Comp3411 Assignment2

Xiangzhuo meng z5042679

1.(a)

The number in the table -> the number of nodes generated during the search

The Mem -> runs out of memory.

Time -> runs for five minutes without producing output.

	Start10	Start12	Start20	Start30	Start40
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)Briefly discuss the efficient of these four algorithms

From the table above, We can see that the UCS(Uniform Cost Search) has the lowest efficiency and the highest space complexity since it runs out of memory from Start12.

The IDS(Iterative Deepening Search) is a bit quick than UCS but still not good. It runs out of time from Start30. And generated 5297410 nodes when it running at Start20 which is too many.

The A* has a good performance on time. It is much more quick than UCS and IDS. However, it has a high space complexity which need cost a lot of memory. The A* runs out of memory when search 30 an 40.

The IDA*(Iterative Deepening A*Search) is the most efficient algorithms among all of the 4 algorithms, which use the least time and memory.

2.

	Start50		Start60		Start64	
IDA*	50	14642512	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),

    %adding part
    W is 1.2 ,
    F1 is (2-W) * G1 + W*H1,
    %adding part

    %delete part
    %F1 is G1 + H1,
    %delete part

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

(d)

Actually , IDA* algorithm is when w =1 and Greedy is when w = 2.

From the table in 2(a), as w increase from 1 to 2, the time needed by the search algorithm decrease which means the speed become faster, but the quality of the solution become worse since the length of the solution path is longer.

3.

(a) Admissible Heuristic dominating straight-line distance heuristic—Manhattan Distance Heuristic:

h(x,y,xG,yG) = |x-xG|+|y-yG|

(b)

(i) In this case, The Straight-Line-Distance heuristic is not admissible.

For example,

State	
	Goal

The h_{SLD} = $\sqrt{2}$, however the agent can take one step either up, down, left, right or diagonally. If the agent moves diagonally from the State to the Goal, the cost will be 1 < $\sqrt{2}$. The SLD heuristic is not admissible.

(ii) My heuristic from part(a) is not admissible as well.

In 3(a) the $h(x,y,x_G,y_G)$=1+1 = 2 > 1. This heuristic is not admissible.

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

Since the diagonal move just costs 1 which is same as the vertical and horizontal move. We just need to find the bigger one of the horizontal distance and the vertical distance

4

(a) Table to record M(n,0) (the minimum number of time steps required to arrive and stop at the Goal.) for 1 <= n <= 21.

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

20	9	+ + + + o - - - -
21	10	+ + + + o - - - o -

(b)

Assume ***S*** is the number of speed up operation(+)

n	M(n,0)	Optima sequence	S	s^2	s*(s+1)	$(s+1)^2$	2*s+1	2*s+2
1	2	+ -	1	1	2	4	3	4
2	3	+ o -	1	1	2	4	3	4
3	4	+ o o -	1	1	2	4	3	4
4	4	+ + - -	2	4	6	9	5	6
5	5	+ + - o -	2	4	6	9	5	6
6	5	+ + o - -	2	4	6	9	5	6
7	6	+ + o - o -	2	4	6	9	5	6
8	6	+ + o o - -	2	4	6	9	5	6
9	6	+ + + - - -	3	9	12	16	7	8
10	7	+ + + - - o -	3	9	12	16	7	8
11	7	+ + + - o - -	3	9	12	16	7	8
12	7	+ + + o - - -	3	9	12	16	7	8
13	8	+ + + o - - o -	3	9	12	16	7	8
14	8	+ + + o - o - -	3	9	12	16	7	8
15	8	+ + + o o - - -	3	9	12	16	7	8
16	8	+ + + + - - - -	4	16	20	25	9	10
17	9	+ + + + - - - o -	4	16	20	25	9	10
18	9	+ + + + - - o - -	4	16	20	25	9	1

19	9	+ + + + - o - - -	4	16	20	25	9	10
20	9	+ + + + o - - - -	4	16	20	25	9	10
21	10	+ + + + o - - - o -	4	16	20	25	9	10

From the table above , we can see that s = $\lfloor\sqrt{n}\rfloor$(s is integer), when s =$\sqrt{n}$, M(n,0) = 2*$\sqrt{n}$.

When s != $\sqrt{n}$, Assume the identity:

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

When s^2 < n <= s*(s+1), there are one “o” operation in the optimal sequence, M(n,0) should be 2*s +1 = $\lceil 2\sqrt{n} \rceil$.

When s(s+1) < n <= (s+1)^2, there are two “o” operation in the optimal sequence, M(n,0) should be 2*s+2 =$\lceil 2\sqrt{n} \rceil$.

In general, M(n,0) = $\lceil 2\sqrt{n} \rceil$.

(c)

M(n,k) means the agent starts from location 0, with velocity k, to the goal n and we need to find the minimum number of time steps required to arrive and stop at the Goal.

K >= 0 and n >= ½*k*(k-1)

The difference between M(n,k) and M(n+x,0) is there are k “+” operation in M(n,k) than M(n+x,0), x is the distance of uniform linear acceleration motion from v = 0 to v = k.

X = (0+k)*(k+1)/2 = k*(k+1)/2

M(n,k) = M(n+x,0)-k = $\left\lceil 2\sqrt{n + k * \frac{k+1}{2}} \right\rceil - k.$

(d)

M(n,k) when k >= 0 and n <= k*(k-1)/2

Since n < K*(k-1)/2. The final velocity can't be 0 even do a uniform linear deceleration motion. We need to reverse the path.

X = (0+k)*(k+1)/2 = k*(k+1)/2

r = 2*(k*(k+1)/2-n)

M(n,k) = M(n+x+r)-k

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

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

$$= \left\lceil 2\sqrt{3 * \frac{k*(k+1)}{2} - n} \right\rceil - k$$

(e)

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

$$M(n,k)\begin{cases} \left\lceil 2\sqrt{n + k * \frac{k+1}{2}} \right\rceil - k & k >= 0 \text{ and } n >= k * (k-1)/2 \\ \left\lceil 2\sqrt{3 * \frac{k*(k+1)}{2} - n} \right\rceil - k & k >= 0 \text{ and } n <= k * (k-1)/2 \end{cases}$$

Type equation here.