Madeline Younes
z5208494

Part 2 – Search

Question 1: Search Algorithms for the 15-Puzzle

a)

	start10	start12	start20	start30	start40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*	29	21	952	17297	112571

b)

Ucsdijkstra:
The uniform cost search dijkstra is simplest search algorithm which from the starting state will choose the next least costly state. The time and memory space complexity is dependant on the cost of the optimal solution and the cost of each step. . With the time complexity in the worst case bring $O\left(b^{\left[\frac{C*}{\varepsilon}\right]}\right)$, where C* = cost of the optimal solution every transition costs at least ε, thereby the cost per step is $\frac{C*}{\varepsilon}$. As well as the space complexity being $O\left(b^{\left[\frac{C*}{\varepsilon}\right]}\right)$, the space complexity reduces to a breadth first search so, $b^{\left[\frac{C*}{\varepsilon}\right]} = b^d$ if all states have the same cost. As confirmed by the results the UCS search is not very space efficient running out of memory from start12 to start40.

Ideepsearch:
The iterative deep search combines benefits of depth first search's space efficiency and breath first search time efficiency. The iterative deep search starts from various initial values calling depth first search with a restricted depth. The time and memory space complexity is dependant on b = branching factor, d = depth of the shallowest solution. With the worst case time complexity of $O(b^d)$ and memory space complexity is $O(bd)$. As shown by the results IDS is not very time efficient running overtime for start30 and start40.

Astar:
The A* search algorithm utilises the path function and the heuristic function to determine the search path. As seen in the results, the A* search is more efficient than UCS and IDS although the memory storage is limit is exceeding using start30 and start40. The A*Search stores all the nodes in the memory making the space complexity equivalent to the amount of nodes which in the worst case is $O(bd)$. Whilst the time complexity is the exponential in relative error in *heuristic h* x length of solution, so in the worst case is $O(bd)$.

Idastar:
IDA* is the most time and space efficient, essentially it is a lower memory variant of A*star, that stops its depth-first search once the current threshold is exceeded, the threshold is increased with each search. Thereby, has the same big O time complexity and space complexity as A*. Which in the worst case the time complexity is $O(b^d)$

and the space memory complexity is $O(bd)$. Looking at the results it is able to run at all start values, although at start20 it reaches the solution using more nodes than A*, thereby showing that the efficiency conclusions made are a trend but not necessarily true in all scenarios.

Question 2: Heuristic Path Search for 15-Puzzle

a)

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2 1.4 1.6	52 66 100	191438 116174 34647	62 82 148	230861 3673 55626	66 94 162	431033 188917 235852
Greedy	164	5447	166	1617	184	2174

b) Added to line 43

```
W is 1.2,
F1 is (2-W)*G1 + W*H1,
```

c) The five heuristic algorithms vary in the weightings of the heuristic function and cost of path function's contribution to the solutions. The IDA* relies equally on the heuristic and the cost of path function, whilst the greedy uses only the heuristic function. Observing the results table the IDA* has greatest quality solution but takes the longest time to run. Whilst, the Greedy algorithm takes the shortest amount of time but produces the lowest quality of solution. Increasing the value of w made the search algorithm more dependant on the heuristic function and thereby more similar to the greedy search algorithm. As the value for w was increased time taken to generate a solution increased but the quality of the solution decreased. All the heuristic functions have the same worst case scenario space complexity of is $O(b^d)$ and time complexity of is $O(bd)$.

$$f(n) = (2 - w)g(n) + wh(n)$$