Programming Assignment #3

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet. If any code you submit is in violation of this policy, you will receive no credit for the entire assignment.

Task A

Problem Description

You are helping your friend Ella pack her suitcase for Mars. The flight regulations dictate that the total weight of items in her suitcase must not exceed W. Ella has several belongings, each with a different sentimental value. She wants to select the amount of items that maximize the total sentimental value to her, while maintaining a total weight up to W. "Hmm", you tell her, "this problem sounds familiar...".

Your task

You must help Ella implement a method `maximizeSentimentalValue()`. This method takes the file `belongings.txt` as input. The file `belongings.txt` contains the value and weight of each item, along with a weight bound W. Your method must output the set of items that Ella should select so that the total sentimental value is maximized, while the total weight does not exceed W.

Task B

Problem Description

Ella has successfully moved to Mars, and is very happy in her new life. Sending texts to her friends is as cheap as can be using a character-by-character, lossless, prefix-free encoding, and with all her favorite items around her she feels as much at home as possible - so that's great. But as the saying goes, one picture is worth a thousand words. Ella wants to show her friends what her new world looks like, and describing it in text just isn't enough. However, sending images is also very expensive, and the cost is determined by the number of pixels in the image.

Ella wants to reduce the width of her images before sending them to her friends to save money, and she considers cropping or resizing. These ideas do not work quite well–with cropping, she could completely lose interesting parts of the image, and with resizing a lot of details become tiny.

One day, a little alien comes up and tells her that there is a good heuristic that computes, for each pixel in an image, the amount of "content" in it. Using this she can reduce the width of the image by strategically discarding low content pixels.

At each round, Ella wants to strategically decrease the width of the image by 1, until she reaches the width she wants. This strategic decrease could be very time-consuming for some people, especially as it is repeated over multiple rounds, but not for Ella: thanks to your crash course when packing her suitcase, she now knows dynamic programming!

Your task

You must help Ella implement a method `cleverWidthReduction()`. This method takes the following as input:

- an image file `Image.type`.
- a positive integer `goalWidth`, smaller than the original width.

This method outputs an edited image with the same height and width equal to `goalWidth`. The method reduces the width of the image by finding a **minimum "content" vertical path**, which can be found using dynamic programming. Your method will then removes this minimum "content" vertical path from the image using the `deleteVerticalPath()` method that has been provided to you. The `cleverWidthReduction()` method continues to do so until the width has been reduced to `goalWidth`.

What is "content"?

Each pixel can be assigned a non-negative, integer "content" value through a certain heuristic method that compares the RGB values of its neighboring pixels. The greater the RGB difference of a pixel's neighbors is, the greater its "content" value is. And the greater its "content" value, the more likely it is that this pixel holds important visual information. You do not have to worry about "calculating" the content of the image.

What is a vertical path?

Given an image with height `h` and width `w`, a vertical path is a vector $[p_1, p_2, ...p_h]$ such that $\forall i$ (a) $p_i \in [0, w-1]$, (b) if p_{i-1} exists, $|p_i - p_{i-1}| \leq 1$, and (c) if p_{i+1} exists, $|p_i - p_{i+1}| \leq 1$.

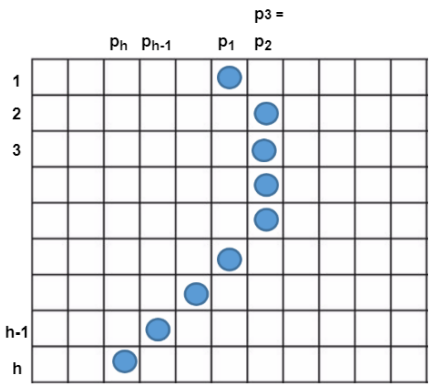

Figure 1: Example of a vertical path.

Description of "MyImage" class

As we don't want you to have to deal with image handling in java, this class will do that part of the dirty work for you. A `MyImage` object consists of two fields: a `BufferedImage image`, and a `double[][] contents`.

The number of rows in the `contents` matrix is equal to the height of the image, and number of columns equal to the width of the image. The `contents` matrix stores, for each pixel, its "content" value, which is calculated according to the magic heuristic provided by the little alien. The `contents` matrix is created when the `MyImage` object is created (the constructor takes in the `BufferedImage image` as input). The method `deleteVerticalPath()` takes as input a path deletes it, updating `image` and `contents` to reflect the changes.

The public methods are the only "interaction" you can have with the object. All of them are "get" functions, except for one.

Part 1: Report [40 points] [Due Date: August 7, 2019]

This programming assignment heavily relies on getting the recursive relation correct. To make sure you write meaningful code for Part 2, we are asking you to determine the recursive relation first. If you get your recursive relation wrong, we will provide you with the correct recursive relation so that the final code you submit is meaningful. In this case, you will not be awarded any points for this section of the assignment.

Your report must include:

(a) [10 points] The recursive relation that you need to use to compute `maximizeSentimentalValue()`.

(b) [30 points] The recursive relation that you need to use to compute `cleverWidthReduction()`.

Part 2: Code [60 points] [Due Date: August 14, 2019]

The methods you need to implement are both located in `Program3.java`. They are are called `maximizeSentimentalValue()` and `cleverWidthReduction()`. The description of the input and desired output for both methods can be found in the comments above them in the code provided.

Instructions

- Download and import the code into your favorite development environment. We will be grading in Java 1.8. **It is YOUR responsibility to ensure that your solution compiles with the standard Java 1.8 configuration (JDK 8).** For most (if not all) students this should not be a problem. If you have doubts, email a TA or post your question on Piazza.
- Carefully study the code provided to you, and ensure that you understand it before starting to code your solution.
- You are only allowed to edit `Program3.java`.

- You can modify `Driver3.java` if it helps you to test or debug your program. However, don't make changes that affect the functionality of your program, as we will be using the original version of `Driver3.java` to grade your submission.
- `Driver3.java` is the main driver program. Use command line arguments to either specify a belongings file, or an image (name only), the image's type(.jpg or .png), and a desired width. Example execution to choose what to pack:
  `java Driver3 myStuff.txt`
  Example execution to reduce width of image:
  `java Driver3 desertLandscape .png 400`
- Do not add a package to your code. If you have a line of code at the top of your Java file that says `package <some package name>;` that is wrong.
- We will be checking programming style. A penalty of up to 10 points will be given for poor programming practices (e.g. do not name your variables foo1, foo2, int1, and int2).

What To Submit

By August 7, 2019 11:59pm

Your report in pdf form, named: `eid_lastname_firstname.pdf`.

By August 14, 2019 11:59pm

A single zip file titled `eid_lastname_firstname.zip` that contains `Program3.java`, as well as any extra `.java` source files you created. Do not put these files in a folder before you zip them (i.e. the files should be in the root of the ZIP archive).