Module 1 Self-Check

605.445 – Artificial Intelligence - Butcher

The purpose of this self-check is to prepare you for the programming assignment.

State Space Search

State Space Search plays a fundamental role in "Good Old Fashioned AI" (GOFAI). State Space Search is typically defined in terms of States, Actions, Transitions and (possibly) Costs. Additionally, for *informed* search you will need a *heuristic* function.

The challenge when solving a real world problem using State Space Search is to identify all of these components, some of which will be *implicit*. Additionally, your program may need to store *metadata* about states as the algorithm progresses.

Consider the general pseudocode for State Space Search (using *graph* search):

```
1       place initial state on frontier
2       initialize explored list
3       while frontier is not empty
4               current-state := next state on frontier
5               return path( current-state) if is-terminal( current-state)
6               children := successors( current-state)
7               for each child in children
8                       add child to frontier if not on explored or frontier
9               add current-state to explored
10      return nil
```

Practice

Imagine we have the following raw data for a "microworld" that a robot must navigate:

```
  0 1 2 3
0 S o o o
1 o x x o
2 o o o o
3 o o x G
```

where "S" is a starting state, "G" is a goal state, "o" is an open position and "x" is a closed/blocked position. The robot's movement is limited to left/right/up/down (no diagonal moves).

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

2. 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?

3. 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?

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

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

6. 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 successor on the frontier to fix this problem? Will it work for all placements of G?

7. We did nothing that enabled us to recover the path. What additional information should we have saved to recover the path?

You should show all your work. For each step, show the frontier (also called the *fringe*) and the explored list. Enumerate the states of the path in order and show the path on the world using "*".

As you complete this assignment, think about the data structures, algorithms and metadata you would need to implement this as a Python program.

Additional Questions

These questions are not required but it will be worth your while to work through them and/or discuss them in your Discussion Groups (although there are no Discussion Groups the first week).

1. What changes would need to be made to make this Breadth First Search? What path would it find?

2. What changes would need to be made to make this Greedy Search? What path would it find? (Assume an action cost of 1).

3. What changes would need to be made to make this A* Search? What path would it find? (Assume an action cost of 1).