Assignment 2

Liangde Li, z5077896

Question 1

(a).

	start10	start12	start20	start30	start40
ucsdijkstra	2565	Mem	Mem	Mem	Mem
ideepsearch	2407	13812	5297410	Time	Time
astar	33	26	915	Mem	Mem
idastar	29	21	952	17297	112571

(b).

1. Uniform cost search is not efficient in both time and memory, as it will explore all the ‘nodes’ which has less cost than the goal.
2. Iterative deepening search is not efficient in time, as it will explore some nodes multiple times, more specifically, the nearer, the more. It is better in memory cost, as it discards some nodes when it moved to other branches.
3. A star search is better than the above two search algorithm, as a good heuristic function will help it to find goal faster, which saves time. But it run out of memory at large size, as it will explore all nodes who has less estimated total cost than the goal’s, to guarantee finding the optimal solution.
4. Iterative deepening A star search is more memory efficient than normal A star search, as its better performance at large size like 30 and 40. The time performance is quite same as A star.

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	148	55626	162	235949
Greedy	164	5447	166	1617	184	2174

(b).
W = 1.2
F1 is 0.8*G1 + 1.2*H1

(c).
W = 1.4
F1 is 0.6*G1 + 1.4*H1
W=1.6
F1 is 0.4*G1 + 1.6*H1

(d).
F = (2-w)*G + w*H
For the five algorithms above, w value are 1, 1.2, 1.4, 1.6 and 2 respectively, meaning the decision are more depends on the expected rest cost to the goal, which makes N values, the explored nodes less, the time to find a path to the goal shorter and shorter. But is not guaranteed to find the optimal solution anymore. It is a trade-off between optimality and time efficiency.

Question 3

(a).
$h(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$

(b).
(i). It is not admissible. For example, if the current position is diagonal neighbour to the goal, the heuristic value is the distance which is $\sqrt{2}$, but it only needs one move.
(ii). It is not admissible. For example, if the current position is diagonal neighbour to the goal, the heuristic value is 2, but it only needs one move.
(iii).

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

Question 4

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

(b).
If $s^2 < n \leq s(s+1)$, it needs s many time to speed up to s, and s many time to speed down to 0, which will run s^2 distance, and then choose any speed from 1 to s to run one more time, so the range is $s^2 < n \leq s(s+1)$, total action is 2s+1.
If $s(s+1) < n \leq (s+1)^2$, it needs at most than s+1 time to speed up, if number of '+' is s+1, which will run $(s+1)^2$ distance. If number of '+' is s, number of '-' is s, so will run s^2 distance, and run two more step, one is at speed s, the other can be from 1 to s-1, so the range is $s(s+1) < n \leq (s+1)^2$, total action is 2s+2.

(c).

Consider a larger path n+$\frac{1}{2}k(k+1)$,

M(n+$\frac{1}{2}k(k+1)$, 0) = $ceiling(2\sqrt{\mathrm{n}+\frac{1}{2}k(k+1)})$,

Minus the first k number of '+', it will be reduced to :

$ceiling(2\sqrt{\mathrm{n}+\frac{1}{2}k(k+1)})$-k

Which is the maximum value, M(n,k).

(d).

So first it will need to keep speeding down to 0.

The distances for all steps are:

k-1, k-2, ….., 1.

Which will travel $\frac{1}{2}k(k-1)$ distance, and is greater than n.

So then it needs to travel in opposite direction of distance:

$$n-\frac{1}{2}k(k-1)$$

So the M(n,k) = k + M($n-\frac{1}{2}k(k-1), 0$)

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

(e).

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