Joanna Klukowska
joannakl@cs.nyu.edu

Programming Project 4: Image Segmentation by Region Growing

Due date: April 14, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission.

You are responsible for every line in your program: you need to know what it does and why.

If you are considering use of any Java API classes that are not discussed in CSCI101 curriculum and have not been covered yet in CSCI102, ask your instructor first. The objective of these projects is to practice the material that has been discussed in class and the solutions are not always the most efficient ones.

In this assignment you will work on the program that performs image segmentation using region growing algorithms.

Image segmentation is a process of partitioning a digital image into multiple segments. Usually, the segmentation simplifies the image and makes it easier to analyze. The segments supposed to represent meaningful regions of the original image.

Region growing approach is a simple approach to image segmentation. It starts with the user selected seed points and grows each of them into a region based on a region membership criterion. In this project a seed is a single pixel and the membership criterion is based on the gray level values of the pixels.

Objectives

The goal of this project is for you to master (or at least get practice on) the following tasks:

- implementation of a reference based stacks,
- implementation of a reference based queue.
- implementation of an algorithms based on their descriptions,
- working with existing code,
- implementing generic classes.

Problem Description

You are given the program that performs image segmentation. The implemented algorithm is based on the depth first search approach to region growing. This solution needs to use a stack to keep track of the pixels that need to be visited and evaluated. Currently it is using the stack implementation from the Java API. **Your first task is to provide your own reference based stack implementation.**

The depth first search algorithm for creation of a single region from a seed is as follows:

```
1 INPUTS:
2 seedX, seedY are the coordinates for a seed
3 value is the gray level value to be used for the region
4 outputImage this is where the region is written to (should be filled with background color,
5 may contain other regions)
6 imputImage the gray level image
7
8 ALGORITHM:
9 regionGrowingDFS ( seedX, seedY, value, outputImage, inputImage )
10    create an empty stack S
```

Joanna Klukowska
joannakl@cs.nyu.edu

```
    determine a threshold to be used

    push seed pixel (seedX,seedY) onto the stack S

    while stack is not empty
        currentPixel = pop the pixel from the stack S
        set currentPixel's gray level to value (in the outputImage)
        for each pixel in the 3x3 neighborhood of the currentPixel
            if the pixel is a background pixel in the outputImage
                  and the pixel is within threshold of the currentPixel (in the inputImage)
                push pixel onto the stack S
```

The above algorithm is already implemented in the method:

```
void getRegionDFS( int x, int y, int value, int [] pixels, int [] imagePixelsCopy )
```

An alternative approach uses a breadth first search approach to region growing algorithm. This solution needs to use a queue to keep track of the pixels that need to be visited and evaluated. Your task is to implement a reference based queue and to implement the second approach.

The breadth first search algorithm for creation of a single region from a seed is as follows:

```
INPUTS:
seedX, seedY are the coordinates for a seed
value is the gray level value to be used for the region
outputImage this is where the region is written to (should be filled with background color,
may contain other regions)
imputImage the gray level image

ALGORITHM:
regionGrowingBFS ( seedX,  seedY,  value,  outputImage,  inputImage )
    create an empty queue Q
    determine a threshold to be used

    enqueu seed pixel (seedX,seedY) onto the queue Q

    while queue is not empty
        currentPixel = dequeue the pixel from the queue Q
        for each pixel in the 3x3 neighborhood of the currentPixel
            if the pixel is a background pixel in the outputImage
                  and the pixel is within threshold of the currentPixel (in the inputImage)
                enqueue pixel onto the queue Q
                set currentPixel's gray level to value (in the outputImage)
```

You need to use this algorithm to provide the implementation for the currently blank method:

```
void getRegionBFS( int x, int y, int value, int [] pixels, int [] imagePixelsCopy )
```

Program Modes

The program can be run in an interactive mode or a test mode.

In the **interactive mode** the user manually selects the seeds using a mouse or a touch pad. The user has the following options selected by single key presses:

r - resets the image to the original

x - select random seed points

Joanna Klukowska
joannakl@cs.nyu.edu

d - run the region growing algorithm (using depth first search) and show the results

b - run the region growing algorithm (using breadth first search) and show the results

In the **test mode** the program uses several images (names provided in the `testFileNames` array) with predefined seeds (seeds provided in the `testSeeds` array) to run either or both of the algorithms (the Boolean flags `RUN_TESTS_DFS` and `RUN_TESTS_BFS` determine which of the algorithms should run). The segmented images are saved to files on disk. No window is launched and no user interaction is possible in this mode.

Programming Tasks

Task 1 implement a reference based stack. Replace the stack used in getRegionDFS with your own implementation. Your stack must be generic. Your stack must provide a reference based implementation. It must provide the following methods:

`boolean empty()` Tests if this stack is empty.

`E peek()` Returns the element at the top of this stack without removing it from the stack.

`E pop()` Removes the element at the top of this stack and returns that element as the value of this function.

`E push(E` item) Pushes an item onto the top of this stack. Returns the item itself.

The stack should also contain a default constructor that creates an empty stack object.

For details of each method you should see `http://docs.oracle.com/javase/8/docs/api/java/util/Stack.html`.

Task 2 implement a reference based queue. Your queue must be generic. Your queue must provide a reference based implementation. It must provide the following methods:

`boolean empty()` Tests if this queue is empty.

`E peek()` Returns the element at the front of this queue without removing it from the queue.

`E dequeue()` Removes the element at the front of this queue and returns that element as the value of this function.

`E enqueue(E` item) Adds an item to the back of this queue. Returns the item itself.

The stack should also contain a default constructor that creates an empty queue object.

Task 3 Implement the method:

```
void getRegionBFS( int x, int y, int value, int [] pixels, int [] imagePixelsCopy )
```

Using Provided Code

This program uses Processing package for the graphical interface and for working with images. You will need to configure your Eclipse to work with Processing. If you are running from the command line, the Processing package need to be included in the class path when the program is compiled and run. See below for details.

Joanna Klukowska
joannakl@cs.nyu.edu

Working with Eclipse

You should download the `proj4_Eclipse.zip` file from the course website. Remember the location to which the file was saved.

Create a new project or select a project to which you want to import the files for this program.

Goto `File` menu, select `Import` and them pick `Archive File` from the `General` list. In the next window select `Browse` and navigate to the saved location of the zip file.

Now, you need to add Processing package `core.jar` to the build path. Right click on the name core.jar, select `Build Path` and then `Add To Build Path`.

You can run the program RegionGrowing as any other program. You may need to select to run it as Java Applet, if prompted the first time you run it. Eclipse should have placed all image files in the correct location so that the program can find them.

Working from a Terminal

You should download the `proj4_terminal.zip` file from the course website. Remember the location to which the file was saved.

Extract the zip file. Lets assume that the content of the zip file is in directory called `proj4_terminal`. You will be running all the commands from within the `proj4_terminal` directory. You should have a file `RegionGrowing.java`, a file `core.jar` and several image files. To compile the program execute

```
javac -cp core.jar *.java
```

To run the program execute:

```
java -cp ".:core.jar" RegionGrowing
```

To run the program in the test mode (with both tests included) execute

```
java -cp ".:core.jar" RegionGrowing test
```

Working on This Assignment

You should start right away! You are given a working program. You should add to it implementation of the task1 and verify that it still works correctly, then move on to tasks 2 and 3.

Grading

Your code will be evaluated based on its implementation (style, efficiency, correctness). Your classes should be general purpose classes that can be used elsewhere (not only for the purpose of this assignment).

30 points - stack implementation

30 points - queue implementation

15 points - breadth first search algorithm implementation

25 points - documentation and program style

An array-based implementation of stack/queue will receive zero credit. An implementation that does not use your own version of stack/queue (but rather uses the classes provided with Java API) will receive zero credit.

Joanna Klukowska
joannakl@cs.nyu.edu

How and What to Submit

Your should submit all source code files (the ones with .java extensions only) in a single **zip** file to NYU Classes. You should have at least 3 different files (classes for `RegionGrowing`, Stack, Queue). You may decide that you need additional classes as well and all of them should be submitted..

Do not submit any of the image files.

Do not submit the `core.jar` file.

Once you submit the program make sure that you receive a confirmation email from NYU Classes. Read through it to double check that correct files were uploaded. Keep this email as your proof of submission until you receive your grade for the assignment.