Assignment: The Rambler's Problem

COM1005 Machines and Intelligence
Semester 2: Experiments with AI Techniques

Heidi Christensen
University of Sheffield
Version 1.0

This assignment carries 30% of the marks for COM1005
DUE DATE: Friday 21st May 2020 at 23:59 (UK Time)

In this assignment you'll experiment with search strategies for solving the Rambler's problem.

The Problem

The problem is to work out the best walking route from a start point to a goal point, given a terrain map for the walking area. A terrain map specifies the height of each point in the area. Figures 1 shows an example of a realistic terrain map.

Figure 1: A realistic Terrain Map

A more simple terrain map is shown in Figure 2.
For a rambler, the best route is the one which involves the least climbing.

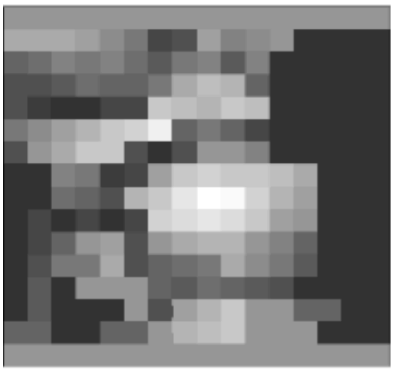

Figure 2: A simple Terrain Map. White is highest, black is lowest. This map is saved in `tmc.pgm`

Representing Terrain Maps

We'll store our terrain maps in Portable Grey Map (pgm[1]) format.
In this format each cell is represented by an `int` from 0 to 255.
You can view and edit pgm files using `irfanviewer` - free download here: `http://www.irfanview.com` or just a normal text editor as it is in fact just a normal text file.
In Figure 3 you can see a screenshot of the content of the TMC.PGM.

```
P2
16 16
255

150 150 150 150 150 150 150 150 150 150 150 150 150 150 150 150
170 170 170 160 140 120 070 080 160 130 140 150 050 050 050 050
100 110 130 120 130 110 090 120 130 090 130 050 050 050 050 050
080 080 090 110 100 100 050 170 190 180 100 050 050 050 050 050
080 060 050 050 070 070 080 190 180 200 190 050 050 050 050 050
050 090 100 100 120 110 100 110 120 100 070 050 050 050 050 050
050 080 110 090 070 080 050 140 150 150 130 050 050 050 050 050
050 050 130 120 060 070 160 200 210 200 200 190 170 050 050 050
050 050 110 100 080 180 200 230 255 250 210 180 160 050 050 050
050 070 050 070 050 070 210 220 230 220 200 160 150 050 050 050
050 070 100 150 160 080 150 170 180 180 150 130 100 050 050 050
050 080 120 120 170 080 100 110 120 170 140 120 090 050 050 050
050 090 050 150 150 150 100 090 110 100 090 080 050 050 050 050
050 090 050 050 050 150 080 160 180 200 150 150 100 100 050 050
100 100 050 050 100 100 150 180 190 200 150 150 150 050 050 050
150 150 150 150 150 150 150 150 150 150 150 150 150 150 150 150
```

Figure 3: This is the content of the file `tmc.pgm` when viewed in a terminal window.

A pgm file contains a header followed by a matrix with the actual data. Figure 4 shows the header information. Figure 5 shows the x- and y-axis definitions.

Code

Code for handling the terrain maps is in the RAMBLERS folder within the assignment folder.

[1] `http://netpbm.sourceforge.net/doc/pgm.html`

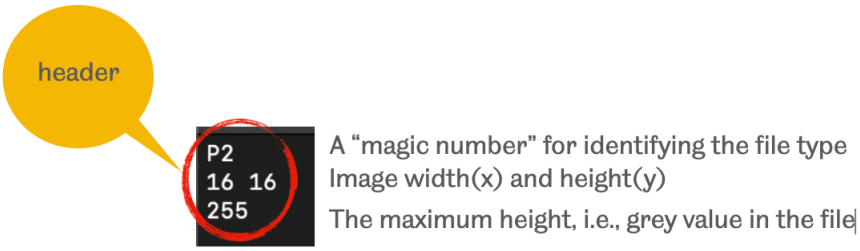

Figure 4: Header information in the pgm file.

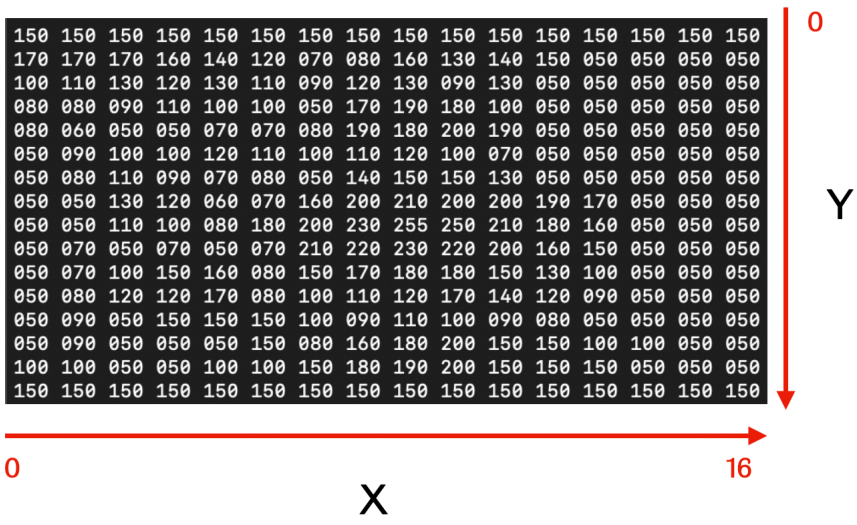

Figure 5: PGM data matrix with x and y orientation.

You are provided with the file TMC.PGM and a java class `TerrainMap` whose constructor is given the filename of a pgm image and reads its contents, i.e.,

```
// defining a new terrain map
    TerrainMap tm = new TerrainMap("tmc.pgm");
```

`TerrainMap` has the following accessors:

```
// accessors
  public int[][] getTmap() {
    return tmap;
  };

  public int getWidth() {
    return width;
  };

  public int getDepth() {
    return depth;
  };

  public int getHeight() {
    return height;
  };
```

Rambler's costs

The Rambler steps one pixel at a time North, South, East or West (not diagonally). An upward step is more costly. The local cost $c(y, x, y', x')$ of a step from (y, x) to (y', x') is:

$$c(y, x, y', x') = \begin{cases} 1 & \text{if } h(y', x') \leq h(y, x) \\ 1 + |h(y', x') - h(y, x)| & \text{otherwise.} \end{cases} \quad (1)$$

where $h(y, x)$ is the height in the terrain map at point (y, x).
NOTE, that y is written before x!

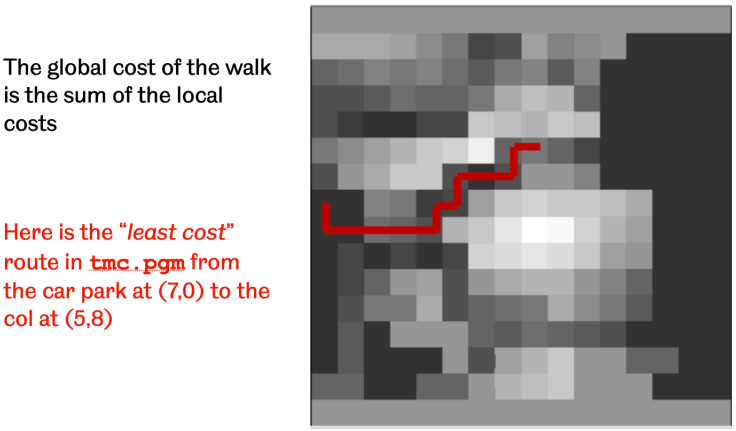

Figure 6: Illustration of a *lowest cost* route found from a start point (car park at (7,0)) and point at (5,8).

What you must do

Using the Java code provided for the assignment do the following **four tasks**:

Task 0: Set up a GitHub (or similar) repository for keeping versions of your code as you develop your solution. Make sure to push updates regularly. The purpose of using a git repository is twofold: i) to develop good habits and practice around version control; ii) in case of a suspicion of unfair means, you have clear evidence that code was developed along the way by yourself. You must add the url of your github repository to your LaTeX report (I've added a nifty little footnote to the title where this info can go).

Task 1: Implement branch-and-bound for the Rambler's problem Working with `search4` code and following the procedure of taking a set of general classes and making a *specific* solution for a particular problem, implement branch-and-bound search for the Rambler's problems. You will need to define a class `RamblersState` and a class `RamblersSearch`. Look at the corresponding classes for Map Traversal (week3) for guidance. You will also need a class for running the tests. Call this `RunRamblersBB`.

Task 2: Assess the efficiency of branch-and-bound Try out a number of start and end points on the tmc map and assess the efficiency of branch-and-bound in this domain. You may also create other Terrain Maps of your own, or make use of `diablo.pgm` in the Rambler's folder which is a terrain map of Mt Diablo in California. Tip: that map is a lot bigger, so if your code takes a long time to run, consider editing the map down.

Task 3: Implement A* search for the Rambler's problem Working from the `search4` code, implement A* search for the Rambler's problem. Remember that for A* you need (under)estimates of the remaining cost to the goal. Experiment with different choices for this heuristics, for example:

1. the Manhattan Distance between the current pixel and the goal $(p + q)$.
2. the Euclidean distance
3. the height difference
4. combinations of these

You may also devise other ways of combining the estimates. You will also need a class for running the tests. Call this `RunRamblersAstart`.

Task 4: Compare the efficiency of branch-and-bound and A* Perform experiments to test the hypothesis that A* is more efficient than branch-and-bound and that the efficiency gain is better for more accurate heuristics.

Task 5: Produce a report Your experimental report should describe your implementations and your results. Your report should include at the very least:

- Description of your branch-and-bound and A* search implementations.
- Presentation of results obtained when testing the efficiency of these two approaches.
- A comparison of the results - can you verify your hypothesis?

A LaTeX report template is provided, with compulsory sections specified. You may add more sections if you wish: include in your report what you think is interesting.

Note 1: *you **must** use the LaTeX template provided for your report. However, you will not be marked on your ability to use LaTeX to format your report (i.e., there are no extra marks for making it look fancy!). That being said, LaTeX is an essential piece of software for communicating research and results in engineering and science, so we want you to get experience using it early on.*

Note 2: *you should not have to make any changes to the Java code provided except to control the amount of printout during a search and perhaps to modify what a successful search returns. It's a good idea to revisit the week 3 lab class for a reminder of how you worked with branch-and-bound and A* searches.*

Code

In the downloaded zip-file you will find a `code/search3` folder. This is your working folder where you should also develop your code. There are the usual classes that implements the search engines (`Search.java`, `SearchState.java` and `SearchNode.java`).
In addition, you will find a class that can read a terrain map (`TerrainMap.java`) as well as a class that easily enables you to handle coordinates (`Coords.java`). Finally, I've given you test class to illustrate how you load a particular pgm file (`TestTerrainMap.java`).

```
/**
  * TestTerrainMap.java
  *
  * Phil Green 2013 version
  * Heidi Christensen 2021 version
  *
  * Example of how you load a terrain map
*/

import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class TestTerrainMap {

  /**
   * constructor, given a PGM image Reads a PGM file. The maximum greyscale value
   * is rescaled to be between 0 and 255.
   *
   * @param filename
   * @return
   * @throws FileNotFoundException
   */

  public static void main(String[] arg) {

    TerrainMap tm = new TerrainMap("tmc.pgm");

    System.out.println(tm.getWidth());
    System.out.println(tm.getTmap()[7][2]);

  }
}
```

Summary of assignment

1. Implement a branch-and-bound and an A* search for the Rambler's problem using the code provided.

2. Run your code with different start and end points and different heuristics (for A*) on the different maps, to assess the efficiency of the two types of search algorithms.

3. Complete your report with the results and conclusions.

What to submit

Submit your solution, as a zipped file called `SURNAME_FIRSTNAME.zip`, where `SURNAME` is your surname (family name), and `FIRSTNAME` is your first name (given name). This zip file should contain:

- a `.pdf` copy of your report named `SURNAME_FIRSTNAME.pdf`
- your code.

Marking Scheme

Percentage points	Categories and example score weights; Marks will be awarded according to these cri
40%	**Implementation of branch-and-bound and A* search**
	30% Sensible implementations of `RamblersState` and `RamblersSearch` and the two respective test classes. Note: marks will be deducted if your code doesn't compile and run on the command line. You can use your favourite IDE but make sure that I can compile & run your code from the command line in the `search3` folder like this "javac RunRamblersBB.java" and "java RunRamblersBB". You will lose marks, if I can't do this without modifying code.
	10% Well documented and well presented code. This is about using comments to ease understanding of more complex parts of the code, and the consistent use of indentations, brackets and appropriate choices of variable names. Note, there isn't much code programming to do in this assignment ...).
40%	**Experimental results**
	15% Good assessment of efficiency of branch-and-bound. More points are given for exploring a good number of different start and end points.
	25% Good assessment of efficiency of A*. More points are given for exploring different estimates of the remaining costs to the goal, and for exploring a good number of different start and end points.
20%	**Documentation - quality of presentation and communication of your experimental results in your report**
	10% Quality of communication of results in e.g. tables and figures
	10% Quality of discussion of results and conclusions
100%	**Total**

Marks and feedback will be returned through Blackboard within 3 weeks.

Rules

Unfair means: This is an individual assignment so you are expected to work on your own. You may discuss the general principles with other students but you must not work together to develop your solution. You must write your own code. All code submitted may be examined using specialized software to identify evidence of code written by another student or downloaded from online resources.