Part 2 - Search

By John Dao (z5258962)

Completed with references to lecture material

Question 1: Search Algorithms for the 15-Puzzle

a)

	Start 10	Start12	Start20	Start30	Start 40
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)

1. UCS (uniform cost search) is seen as the poorest of the four algorithms as given the 5 start positions, it was only able to complete start10 with all others resulting in MEM errors. UCS attempts to find the optimal solution which is the minimum cost. Space grows exponentially and this results in UCS' MEM error as start grows in size.

   Cost of a path is the sum of the costs of its arcs:

   $$cost(< n_0, \ldots, n_k >) = \sum_{i=1}^{k} cost(< n_{i-1}, n_i >)$$

   Space Complexity (worst case):

   $$O(b^{[C^*/e]}),\ b^{[C^*/e]} = b^d$$

   Time Complexity (worst case):

   $$O(b^{[C^*/e]})$$

   Where
   C* = Cost of optimal solution
   E = transition cost

2. IDS (Iterative deepening search) can be seen as an improvement from UCS. IDS attempts to combine depth-first and breadth-first search methods to obtain an optimal and complete solution. As it was able to run start10, start12 and start 20, but failed

start30 and start40 due to TIME it's improvement can only be said as slight. While it is possible to solve the start in higher numbers, it takes a while and this is reflected in its exponential time complexity.

Space Complexity (worst case):

$$O(bd)$$

Time Complexity (worst case):

$$O(b^d)$$

3. A* can be seen as an improvement over IDS. It attempts to use both cost of path generated and estimate of goal to order nodes. As IDS incurs TIME instead of MEM error for start 30 and start 40 and has a much improved path length, its improvement can be described as said as substantial over IDS. As A* utilises a priority queue, MEM errors can occur as the search space increases and is evident in the cases for start30 and start40.

   Space Complexity (worst case):

   $$O(b^d)$$

   Time Complexity is dependant on the heuristic function $f(n) = g(n) + h(n)$

   $$O(b^m) \text{ or } O(b^{\epsilon d})$$

   (worst case where space is a tree, time complexity is polynomial)

4. IDA* can be seen as the best algorithm out of the four. IDA* attempts to use a heuristic function with A* to calculate the cost to get to the goal state. As IDA* has successfully calculated all start functions, with marginal improvements in start10 and start12 and a marginal regression in start20, it can be considered a good improvement over A*. Due to the depth limited upon $f(n) = g(n) + h(n)$, it does not incur memory errors like A* and can continue the search, ultimately completing start30 and start40.

   Space Complexity (worst case):

   $$O(bd)$$

   Time Complexity is dependant on the heuristic function $f(n) = g(n) + h(n)$

$O(b^m)$ *or* $O(b^{\epsilon d})$

(worst case where space is a tree, time complexity is polynomial)

Question 2: Heuristic Path Search for 15-Puzzle

a) and 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)

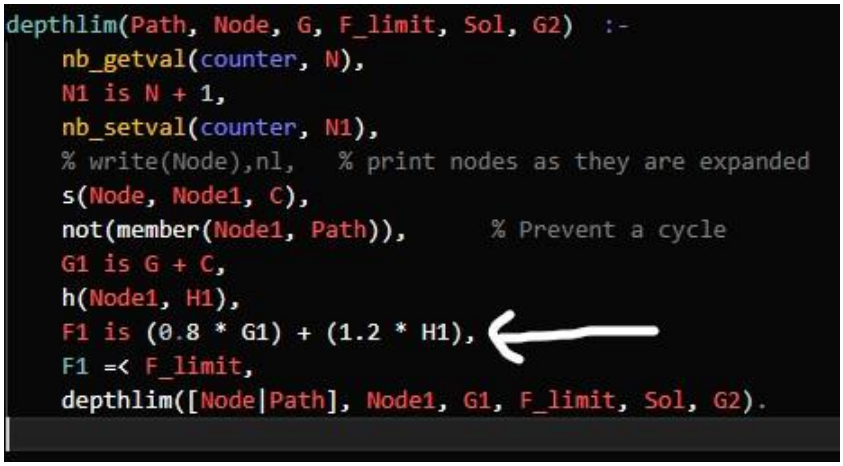
Given the objective function:

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

Where w = 1.2

We can replace the following code like below;

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



d)

In the comparison of these 5 algorithms, it can be established that there is a tradeoff between speed and quality which can be represented as w moves towards 2.We can see this when we compare w = 1 and w = 1,2.

In w = 1, which is utilised in IDA*, We can see that there is a high quality of solution, where the path length is shorter, but lower performance in comparison to w = 1.2 where the solution is of a lower quality, where the path length is longer, but is indeed quicker in speed. The trend is much the same when we increment towards w = 2 which is utilised in [greedy] where in that case, the highest speed is achieved, but the quality of the solution is poorer than what was achieved in w = 1 (IDA*).

We can see this when we actually substitute the values into the objective function where
For w = 1 (IDA*):

$$f(n) = (2 - 1) \cdot g(n) + 1 \cdot h(n) = g(n) + h(n)$$

And for w = 2

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

In this comparison, we can see the correlation between speed and quality in terms of w, where in two ends of the spectrum,

- IDA* represents g(n) + h(n) in its simplest form (higher quality)
- Greedy represents h(n) in its simplest form (lower quality)

Thus as w increments towards 2 from a baseline of 1 in IDA*, we discover that there is a tradeoff of quality for an increase in performance.