Depth	**BFS**		**IDS**		**A*: h1**		**A*: h2**		**A*: h3**	
	Avg run time (sec)	**Avg #nodes Explr'**	**Avg run time (sec)**	**Avg #nodes Explr'**	**Avg run time (sec)**	**Avg #nodes Explr'**	**Avg run time (sec)**	**Avg #nodes Explr'**	**Avg run time (sec)**	**Avg #nodes Explr'**
8	**0.00367**	**135**	**0.00590**	**180**	**0.00043**	**14**	**0.00062**	**13**	**0.00096**	**18**
15	**1.58426**	**4265**	**0.39194**	**4614**	**0.02380**	**296**	**0.00731**	**116**	**0.03161**	**316**
24	**500.75**	**107117**	**65.05**	**85216**	**36.75**	**13548**	**0.49424**	**1454**	**31.8**	**11527**

Conclusion

The above shows different algorithms being run on different levels. Among all the algorithms, it is seen the A* with misplaced tile heuristic and A* with Manhattan distance heuristic performs much better than the other algorithms in terms of the run time. The IDS did relatively poorly as it times up most of the time because it is exploring too many nodes, and was going back to its ancestor node from its child node. Because the cycle from the child to its ancestor nodes were checked, this provides a much more efficient IDS algorithm. The BFS performs the poorest among all 5 algorithms as the level increases and also generates the most nodes to explore while solving the puzzle. A* Manhattan distance heuristic performs the best in terms of run time and also nodes generated because they produce the least nodes to reach the goal state, making it the most optimal algorithm to use to solve the puzzle. The modified A* Manhattan distance heuristic, which is h3 did relatively poorer as compared to the original A* Manhattan distance heuristic. As a conclusion, all three A* search performs better than BFS and IDS in terms of run time and nodes explored, suggesting that A* algorithms are more efficient.