Kylee Bowers

z5304138

Assignment 1 – Prolog and Search

2.1

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)

UCS (Uniform Cost Search): USC is both time inefficient and memory inefficient. Its time complexity is $O(b^{1+(C/e)})$, where b is its branching factor, C is the destination cost, and e denotes the steps taken to reach the destination. In terms of memory usage, USC performed the worst. This can be observed from the table in part a. It ran out of memory from start12 and onward, possibly due to the algorithm's exponential memory requirements.

IDS (Iterative-Deepening Search): IDS is time inefficient and memory efficient. Its worst-case time complexity is $O(b^d)$, where b is its branching factor and d is its depth. In the table above, IDS timed out from start30 onwards possibly due to its exponential time complexity. The Space required for IDS is only $O(d)$, hence the algorithm did not run out of memory during the testing as we can see from the table above.

A* (A-Star): A* is time efficient and memory inefficient. The time complexity of A* is depending on heuristics. Its worst-case time complexity is $O(b^d)$, where b is its branching factor and d is its depth. Its space complexity is also the same. It stores all better paths to the memory, which is why we see that it ran out of memory from start30 and onwards.

IDA* (Iterative-Deepening A-Star Search): IDA* is both time efficient and memory efficient. Because it is a combination of IDS and A*, it has both advantages of IDS and A*. As we can see from the table above, IDA* has the best performance during our testing. IDA* does not store all better paths to the memory like A*, hence it is memory efficient. The time complexity of IDA* is depending on heuristics.

2.2

a)

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2 **1.4** **1.6**						
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),
    %F1 is G1 + H1
    W is 1.2,
    F1 is (2 - W)*G1 +  W*H1,
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

c)

	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**

d)

IDA*: IDA* is the slowest among the five algorithms. However, it produced the most optimal solution (generated the shortest path solution).

Heuristic: The speed and quality of the solution tend to vary for different values of *w*. A *w* increases, the time complexity of the algorithm is faster, but the quality of the solution gets worse.

Greedy: Greedy search produced the least optimal solution (longest path solution). However, it has the best speed among the five algorithms.