CS561 - ARTIFICIAL INTELLIGENCE LAB

ASSIGNMENT-1: A* Search, BFS

Group Name: **1801cs31_1801cs32_1801cs33**

Students:

Names	Roll No.	Batch
M **Maheeth Reddy**	1801CS31	B.Tech.
M **Nitesh Reddy**	1801CS32	B.Tech.
Nischal A	1801CS33	B.Tech.

Answer 3:

Results Comparison between A* and Best First Search for given Heuristics

Test Case 1:

Algorithm			h1 = number of tiles displaced from their destined position (Hamming Distance)		h2 = sum of Manhattan distance of each tile from the goal position	
	Start State	Goal State	Optimal Path Cost	Optimal Path	Number of States to Optimal Path	Optimal Path
A*	[2, 8, 1, 4, 6, 3, 7, 5, 0]	[1, 2, 3, 8, 0, 4, 7, 6, 5]	12	[[2, 8, 1, 4, 6, 3, 7, 0, 5], [2, 8, 1, 4, 0, 3, 7, 6, 5], [2, 8, 1, 0, 4, 3, 7, 6, 5], [0, 8, 1, 2, 4, 3, 7, 6, 5], [8, 0, 1, 2, 4, 3, 7, 6, 5], [8, 1, 0, 2, 4, 3, 7, 6, 5], [8, 1, 3, 2, 4, 0, 7, 6, 5], [8, 1, 3, 2, 0, 4, 7, 6, 5], [8, 1, 3, 0, 2, 4, 7, 6, 5], [0, 1, 3, 8, 2, 4, 7, 6, 5], [1, 0, 3, 8, 2, 4, 7, 6, 5], [1, 2, 3, 8, 0, 4, 7, 6, 5]]	12	[[2, 8, 1, 4, 6, 3, 7, 0, 5], [2, 8, 1, 4, 0, 3, 7, 6, 5], [2, 8, 1, 0, 4, 3, 7, 6, 5], [0, 8, 1, 2, 4, 3, 7, 6, 5], [8, 0, 1, 2, 4, 3, 7, 6, 5], [8, 1, 0, 2, 4, 3, 7, 6, 5], [8, 1, 3, 2, 4, 0, 7, 6, 5], [8, 1, 3, 2, 0, 4, 7, 6, 5], [8, 1, 3, 0, 2, 4, 7, 6, 5], [0, 1, 3, 8, 2, 4, 7, 6, 5], [1, 0, 3, 8, 2, 4, 7, 6, 5], [1, 2, 3, 8, 0, 4, 7, 6, 5]]
BFS			36	[[2, 8, 1, 4, 6, 3, 7, 0, 5], [2, 8, 1, 4, 0, 3, 7, 6, 5], [2, 8, 1, 0, 4, 3, 7, 6, 5], [0, 8, 1, 2, 4, 3, 7, 6, 5], [8, 0, 1, 2, 4, 3, 7, 6, 5], [8, 4, 1, 2, 0, 3, 7, 6, 5], [8, 4, 1, 0, 2, 3, 7, 6, 5], [0, 4, 1, 8, 2, 3, 7, 6, 5], [4, 0, 1, 8, 2, 3, 7, 6, 5], [4, 1, 0, 8, 2, 3, 7, 6, 5], [4, 1, 3, 8, 2, 0, 7, 6, 5], [4, 1, 3, 8, 0, 2, 7, 6, 5], [4, 1, 3, 0, 8, 2, 7, 6, 5], [0, 1, 3, 4, 8, 2, 7, 6, 5], [1, 0, 3, 4, 8, 2, 7, 6, 5], [1, 3, 0, 4, 8, 2, 7, 6, 5], [1, 3, 2, 4, 8, 0, 7, 6, 5], [1, 3, 2, 4, 0, 8, 7, 6, 5], [1, 0, 2, 4, 3, 8, 7, 6, 5], [1, 2, 0, 4, 3, 8, 7, 6, 5], [1, 2, 8, 4, 3, 0, 7, 6, 5], [1, 2, 8, 4, 0, 3, 7, 6, 5], [1, 2, 8, 0, 4, 3, 7, 6, 5], [0, 2, 8, 1, 4, 3, 7, 6, 5], [2, 0, 8, 1, 4, 3, 7, 6, 5], [2, 8, 0, 1, 4, 3, 7, 6, 5], [2, 8, 3, 1, 4, 0, 7, 6, 5], [2, 8, 3, 1, 0, 4, 7, 6, 5], [2, 8, 3, 0, 1, 4, 7, 6, 5], [0, 8, 3, 2, 1, 4, 7, 6, 5], [8, 0, 3, 2, 1, 4, 7, 6, 5], [8, 1, 3, 2, 0, 4, 7, 6, 5], [8, 1, 3, 0, 2, 4, 7, 6, 5], [0, 1, 3, 8, 2, 4, 7, 6, 5], [1, 0, 3, 8, 2, 4, 7, 6, 5], [1, 2, 3, 8, 0, 4, 7, 6, 5]]	34	[[2, 8, 1, 4, 6, 3, 7, 0, 5], [2, 8, 1, 4, 0, 3, 7, 6, 5], [2, 0, 1, 4, 8, 3, 7, 6, 5], [2, 1, 0, 4, 8, 3, 7, 6, 5], [2, 1, 3, 4, 8, 0, 7, 6, 5], [2, 1, 3, 4, 0, 8, 7, 6, 5], [2, 1, 3, 0, 4, 8, 7, 6, 5], [0, 1, 3, 2, 4, 8, 7, 6, 5], [1, 0, 3, 2, 4, 8, 7, 6, 5], [1, 4, 3, 2, 0, 8, 7, 6, 5], [1, 4, 3, 2, 8, 0, 7, 6, 5], [1, 4, 0, 2, 8, 3, 7, 6, 5], [1, 0, 4, 2, 8, 3, 7, 6, 5], [0, 1, 4, 2, 8, 3, 7, 6, 5], [2, 1, 4, 0, 8, 3, 7, 6, 5], [2, 1, 4, 8, 0, 3, 7, 6, 5], [2, 0, 4, 8, 1, 3, 7, 6, 5], [2, 4, 0, 8, 1, 3, 7, 6, 5], [2, 4, 3, 8, 1, 0, 7, 6, 5], [2, 4, 3, 8, 0, 1, 7, 6, 5], [2, 0, 3, 8, 4, 1, 7, 6, 5], [0, 2, 3, 8, 4, 1, 7, 6, 5], [8, 2, 3, 0, 4, 1, 7, 6, 5], [8, 2, 3, 4, 0, 1, 7, 6, 5], [8, 2, 3, 4, 1, 0, 7, 6, 5], [8, 2, 0, 4, 1, 3, 7, 6, 5], [8, 0, 2, 4, 1, 3, 7, 6, 5], [8, 1, 2, 4, 0, 3, 7, 6, 5], [8, 1, 2, 0, 4, 3, 7, 6, 5], [0, 1, 2, 8, 4, 3, 7, 6, 5], [1, 0, 2, 8, 4, 3, 7, 6, 5], [1, 2, 0, 8, 4, 3, 7, 6, 5], [1, 2, 3, 8, 4, 0, 7, 6, 5], [1, 2, 3, 8, 0, 4, 7, 6, 5]]

Test Case 2:

Algorithm			h1 = number of tiles displaced from their destined position (Hamming Distance)		h2 = sum of Manhattan distance of each tile from the goal position	
	Start State	Goal State	Optimal Path Cost	Optimal Path	Number of States to Optimal Path	Optimal Path
A*	[1, 2, 3, 0, 4, 5, 6, 7, 8]	[1, 2, 3, 4, 5, 8, 0, 6, 7]	5	[[1, 2, 3, 4, 0, 5, 6, 7, 8], [1, 2, 3, 4, 5, 0, 6, 7, 8], [1, 2, 3, 4, 5, 8, 6, 7, 0], [1, 2, 3, 4, 5, 8, 6, 0, 7], [1, 2, 3, 4, 5, 8, 0, 6, 7]]	5	[[1, 2, 3, 4, 0, 5, 6, 7, 8], [1, 2, 3, 4, 5, 0, 6, 7, 8], [1, 2, 3, 4, 5, 8, 6, 7, 0], [1, 2, 3, 4, 5, 8, 6, 0, 7], [1, 2, 3, 4, 5, 8, 0, 6, 7]]
BFS			5	[[1, 2, 3, 4, 0, 5, 6, 7, 8], [1, 2, 3, 4, 5, 0, 6, 7, 8], [1, 2, 3, 4, 5, 8, 6, 7, 0], [1, 2, 3, 4, 5, 8, 6, 0, 7], [1, 2, 3, 4, 5, 8, 0, 6, 7]]	5	[[1, 2, 3, 4, 0, 5, 6, 7, 8], [1, 2, 3, 4, 5, 0, 6, 7, 8], [1, 2, 3, 4, 5, 8, 6, 7, 0], [1, 2, 3, 4, 5, 8, 6, 0, 7], [1, 2, 3, 4, 5, 8, 0, 6, 7]]

Why one search technique is better than the other one?

Best First Search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is

$$f(n) = h(n)$$

A* search is a special case of the Best First Search algorithm which evaluates nodes by combining the cost g(n) to reach a node **n** and the cost h(n) to get from the node **n** to the goal.

$$f(n) = g(n) + h(n)$$

The evaluation function f(n), is the estimated cost of the cheapest solution through node **n**.

Clearly, the A* Search Algorithm differs from Best First Search in its evaluation function. The BFS algorithm first expands the node whose estimated distance to the goal is the smallest. So, only the nodes in the fringe are kept in memory for making the next decision and nodes that have already been expanded are discarded. But, A* Search keeps all the nodes in memory, not just the ones in the fringe. This helps it in making more informed suggestions.

So, this is the reason why the A* Search Algorithm is optimal and Best First Search isn't. A* Search has a lesser branching factor than Best First Search and doesn't spend much time in evaluating the undesired paths.

Time Complexity of Best First Search is $O(b^m)$, where b is the branching factor and m is the maximum depth of the search tree.
Time Complexity of A* Search Algorithm is exponential with the path length. The worst case complexity is $O(b^d)$ where b is the branching factor and d is the depth of the shortest path.

Space Complexity of Best First Search is $O(b^m)$, where b is the branching factor and m is the maximum depth of the search tree.
Space Complexity of A* Search Algorithm in the worst case is $O(b^d)$ where b is the branching factor and d is the depth of the shortest path.

Comparison between the results of BFS and A* Search Algorithms

In the context of the results obtained by us, we can see that A* algorithm performs better than Best First Search with both heuristics for the first input-output pair. We can see that the difference between the two algorithms occurs when the search reaches the node with value [8, 0, 1, 2, 4, 3, 7, 6, 5]. At this point Best First search chooses [8, 4, 1, 2, 0, 3, 7, 6, 5] as the value of h(n) is least for this. However A* algorithm choses [8, 1, 0, 2, 4, 3, 7, 6, 5] as the next state since the overall value of f(n) = g(n) + h(n) is lesser and accounts for the least cost for reaching this state from the start state. This decision helps A* to perform better than BFS. Hence, A* Search is better in general.

The End