CS 271 - Project 1 README

First Semester, AY 2018 - 2019

Tabios, Karlo Justin V.
Wenceslao, Stephen John Matthew C.

ISLAND-HOPPING AGENT

PEAS Description

Performance Measure

- Agent reaches specified goal

Environment

- Multiple islands with connections among each other, represented by a directed graph through LinkedLists

Actuators

- Agent can move to another island as if it is migrating or “island-hopping”
- Agent has 8 directions to move by standing on an island: "north", "north west", "west", "south west", "south", "south east", "east", and "north east"

Sensors

- Agent can see the adjacent islands from any island it is located
    - Note: not all directions from 1 island’s perspective have an adjacent island (an island may have at least 1 connected island, up to 8 connected islands to itself)

How it works

1. `IslandHopper.java` contains the main function of the program. When run, an `input.txt` file is initially read by the program. The input file should specify the algorithm to be used (i.e., BFS or IDS), the initial state, the goal state, and the “islands.”
    a. In the islands section of the input file, each row corresponds to the single island. The number of rows determine how many islands there are in the environment. In the program, there are eight islands by default. This number is arbitrarily chosen.
    b. In each row, there are numbers ranging from -1 to 7. Any number greater than or equal to 0 and less than 7 denote a connection with another island of that number or index. For example, in the first island (row 1, zeroth island) has connections to the second (row 2, first island), the third, and the fourth island.
        i. The order in which each of the numbers in the rows is located is important since they will dictate the direction that has be taken to get

to that island. Reading the numbers from left to right, the corresponding direction for whatever number is as follows: "north", "north west", "west", "south west", "south", "south east", "east", and "north east".

ii. Example for zeroth island as shown in the `input.txt` file:
"`-1 -1 1 2 3 4 -1 -1`"
Verbalizing this notation, we say that the agent can go to island 1 by moving west, go to island 2 by moving south west, go to island 3 by moving south, and go to island 4 by moving south east.

c. In each row, any number equal to -1 denote a disconnection or an absence of a connection from an island. Therefore in the above example given "`-1 -1 1 2 3 4 -1 -1`" , the agent cannot go to north (which is the 1st number "`-1`") since there is not island to go to, and so on.

2. After reading the input.txt file, the program now creates the directed graph using the `DirectedGraph` class, and the agent using the `RouteFindingAgent` class. The strategy parameter is passed to the agent, so it knows whether to use BFS or IDS at any given moment.
3. BFS or breadth-first search works by expanding all the internal nodes or leaves of the current level. By using a fringe to goal-check each element, the agent performs a "shallow" checking of the goal at each level. The fringe is filled up by all the nodes at each level with valid integer values corresponding to the connections (i.e., values between 0 and `size`, inclusive).
4. IDS or iterative deepening search works by expanding first the left hand-side of the tree traversal until a given depth bound, from which it would backtrack and expand the rest of the ancestors' children until the depth bound. If the goal is not found on the initial run with the initial depth bound, the depth bound is incremented by a certain number. IDS will be performed again until it reaches the goal, this time with an increased depth bound.