ASSIGNMENT 2

COMP9414
ARTIFICIAL INTELLIGENCE
SEMESTER 1,2018

STUDENT ID:z5179974(z5179974@student.unsw.edu.au)

STUDENT NAME: RAGAVENDRAN LAKSHMINARASIMHAN

Question 1: Search Algorithms for the 15-Puzzle

(a)

	Start10	Start12	Start20	Start30	Start40
Uniform Cost Search	2565	Mem	Mem	Mem	Mem
Iterative Deepening Search	2407	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
Iterative Deepening A*	29	21	952	17297	112571

(b)

Criteria	Uniform Cost	Iterative Deeping	A*	Iterative Deepening A*
Time	$O(b^{\lceil C*/\varepsilon \rceil})$	$O(b^d)$	$O(b^d)$	$O(b^d)$
Space	$O(b^{\lceil C*/\varepsilon \rceil})$	$O(bd)$	$O(b^d)$	O(d)
Complete	YES[2]	YES[1]	YES	YES
Optimal	YES	YES[3]	YES	YES

b = branching factor, d = depth of the shallowest solution,
m = maximum depth of the search tree, k = depth limit.
1 = complete if b is finite.
2 = complete if b is finite and step costs $\geq$ e with e > 0.
3 = optimal if actions all have the same cost.

- The Uniform Cost Search is the least memorial efficient since its space complexity is $O(b^{\lceil C*/\varepsilon \rceil})$ which grows exponentially.
- The Iterative Deepening Search's space complexity is only $O(b$d) that makes it the most memorial efficient. However, it is time consuming and even sometimes not guaranteed to be complete.
- The heuristic estimation has made A* more time efficient, but the memory is still an issue.
- The Iterative Deepening Search can solve the memory issue of A*. On the other hand, A* can also help eliminate time issues of the Iterative Deepening Search.

Question 2: Heuristic Path for the 15-Puzzle

a)

It seems the Greedy Heuristic does find out solutions to the goals quickly, but they are NOT optimal (value of G is usually larger than actual depth of the goal).

	Start50		Start60		Start64	
	G	N	G	N	G	N
Iterative Deepening A*	50	1462512	60	321252368	64	1209086782
Greedy	164	5447	166	1617	184	2174

(b)

The modified part of the source code is in RED colour.

```
% Otherwise, use Prolog backtracking to explore all successors
% of the current node, in the order returned by s.
% Keep searching until goal is found, or F_limit is exceeded.
depthlim(Path, Node, G, F_limit, Sol, G2)  :-
nb_getval(counter, N),     N1 is N + 1,
   nb_setval(counter, N1),
    % write(Node),nl,   % print nodes as they are expanded
   s(Node, Node1, C),
   not(member(Node1, Path)),      % Prevent a cycle
  G1 is G + C,
  W =  1.2,
   ,h(Node1, H1),
   F1 is (2-W) * G1 + W * H1,
    F1 =< F_limit,
   depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

I changed the code of predicate ***depthlim***. The original code is "$F1\ is\ G1+H1$", which represents $F(n)=G(n)+H(n)$. Now, we get a value of ω, and let "$F1\ is\ (2-\omega)*G1+\omega*H1$". Then the $F(n)=(2-\omega)*G(n)+\omega*H(n)$, which is the result we want.

(c)

	Start50		Start60		Start64	
	G	N	G	N	G	N
IDA*	50	1462512	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

(d)

More weights for the $f(n)$ may speedup the solution, but could may make the solution less optimal. It is also possible that large ω value like 1.6 may even lead to less accurate solution than Greedy Search because it is closer to a double Greedy Search that is $2f(n)$. IDA* is actually the situation when ω=1 and Greedy is the situation when ω=2. In general cases, as ω $grows\ from$ 1 to 2, the search algorithm runs faster, but the solution path become longer. Hence the speed is faster but quality is worse as ω increase.
When the W is growing from 1.2 to 1.6, the length of the path is growing while the number of nodes expanded is getting less and less. So the quality is becoming bad but the speed is getting faster when W changes from 1.2 to 1.6

Question 3: Maze Search Heuristics

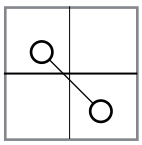
a)

Manhattan-Distance dominates the Straight-Line-Distance, and the formula for it is:

$$h(x, y, x_G, y_G) = |x\text{-}x_G| + |y\text{-}y_G|$$

(b - i)

No.Straight-Line-Distance heuristic is not admissible. In the question we consider the agent having the same cost when it move up, down, left, right or diagonally, while the actual cost when the agent move from one gird centre to the other grid centre, it cost √2, which is larger than 1.The definition of heuristics, the actual cost must lager than the calculated cost, while 1 <= √2, so the Straight-Line-Distance heuristic is not admissible.



(b - ii)

No. If the agent is considered to have the same cost with diagonally, horizontal and vertically, then the diagonally travel will be the same as only travel horizontally or vertically, while the Manhattan-Distance will be a sum of the travel horizontally and vertically.

(b - iii)

$$h(x, y, x_G, y_G) = \begin{cases} |x_G - x| \ \ if \ \ x_G - x \geq y - y_G \\ |y - y_G| \ if \ \ x_G - x < y - y_G \end{cases}$$

Question 4 - Graph Paper Grand Prix

(a) Where $k = 0$ the following is the output from $M(n,0)$ for $1 < n < 21$:

n	optimal sequence	+	o	-	t	$M(n,0)$
0		0	0	0	0	0
1	+ -	1	0	1	2	2
2	+ o -	1	1	1	3	3
3	+ o o -	1	2	1	4	4
4	+ + - -	2	0	2	4	4
5	+ + - o -	2	1	2	5	5
6	+ + o - -	2	1	2	5	5
7	+ + o - o -	2	2	2	6	6
8	+ + o o - -	2	2	2	6	6
9	+ + + - - -	3	0	3	6	6
10	+ + + - - o -	3	1	3	7	7
11	+ + + - o - -	3	1	3	7	7
12	+ + + o - - -	3	1	3	7	7
13	+ + + o - - o -	3	2	3	8	8
14	+ + + o - o - -	3	2	3	8	8
15	+ + + o o - - -	3	2	3	8	8
16	+ + + + - - - -	4	0	4	8	8
17	+ + + + - - - o -	4	1	4	9	9
18	+ + + + - - o - -	4	1	4	9	9
19	+ + + + - o - - -	4	1	4	9	9
20	+ + + + o - - - -	4	1	4	9	9
21	+ + + + o - - - o -	4	2	4	10	10

(b) In the above table the t column is the length of the optimal sequence and $M(n,0)$ was calculated with the following formula

$$M(n,0) = 2[\sqrt{n}] \qquad (1)$$

We can see that t agrees with the $M(n,0)$ all of the time. If we assume the following identity

$$\lceil 2\sqrt{n} \rceil = \begin{cases} 2s+1 \text{ if } & s^2 < n \leq s(s+1) \\ 2s+2 \text{ if } & s(s+1) < n \leq (s+1)^2 \end{cases} \qquad (2)$$

where s is the maximum speed (the number of '+'s, + column), then the identity holds for $s(s+1)$ when there is one rest (i.e. a 'o') and $s(s+1)$ when there are two rests.

Exception to this is when n is 2, 4, 9 or 16 i.e. n is a perfect square. Then the value from the identity is too high by 1 since $s^2 = n$.

(c) If $k \geq 0$ then we are starting with some k at S and if $n \geq k(K-1)$ then we stop at or before G (distance of n). So if we assume this roughly the second half of $M(n,0)$ i.e. the deceleration time, then equation (1) becomes the total distance x less the time it took to accelerate to velocity k that is

$$M(x,0) = [2\sqrt{x}] - k \quad (3)$$

where x is the total distance to accelerate to velocity k given by the Gaussian summation formula

$$\frac{k(k+1)}{2} \quad (4)$$

and n. Substituting these into (3) for x gives

$$M(n,k) = \left\lceil 2\sqrt{n + \frac{k(k+1)}{2}} \right\rceil - k \quad (5)$$

(d) From the same approach as (c) except $n < k(K-1)$ thus we overshoot the goal G and have to reverse to it. So $x > n$ and the total distance is now

$$\begin{aligned} M(n,k) &= \text{total overshoot time} + \text{reverse time} - \text{acceleration time} \\ M(n,k) &= \left\lceil 2\sqrt{\tfrac{k(k+1)}{2} + \tfrac{k(k-1)}{2}} \right\rceil + \left\lceil 2\sqrt{\tfrac{k(k-1)}{2} - n} \right\rceil - k \end{aligned} \quad (6)$$

the first term simplifies down to $2k$ (which makes sense) leaving us with

$$M(n,k) = \left\lceil 2\sqrt{\frac{k(k-1)}{2} - n} \right\rceil + k \quad (7)$$

(e) From the derivation above from 1-dimensional to 2-dimensional for the GPGP game we get

$$\boldsymbol{h(r,c,u,v,r_G,c_G) = max(M(r_G - r,u),M(c_G - c,v))} \quad (8)$$

where M is given by the equation (5) above.

This an admissible heuristic as it only considers one dimension(the longest) into account thus under estimates the vehicle performance. Even if it were to consider the hypotenuse distance this would, most probably, be in accurate as the GPGP is not normally a straight line.