COMP3702/COMP7702 Artificial Intelligence (Semester 2, 2020)

Assignment 1: Search in LaserTank – **Report Template**

**Name**: JIANWEI LI

**Student ID**: 45555408

**Student email**: Jianwei.li@uqconnect.edu.au

Note: Please edit the name, student ID number and student email to reflect your identity and **do not modify the design or the layout in the assignment template**, including changing the paging.

**Question 1** (Complete your full answer to Question 1 on the remainder page 1)

|  |  |
| --- | --- |
| **Dimension** | **Values** |
| **Modularity** | **Flat**  The game only has one level of abstraction,there is no organizational structure or interacting modules. |
| **Planning horizon** | **Indefinite stage**  The agent who wants to get to the goal does not know a priori how many steps it will take to get there. |
| **Representation** | **States**  Each of the different ways the grid could be, would affect what the agent should do next. |
| **Computational limits** | **Perfect rationality**  The agent generates an optimal path without taking into account its limited computational resources. |
| **Learning** | **Knowledge is given**  The knowledge needed to decide what to do is provided as part of the codes. |
| **Sensing uncertainty** | **Fully observable**  The agent knows the state of the grid from the code of Lasertankmap. |
| **Effect uncertainty** | **Deterministic**  The resulting state can be determined from the action that agent takes and the grid. |
| **Preference** | **Goals**  The position of flag is the goal to achieve. |
| **Number of agents** | **Single agent**  There’s only one agent taking actions here. |
| **Interaction** | **Offline**  All the states are generated on local machine, no need to interact with environment. |

**Question 2** (Complete your full answer to Question 2 on page 2)

**Action Space:** taking a set of actions [MOVE\_FORWARD, TURN\_LEFT, TURN\_RIGHT, SHOOT\_LASER]

A = ['f', 'l', 'r', 's']

**Percept Space:** grid data, coordinate of tank (player\_x, player\_y), heading of tank (player\_heading)

**State Space: S = P** since the game is fully observable

**Transition Function:** given a state **S** and performing an action **A** results in a new state **S’**.

e.g. **S**[grid\_data, (x, y), heading] x **A**[‘s’] = **S’**[new\_grid\_data, (x, y), heading]

**Utility Functions:** steps g(n) - each step costs 1 constantly

heuristic h(n) - Manhattan distance between starting node and goal node

**Question 3** (Complete your full answer to Question 3 on page 3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Nodes Generated** | **Nodes on Fringe When Terminates** | **Nodes Explored When Terminates** | **Run Time**  **(in seconds)** | **Test Cases** |
| **UCS** | 51305 | 408 | 19289 | 1.459 | **t1\_bridgeport** |
| **A\*** | 45108 | 480 | 16954 | 1.326 |
|  | | | | | |
| **UCS** | 12326 | 1 | 4685 | 0.366 | **t1\_crossfire** |
| **A\*** | 12326 | 1 | 4685 | 0.481 |
|  | | | | | |
| **UCS** | 2504 | 158 | 976 | 0.250 | **t1\_ice\_maze** |
| **A\*** | 353 | 43 | 133 | 0.121 |
|  | | | | | |
| **UCS** | 34644 | 2568 | 12647 | 1.181 | **t2\_brickyard** |
| **A\*** | 3146 | 351 | 1138 | 0.357 |
|  | | | | | |
| **UCS** | 48983 | 665 | 19406 | 2.122 | **t2\_puzzle\_maze** |
| **A\*** | 30967 | 511 | 12240 | 1.373 |
|  | | | | | |
| **UCS** | 24281 | 792 | 9547 | 1.392 | **t2\_shortcut** |
| **A\*** | 4665 | 233 | 1835 | 0.732 |
|  | | | | | |
| **UCS** | 8003 | 14 | 3201 | 2.717 | **t3\_labyrinth** |
| **A\*** | 5353 | 62 | 2135 | 1.750 |
|  | | | | | |
| **UCS** | 36691 | 391 | 14485 | 3.615 | **t3\_the\_river** |
| **A\*** | 18654 | 411 | 7338 | 1.864 |

**e)**

Both UCS and A\* algorithm generated an optimal solution in the case of that the solution exists.

UCS algorithm generated and explored more nodes than A\* algorithm, and also in most cases, UCS took more time compare to A\*.

But there’s one exception, when each node in a tree had very few branching factors, on the contrary, the UCS algorithm was faster than A\* algorithm, showed better performance. (e.g. the test case t1\_crossfire)

**Question 4** (Complete your full answer to Question 4 on pages 4 and 5, and keep page 5 blank if you do not need it)

Assume that f(*n*) = g(*n*) + h(*n*) is the utility function for all the map elements.

(where *n* is the next node on the path, g(*n*) is the cost of the path from the start node to *n*, and h(*n*) is a heuristic function that estimates the cost of the cheapest path from n to the goal, here is Manhattan distance.)

Since teleporter and ice tiles would let our tank span a greater distance, it’s possible to reach the flag faster, so the sooner we explore the node contains teleporter or ice the better.

1. h(*n*) = - g(*n*) , f(*n*) = g(*n*) - g(*n*) = 0 for teleporter tiles and ice tiles,

remain f(*n*) = g(*n*) + h(*n*) unchanged for the other map elements.

Although h(*n*) is not admissible, but efficient. As soon as the agent explores the parent node of child node that contains teleporter or ice, the child node will be pushed into the fringe with the highest – priority, and will be the first to be popped out from fringe, the first to be explored.

1. h(*n*) = 0 , f(*n*) = g(*n*) + 0 = g(*n*) for teleporter tiles and ice tiles,

remain f(*n*) = g(*n*) + h(*n*) unchanged for the other map elements.

The child node that contains teleporter or ice tile will be pushed into the fringe with higher – priority, and will be popped out from fringe and be explored sooner.

1. Divide the whole path into two subsegments by one of the teleporter pairs (depends on which one is closer),

the 1st half: f(*n*) = g(*n*) + h(*n*) for all map elements,

h(*n*) is Manhattan distance between **starting node** and **teleporter node**, treat the teleporter as a temporary goal node, upon arrival, save the 1st half path, then

the 2nd half: f(*n*) = g(*n*) + h(*n*) for all map elements,

h(*n*) is Manhattan distance between **the other teleporter node** and the final **goal** **node**, upon arrival, save the 2nd half path, then [the 1st half path] + [the 2nd half path] will be the whole path.

Plan the searching route manually, reduce the number of nodes explored.