Data Structures and Algorithms

COMP 251, Winter 2013

Assignment 1

Due date: Monday, January 28, 2013
6pm

All coding for the assignment must be in Java. All code should be well-commented, in a professional style, with appropriate variables names, indenting, etc. Your code must be clear and readable. **Marks will be very generously deducted for bad style or lack of clarity.**

You will also need to do experimental work. In this you should endeavour to minimize sources of noise and ensure your results are meaningful despite any variance. Graphed output will be required; you may use any graphing software you wish, and provide output in any common graphics format.

Here you will develop and analyze an algorithm for finding simple geometric shapes in images. You need to write an application that accepts either three or five command-line arguments:

```
java Assig1 filename minr minr [maxw maxh]
```

The *filename* is an image file, *minr* and *maxr* are min/max radii (inclusive), and *maxw* and *maxh* are optional parameters that specify the maximum width and height of the image you should analyze (if not specified, you analyze the entire image).
You can assume sensible values for all parameters, and do not need to error-check parameter validity. You *will* need to determine whether the final two parameters have been specified.
Your application should emit textual output to `stdout` as well as creating an output image file with a fixed filename and format: "`outputimage.png`". Please restrict yourself to this exact interface description.

1. In a computer context images can be represented as 2-dimensional arrays of pixels, each of which is encoded as an integer value representing the pixel's Red-Green-Blue (RGB) values. Within this context, a fundamental problem in image analysis is to recognize simple geometric shapes, such as circles. For circles, one way of doing this is to take advantage of symmetry, the basic idea of which is as follows. **10**

   Assume each point in the array is potentially the center of a circle of radius r. If so, then from the center of the circle you should find a circular boundary r pixels away; if not, no circle exists at that center and radius.

   There are two properties you need to be concerned with if a circle may be present. First, the circle itself should consist of pixels that all have the exact same colour (RGB value). Second, we are only interested in detecting circles of a single line thickness and not within arbitrary shaped patches of the same colour; circles of radius $r+1$ and $r-1$ around the same center should *not* contain pixels of the same colour as your potential circle.

   Implement this algorithm. For each circle found, textually emit its center and radius and (re-)draw the circle on the image in red (the modified image should be the output image).

   Each circle should be reported (and drawn) exactly once; if necessary, choose an appropriate data structure to ensure that is the case. Note that some circles are impossible given a center location and radius—avoid testing these.

   For uniformity, you must use *Bresenham's* algorithm to trace out circles (and draw them). An implementation of this is provided in the sample code, as is a small example of loading an image, accessing and changing a pixel, and writing out the resulting image. Several image files are also provided for you to test your code. Note that grading of your implementation will use those files, and may include arbitrary other files as well.

2. Clearly (and succinctly) describe your algorithm, using words and/or high-level pseudo-code. Assume that Bresenham's algorithm takes time $2\pi r$ for a given r. For an $n \times n$ image input and an arbitrary but constant single radius input (ie r =*maxr*=*minr*):

   (a) Analyze the worst-case time complexity of your approach in terms of n. You must provide a clear description of how you derive the complexity. Provide proof of an appropriate complexity class. **10**

   (b) Analyze the best-case time complexity of your approach in terms of n. You must provide a clear description of how you derive the complexity. Provide proof of an appropriate complexity class. **5**

   (c) Suppose that r is not a constant, but rather a function of n. Is there an r that maximizes the worst-case time complexity for a given n? Justify your answer. **5**

3. (a) Experimentally measure the performance of your algorithm in relation to n. Use image 3 as input, and vary the last (two) command-line parameter(s) as a proxy for a range of image inputs of increasing size. Keep radius constant at 25. Be sure to measure only time to detect the circles, not the time to emit output. Discuss and explain the behaviour—does it match your calculated time complexity? **10**

   (b) Now, experimentally measure performance in terms of radius. Use image 3 again, analyzing the entire image, but varying radius from 4 up to 200. Again, plot performance and discuss the results. **5**

4. The algorithm given is quite fragile, and does not always find things which look circular to humans. Come up with a more robust way of tracing out the circles that detects as many of what a human observer might consider circles in images 1 and 3 (small) as possible. Concisely describe your algorithm changes. **5**

What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 6pm**.

Where possible hand in only **source code** files containing code you write. Do not submit compiled binaries or .class files. For the written answer questions submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

Note that for written answers you must show all intermediate work to receive full marks.

This assignment is worth 6% of your final grade. **50**