UNSW AUSTRALIA

SCHOOL OF ENGINEERING

ASSIGNMENT COVER SHEET

[INDIVIDUAL SUBMISSION]

For submission to: Alan Blair

Course code : 9414

Course name : Artificial Intelligence

Assignment : Assignment2

ACADEMIC REQUIREMENTS

Before submitting this assignment, students are strongly advised to:

- Review the assessment requirements contained in the briefing document for the assignment.
- Review the various matters related to assessment in the relevant Course Outline.
- Review the Plagiarism and Academic Integrity website at https://student.unsw.edu.au/plagiarism to ensure they are familiar with the requirements to provide appropriate acknowledgement of source materials.
- Retain a copy of this assessment for their records and in case it is misplaced and has to be re-submitted.

If after reviewing this material there is any doubt about assessment requirements then in the first instance the student should consult with the Course Coordinator and then if necessary with the Director – Undergraduate Teaching and Learning Committee.

While students are generally encouraged to work with other students to enhance learning, all assignments submitted for assessment by a student must be their entire own work and they may be required to explain any or all parts of the assignment to the Course Coordinator or other authorised persons. Collusion is where another person (s) assist in the preparation of an assignment without the consent of knowledge of the Course Coordinator.
Plagiarism and Collusion are considered as Academic Misconduct and will be dealt with according to University Policy.

STUDENT DECLARATION OF ACADEMIC INTEGRITY

I declare that:

- This assessment item is entirely my own original work, except where I have acknowledged use of source material [such as books, journal articles, other published material, the Internet, and the work of other student/s or any other person/s].
- This assessment item has not been submitted for assessment for academic credit in this, or any other course, at UNSW or elsewhere.

I understand that:

- The assessor of this assessment item may, for the purpose of assessing this item, reproduce this assessment item and provide a copy to another member of the University.
- The assessor may communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).

Student first name: TANG **Surname:** NANYANG

Student number: z5103095 **Date:** 27/04/2018

Unless stated otherwise, assignments must be lodged in the nominated Assignment Box, no later than 5.00pm on the assignment due date.

Assignment 2

Question1:

(a) :

	start 10	start 12	start 20	start 30	start 40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*	29	21	952	17297	112571

Table 1

(b) :

UCS(uniform cost search):

As can be seen from Table 1, UCS requires more memory than other methods in solving 15-Puzzle. In fact, its space and time complexity are $O(b^{(C^*/\varepsilon)})(b^{(C^*/\varepsilon)} = b^d \; if \; all \; step \; costs \; are \; equal)$ and $O(b^{(C^*/\varepsilon)})$, respectively.

IDS(iterative deepening first search):

It is obvious that, IDS also occupies a lot of spaces, only less than UCS. In addition, its time complexity is also high. Its space complexity is $O(bd)$ and time complexity is $O(b^d)$. According to formula, the IDS utilizes less memory than UCS, however, the time complexity of IDS is same as UCS to some extent.

A*(A* search):

A* occupies far less memory than UCS and IDS. It combines the advantages of UCS and greedy search. Therefore, its formula is $f(n) = g(n) + h(n)$. However, memory usage is still a concern for A*. This point could be seen from start30 and start40 in Table 1. For time complexity, A* is faster than UCS and IDS obviously.

IDA*(iterative deepening A*):

IDA* is a low-memory variant. It occupies less space than A* although the time it consumed may be lightly more than A*.

Question2:

	start50		start60		start64	
	G	N	G	N	G	N
IDA*	50	14642512	60	321252386	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

Table 2

(a):

The answer is shown in the row named Greedy in Table 2.

(b):

```
% 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,
    h(Node1, H1),
    F1 is 0.8*G1 + 1.2*H1, %   f(n) = (2-w)*g(n) + w*h(n) where w=1.2
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

Figure 1

From figure 1, F1 = G1+H1 has been changed into F1 = 0.8*G1+1.2*H1. Actually, this code delegates the evaluation function: $f(n) = (2 - w)g(n) + w\,h(n)$, where $0 \leq w \leq 2$. In this code, the function is modified from $f(n) = g(n) + h(n)$ into $f(n) = 0.8g(n) + 1.2\,h(n)$.

(c):

The results are demonstrated in Table 2.

(d):

When w=0, the function is greedy search. When w=1, the function is A* search. From the data in Table 2, we could know that the length of path would increase and the total number of states expanded would decrease with the increase in w. This means that the quality is worse, however, speed is quicker with the growth in w.

Question3:

(a):

$$h_2(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$$

The heuristics is the Manhattan distance of the maze. It delegates the least total steps required from current position to Goal position. Also, compared with h_{SLD}, $h_2 \geq h_{SLD}$ .This means that h_2 dominates h_{SLD} and h_2 would produce less or equal number of nodes compared with h_{SLD}.

(b):

(i): The Straight-Line-Distance heuristic is not admissible. This is because the step to diagonal position is still regarded as 1 step. However, if we arrived the same position employing Straight-Line-Distance heuristic, it would take $\sqrt{2}$ steps. Therefore, the h_{SLD} is more than the h^* where h^* is the true cost from current position to goal. The figure2 blow shows the consequence.

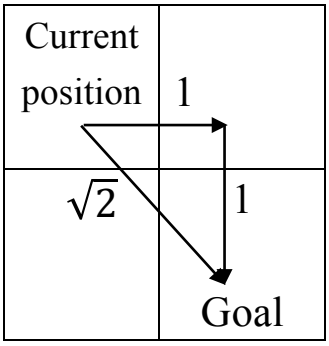

Figure 2

(ii): The $h_2 \geq h_{SLD}$, as a result, the cost consumed by h_2 is more than h^*. This is because h_2 would take 2 steps compared to h^* consuming 1 step. For this reason, h_2 is not admissible.

(iii): $h_3(x, y, x_G, y_G) = \max(|x - x_G|, |y - y_G|)$

When $|x - x_G| = |y - y_G|$, the most cost consumed is $|x - x_G|$. When $|x - x_G| \neq |y - y_G|$, the most cost consumed is max($|x - x_G|, |y - y_G|$). Figure 3 shows different conditions for the heuristics.

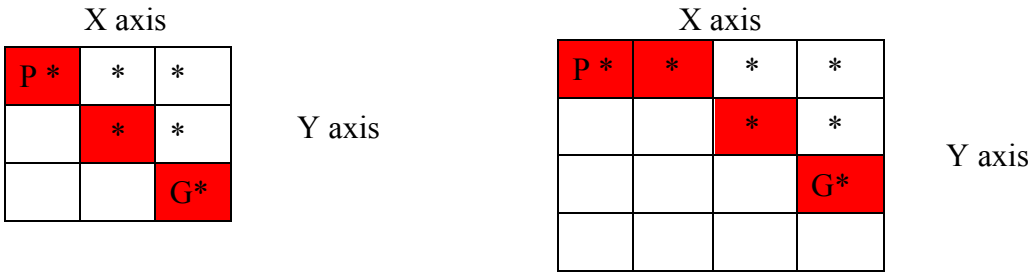

Figure3

Note: The red square delegates the actual path.

Question4:

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

Table 3

(a):

As can be seen from Table 3, M(n, 0) for 1 ⩽ n ⩽ 21 is shown by writing down the optimal sequence of actions for all n between 1 and 21.

(b):

In fact, $M(n,0) = \lceil 2\sqrt{n} \rceil$. It follows the principle below:

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

where s is the maximum speed from S position to n position.

Firstly, I assume $n = s^2$. This means in the optimal sequence, velocity keeps change and [o] is not in the optimal sequence. Then the path length= $\frac{2s(s+1)}{2} - s = s^2$. Fastest time is equal to 2s. Next, I assume $n = s^2 + 1$, it is observed that I only need add a step that length is equal to 1 into the total distance of $n = s^2$. During the process, time is equal to 2s+1. Similarly, when $n = s^2 + k\ where\ 1 \le k \le s$, the time consumed is only 1 second more than $n = s^2$. Therefore, when $s^2 < n \le s(s+1)$, fastest time is $2s+1$. Now, we assume $n = s^2 + z\ where\ s < z \le 2s$. This means that we only add a step that length is no more than s into $n = s^2 + k$. However, the time consumed by this step is also 1 second. Therefore, the fastest time takes $2s+1+1 = 2s+2$ second. When $n = s^2 + z\ where\ z = 2s+1$, it would return initial state which is $n = (s+1)^2$ and its algorithm logic is same as $n = s^2$, so the time consumed is $2(s+1) = 2s+2$. For this reason, I revised this formula:

$$M(n,0) = \lceil 2\sqrt{n} \rceil = \begin{cases} 2s+1, & if\ s^2 < n \le s(s+1) \\ 2s+2, & if\ s(s+1) < n \le (s+1)^2 \\ 2s, & if\ s^2 = n \end{cases}$$

Now, the formula should be appropriate for all cases.

(c):

If we regard every move from velocity 0 to velocity k which is maximum speed as an arithmetic progression, the total distance of acceleration would be $n\frac{(a_1+a_n)}{2} d\ where\ n = k, d = 1, a_1 = 1, a_n = k$. It could become $\frac{k(k+1)}{2}$ by simplifying. When the vehicle arrived firstly at velocity k, we add the total distance of acceleration into the total distance from S to G. This means we start to move again from S to G at velocity=k. This the time consumed in process is same as $M\left(n + \frac{k(k+1)}{2}, 0\right) - t\ where\ t = k\ is\ the\ acceleration\ time$. Therefore,

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

The figure 4 below show the process.

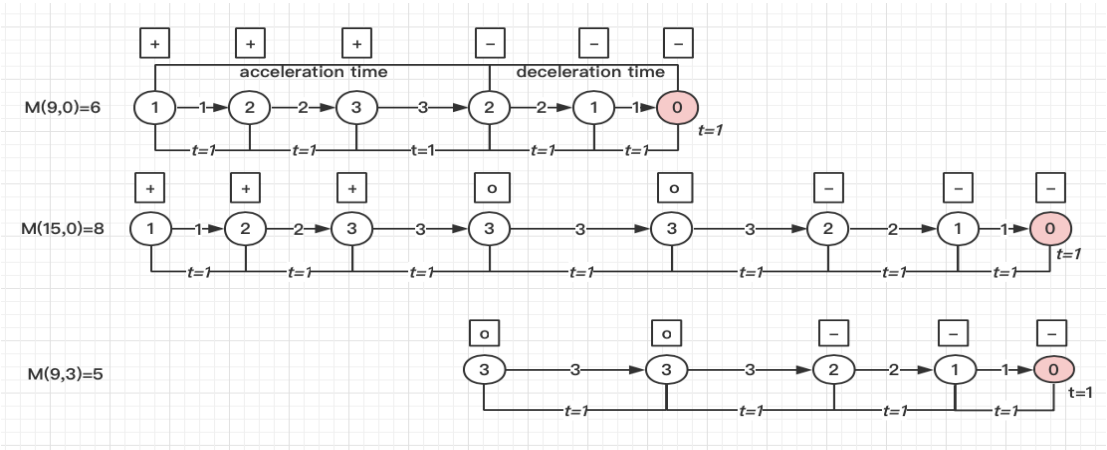

Figure 4*(The circle with red is G)*

(d):

when $n < \frac{1}{2}k(k-1)$, it means G is in the place at speed firstly $\neq 0$. In order to satisfy the requirement that velocity is 0 at G, a path reverse is necessary.

$M(n,k) = time\ that\ the\ velocity\ firstly\ from\ 1\ to\ 0 - acceleration\ time + reverse\ time.$

The distance from k=1 to maximum speed k and the distance from maximum speed to k=0 are regarded as two arithmetic progressions, respectively. They are acceleration distance:$\frac{k(k+1)}{2}$ and deceleration distance:$\frac{k(k-1)}{2}$. Reverse time: $M(n,k) = \left\lceil 2\sqrt{-n + \frac{k(k-1)}{2}} \right\rceil$.

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

$$M(n,k) = k + \left\lceil 2\sqrt{-n + \frac{k(k-1)}{2}} \right\rceil$$ The figure 5 shows the process.

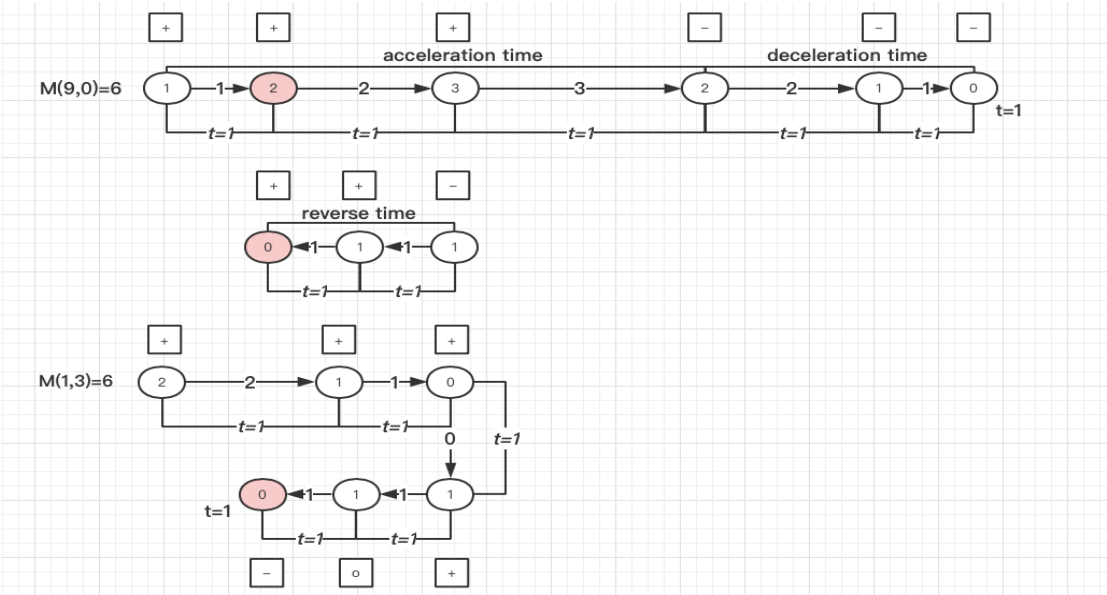

Figure5*(The circle with red is G)*

(e):

$$M(n,k) = \begin{cases} \left\lceil 2\sqrt{n + \frac{k(k+1)}{2}} \right\rceil - k & n \geq \frac{k(k-1)}{2} \\ k + \left\lceil 2\sqrt{-n + \frac{k(k-1)}{2}} \right\rceil & n < \frac{k(k-1)}{2} \end{cases}$$

$h(r, c, u, v, r_G, c_G) = \max\left(M\big((r_G - r), u\big), M\big((c_G - c), v\big)\right)$

This heuristic is admissible. This is because when the speed of vertical or horizontal direction arrived 0, we need only consider whether the speed of other direction reaches 0. Therefore, maximum time taken by $M\big((r_G - r), u\big)$ or $M\big((c_G - c), v\big)$ is reasonable.