3. Analysis:

1. Length of the solutions across the different algorithms and heuristics.

We created 50 puzzles and divided them into 10 different categories. These different categories were based off of the number of cars on the board including the ambulance inside this number. The numbers vary from 4 cars per board to 13 cars per board. Then we applied the different algorithms with the different heuristics in order to come up with the length of the solution if any, length of the search path and the execution time. These values are inside a table that we generated using our code and the analysis is done according to this.

Length of the solutions across algorithms:

When it comes to the length of the solutions across the different algorithms, we noticed an interesting pattern. No matter what algorithm was used, the one with the lowest length in solution was always the uniform cost search (UCS). This does not mean that none of our other algorithms got the same length for the solution as the UCS, but rather that UCS always guarantees the lowest cost as learned in class. This makes total sense because even though UCS is slow as seen by the higher execution time due to a lot of backtracking and thus higher length for the search path, it still guarantees the lowest cost. The reason for this is because the UCS keeps track of an open list that is a priority queue based off of the costs of each move. It orders them from the lowest to the highest cost and always assures that the lowest cost node is chosen. There is a trade-off with it because this results in having a longer length for the search path due to the frequent backtracking. Furthermore, another observation we did is that as the number of cars increases, so does the length for the solutions. This makes sense because the more cars there are in the puzzle, the higher are the odds of getting cars blocking other cars and therefore more moves required in order to reach the final state of the game. Moreover, when it comes to the same puzzle, the differences between the length of the solutions are fairly similar with the one guarantee that UCS is the shortest. However, we did notice a trend that whenever we did have a difference in the length of the solutions, Greedy Best First Search (GBFS) tended to be the higher one in comparison to UCS and A/A* and this makes sense since as we learned in class, GBFS tends to be quick but might not give the optimal solution meaning the lowest cost. As we can see in the following picture, for puzzle 17, GBFS had in general higher solution length than the others. All this makes sense and goes perfectly with the theory learned in class.

17	UCS	NA	8	1044	2.22
17	GBFS	h1	9	138	0.21
17	GBFS	h2	9	138	0.2
17	GBFS	h3	9	138	0.2
17	GBFS	h4	8	1044	2.28
17	A/A*	h1	8	741	1.58
17	A/A*	h2	8	741	1.59
17	A/A*	h3	9	138	0.24
17	A/A*	h4	8	1044	2.3

Lastly, we did notice that when there was no solution for a puzzle, all the algorithms spotted this, thus confirming that our algorithms are correct as they are getting the same result.

Length of the solutions across heuristics:

When it comes to the length of the solutions across the different heuristics, we noticed that there wasn't much of a difference between them. One key point to note is that depending on how good or bad the chosen heuristic function is, it changes the expected outcome for the length of the solution. This can be demonstrated by our heuristic 4 which we purposely chose one that is not that good in order prove this point. The heuristic 4 that we chose is basically using the number of empty spaces in each state and since this is basically staying constant throughout the states, it is imitating an uninformed search thus giving us similar if not the same value as UCS which itself is an uninformed search algorithm. Here is an image from our data showing how the different heuristics are giving fairly similar values with the exclusion of heuristic 4.

Puzzle Number	Algorithm	Heuristic	Length of the Solution	Length of the Search Path	Execution Time (in seconds)
1	UCS	NA	3	87	0.13
1	GBFS	h1	3	10	0.01
1	GBFS	h2	3	10	0.01
1	GBFS	h3	3	10	0.01
1	GBFS	h4	3	87	0.14
1	A/A*	h1	3	66	0.1
1	A/A*	h2	3	66	0.1
1	A/A*	h3	3	10	0.01
1	A/A*	h4	3	87	0.14

2. The admissibility of each heuristic and its influence on the optimality of the solution.

When talking about heuristics, there are two essential topics we would like to discuss about for each algorithm. These two topics are the admissibility of the heuristic and the informedness of it. These two together are a good way to evaluate the heuristics and determine the optimality of the solution at hand. The admissibility of the heuristic makes sure that at every node, we never overestimate the actual cost of reaching the goal state from that node. Furthermore, the informedness measures the quality of the heuristic by determining the amount of backtracking required. When we are more informed about the heuristic, there is consequently less backtracking and therefore a shorter search path. When it comes to the admissibility of each heuristic for our 50 puzzles, we can see that they are admissible as shown in the image below.

Puzzle Number	Algorithm	Heuristic	Length of the Solution	Length of the Search Path	Execution Time (in seconds)
1	UCS	NA	3	87	0.13
1	GBFS	h1	3	10	0.01
1	GBFS	h2	3	10	0.01
1	GBFS	h3	3	10	0.01
1	GBFS	h4	3	87	0.14
1	A/A*	h1	3	66	0.1
1	A/A*	h2	3	66	0.1
1	A/A*	h3	3	10	0.01
1	A/A*	h4	3	87	0.14
2	UCS	NA	4	63	0.05
2	GBFS	h1	4	14	0.01
2	GBFS	h2	4	14	0.01
2	GBFS	h3	4	14	0.01
2	GBFS	h4	4	63	0.05
2	A/A*	h1	4	57	0.05
2	A/A*	h2	4	57	0.05
2	A/A*	h3	4	14	0.01
2	A/A*	h4	4	63	0.05

Furthermore, in that picture taken from our data, we also see that the heuristics have informedness as seen by the less backtracking identified by the shorter search path when compared to the uninformed search. In addition, we did notice that heuristic 4 although was admissible, didn't pass the informedness criteria for evaluating the heuristics thus being not as optimal as the rest of the heuristics. All this shows the importance of the different evaluation criteria for heuristics, namely admissibility and informedness and how they can influence on the optimality of the solution.

3. The execution time across the different algorithms and heuristics.

Execution time across algorithms:

As we learned in class, if we have a good heuristic function, Greedy Best First search (GBFS) has the potential to find the solution very quickly. This is clearly seen in our generated data from the execution times column for the different algorithms and heuristics. Here is an example from our data, the last column being the execution time, as we can see the GBFS is much quicker than the other algorithms.

4	UCS	NA	3	60	0.07
4	GBFS	h1	3	6	0.01
4	GBFS	h2	3	6	0.01
4	GBFS	h3	3	6	0.01
4	GBFS	h4	3	60	0.07
4	A/A*	h1	3	41	0.04
4	A/A*	h2	3	41	0.04
4	A/A*	h3	3	6	0.01
4	A/A*	h4	3	60	0.07

Furthermore, for that specific puzzle which is puzzle 4 (noted by the first column), we can see that A/A* for this case is slower than GBFS. However, this is not always the case and there are times where A/A* can be as fast as or even faster than GBFS. Moreover, it makes sense that some times, GBFS can be faster than A/A* and this is because as we learned in class, GBFS does not use g(n) whereas A/A* does and computing g(n) can take time. This phenomenon is clearly observed for our case as seen from the higher execution time for A/A*.

No, an informed search is not always faster than an uninformed search and this is seen by our heuristic 4 which was chosen purposely to be a not so good heuristic function to prove some points. One of the main reasons for this can be the different heuristic functions used, along with other reasons such as the size of the state space, on the branching factors and on how deep the solution is. As we can see in the following image from our data, the algorithm of UCS which is an uninformed search is slightly quicker in the execution time than the A/A* algorithm which is an informed search algorithm. Therefore, showing that an informed search is not always necessarily quicker than an uninformed search algorithm.

45	UCS	NA	15	2880	13.93
45	GBFS	h1	16	427	1.78
45	GBFS	h2	16	427	1.71
45	GBFS	h3	16	427	1.72
45	GBFS	h4	15	2880	14.45
45	A/A*	h1	15	2120	9.75
45	A/A*	h2	15	2120	9.66
45	A/A*	h3	16	422	1.88
45	A/A*	h4	15	2880	14.59

Execution time across heuristics:

As we learned in class, depending on the heuristic function chosen, an algorithm can be quicker or slower. This is because it can potentially allow more backtracking and thus making it slower. Furthermore, since h(n) is the estimated cost to reach the goal state from the current node, if this estimation is poorly done, it makes sense that it will make the algorithm slower by causing this increase in backtracking. In order to demonstrate all this, we have purposely chosen a bad heuristic 4. The reason why it is a bad heuristic is due to the fact that since we are basing it off of the number of empty tiles on the field and that is staying relatively constant, it is basically acting like the UCS as seen by the similar execution time for UCS and H4 heuristic. We can see in the following image from our data that for all the algorithms with heuristic 4, the execution time is slower than the rest.

47	UCS	NA	14	28324	458.16
47	GBFS	h1	17	3652	16.28
47	GBFS	h2	17	3652	16.6
47	GBFS	h3	17	3652	16.82
47	GBFS	h4	14	28324	464.02
47	A/A*	h1	14	13801	147.63
47	A/A*	h2	14	13801	143.18
47	A/A*	h3	17	2777	12
47	A/A*	h4	14	28324	504.52

Even though this example is taken from puzzle 47, we did notice this same trend for all 50 puzzles. The trend being that every algorithm using heuristic 4 had significantly higher execution time than the rest of the algorithms with different heuristics. This demonstrates the theory learned in class, where depending on the heuristic chosen, the performance and efficiency can vary.