Part 2 – Search

Yiting Liu_z5211008

Question 1: Search Algorithms for the 15-Puzzle

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: (Time inefficient and memory inefficient)
From the table in a), it's can be found that UCS is the worst, for it runs out the memory at start12, 20, 30 and 40. That's because while the UCS is trying to find the optimal path to solve the puzzle, it needs to take a lot of time to expand a lot of nodes and keep all better paths in the memory. In this way, UCS is time inefficient and memory inefficient.

IDS: (Time inefficient and memory efficient)
Although both of UCS and IDS are uninformed search, IDS has better memory efficiency than UCS. That's because IDS just store the optimal path to memory. However, IDS still has some problems of time efficiency, for it takes a long time at start30 and 40. From the table in a), it can be found that the time taken of IDS grows exponentially.

A *: (Time efficient and memory inefficient)
From the table in a), it's shown that A *Search runs out the memory at start 30 and 40. That's because of A* search store all better paths to the memory like UCS, so it won't have enough memory while solving complex puzzles, which is memory inefficient. However, as we can see, the number of nodes generated by A* is much less than UCS and IDS, so it's much quicker and efficient in time. The time complexity of A* is depending on heuristics.

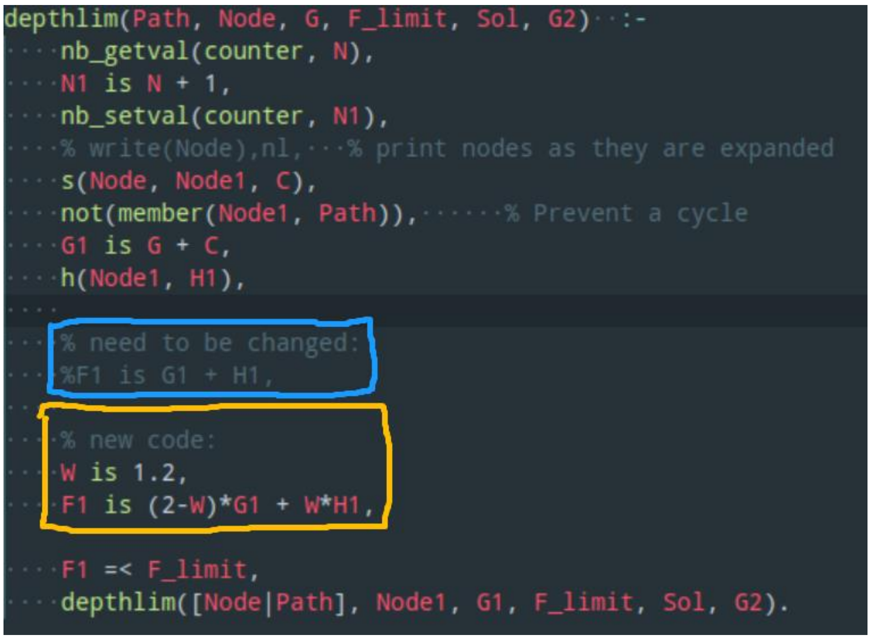
IDA *: (Time efficient and memory efficient)
Because IDA* is a combination of IDS and A*, so it has both advantages of IDS and A*, which is time efficient and memory efficient. IDA* doesn't store all better paths to the memory like A*, so it is memory efficient. As both of IDA* and A * are informed search, so the time complexity of IDA* is depending on heuristics as well.

Question 2: Heuristic Path Search for 15-Puzzle

a) & 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

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),

    % need to be changed:
    %F1 is G1 + H1,

    % new code:
    W is 1.2,
    F1 is (2-W)*G1 + W*H1,

    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

d)

According to tutorial 3, IDA* Search has the same performance with w = 1, and Greedy Search has the same performance with w = 2. As we can see from the table above, which w increases from 1 to 2, the time taken become less and less, but the number of nodes generated become more and more. Therefore, while w is getting bigger since 1 to 2, the speeds of algorithms are getting faster, but the qualities of solutions are getting worse.