Name : Haiyu LYU zNumber: z5142340

Question 1: Search Algorithms for the 15-Puzzle

(a)

Algorithm\Start	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

(b)

1.UCS: From the table we can see clearly that it is not a efficient algorithm. And UCS has a large space complexity, because this algorithm is easy to runs out of memory.

2.IDS: The efficiency of IDS is not high as it generates a number of nodes. At the same time, it has the largest time complexity(runs for a long time and no result).

3.A*: It has better efficiency when start position is 10, 12 and 20. But as depth increases, it will go out of memory.

4.IDA*: Compared to the above three algorithms, IDA* has the best efficiency. Since it combines A* and IDA, he can calculate the result of Start40.

Question 2: Heuristic Path Search for 15-Puzzle

(a)

	Start 50		Start 60		Start 64	
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)

When w = 1.2

```
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)
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

(c)
When w = 1.4

```
F1 is 0.6*G1 + 1.4*H1,
% f(n) = (2-w)*g(n) + w*h(n)
```

When w = 1.6

```
F1 is 0.4*G1 + 1.6*H1,
% f(n) = (2-w)*g(n) + w*h(n)
```

(d)
1.When w = 1, the length of the path is shortest and the number of nodes is very big.
2.When w = 1.2, the we can see that all paths become longer, while the nodes are more smaller than before. Hence we can complete the search in less time.
3.When w=1.4, we can see from the table that it needs 116342 nodes in Start50. However, in Start60, it costs 4432 nodes. Hence, we can think that when the paths become longer, the generated nodes does not necessarily grow.
4.When w=1.6,in Start60, it generates 55626 nodes. This is a lot more than the nodes(4432) generated in Start60 when w=1.4. Therefore, when the weight of heuristic function becomes larger, the search is not optimal.
5.The greedy algorithm uses less time than the IDA* algorithm because it produces few nodes. Although its search is not complete, it is very efficient.

Question 3: Maze Search Heuristics

(a)
Another admissible heuristic could be Manhattan Distance:

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

(b)
(i) No
We can assume that we move from (0,0) to (1,2). Firstly, we need to move (0,0) to (1,1) and the cost is 1 moves. Then the cost moving from (1,1) to (1,2) is 1 moves. Hence h(n)* is 2.However, h(n) = √5. Because of h(n)>h(n)*, SLD heuristic is not admissible.

(ii) No
We use the last method that move from (0,0) to (1,2). We calculate that h(n)* is still equal to 2, while h(n) is 3. As h(n) is greater than h(n)*, Manhattan Distance heuristic is not admissible.

(iii) The best admissible heuristic:

$$\mathbf{h(x,y,x_G,y_G) = max(|x_G - x| , |y_G - y|)}$$

Question 4: Graph Paper Grand Prix

(a)

When k=0,1<=n<=21:

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

(b)

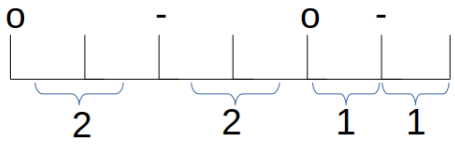
$$M(n,0) = \left\lceil 2\sqrt{n} \right\rceil,$$

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

S is maximum speed. This means that s is the number of '+'. For example, in [+ + o - o -] , s= 2.
When optimal sequence has one 'o',M(n,0) = 2s+1.
When optimal sequence has two 'o',M(n,0) = 2s+2.
When there is no 'o' in optimal sequence, M(n,0) = 2s.

(c)

Let k = 2,s = 0 and n=6.They meet the conditions: $n \ge \frac{1}{2}k(k-1)$



The optimal sequence is [o o - -].

m(6,2)=4

Use formula we can calculate m(6,2):

$$m(6,2)=\lceil 2*\sqrt{(2*(2-1)\div 2)+6}-2\rceil=4$$

(d)

(e)