Z5174741

GUANQUN ZHOU

Question1

Search Algorithms for the 15-Puzzle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Start10 | Start12 | Start20 | Start30 | Start40 |
| UCS | 2565 | Mem | Mem | Mem | Mem |
| IDS | 2407 | 13812 | 5297410 | Time | Time |
| A\*(Man) | 33 | 26 | 915 | Mem | Mem |
| IDA\* | 29 | 21 | 952 | 17297 | 112571 |

(b)

Based on this result ,we can conduct that Informed search strategy is more efficient than uninformed search strategy.

We can find that UCS IDS as uninformed search algorithms are limited by memory and computing time. And UCS has lowest efficiency and highest space complexity.

IDS performed better , but it is still limited by computing time. IDS will run out of time when start30 and start40,and IDS has 5297410 nodes when start20,which is too expensive.

A\* is a good informed search strategy, it can get the result more quickly than UCS and IDS when start10,start20,start30.but it will cost too much memory when start 30 and start 40.

IDA\* seems to be the best algorithm for puzzle problems, It boost A\*, saving the memory, so it can get all the result in the table.

On the other hand, the quality of heuristic function could effect the performance of algorithm. In this case, Manhattan distance is more efficient than misplaced distance, so when we use both distance in IDA\* to compare the algorithms, we can find Manhattan distance as part of heuristic function performed better than misplaced distance.

Question2 Heuristic Path Search for 15-Puzzle

(a)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Start50 | | Start60 | | Start64 | |
| IDA\* | 50 | 1462512 | 60 | 321252368 | 64 | 1209086782 |
| 1.2 | G=52 | N=191438 | G=62 | N=230861 | G=66 | N=431033 |
| 1.4 | G=66 | N=116174 | G=82 | N=3673 | G=94 | N=188917 |
| 1.6 | G=100 | N=34647 | G=148 | N=55626 | G=162 | N=235852 |
| Greedy | G=184 | N=5447 | G=166 | N=1617 | G=184 | N=2174 |

(b)

Original code: F1 is G1 + H1

New code: F1 is (2-1.2)\*G1 + 1.2\*H1

(c)

As we can see from the above table

(d)

All of them are informed searches, and the different are the

f(n) = (2-W)·G + W·H. Which means the w in heuristic function.

The N is decreasing with W is increasing, which can help reduce memory expense and be benefit to speed.

However, the length of path is increasing with W is increasing, which means the result’s quality is going bad.

Actually, when the W increased to 2, it became Greedy algorithm with f(n)=h(n), and Greedy can’t get the optimal and we cannot get shorter distance in this algorithm.

User’s choice will decide the balance of speed and quality,and like w=1.4 or 1.6 could be balanced options.

Question3

Maze Search Heuristics

h(x,y,xG,yG)=|x-xG|+|y-yG|

Because it is not allowed to move oblique, suitable formula should be like this one.

(b)

(i)

I think use:( (Straight-Line-Distance)／ as heuristic function is admissible, because it allows to move oblique, so Euclidean distance will find out which path is the best one in Maze. In this case, the minimal cost of moves bases on moves’ number, so it should be 1 as minimum.

So h(x,y,xG,yG)= could be admissible.

(ii)

No. It is not the best heuristic function in 8 directions’ condition, so it never be admissible for part(a) heuristic. There is a solution path which never be better than (Euclidean distance) ／ .

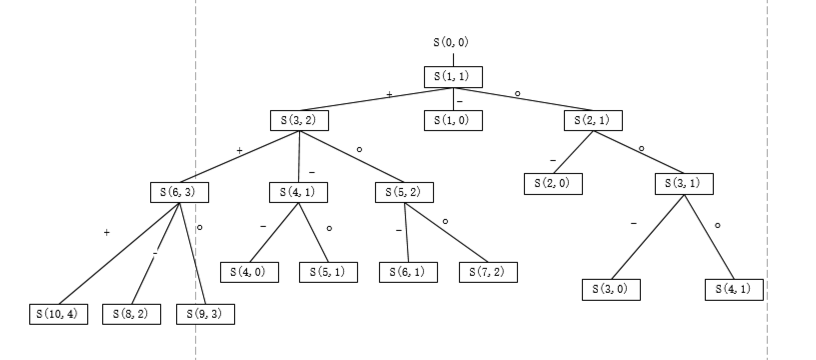
(iii)

Maze allows 4 directions of movement, use Manhattan distance(like (a)).h(x,y,xG,yG)=|x-xG|+|y-yG|. Maze allows 8 directions of movement, use Straight-Line-Distance

hSLD(x,y,xG,yG)=

If Maze allows every directions of movement and estimate the cost by calculating distance, Straight-Line-Distance would be very good as H function.

Question4

（a）

The logic process should be like this ,and in this way, we can find results we need ,and we can select the best method. Plus, we can also find the process of ‘+’, ‘\_’, ‘o’ and the number and order of each operation.So, the answer of 1 ≤ n ≤ 21 should be contained in the blew table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | M(n,0) | sequence | 2s+1 | 2s+2 |
| 1 | 2 | [+ -] |  | 2 |
| 2 | 3 | [+ o -] | 3 |  |
| 3 | 4 | [+ o o -] |  | 4 |
| 4 | 4 | [+ + - -] |  | 4 |
| 5 | 5 | [+ + - o -] | 5 |  |
| 6 | 5 | [+ + o - -] | 5 |  |
| 7 | 6 | [+ + o – o - ] |  | 6 |
| 8 | 6 | [+ + o o - -] |  | 6 |
| 9 | 6 | [+ + + - - -] |  | 6 |
| 10 | 7 | [+ + + - - o -] | 7 |  |
| 11 | 7 | [+ + + - o - -] | 7 |  |
| 12 | 7 | [+ + + o - - -] | 7 |  |
| 13 | 8 | [+ + + o - - o -] |  | 8 |
| 14 | 8 | [+ + + o - o - -] |  | 8 |
| 15 | 8 | [+ + + o o - - -] |  | 8 |
| 16 | 8 | [+ + + + - - - -] |  | 8 |
| 17 | 9 | [+ + + + - - - o -] | 9 |  |
| 18 | 9 | [+ + + + - - o - -] | 9 |  |
| 19 | 9 | [+ + + + - o - - -] | 9 |  |
| 20 | 9 | [+ + + + o - - - -] | 9 |  |
| 21 | 10 | [+ + + + o - - - o -] |  | 10 |

(b)

M(n,0) is the minimum number of time steps . n is the distance , and in this question: the range of change of vector each time is a set {−1, 0, +1}the least time used is: M(n, 0) = 2. M(n,0) is the minimum number of time steps ,so M(n,0) must be integer : M(n, 0) = ⌈2⌉. And we also can find that the conclusion is correct from the last two column of above table.

(c)

According to (b),we know that M(n,0) = ⌈2⌉. In this question, the original vector is K0, and n so we can consider this question in this way:

if n =, according to Arithmetic sequence summation formula, M(n,k) = k

if n >, we can assume that this path is a part of a larger path:

First step: keep accelerating to K, according to Arithmetic sequence summation formula, this S1 = ,and the number of steps is K.

Second step: keep the process with vector is K and target distance is n.

From (b),we know that M(n,0) = ⌈2⌉, so for the larger path, distance is S1+n, M(S1+n,0) = ⌈⌉. And accelerating process, number of steps is k,

so M(n,k)=M(S1+n,0)-k = ⌈⌉ - k

(d)

In this question: the original vector is K0, and n,

According to Arithmetic sequence summation: if keep slowing down, the distance should be , but n is less than .so the whole process should be like this:

First step: keep slowing down until k became 0,during the path, agent will go pass the n point, and stop at the point which is further than n point, and the extra distance is .

Second step: after stop at the further point, turn back move to n point with original vector is 0 and the target distance is , so the number of steps of this part is M( which is equal to ⌈2⌉. So the number of the whole process is M(n,k) = M(+k = ⌈2 + k.

（e）

h(𝑟, 𝑐, 𝑢, 𝑣, 𝑟G ,𝑐G )=max (M(𝑟G −𝑟,𝑢),𝑀(𝑐G −𝑐,𝑣))