% Yiyan Yang
% z5183946
% Assignment 1-Prolog and Search

Question 1

	start10	start12	start20	start30	start40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13812	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*	19	21	952	17297	112571

The simplest algorithm (in this assignment) to find a shortest path is Dijkstra, which is implemented using a priority queue. While the complexity of Dijkstra is $O(V + E \log V)$ where E is equal to V (each state can be converted to two states, so degree of each vertex is 2). Hence the complexity is $O(VlogV)$, which is relatively high. And the space complexity is $O(N^2)$, which is still high. Because the memory grows faster than time needed, Mem occurred in the table.

Iterative deepening search uses depth first search with an increasing depth bound. As a DFS is performed in each depth, the time complexity $O(b^d)$ is rather higher than that of Dijkstra, where b is estimated to be 2 and d is the depth. The time needed to find the appropriate path is growing exponentially according to the minimum number of moves. However, the growth rate of it is just linear, which is $O(bd)$, so we haven't seen a Mem is the table.

A* algorithm uses heuristic function to guide the algorithm to choose nodes with higher expectation. The time complexity of A* is polynomial if appropriate heuristic is used, as the heuristic usually eliminates most of the bad choices. And the space required is $O(V) = O(b^d)$, which is determined by the depth.

Iterative deepening A* algorithm is a modified version of A* with an increasing depth bound. So it has exactly the same time and space complexity as that of A*. But with the help of iterative deepening bound, this algorithm can avoid searching to deep and exceed the memory limit, note that the space complexity is growing exponentially as the depth grows. Thus, we have Mem for A* but finally found all the answers for IDA*.

Question 2

	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	217

```
G1 is G + C,
h(Node1, H1),
F1 is G1 + H1,      You, 11 minutes ago • New progress and
F1 =< F_limit,
depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

The line highlighted is the code that is modified, which is the heuristic with w equals to 1:

$$F1 = (2 - w)G1 + w * H1$$

For example, when $w = 1.2$, the heuristic then becomes:

$$F1 \; is \; 0.8 * G1 + 1.2 * H1$$

With higher w, the heuristic is “greedier” as it considers more on *expected* length rather than *passed* length. So, the algorithm is more likely to terminate and tend to backtrack less frequently which makes it faster. As a consequence, the accuracy is lowered as cost of high speed. Hence when accuracy is not that important, a higher w is considerable, conversely, if a optimal answer is required, IDA* should be chosen.