Question 1 - Search Algorithms for the 15-Puzzle

(a)

	Start 10	Start 12	Start 20	Start 30	Start 40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2047	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*	29	21	952	17297	112571

(b)

For the UCS, the efficiency is the lowest, position start from 10 generates most nodes among four and for the position start from 12,20,30,40, memory is run out of.
For the IDS, the efficiency is the second lowest, position start from 10, 12, 20 generates second largest nodes among four, and for position 30, 40, the time complexity is high.
For A*, it has good efficiency, it performs good when it search 10, 12, 20 depth, but it cost much memory for position 30, 40, the time costs is ok.
For IDA*, it has the best efficiency among all algorithm, and it is the only one which can get the data for the depth 40, however, it produces more nodes as position increase, and time seems cost more.

Question 2 – Heuristic Path Search for the 15-Puzzle

(a)&(c)

	Start50		Start60		Start64	
IDA*	50	1462512	60	321252368	64	1209086782
1.2	52	191438	62	230861	66	431033
1.4	66	116174	82	3673	94	188917
1.6	100	34647	148	55626	162	235852
Greedy	164	5447	166	1617	184	2174

(b)

```
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),
    W is 1.2, % W = 1.2/1.4/1.6/1.8
    F1 is (2-W) * G1 + W * H1,
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

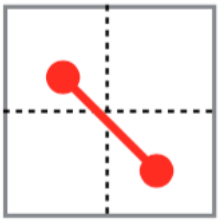
The changed code is to add W as a parameter when calculate the F. The objective function is

f(n) = (2-w) * g(n) + w * h(n).

(d)
For IDA*, the solution is the best, the path is always the shortest, for the time consuming, it cost the most
For w from 1.2 to 1.6, the solution become less effective and time cost decrease.
For Greedy search, the time cost is the shortest while the solution is not as good as others.

Question 3 – Maze Search Heuristics
(a)
The Manhattan-Distance dominates the Straight-Line-Distance, and the formula for it is:
$H(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$

(b)
(i)
No. The 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. Based on 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.



(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.

(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)	2s+1	2s+2
0		0	0	0	0	0	1	2
1	+ -	1	0	1	2	2	3	4
2	+ o -	1	1	1	3	3	3	4
3	+ o o -	1	2	1	4	4	3	4
4	+ + - -	2	0	2	4	4	5	6
5	+ + - o -	2	1	2	5	5	5	6
6	+ + o - -	2	1	2	5	5	5	6
7	+ + o - o -	2	2	2	6	6	5	6
8	+ + o o - -	2	2	2	6	6	5	6
9	+ + + - - -	3	0	3	6	6	7	8
10	+ + + - - o -	3	1	3	7	7	7	8
11	+ + + - o - -	3	1	3	7	7	7	8
12	+ + + o - - -	3	1	3	7	7	7	8
13	+ + + o - - o -	3	2	3	8	8	7	8
14	+ + + o – o - -	3	2	3	8	8	7	8
15	+ + + o o - - -	3	2	3	8	8	7	8
16	+ + + + - - - -	4	0	4	8	8	9	10
17	+ + + + - - - o -	4	1	4	9	9	9	10
18	+ + + + - - o - -	4	1	4	9	9	9	10
19	+ + + + - o - - -	4	1	4	9	9	9	10
20	+ + + + o - - - -	4	1	4	9	9	9	10
21	+ + + + o - - - o -	4	2	4	10	10	9	10

(b)

From the table above, t column is the length of the optimal sequence and M(n, 0) was calculated with the formula: M(n, 0) = $\lceil 2\sqrt{n} \rceil$, and t agrees with the M(n, 0) all of the time if we assume the following identity

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

for this, 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.

However, there is exception that 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 >= 0 then it can be started with some k at S and if n >= k(K-1) then it can be stopped at or before G(distance of n). So if assume this roughly the second half of M(n, 0) i.e. the deceleration time, then equation M(n, 0) = $\lceil 2\sqrt{n} \rceil$ becomes the total distance x less the time

it took to accelerate to velocity k that is M(x, 0) = $\lceil 2\sqrt{x} \rceil$ - k where x is the total distance to accelerate to velocity k: k(k+1)/2 (Gaussian summation formula) and n.

So from the formulas above, the formula $M(n,k) = \left\lceil 2\sqrt{n + \frac{1}{2}k(k+1)} \right\rceil - k$ can be proved.

(d)

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

M(n, k) = total over shoot time + reverse time – acceleration time

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

(e)

h(r, c, u, v, r_G, c_G) = max(M(r_G −r, u), M(c_G − c, v))