COMP3411 Assignment1 Part2

Tong Zheng
z5142003

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)

According to the results above,
Uniform-Cost Search is the least efficient algorithm. The time and space complexity are both O(b[C*/ϵ]).

Iterative Deepening Search has space complexity of O(bd) and time complexity of O(b^d), causing the “time out” at start30 and start40.

A star Search is more efficient than the previous two algorithms. Since it uses the evaluation function f(n) = h(n) + g(n), taking both the cost to current node and cost from current node to the goal state into consideration. While trying minimize the cost, it searches with rather fast step. However, it struggles dealing with long term searching, since after expanding a great number of nodes, it takes much more time and space to keep searching, which cause it ends in running out memory in start30 and start40 since its space complexity can go to O(b^d).

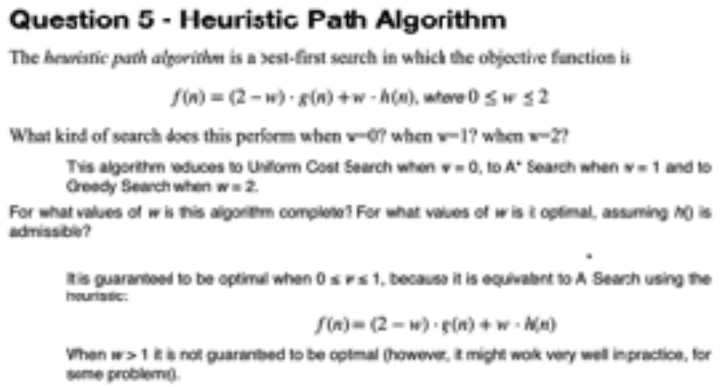
Iterative Deepening A-Star Search is the most efficient algorithm among all above. It combines IDS and A* search. Each performance it cuts off each search when f exceeds current threshold. With time complexity of O(b^ϵd) and space complexity of O(bd), it dose not run out of memory at start30 or start40.

Question 2

a) c)

	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	235848
Greedy	164	5447	166	1617	184	2174

b)



Question 5 - Heuristic Path Algorithm

The *heuristic path algorithm* is a best-first search in which the objective function is

$$f(n) = (2 - w) \cdot g(n) + w \cdot h(n), \text{ where } 0 \le w \le 2$$

What kind of search does this perform when w=0? when w=1? when w=2?

This algorithm reduces to Uniform Cost Search when $w = 0$, to A* Search when $w = 1$ and to Greedy Search when $w = 2$.

For what values of w is this algorithm complete? For what values of w is it optimal, assuming $h()$ is admissible?

It is guaranteed to be optimal when $0 \le w \le 1$, because it is equivalent to A Search using the heuristic:

$$f(n) = (2 - w) \cdot g(n) + w \cdot h(n)$$

When $w > 1$ it is not guaranteed to be optimal (however, it might work very well in practice, for some problems).

According to the function f(n) = (2-w)*g(n) + w*h(n) where 0<=w<=2.
When w = 1.2, we get f(n) = 0.8g(n) + 1.2h(n).
Hence, replace the “F1 is G1 + H1” with “F1 is 0.8*G1 + 1.2*H1”

```
31  % Otherwise, use Prolog backtracking to explore all successors
32  % of the current node, in the order returned by s.
33  % Keep searching until goal is found, or F_limit is exceeded.
34  depthlim(Path, Node, G, F_limit, Sol, G2)  :-
35      nb_getval(counter, N),
36      N1 is N + 1,
37      nb_setval(counter, N1),
38      % write(Node),nl,   % print nodes as they are expanded
39      s(Node, Node1, C),
40      not(member(Node1, Path)),      % Prevent a cycle
41      G1 is G + C,
42      h(Node1, H1),
43      F1 is G1 + H1,
44      F1 =< F_limit,
45      depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
46
```

```
% Otherwise, use Prolog backtracking to explore all successors
% of the current node, in the order returned by s.
% Keep searching until goal is found, or F_limit is exceeded.
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 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

d)

These five algorithms are all based on formula f(n) = (2-w)*g(n) + w*h(n) where 0<=w<=2.
When we have w = 0, it's greedy search which try to minimize the cost at each step. When we have w = 1, it's IDA* search which minimize both current cost and future cost.
It takes greedy search less time to find the path, however, the path might not be optimal. When w decreases towards 1, the nodes expanded would increase, but the path will get more close to optimal path, path length will decrease.