Özyeğin University
CS551 – Introduction to Artificial Intelligence
Assignment-1

Solving N-Puzzle Problem with Search Algorithms

Furkan Cantürk

03.30.2021

1. PROBLEM

n-puzzle game consists of a $n \times n$ board holding $n^2 - 1$ tiles numbered from 1 to $n^2 - 1$ and one blank space which is represented as 0 in this assignment. Given an initial state and goal state of the board, the combinatorial search problem is to find a sequence of moves that transitions this state to the goal state. Actions that can be taken to solve the problem is swapping the blank space in one of the four directions $\{\text{Up}, \text{Down}, \text{Left}, \text{Right}\}$. Moving from a state to its one of successor state has a unit cost. Thus, the total cost of path from the initial state to the goal state is the number of moves.

2. ALGORITHMS

2.1. Breadth-First Search (BFS)

Breadth-first search always expands the shallowest unexpanded node. a First-in First-out (FIFO) queue is used for the frontier. The newly generated nodes always go to the back of the queue, while the older nodes get expanded first. Goal test is applied for each successor node before inserting into the frontier.

BFS is complete. BFS is optimal if the path cost is a non-decreasing function of depth.

Its both time and space complexity are $O(b^d)$ where b is the branching factor and d is tree depth which goal state is reached at.

2.2. Uniform Cost Search (UCS)

UCS expands the node with least path cost. A Priority Queue based on cost is used for the frontier. Goal test on the node is performed when it is selected for expansion. This is because the first node generated could be a sub-optimal path.

Completeness is guaranteed only if cost of every step is positive.

Uniform-cost search is optimal. At every step the path with the least cost is chosen, and paths never gets shorter as nodes are added, ensuring that the search expands nodes in the order of their optimal path cost.

Its both time and space complexity are $O(b^{1+\frac{C^*}{e}})$ where $C^*$is optimal path cost and e is least cost at a step.

2.3. Depth-First Search (DFS)

Depth -first search always expands the deepest unexpanded node. a Last-in First-out (LIFO) queue (i.e. a stack) is used for the frontier. Goal test is applied for each successor node before inserting into the frontier.

DFS is not complete and non-optimal in nature.

Its both time and space complexity are $O(bm)$ where m is the maximum depth of the search tree.

2.4. A* Search

A* search is an informed search algorithm. When a search algorithm is guided by an information obtained during search, which is generally heuristic, it becomes an informed search. It combines behaves of Dijkstra's algorithm prioritizing states that are close to the starting point and Best-First-Search prioritizing states that are closer to the goal.
Let g(n) represent the cost of the path from the initial state to state n, and h(n) represent an admissible heuristic estimated cost of the path from n to the goal. A heuristic information is admissible if it does not overestimate cost to reach goal. A* builds a priority queue of nodes ordered on f(n) = g(n) + h(n) which is the total estimated cost through n. Goal test is applied for a node when it is expanded. In this assignment, Manhattan Distance is used as the heuristic for A*.

A* search has a time complexity of $O(b^d)$ and keeps all generated nodes in memory.

3. RESULTS

In this assignment, the algorithms benchmarked for 8-puzzle problem. I created 50 random initial states and got goal states with some random number (between 10 and 50) of random moves starting from given initial states. All tests are done with Python 3.8.3. The following table presents number of expanded nodes, reached tree depth (number of moves from initial state to goal state), and solve time in milliseconds by each algorithm for each problem instance. The last row of the table is the average values of each column.

DFS can find a solution within 1181 ms at average but its solution cost is too away from optimal cost. Since it is not guided based on any cost information, it follows a random path from initial state to goal state.

A* can find the optimal solution in 2 ms at average and it outperforms BFS and UCS by a substantial difference. A* proceeds an informed search using a heuristic so that it does not make exhaustive search and goes on one direction during the search until f value of current path increases. In the comparison of BFS and UCS, BFS expands less nodes and so can find the solution earlier than UCS since BFS applies the goal test before inserting nodes into the frontier.

Instance	BFS			DFS			UCS			A*		
No	depth	expanded	time	depth	expanded	time	depth	expanded	time	depth	expanded	time
1	13	1618	38	5285	5355	187	13	2742	94	13	39	2
2	3	8	0	11	11	0	3	15	16	3	4	1
3	4	8	16	1304	1317	31	4	16	0	4	5	0
4	6	37	0	35534	36420	1507	6	88	0	6	7	0
5	14	2431	69	77164	83223	3252	14	5198	206	14	19	1
6	5	16	16	1329	1343	43	5	41	2	5	6	0
7	7	81	1	90865	103771	4061	7	118	4	7	10	0
8	20	47120	3157	64562	67742	2675	20	99878	5246	20	726	40
9	15	5282	201	24801	25254	948	15	10208	469	15	61	3
10	6	45	0	1328	1343	41	6	84	3	6	9	1
11	14	4070	179	38198	39114	1580	14	6083	300	14	64	3
12	3	6	1	1317	1330	41	3	11	1	3	4	0
13	10	428	14	1302	1316	41	10	665	26	10	12	1
14	4	19	1	14	14	0	4	23	1	4	5	0
15	17	11460	517	50209	51955	2041	17	22048	1026	17	19	1
16	2	2	0	1314	1327	43	2	13	0	2	3	1
17	7	65	2	18783	19081	728	7	145	6	7	8	0
18	5	25	0	1327	1342	63	5	42	2	5	6	1
19	7	76	0	18813	19111	843	7	146	5	7	8	0
20	4	10	0	8	8	0	4	29	1	4	5	0
21	6	55	0	41026	42149	1638	6	90	4	6	7	0
22	13	1856	62	90511	102640	3994	13	3152	165	13	76	4
23	9	223	31	1407	1422	41	9	429	18	9	10	1
24	6	58	0	93316	108850	4314	6	63	3	6	9	0
25	5	21	0	47561	49049	1856	5	43	1	5	6	1
26	3	7	8	57179	59618	2364	3	15	1	3	4	0
27	8	134	0	18886	19189	684	8	267	9	8	10	1
28	3	9	0	3	3	0	3	19	1	3	4	0
29	4	15	0	4	4	0	4	26	1	4	5	0
30	16	6460	226	74934	80173	3176	16	9524	427	16	144	7
31	12	858	47	11848	12006	421	12	1866	72	12	22	1
32	5	26	0	91	91	10	5	68	3	5	7	1
33	13	1522	63	22393	22820	895	13	2122	83	13	21	1
34	12	1027	25	4014	4066	122	12	1959	104	12	21	1
35	5	23	0	1307	1320	73	5	42	2	5	6	0
36	21	57587	4173	41561	42742	1707	21	112036	6296	21	360	18
37	2	4	0	2	2	2	2	8	0	2	3	0
38	6	54	0	34860	35666	1478	6	101	4	6	7	1
39	12	1354	34	1336	1353	47	12	2166	95	12	36	2
40	11	742	26	5011	5072	155	11	964	41	11	22	1
41	11	495	16	45721	47135	1919	11	1191	50	11	31	2
42	10	404	16	20738	21115	782	10	731	28	10	15	1
43	10	516	16	22686	23089	988	10	817	34	10	29	1
44	11	580	12	30395	31004	1172	11	1296	55	11	29	2
45	9	259	16	80063	87146	3379	9	416	15	9	16	1
46	12	928	16	14966	15179	536	12	1937	105	12	25	1
47	13	1751	62	91531	105032	4067	13	2964	119	13	46	2
48	6	37	7	57162	59601	2220	6	63	3	6	7	0
49	6	54	0	64568	67797	2861	6	77	3	6	7	0
50	9	248	0	1307	1321	47	9	420	15	9	10	1
Avg:	**8.7**	**3002.3**	**181**	**28197**	**30140.6**	**1181**	**8.7**	**5849.3**	**303**	**8.7**	**40.3**	**2**

Table 1. Expanded nodes, search depth, and solution time in ms by each search algorithm