Homework 2: Searching

COEN166/266, Spring 2016

Description

For this assignment, you will implement an A* search algorithm in scheme.

Consider an agent occupying a single location in a 10x10 grid. This grid is comprised of three types of spaces:

Empty (`'empty`)
Barrier (`'barrier`)
Goals (`'(goal <value>)`)

For example, consider this grid:

E	E	E	E	E	E	E	E	E	E
B	E	B	B	E	E	B	B	B	E
B	E	B	B	E	B	B	E	E	E
B	E	B	B	E	B	B	E	B	E
B	E	G 100	B	E	B	B	G 1000	B	E
B	E	B	B	E	B	B	B	B	E
E	E	E	B	E	E	E	E	E	E
E	B	B	B	B	B	B	E	B	B
E	B	B	B	B	B	B	E	E	E
E	E	E	E	G 20	B	B	B	B	E

The grid would be represented by the following Scheme input:

```
((empty empty empty empty empty empty empty empty empty empty)
 (barrier empty barrier barrier empty empty barrier barrier barrier empty)
 (barrier empty barrier barrier empty barrier barrier empty empty empty)
 (barrier empty barrier barrier empty barrier barrier empty barrier empty)
 (barrier empty (goal 100) barrier empty barrier barrier (goal 1000) barrier empty)
 (barrier empty barrier barrier empty barrier barrier barrier barrier empty)
 (empty empty empty barrier empty empty empty empty empty empty)
 (empty barrier barrier barrier barrier barrier barrier empty barrier barrier)
 (empty barrier barrier barrier barrier barrier barrier empty empty empty)
 (empty empty empty empty (goal 20) barrier barrier barrier barrier empty))
```

The agent will always start in location 0,0 (the bottom left-hand corner of the map), facing upward (towards positive Y coordinates). It will be given a map of the environment and the agent's starting energy level. For any single move, the agent can perform the following actions, each of which has an associated cost:

`MOVE-1`	(costs 10 units of energy)
`MOVE-2`	(costs 15 units of energy)
`MOVE-3`	(costs 18 units of energy)
`TURN-RIGHT`	(costs 5 units of energy)
`TURN-LEFT`	(costs 5 units of energy)

The agent cannot move through a barrier, cannot move off of the map, nor can it move through a goal state.

The performance metric will be: energy remaining + value of goal. In other words, if an agent starts with an energy level of 100 and uses 30 units of energy to reach a goal with a value of 50, its overall score would be 120 (70 energy remaining + 50 points for the goal reached). The search algorithm should not return a path to a goal that cannot be reached without running out of energy.

Your A* algorithm must include a heuristic function that optimistically estimates the cost remaining to reach a goal. It should take into consideration (in some form) the number of moves and number of turns that are required to reach the goal. How you use these is up to you, but keep in mind that the heuristic should balance accuracy and efficiency. Particularly clever approaches may merit extra credit.

Assignment Variants

For this assignment, there will be three options:
1. Implement an A* tree search algorithm.
2. Implement #1, **plus** an A* graph search algorithm.
3. Implement #1, #2, **plus** an IDA* or an A* bream search algorithm (your choice).

Option 1 will receive a grade of up to 90% of the full credit possible.
Option 2 will receive a grade of up to 100% of the full credit possible.
Option 3, if completed, will receive extra credit.

API

For option 1, implement the following functions:

`heuristic-function`

Parameters

Takes two parameters:
1. A 10x10 environment map (as described above)
2. The current state. This should be a list containing: the X and Y coordinates of the agent, plus the direction as a symbol (N, S, E, or W), not a string. For example the starting state would be represented as: `(0 0 N)`

Returns

An optimistic estimate of the cost for the agent to reach a goal state from that position.

`a*-tree-search`

Parameters

Takes two parameters:
1. A 10x10 environment map (a list of 10 lists, each comprised of 10 elements)
2. A value indicating the agent's current energy level

Returns

A list of steps (in order!) to reach a goal state with a maximum performance metric score. **These should be the actions described above, as strings.** If no goal state is reachable with the available energy, returns false.

For option 2, also implement:

`a*-graph-search`

The inputs and outputs should be identical to the A* tree search. Its only difference should be internal. The A* graph search will maintain a list of visited states, and will not re-consider those states once they have already been visited.

For option 3, also implement **one** of the following:

`ida*-tree-search`

Has the same parameters and return value as the other A* search functions, but performs a depth-first search of the space, limited by the lowest A* value not seen in the previous search iteration.

`a*-beam-search`

Has the same parameters and return values as the previous A* searches, but also accepts a parameter indicating the maximum size of the frontier.

Documentation

Include a brief description of the algorithm that your heuristic function uses to estimate the cost to goal. Do not include things that the code submitted does not do (I am not looking for information on what ideas you discarded or what you wanted to implement, but didn't).

Due Date

Monday, May 2nd

Grading

If you want to get full credit on this assignment, be sure to:

- Follow the API provided exactly.
- Submit your code and the explanation of your heuristic to Camino.
- Try the examples provided to make sure they work.
- Test your code with other examples that you write (these don't need to be turned in).
- Make sure your code works on the design center workstations.
- Comment your code.
- Indent your code for readability.