**Assignment 2**

**Question1:**

1. **:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **start 10** | **start 12** | **start 20** | **start 30** | **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 |

*Table 1*

1. **:**

**UCS(uniform cost search):**

As can be seen from Table 1, UCS requires more memory than other methods in solving 15-Puzzle. In fact, its space and time complexity are and , respectively.

**IDS(iterative deepening first search):**

It is obvious that, IDS also occupies a lot of spaces, only less than UCS. In addition, its time complexity is also high. Its space complexity is and time complexity is . According to formula, the IDS utilizes less memory than UCS, however, the time complexity of IDS is same as UCS to some extent.

**A\*(A\* search):**

A\* occupies far less memory than UCS and IDS. It combines the advantages of UCS and greedy search. Therefore, its formula is . However, memory usage is still a concern for A\*. This point could be seen from start30 and start40 in Table 1. For time complexity, A\* is faster than UCS and IDS obviously.

**IDA\*(iterative deepening A\*):**

IDA\* is a low-memory variant. It occupies less space than A\* although the time it consumed may be lightly more than A\*.

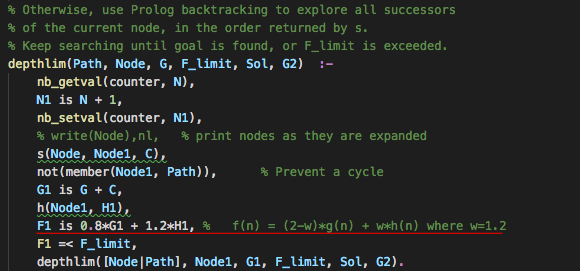
**Question2:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | start50 | | start60 | | start64 | |
|  | G | N | G | N | G | N |
| IDA\* | 50 | 14642512 | 60 | 321252386 | 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 |

*Table 2*

**(a):**

The answer is shown in the row named Greedy in Table 2.

**(b):** 

*Figure 1*

From figure 1, F1 = G1+H1 has been changed into F1 = 0.8\*G1+1.2\*H1. Actually, this code delegates the evaluation function:, where . In this code, the function is modified from into .

**(c):**

The results are demonstrated in Table 2.

**(d):**

When w=0, the function is greedy search. When w=1, the function is A\* search. From the data in Table 2, we could know that the length of path would increase and the total number of states expanded would decrease with the increase in w. This means that the quality is worse, however, speed is quicker with the growth in w.

**Question3:**

**(a):**

The heuristics is the Manhattan distance of the maze. It delegates the least total steps required from current position to Goal position. Also, compared with , .This means that dominates and would produce less or equal number of nodes compared with .

**(b):**

(i): The Straight-Line-Distance heuristic is not admissible. This is because the step to diagonal position is still regarded as 1 step. However, if we arrived the same position employing Straight-Line-Distance heuristic, it would take steps. Therefore, the is more than the where is the true cost from current position to goal. The figure2 blow shows the consequence.

|  |  |
| --- | --- |
| Current  position | 1 |
|  | 1  Goal |

***Figure 2***

(ii): The , as a result, the cost consumed by is more than . This is because would take 2 steps compared to consuming 1 step. For this reason, is not admissible.

(iii):

When , the most cost consumed is . When , the most cost consumed is max(). Figure 3 shows different conditions for the heuristics.

X axis X axis

Y axis

Y axis

|  |  |  |
| --- | --- | --- |
| P \* | \* | \* |
|  | \* | \* |
|  |  | G\* |

|  |  |  |  |
| --- | --- | --- | --- |
| P \* | \* | \* | \* |
|  |  | \* | \* |
|  |  |  | G\* |
|  |  |  |  |

***Figure3***

**Note: The red square delegates the actual path.**

**Question4:**

****

***Table 3***

**(a):**

As can be seen from Table 3, M(n, 0) for 1 ≤ n ≤ 21 is shown by writing down the optimal sequence of actions for all n between 1 and 21.

**(b):**

In fact, . It follows the principle below:

**where s is the maximum speed from S position to n position.**

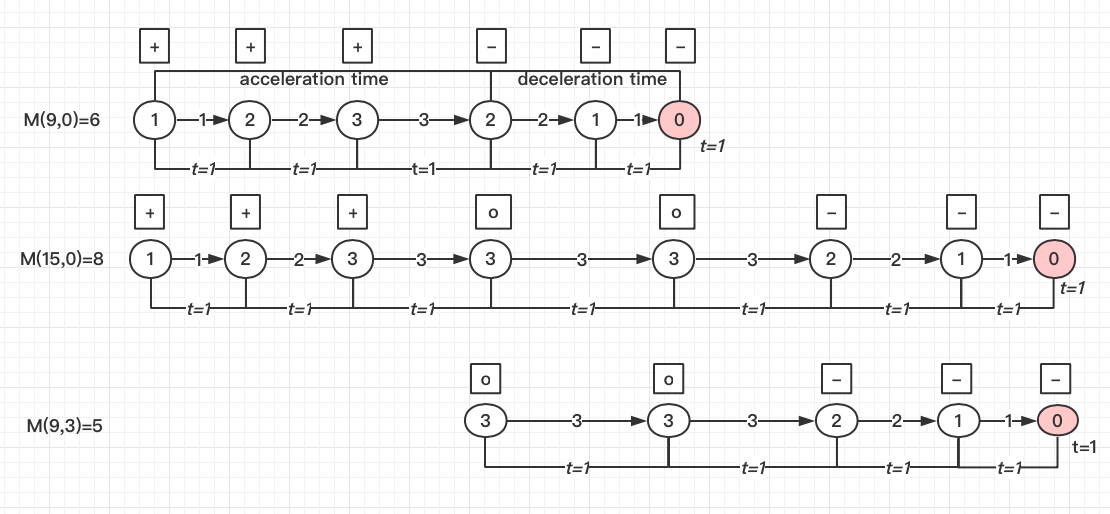
Firstly, I assume . This means in the optimal sequence, velocity keeps change and [o] is not in the optimal sequence. Then the path length. Fastest time is equal to 2s. Next, I assume , it is observed that I only need add a step that length is equal to 1 into the total distance of . During the process, time is equal to 2s+1. Similarly, when , the time consumed is only 1 second more than . Therefore, when , fastest time is . Now, we assume . This means that we only add a step that length is no more than s into . However, the time consumed by this step is also 1 second. Therefore, the fastest time takes second. When , it would return initial state which is and its algorithm logic is same as , so the time consumed is . For this reason, I revised this formula:

Now, the formula should be appropriate for all cases.

**(c):**

If we regard every move from velocity 0 to velocity k which is maximum speed as an arithmetic progression, the total distance of acceleration would be . It could become by simplifying. When the vehicle arrived firstly at velocity k, we add the total distance of acceleration into the total distance from S to G. This means we start to move again from S to G at velocity=k. This the time consumed in process is same as . Therefore,

The figure 4 below show the process.



***Figure 4(The circle with red is G)***

**(d):**

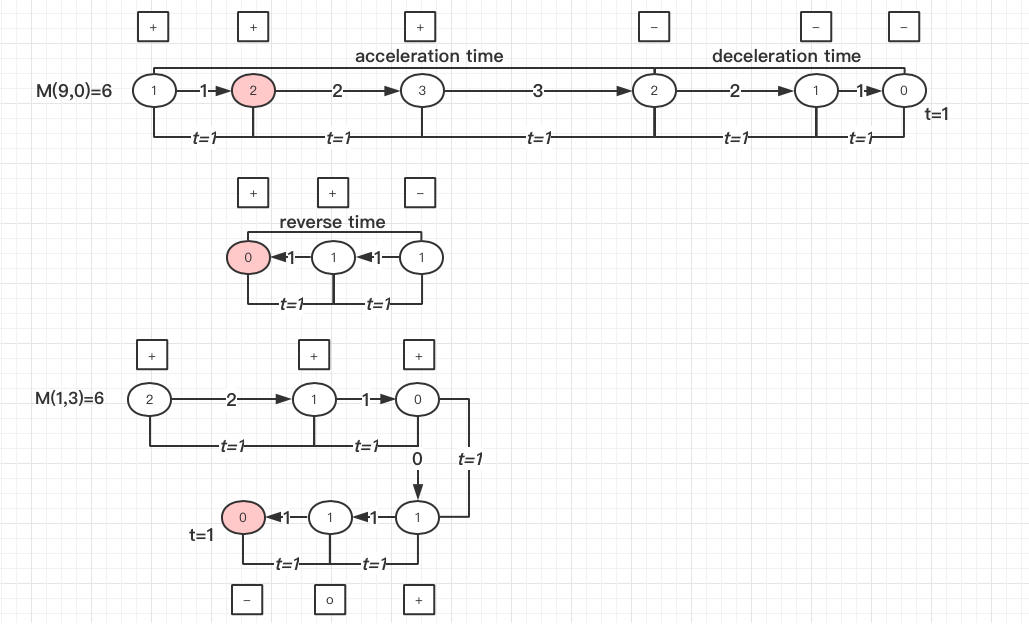
when , it means G is in the place at speed firstly . In order to satisfy the requirement that velocity is 0 at G, a path reverse is necessary.

.

The distance from k=1 to maximum speed k and the distance from maximum speed to k=0 are regarded as two arithmetic progressions, respectively. They are acceleration distance: and deceleration distance:. Reverse time:.

Therefore,

The figure 5 shows the process.



***Figure5(The circle with red is G)***

(e):

This heuristic is admissible. This is because when the speed of vertical or horizontal direction arrived 0, we need only consider whether the speed of other direction reaches 0. Therefore, maximum time taken by or is reasonable.