Heuristics and Search

COMP9814 Assignment 2

Tianpeng Chen z5176343

Question 1

a)

	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)

	Time Efficiency	**Memory Usage**
UCS	Medium	High (Out of stack when node >= 12)
IDS	High when node < 20 Low when node >= 20	Low
*A**	High	Medium (Out of stack when node >= 30)
*IDA**	High	Low

UCS is complete (if b is finite) and optimal, but it exhaustively expands all nodes closer to the initial state. Thus, it will consume much time and pretty much memory to do so.

IDS is also complete and optimal, but it's not time efficient (runs in $O(b^m)$).

A* Search and **IDA*** Search are similar. They both run very fast when node <= 20. As number of states grows, A* will consume more memory as it keeps all nodes in memory; whereas **IDA*** works perfectly.

Question 2

a)

	Start50		**Start60**		**Start64**	
*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	142	55626	162	235848
Greedy	164	5447	166	1617	184	2174

b)

Simply change this section of heuristic.pl (copied from idastar.pl)

```
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,        % F1 is (2-w)G1 + wH1

F1 =< F_limit,

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

d) IDA* is a special case of heuristic search (with w=1), and greedy search is also a special case of heuristic search (with w=2). As w grows, this algorithm becomes more ‘greedy’, i.e. care more about local optimal other than global optimal. As a consequence, the algorithm runs faster, but the quality of answer decreases.

Question 3

a) Let $\mathrm{h}(x, y, x_G, y_G)$ be the **Manhattan-Distance** heuristic:

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

Manhattan-Distance, also known as taxicab-distance, or L1 distance, which calculates the total travel distance between two points in a grid based on strictly horizontal and/or vertical path (along the grid lines). Manhattan-Distance is equal in either directions. If there are no obstacles, this distance is clearly exact, and introducing obstacles can only increase the length of the optimal path from the initial point to the goal, which means it's always not higher than the actual cost. Thus, this heuristic is admissible.

Manhattan-Distance heuristic dominates the Straight-Line-Distance heuristic because it will always be not lower than the Straight-Line-Distance (which is the shortest distance between two points). But especially for grid-like spaces (e.g. a maze), Manhattan-Distance heuristic is a better heuristic. Because Straight-Line-Distance between 2 points is always underestimate in a maze.

b)

i) No. Assume we have two points A (0, 0), B (1, 1). The actual distance from A to B will be 1 (diagonal move cost is 1), but the Straight-Line-Distance is $\sqrt{2} > 1$. Thus, it's not admissible.

ii) No. Imagine the same points A (0, 0) and B (1, 1). The actual distance is 1, but the Manhattan-Distance is 1+1=2 > 1. Thus, it's not admissible.

iii) In this case, let $\mathrm{h}(x, y, x_G, y_G)$ be the **Chebyshev-Distance** heuristic:

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

Chebyshev-Distance (also known as L∞ Distance), is similar to Manhattan-Distance, but the move cost of one horizontal, vertical or diagonal step is 1. The Chebyshev-Distance from two points will be the sum of any horizontal, vertical and diagonal steps.

If there are no obstacles, the actual distance between any two points will be exact as Chebyshev-Distance;

With obstacles, the actual distance will be greater than the Chebyshev-Distance;

Thus, this heuristic function is admissible. In fact, it's the best admissible heuristic function for this situation.

Question 4

a) The following chart shows output of $M(n, 0)$ for $1 < n < 21$, where $k = 0$:

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

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

We can see that t agrees with $M(n, 0)$ all the time. Assume the following identity:

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

s is the maximum speed (can be simply indicated by number of '+'s, we can see the identity $2s+1$ holds for all the situations where there's one rest (1 'o'), and $2s+2$ holds for all the situations where there are two rests (2 'o's).

We also notice that when *n* is a perfect square (i.e. *n*=0, 1, 4, 9, 16… see the shaded rows in the above chart), there is no rest in the optimal sequence, and the identity function above don't hold for these situations. In this case, since *n* is a perfect square, we can calculate $2\sqrt{n}$ straightforward for $M(n, 0)$.

c) Given $k \geq 0 \; and \; n \geq \frac{1}{2}k(k-1)$, we are starting from *S'* with velocity *k*. Assume that we started from *S* with velocity 0 and keep accelerating to *k* until we arrive to *S'* (with no rests). The time consumed on this distance is *k,* the distance will be $k + k - 1 + k - 2 + \cdots + 0 = \frac{k(k+1)}{2}$. After that, if we want to stop at *G*, we have to deaccelerate from *k* to 0 (with some rests), the time consumed in this part is $M(n, k)$. Finally, the total time we need from *S* to *G* is $M(x, 0)$.

Thus, we have:

Time from *S' to G* = Time from S to G – Time from S to S'

$$M(n, k) = M(x, 0) - k$$

where *x* is $n + \frac{k(k+1)}{2}$,

Finally, we have:

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

d) If $n < \frac{k(k-1)}{2}$, we can't stop at *G* but a certain spot *G'* after *G*. That is, we overshoot the goal *G* and need reverse to it. Let *x*>*n* be the total distance:

$$M(n, k) = total\ time + reverse\ time - acceleration\ time$$

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

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

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

where *M* donates the total time from one point with some velocity to the goal point with no velocity.

This heuristic is admissible because if the car goes horizontally or vertically, it will be exact as the actual cost; otherwise it's always not higher than the actual cost.