COMP3411 : Assignment 2

Kevin Luo z5061845

1 Question 1: Search algorithms for 15-puzzle

1.1

Table 1: States expanded

	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	186115

1.2

- **UCS** UCS is the worst algorithm. It exceeded its memory at start12. **Slow**
- **IDS** This performed slightly better than UCS but its memory also grew exponetially. However it never ran out of memory, instead its time expired. **Slow**
- **A*** A* has a very similar time to IDA* up to start20, afterwards, its memory usage blows up and is unable to calculate larger values. **Medium Speed**
- **IDA*** It is clear that IDA* is the most efficient in terms of time and memory usage, due to it being the only one to be able to calculate past start30. **Medium Speed**

2 Heuristic Path Search

2.1

Table 2: Search Path

	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	2358520
Greedy	164	5447	166	1617	184	2174

2.2

```
1  depthlim(Path, Node, G, F_limit, Sol, G2) :-
2      nb_getval(counter, N),
3      N1 is N + 1,
4      nb_setval(counter, N1),
5      % write(Node),nl,   % print nodes as they are expanded
6      s(Node, Node1, C),
7      not(member(Node1, Path)),       % Prevent a cycle
8      G1 is G + C,
9      h(Node1, H1),
10     % F1 is G1 + H1,
11     W is 1.2,
12     F1 is (2-W)*G1 + W*H1,
13     F1 =< F_limit,
14     depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

Line removed is line (10) and the lines added are (11-12). These introduce a new var W, and is used to modify F1 depending on the equation given.

2.3

Refer to table in 2.1

2.4

- **IDA*** Slowest algorithm by far. It only adds the cost of the heuristic. Has quickest path.
- **1.2** We implemented the Heuristic Path Search with an Iterative Deepening solution. It is able to generate the graph fairly quickly while its solution is just below optimal.
- **1.4** Same as above, runs faster than 1.2 and has fewer nodes expanded but its solution is no longer optimal.
- **1.6** Same as 1.2, with even fewer nodes expanded but has very bad optimal path.
- **Greedy** Fastest solution while generating the least amount of nodes but it resulted in the longest path solution.
- **Outlier** start60 produced some weird solutions where the nodes expanded were less than their start 50 cousins.

3 Maze Search Heuristics

3.1

Manhattan heuristic

$$h(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$$

3.2

3.2.1

No, a heuristic must always underestimate or be equal to the cost of a goal. Below is a counter example where the quickest route with diagonal movement is 2 moves but the straight line heuristic gives:

$$\sqrt{2^2 + 1^2} = \sqrt{5} = 2.23$$

Table 3: Counter example

		G
S		

3.2.2

No. The Manhattan heuristic is no longer admissable in the case of diagonal movement due to the pythagoras theorem - the root of the two squared sides is longer than the hypothenus.

3.2.3

Chebyshev distance

$$h(x, y, x_G, y_G) = max(|x - x_G|, |y - y_G|)$$

4 Graph Paper Grand Prix

4.1

Table 4: My caption

n	Sequence	Number of actions
1	+-	2
2	+o-	3
3	+oo-	4
4	++- -	4
5	++-o-	5
6	++o- -	5
7	++o-o-	6
8	++oo- -	6
9	+++- - -	6
10	+++- -o-	7
11	+++-o- -	7
12	+++o- - -	7
13	+++o- -o-	8
14	+++o-o- -	8
15	+++oo- - -	8
16	++++- - - -	8
17	++++- - -o-	9
18	++++- -o- -	9
19	++++-o- - -	9
20	++++o- - - -	9
21	++++o- - -o-	10

4.2

Let Number of Actions (NoA) from the table above be the length of the sequence. NoA follows the given identity where s is the maximum speed - number of +. This holds true for $s(s+1)$ no matter how many o's (rests) are present. We then have the identity s^2 for the rest of the sequences, ie when n is a perfect square.

4.3

We start with some velocity $k \geq 0$ and a destination n such that $n \geq \frac{k(k-1)}{2}$. We then have that the deceleration time for $M(n, 0)$ would be the total distance up to some x, minus the time it too to accelerate to k:

$$M(x, 0) = \lceil 2\sqrt{x} \rceil - k$$

We have that x is the total distance to accelerate to k.

$$\frac{k(k+1)}{2} + n$$

Substituting into the original at x, k, we have:

$$M(n, k) = \left\lceil 2\sqrt{n + \frac{k(k+1)}{2}} \right\rceil - k$$

4.4

Similar to 4.3, we have some velocity $k \geq 0$ but a destination n such that $n < \frac{k(k-1)}{2}$. Our destination has extended over the goal and we have to reverse back towards G. We have $x > n$ such that
$M(n, k)$ = time extended past goal + time to reverse - time to accelerate (k).
This gives us:

$$M(n, k) = \left\lceil 2\sqrt{\frac{k(k+1)}{2} + \frac{k(k-1)}{2}} \right\rceil + \left\lceil 2\sqrt{\frac{k(k-1)}{2} - n} \right\rceil - k$$

$$M(n, k) = \left\lceil 2\sqrt{\frac{k(k-1)}{2}} \right\rceil + k$$

4.5

Similarly to the Chebyshev distance but without the absolute values since we can have negative acceleration , we have:

$$h(r, c, u, v, r_G, c_G) = \max(M(r_G - r, u), M(c_G - c, v))$$

where M is the equation from 4.3