Part 2 – Search

Question 1: Search Algorithms for the 15-Puzzle

(a)

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

(b) The uniformed-cost search with Dijkstra's algorithm (UCS) is the least memory and time efficient as it has space complexity big O notation of $O(b^{\frac{C*}{\varepsilon}})$, which resulted in this algorithm to run out of memory as increasing G exponentially increase memory usage. Given enough memory the algorithm would also run out of time as the time complexity of the algorithm is also given as $O(b^{\frac{C*}{\varepsilon}})$. The trade-off of this is that this algorithm is both complete and optimal.

The Iterative Deepening Search (IDS) tries to combine depth first and breadth first advantage of low memory and optimal/complete. It has a time complexity $O(b^d)$ and a space complexity of $O(bd)$ this is shown in the table as there were enough memory but at a larger G the amount of time needed increase exponentially. This algorithm is also complete given the branches are finite as well as optimal if the cost of the path are identical. Which means that this algorithm is better than UCS given a few conditions such as equal cost path and finite branches as it can search with less memory usage than UCS.

The A* search using the Manhattan Distance heuristic (A*) is an informed search algorithm which has a time complexity of $O(b^d)$ and a space complexity of $O(b^d)$ from maintaining a priority queue which resulted in the program exceeding the memory limit. This algorithm uses UCS minimization of cost as well as minimizing the estimate to the goal using the greedy search method.

The Iterative Deepening A* Search (IDA*) is an informed search algorithm which has an exponential time complexity $O(b^d)$ as well as a linear space complexity $O(bd)$ compared to the A* search. This algorithm is the most efficient as it stops when it reaches its current threshold from the formula $f(n) = g(n) + h(n)$ as well as combining the A* depth first search.

Since both UCS and IDS are uninformed search methods compared to the A* and IDA* informed search algorithms they take more time to complete as well as more memory usage but are complete and optimal with less conditions. The informed search algorithm are given heuristics and are able to use the information to search with better efficiency compared to uninformed search.

Question 2: Heuristic Path Search for 15-Puzzle

(a) & (c)

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2	52	191438	62	230861	66	431033
1.4	66	116342	82	4432	94	190278
1.6	100	33504	148	55626	162	235848
Greedy	164	5447	166	1617	184	2174

(b)

```
31  % Otherwise, use Prolog backtracking to explore all successors
32  % of the current node, in the order returned by s.
33  % Keep searching until goal is found, or F_limit is exceeded.
34  depthlim(Path, Node, G, F_limit, Sol, G2)  :-
35      nb_getval(counter, N),
36      N1 is N + 1,
37      nb_setval(counter, N1),
38      % write(Node),nl,    % print nodes as they are expanded
39      s(Node, Node1, C),
40      not(member(Node1, Path)),      % Prevent a cycle
41      G1 is G + C,
42      h(Node1, H1),
43      F1 is G1 + H1,
44      F1 =< F_limit,
45      depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
46
```

```
31  % Otherwise, use Prolog backtracking to explore all successors
32  % of the current node, in the order returned by s.
33  % Keep searching until goal is found, or F_limit is exceeded.
34  depthlim(Path, Node, G, F_limit, Sol, G2)  :-
35      nb_getval(counter, N),
36      N1 is N + 1,
37      nb_setval(counter, N1),
38      % write(Node),nl,    % print nodes as they are expanded
39      s(Node, Node1, C),
40      not(member(Node1, Path)),      % Prevent a cycle
41      G1 is G + C,
42      h(Node1, H1),
43      F1 is (2-w)*G1 + (w*H1),      % Change here using heuristic path algorithm
44      F1 =< F_limit,
45      depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

In the second picture I changed line 43 in order to match the heuristic path algorithm:

$$f(n) = (2 - w) * g(n) + w * h(n), where\ 0 \leq w \leq 2.$$

Then I would sub in $w = 1.2, 1.4\ and\ 1.6$ to change the weighting of the algorithm.

(d) As shown in the table as we increase the weight of the formula 'w' by an increment of 0.2 there is an increase in speed but also an increase in path length which is a decrease in quality. Since when 'w' approaches to two we get $f(n) = (2 - 2) * g(n) + 2 * h(n) = 2 * h(n)$ which is the same as a greedy algorithm. While if 'w' approaches to one we get $f(n) = (2 - 1) * g(n) + 1 * h(n) = g(n) + h(n)$ which is the same as an Iterative Deepening A* search.