 1	test 2	Test 3	Test 4	Test 5	Test 6	Test 7
BFS	3	95	68	25	5020	106	43
Limited Depth Search	4	3	7	6	6	7	7
Iterative DLS	3	5	7	6	11	7	7
A*	3	94	6	6	202	9	8

Result Analysis and discussion

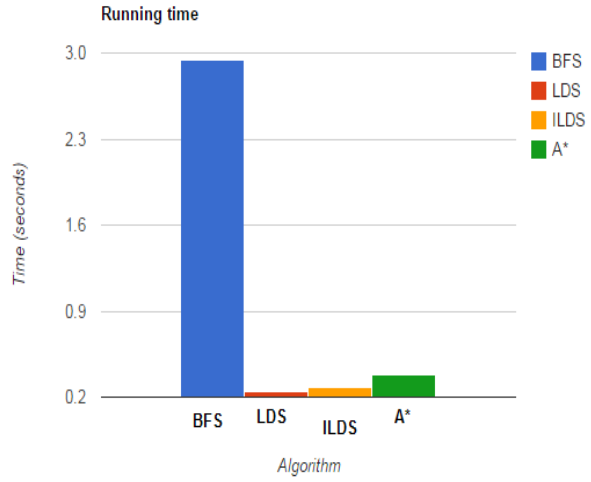
1. It appears that A* is a worse algorithm overall (speed and memory usage) if the board is quite small and the solution is only a few moves long. We believe this is because the calculation of the heuristic function only really pays off if the board is decently large and the solution is longer than a few moves.
And this correlates with the fewer b* branches that are generated from using A* search.

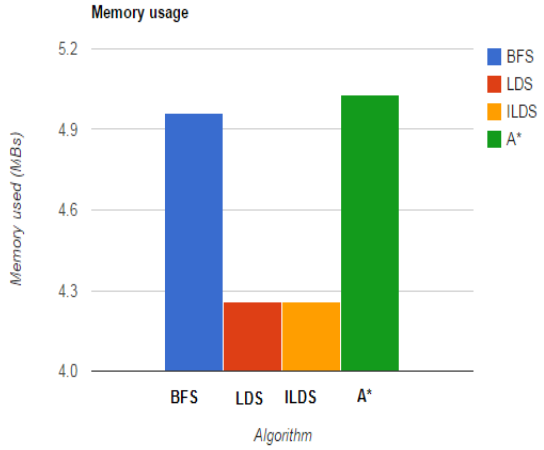
2. Limited Depth Search and Iterative DLS both have nearly identical results. This is expected since they are Iterative DLS is a small modification.

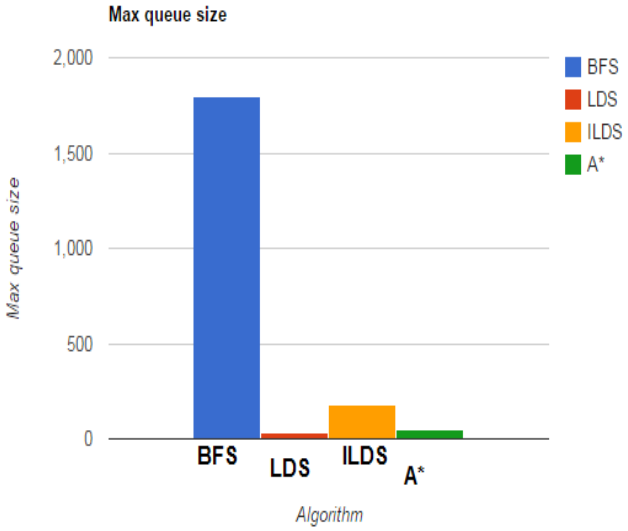
3. The is an interesting pattern when comparing the memory usage of all 4 algorithms. In terms of overall efficiency: DLS = IDLS > A* > BFS, which is consistent to the big O worst case for memory in the book.

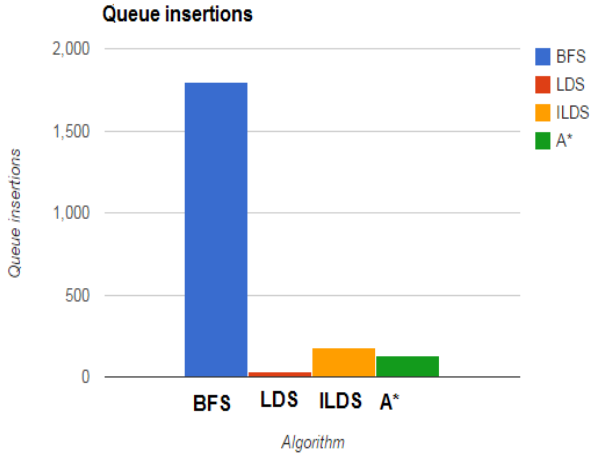
4. It's interesting how DLS will sometimes beat IDLS in running time and vice versa depending on the test. This has to do with the solution being shorter than 5 moves long, in which IDLS should find it faster than DLS.

The following graphs represent the average values for each algorithm:
Note: Please keep in mind that we conducted tests with solutions of 5 moves or less, for larger solutions A* would dominate the other algorithms.









Author contributions

Rene: Algorithm design, coding, comments, report writing, optimization.

I learned how to program in Python 2.7, what a heuristic function is and all about the various fields of machine learning. This assignment exposed me to a different style of programming that I wouldn't have encountered otherwise, and made me privy to some very odd features of python optimization.

Chris: Debugging, analyzing, test runner, test design, optimization.

Besides the standard machine learning information and various algorithm backgrounds and design, I learned a lot about Python profiling and debugging. This assignment required far more rigorous testing routines than I've ever encountered in my own forays into the world of Python, but at the end of the day I feel more prepared to perform this level of stringent examination on my own personal projects.

Divyansh: Debugging, report writing, code reviewer, optimization.

I learned a lot about Python, machine learning, and debugging across different platforms. Code review was a new process for me, and it was very exciting to see other people's thought processes on similar lines of code. Additionally, CS write-ups have never been all that enticing to me, however I had the time of my life writing this report. 10/10, would report again.