

## Assignment 3: Due January 30

### 1. (50 points) Dijkstra's Algorithm.

Dijkstra's algorithm is a well known graph algorithm which performs essentially the same task as *Uniform Cost Search*. The standard algorithm is applied using a graph,  $G$ , a weighting function  $w$  that returns values *greater or equal to 0*, and a source vertex/node  $s$ .

```

DIJKSTRA( $G, w, s$ )
1 INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  $S \leftarrow \emptyset$ 
3  $Q \leftarrow V[G]$ 
4 while  $Q \neq \emptyset$ 
5     do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
6          $S \leftarrow S \cup \{u\}$ 
7         for  $v \in \text{Adj}[u]$ 
8             do RELAX( $u, v, w$ )

```

In line 1, a data structure containing  $(d, \pi)$  values for each vertex/node is created which stores cost and predecessor information respectively for each vertex/node in the graph. The initialization sets these values such that  $\pi[v] = \text{NIL}$  for all  $v \in V$ , ( $V$  is the set of vertices/nodes in the graph)  $d[s] = 0$  and  $d[v] = \infty$  for all  $v \in V - \{s\}$ .

The function RELAX updates the  $d, \pi$  values by checking the impact of a newly found edge between  $u$  and  $v$  with a weight  $w$  as follows:

```

RELAX( $u, v, w$ )
1 if  $d[v] > d[u] + w(u, v)$ 
2     then  $d[v] \leftarrow d[u] + w(u, v)$ 
3          $\pi \leftarrow u$ 

```

So, if a new path with a lower cost to  $v$  is found through  $u$ , then update the data about  $u$  to reflect this better path.

It should be clear that Dijkstra's can also be equally applied to a single destination problem instead of a single source problem. In this case, the result of the computation can be used to find the best path to the destination from any node on the graph.

For this assignment, you will implement Dijkstra's algorithm to create Single Destination Route maps that will guide insects across the map. To do this, you'll need to implement your own SDRouteMap.

Begin by downloading the RobotDefense files (ps3.zip) from the course website. 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 have 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 “RouteMap” at the top of the screen, and on the right a series of tiles including *Grass*, *Mud*, etc.

The island is a regular grid (like a chess board), and your task will be to create a route map that takes bugs from the hole on the left side of the island to the corresponding hole on the right side of the island. The cost of traversing a square on the grid is based on what type of terrain occupies the specified location. The grid itself can be viewed as a Graph where the center point on each grid cell is a vertex and weighted edges connect adjacent cells. The weight of the edge between two cells is calculated by directly from the cost of the terrain in the two connected cells (note that since the path is longer between diagonally adjacent cells, so is the corresponding edge weight).

### Game Controls

There are four controls you may care about:

- The label at the top of the screen entitled “RouteMap” indicates what class is being used to build the route between the initial and goal locations. 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 clicking on the hole on the left side of the screen you can activate and deactivate route and cost information from RouteMap class displayed in the label at the top of the screen. When the route information is shown for the stub classes, you will see bogus path costs from all tiles (-1 in this case) and direction indicators (little dots on each tile) that point in random directions. When you use your own, correctly coded *SDRouteMap*, the path costs and direction indicators should be more meaningful.
- You can modify the terrain on the map by selecting one of the tiles on the right side of the screen and then right clicking the map. If your route map is being displayed when you do this, you should see path costs recalculated auto-magically.

### Writing Your Code

- Download *ps3.zip* from the course web site.
- Unpack the contents in a working directory (let’s call it *ps3*)
- Begin by testing the system. You should be able to run `java -jar robotdefense728.jar`, play the game (its boring!) and in the upper left corner click the *SDRouteMap* box to select between *StubSDRouteMap* and *StudentSDRouteMap*.
- Next, change the name of *StudentSDRouteMap* 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 *StudentSDRouteMap* with the new class name.
  3. Open the file `jig.misc.rd.SDRouteMap` in `ps3/META-INF/services` and replace the *StudentSDRouteMap* entry with your new class name.

4. Remove the `StudentSDRouteMap.class` file
5. Recompile your new source, for example by issuing the command `javac -cp robotdefense745.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 `SDRouteMap`.

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.SDRouteMap`

**To Turn In:**

- A single source file with your implementation of the `SDRouteMap` 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).
- RouteMap 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 `SDRouteMap` 3) Press the 0 key to save route information 4) rename the output file `route.dat` to `route-<n>.dat`
- zip all of this up in a file called `<your last name>.zip` and email it to me by the due date