ECE 646/ COMP 465: Artificial Intelligence (Spring 2019) Programming Assignment 1 (Max Points: 100)

Assigned Saturday February 23, due midnight Saturday March 9. Turn in your code using the canvas. Please comment your code extensively so we can understand it, and use sensible variable names.

In this assignment, you will (i) familiarize yourselves with SEPIA and (ii) implement the A* search algorithm for pathfinding in SEPIA. SEPIA is based on Warcraft, a real-time strategy (RTS) game (though we will not use the real-time aspects in these assignments). RTS games generally have "economic" and "battle" components. In the economic game, the goal is to collect different types of resources in the map. Typical resources are "Gold" and "Wood." Resources are collected using units called "Peasants." Having resources allows the player to build other buildings (Peasants can be used to build things) and produce more units. Some of these units are battle units that can be used to fight the opponent. Games generally end when one player has no more units left; however, in SEPIA, a variety of victory conditions can be declared through XML configuration files. For example we can declare a victory condition to be when a certain amount of Gold and Wood have been collected, some number of units of a certain type built, etc.

On the website/Canvas I showed you Lecture 0, you will find the Sepia.jar file, some agent source code, documentation and maps for this assignment in PA1.zip.

Familiarization with SEPIA (20 points)

Starting with the code for ResourceCollectionAgent, write a SEPIA agent for the ResourceCollection.xml map that executes the following actions: (i) collect enough Gold to build two Peasants (400 Gold each), (ii) build two Peasants (Townhalls can do this), (iii) collect enough Gold and Wood using all three Peasants to build a Farm (500 Gold, 250 Wood), (iv) build a Farm (Peasants can do this), (v) collect enough Gold and Wood using all three Peasants to build a Barracks (700 Gold, 400 Wood), (vi) build a Barracks (Peasants can do this), (vii) collect enough Gold using all three Peasants to build two Footmen (600 Gold each), (viii) build two Footmen, (ix) defeat the enemy footman on the map. You can use the provided CombatAgent code to figure out how to do (ix). Call your agent RCAgent. Check the execution of the sequence of actions for your agent using VisualAgent.

Pathfinding (80 points)

Write an agent that can move around a given map by implement the A* search algorithm discussed in class. Note that SEPIA has built in COMPOUNDMOVE/ createCompoundMove() actions that can handle pathfinding; **do not** use these actions in your code! Instead, use

PRIMITIVEMOVE/createPrimitiveMove()s and implement your own pathfinding solution. For A*, you will need a heuristic function. The Chebyshev distance is a good heuristic for this purpose. This is defined as follows: $D((x_1,y_1),(x_2,y_2))=max(|x_2-x_1|,|y_2-y_1|)$. To test your algorithm, use the maze maps provided. In each map, a Footman is trapped in a maze. Somewhere in the maze is an enemy Townhall. Use your implementation to guide the Footman to the Townhall and attack it. When the Townhall is destroyed, the game will end. If it is not possible to guide the Footman to the Townhall, print "No available path." in the terminal and quit. Use VisualAgent to check that your agent is behaving correctly in all cases. (You can run the maze maps just using VisualAgent, left click on the Footman and right click on the Townhall to see the solution according to the built in pathfinding routines.) Note that we will test your code on other maps as well, so ensure nothing is specific to the maps provided.

Code and Data Structures

Call your main agent containing the A* search code SearchAgent.java. Remember to have SEPIA.jar in the classpath when you compile this. Also remember to comment out the package line if starting with one of the provided agents as a template; import the abstract Agent class, update the config file to use your agent when running your code, and make sure the classpath includes wherever you have put the agent class file when running. To implement A*, it is fine to use basic data structures, such as arrays and lists, to keep track of what you need. You need not optimize for space or runtime for any of the methods (of course, you are welcome to do so if you wish).

What to turn in

Prepare a ZIP file with your agent code (do NOT include any class files, or any SEPIA code). Include a README with your name(s) and ID(s) and any other comments you have, such as special compilation instructions if any. Name your file as "yourname_PA1.zip" and use Canvas to submit it.