Name: Pratik Jasani

Course: EN 605.645

**State Space Search Self Check**

1. **What is a state in this problem? How would you characterize it (what data is needed)? What is the set of states, S?**

* A state is a location that the robot can be at in the environment . A set of all possible locations in the environment that the robot can be at is called set of states. The data about the environment and the location of the robot is needed to define a state for this practice problem. The environment can tell us what the microworld looks like or if an action is valid or not.

1. **How is the set of transitions, T, related to the successors function? Do we need specify the set of transitions explicitly? If the robot can only move up, down, left or right by one square, would the successors function produce for the starting state, S?**

* The successor function returns the set of transitions. The set of transitions are the valid actions that the robot can take. The successor function will never produce the starting state again because after the initial step it will be marked as explored. We do need to specify the set of transition explicitly. If we have 4 transitions, we can say which transitions are blocked and which are not. We can also make sure that our successor function only returns transitions which are not blocked/closed.

1. **Draw the graph that is generated by your successor function/transitions set. Based on the**

**algorithm above, is the actual graph generated all at once or piecemeal?**

* The graph is not generated all at once. It searches only what is needed.
* For the below graph I followed the given algorithm. The algorithm terminates after finding the path, so the below graph ends after goal was found (3,3). Below D = down, R = right, L = Left, R = Right.

Start

R

D

D

R

R

U

R

R

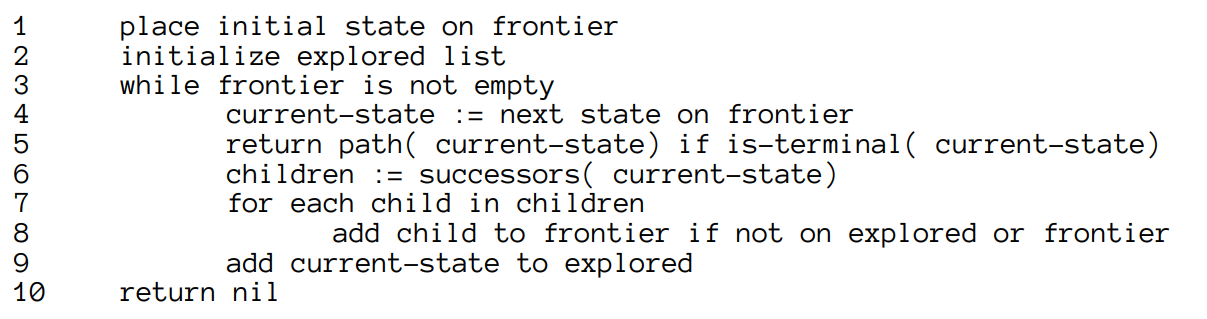
Goal

D

1. **Solve the above State Space Search problem of moving from S to G using the Depth First**

**Search (DFS) algorithm.**

* I used (column, row) notation below. At each state, I checked direction in following order below, right, up, left. This means the transition to the left of current state was added to frontier if it was valid (not blocked ) then transition above current state and so on.
* I followed the code shown below.



**The following steps will be taken by the DFS.**

Step 3

Step 1

Step 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | **\*** | x | x | o |
| **2** | o | o | o | o |
| **3** | o | o | x | G |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | **\*** | o | o | o |
| **3** | o | o | x | G |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | **\*** | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | o |
| **3** | o | o | x | G |

Step 6

Step 4

Step 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | o |
| **3** | **\*** | o | x | G |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | o |
| **3** | o | **\*** | x | G |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | O | o | o |
| **1** | o | x | x | o |
| **2** | o | **\*** | o | o |
| **3** | o | o | x | G |

Step 7

Step 9

Step 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | o |
| **3** | o | o | x | **\*** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | **\*** |
| **3** | o | o | x | G |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | **\*** | o |
| **3** | o | o | x | G |

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
| 0,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  |  |
|  |  |
| 0,1 |  |
| 1,0 |  |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  |  |
| 0,2 |  |
| 1,0 |  |

|  |
| --- |
| F |
|  |
|  |
|  |
| 0,1 |
| 1,0 |

|  |
| --- |
| F |
|  |
|  |
|  |
| 0,2 |
| 1,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
| 0,3 | 0,2 |
| 1,2 |  |
| 1,0 |  |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
| 1,3 |  |
| 1,2 |  |
| 1,0 |  |

**blue = current state, red = explored state, green = states in the frontier**

Step 4

Step 3

Step 2

Step 1

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
| 0,3 |
| 1,2 |
| 1,0 |

Step 5

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
| 1,3 |
| 1,2 |
| 1,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
|  | 1,3 |
| 1,2 |  |
| 1,0 |  |

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
|  |
| 1,2 |
| 1,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
|  | 1,3 |
| 2,2 | 1,2 |
| 1,0 |  |

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| 2,2 |
| 1,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
|  | 1,3 |
|  | 1,2 |
|  | 2,2 |
| 3,2 |  |
| 1,0 |  |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
|  | 1,3 |
|  | 1,2 |
|  | 2,2 |
| 3,3 | 3,2 |
| 1,0 |  |

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| 3,2 |
| 1,0 |

Step 8

Step 7

Step 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | S | o | o | o |
| **1** | o | x | x | o |
| **2** | o | o | o | o |
| **3** | o | o | x | **\*** |

Final Path

Step 9

|  |
| --- |
| F |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| 3,3 |
| 1,0 |

|  |  |
| --- | --- |
| F | E |
|  | 0,0 |
|  | 0,1 |
|  | 0,2 |
|  | 0,3 |
|  | 1,3 |
|  | 1,2 |
|  | 2,2 |
|  | 3,2 |
| 1,0 |  |

Note: The location 3,3 is our goal.

Path = (0,0) -> (0,1) -> (0,2) -> (0,3) -> (1,3) -> (1,2) -> (2,2) -> (3,2) -> (3,3)

1. **Is this the optimal (lowest cost) path? What is the optimal path?**

* Each pair is in format (column, row). My path is (0,0) -> (0,1) -> (0,2) -> (0,3) -> (1,3) -> (1,2) -> (2,2) -> (3,2) -> (3,3). This is not the optimal path.
* There are 2 optimal paths.
  + Path 1: (0,0) -> (1,0) -> (2,0) -> (3,0) -> (3,1) -> (3,2) -> (3,3)
  + Path 2: (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (3,2) -> (3,3)

1. **Notice the fold at (0, 2) -> (0, 3) -> (1,3) -> (1, 2) -> (2, 2) instead of going directly from (0, 2) to (1, 2). What caused this? Was it DFS or the order that we placed successor states on the frontier? Is there a different order we can use to place successors on the frontier to fix this problem? Will it work for all placements of G?**

* The path going directly from the (0,2) to (1,2) was taken because of the order that we placed the successor states on the frontier. I chose to put the successor in the frontier in order left, up, right, down. This means that top of the frontier will be the successor that was found below the current state. If this order was different then we would have seen a different path.
* Let’s say we are taking the order right, bottom, left, up. This order will help make sure that we move to (1,2) from (0,2). This should work for all placements of G.

1. **If we are in Case 1, we did nothing that enabled us to recover the path (no bookkeeping). What additional information should we have saved to recover the path?**

* We should have kept track of the path from the start state to the current state.