

## Assignment 4: Due February 6

### 1. (50 points) Graph Search.

Point to point route planning is a classic search problem. In this programming assignment, you'll add route planning capabilities to the Robot Defense game enabling the technology center's small robots to travel the map quickly and intelligently.

The pseudo code for graph search is available in your book on page 83.

Begin by downloading the RobotDefense files (ps4.zip) from the course website. As in the previous assignment, you'll need Java 1.5, or 1.6\_02 or above (release 1 of Java 1.6 has a critical bug). The code should work on the lab computers without an issue.

#### The Game

Before you begin coding, run the game so you understand the problem domain. When the game is launched, you will see a green island on the left side of the screen, a label indicating "SearchFactory" at the top of the screen, and on the right a series of tiles including *Grass*, *Mud*, etc.

You should also note that in this version of the game, you can select both the `SDRouteMap` and the `SearchFactory` using the click labels in the upper left corner. You can use your `SDRouteMap` implementation from PS 3 with this version of the game, but it's not strictly necessary. Here we will be focusing on the `SearchFactory` implementation, and we won't really need to use the `RouteMap` at all. As before, two stub classes have been provided for you.

#### Game Controls

There are four controls you may care about:

- The label at the top of the screen entitled "SearchFactory" indicates what class is being used to plan the path between points. Initially, you will have two choices, both of which are really just simple stub classes. By clicking the label, you can select between them. Eventually you will replace one of the stub classes with your own implementation.
- By left clicking on the Robot Control Center (the gray building) you can enable the control of the center's robot. Once the Robot Control Center is selected, subsequent right clicks will request the robot to plan a path from its current location to the cell on the map that was clicked. Note that the Stub implementation is really poor, and can only plan extremely short paths! Your implementation should be able to plan arbitrarily long paths.
- You can enable path recording by pressing '8'.
- You can turn path recording off by pressing '9'.

#### Writing Your Code

A Factory class is one whose main purpose is to build other objects. It is common to use factory classes with the Java service provider scheme for two reasons. First, factories make good lightweight proxies for the classes that do the real work. So by making a simple factory the nominal service provider, complex class loading can be deferred until the service is actually needed. Second, since reflection requires the use of default constructors, factories provide a convenient way to invoke more complex constructors in the classes they do the real work.

For this assignment, you will end up implementing three classes: a `SearchFactory`, a `SearchProblem` and a `SearchNode`. The `SearchFactory` must allow the game to create a search problem (defined on a graph with specified initial and goal states), and to perform A\* search to solve such a problem. The result of a successful search should be a `SearchNode` that points to a goal state, and whose `getParent()` method can be called successively to unravel the path back to the initial state. Thus, if we call `getParent()` repeatedly beginning with the node returned by the search, when the method finally returns `null`, we should be at a `SearchNode` that wraps the initial state (starting location). Additionally, note that the `getCost()` method should return the total path cost when invoked on the `SearchNode` wrapping the destination. If the search cannot find a path from the initial state to the goal state, it should return `null`. (Currently, robots cannot re-enter the control center, so that is one way to test this functionality).

- Download `ps4.zip` from the course web site.
- Unpack the contents in a working directory (let's call it `ps4`)
- Begin by testing the system. You should be able to run `java -jar robotdefense792.jar`, play the game (its boring!) and in the upper left corner click the `SearchFactory` box to select between `StubSearchFactory` and `StudentSearchFactory`.
- Next, change the name of `StudentSearchFactory` by replacing `Student` with your last name. You'll need to do five things:
  1. Change the file name
  2. Open the file and globally replace `StudentSearchFactory` with the new class name. You may also want to change the associated class names as well.
  3. Open the file `jig.misc.rd.route.SearchFactory` in `ps3/META-INF/services` and replace the `StudentSDRouteMap` entry with your new class name.
  4. Remove the `StudentSearchFactory.class` file
  5. Recompile your new source, for example by issuing the command `javac -cp robotdefense792.jar <your new java file>`
- You should be able to rerun the jar with the exact same results except noting that the name of the `StudentSDRouteMap` has changed appropriately.
- Now take a deep breath, and perform your own implementation of `SearchFactory`, `SearchNode` and `SearchProblem`.

A correct implementation will be sufficient for full credit on this assignment.

The bulk of the classes in the jar file will be of little interest to you. Focus your attention on classes and interfaces associated with the routemap and the graph data structure. Namely:

- `jig.misc.Edge`
- `jig.misc.EnumerableGraph`
- `jig.misc.Node`
- `jig.misc.rd.route.SearchFactory`
- `jig.misc.rd.route.SearchNode`
- `jig.misc.rd.route.SearchProblem`

**To Turn In:**

- A single source file with your implementation of the required classes and whatever other helper classes are needed to make it run (keep them all in a single file please).
- A javadoc comment at the top of your source file indicating your full name, and a statement about the status of your code (known problems, particular points of beauty, etc).
- Path data from each of the four included levels. You should obtain this by opening the `levels` directory and then for each file named `level-<n>.dat` do: 1) copy the file to `level.dat` 2) run the game using your `SearchFactory` 3) Press the 8 key to save path information 4) maneuver your robot from its initial location to each of the purple crystals labeled 1-5 in order 4) Press the 9 key to close the path information file 5) rename the output file `path.dat` to `path-<n>.dat`
- zip all of this up in a file called `<your last name>.zip` and email it to me by the due date