Programming Assignment 2—Dynamic Programming and Image Processing
Points: 400

Due: Dec 12, 11:59PM
Late Submission Dec 13, 11:59PM (1% penalty, yes it is 1%)
Early submission: Dec 11, 11:59PM (10% Bonus)

Description of a programming assignment is not a linear narrative and may require multiple readings before things start to click. You are encouraged to consult instructor/Teaching Assistants for any questions/clarifications regarding the assignment. Your programs must be in Java, preferably Java 8.1.

All your classes must be in the package named **pa2**. For this PA, you **may work in teams of 2**. It is your responsibility to find a team member. If you can not find a team member, then you must work on your own. Only **one submission** [er group.

1 Matrix Cuts

Let M be a $n \times m$ integer matrix (with n rows and m columns). Assume that rows are numbered $0, 2, \cdots, n-1$ and columns are numbered $0, 2, \cdots m-1$. Let $M[i, j]$ refers to the cell in ith row and jth column. We will define two types of matrix cuts, *width cut* and *stitch cut*.

Consider a path in the matrix that starts in a cell in the zeroth row of the matrix. When you are at cell (i, j) (ith row and jth column), in the next step you can go to cell $(i+1, j-1)$ or $(i+1, j)$ or $(i+1, j+1)$. Such a path is called width cut. Below is a more formal definition.

A *width cut* V of M is a sequence of n tuples $[\langle 0, y_0\rangle, \langle 1, y_1\rangle, \cdots, \langle n-1, y_{n-1}\rangle]$ such that every y_i lies between 0 and $m-1$ and the following holds

$$\forall\ i\ , 0 \le i < n-1, y_i \in \{y_{i-1}, y_{i-1}-1, y_{i-1}+1\}$$

The *cost of a width cut* $V = [\langle 0, y_0\rangle, \langle 1, y_1\rangle, \cdots, \langle n-1, y_{n-1}\rangle]$ is defined as

$$Cost(V) = M[0, y_0] + M[1, y_1] + \cdots + M[n-1, y_{n-1}]$$

Given a matrix, the *min-cost width cut*, denoted $MinWC(M)$, is a width cut whose cost is the smallest. If there are multiple min-cost width cuts whose cost is the same, then we take this to be the min-cost width cut that ends in the left most cell of the bottom row.

We next define the notion of *stitch cut* which is similar to the notion of *width cut*. This is also a path that starts at a cell in the zeroth row. When you are at cell (i, j), in the next step you can go to $(i, j+1)$, or $(i+1, j)$ or $(i+1, j+1)$. Below is a formal definition.

A *stitch cut* S of M is a sequence of tuples $[\langle r_0, c_0\rangle, \langle r_1, c_1\rangle \cdots \langle r_k, c_k\rangle]$ such that the following conditions hold

- $r_0 = 0$ ad $r_k = n - 1$. (The path start in top row and end in bottom row).
- For every i, $0 \leq i \leq k$, $0 \leq c_i \leq m - 1$
- For every i, $1 \leq i \leq k$, $c_i \in \{c_{i-1}, c_{i-1} + 1\}$
- For every i, $0 \leq i \leq k$, $0 \leq r_i \leq n - 1$
- For every i, $1 \leq i \leq k$, $r_i \in \{r_{i-1}, r_{i-1} + 1\}$

The *cost of a stitch cut* S is

$$\sum_{\langle r,c\rangle \in S} M[r, c]$$

Min cost stitch cut is a stitch cut whose cost is the smallest. If there are multiple stitch cuts whose cost is the same, then we take this to be a min-cost stitch cut that ends in the left most cell of the bottom row.

Note that, for a matrix with n rows, while width cut will have exactly n tuples, a stitch cut might have more than n tuples. Consider the following matrix

```
5    7    9    4    5
2    3    1    1    2
2    0    49   46   8
3    1    1    1    1
50   51   25   26   1
```

Mincost stitch cut for this matrix is

$$\langle 0,0\rangle, \langle 1,0\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 3,3\rangle, \langle 4,4\rangle$$

with a cost of 10, and mincost width cut is

$$\langle 0,3\rangle, \langle 1,3\rangle, \langle 2,4\rangle, \langle 3,3\rangle, \langle 4,4\rangle$$

with a cost of 15

1.1 Task1: MatrixCuts

You are provided a class named `Tuple` that implements tuples.

Design a class named `MatrixCuts` with following `static` methods.

`static ArrayList<Tuple> widthCut(int[][] M)`: Returns the min-cost width cut of M and its cost. The return type is an arraylist of Tuples (`ArrayList<Tuple>`. First entry of this list is a tuple of the form $\langle x - 1\rangle$, where x is the cost of the min-cost width cut. Rest of the tuples represent the min cost width cut. For example, on the above matrix, this method returns the following array list.

$$\langle 15, -1\rangle, \langle 0,3\rangle, \langle 1,3\rangle, \langle 2,4\rangle, \langle 3,3\rangle, \langle 4,4\rangle$$

Note that if M has n rows, the the returned array list has exactly $n+1$ Tuples.

`static ArrayList<Tuple> stitchCut(int[][] M)`: Return the min-cost stitch cut of M and its cost. The return type is an *ArrayList of Tuples.* (`ArrayList<tuple>`). First entry of this list is a tuple of the form $\langle x-1\rangle$, where x is the cost of the min-cost stitch cut. Rest of the tuples represent the min cost stitch cut. For example, on the above matrix, this method returns the following array list.

$$\langle 10,-1\rangle, \langle 0,0\rangle, \langle 1,0\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 3,3\rangle, \langle 4,4\rangle$$

Note that if M has n rows, the the returned array list has can have more than $n+1$ Tuples.

You **must** use dynamic programming paradigm to arrive at your code. For this, first define the recurrence relation. Then arrive at an iterative solution. Your code **must be iterative, not recursive and should not use use memoization**. Otherwise you will receive **zero** credit.

2 Applications to Image Processing

Width and stitch can be used in image processing, in reducing the image width and in stitching images that partially overlap.

2.1 Reducing Image Width

Recall that an image is a 2-dimensional array of pixels. Each pixel has represented as 3-tuple in the RGB format. Given an image of width W and height H, it is represented by a $H \times W$ matrix (2-D array). For example an image of width 4 and height 3 is represented as the following 2-D array of pixels M:

```
[98, 251, 246]  [34, 0, 246]  [255, 246, 127]  [21, 0, 231]
[25, 186, 221]  [43, 9, 127]  [128, 174, 100]  [88, 1, 143]
[46, 201, 132]  [23, 5, 217]  [186, 165, 147]  [31, 8, 251]
```

In the above $M[i,j]$ refers to the pixel in the ith row and jth column. For example $M[2,3]$ is $[31,8,251]$ and $M[0,0]$ is $[98,251,246]$.

Suppose you would like to reduce the width of an image, standard techniques such as cropping may remove some prominent features of the image. Ideally, we would like to preserve the most important features of an image even while reducing the width. Suppose that width of an image is W and we would like to re-size the width to $W-1$. Thus we would like to remove one pixel from *each row* of the image, so that the width becomes $W-1$. Which pixels should be removed from each row? Ideally, pictures that are not *important.* How do we decide the importance of a pixel? While there are several approaches, for this PA we consider the following approach to decide importance of a pixel.

Given two pixels $p=[r_1,g_1,b_1]$ and $q=[r_2,g_2,b_2]$, let

$$Dist(p,q) = (r_1-r_2)^2 + (g_1-g_2)^2 + (b_1-b_2)^2$$

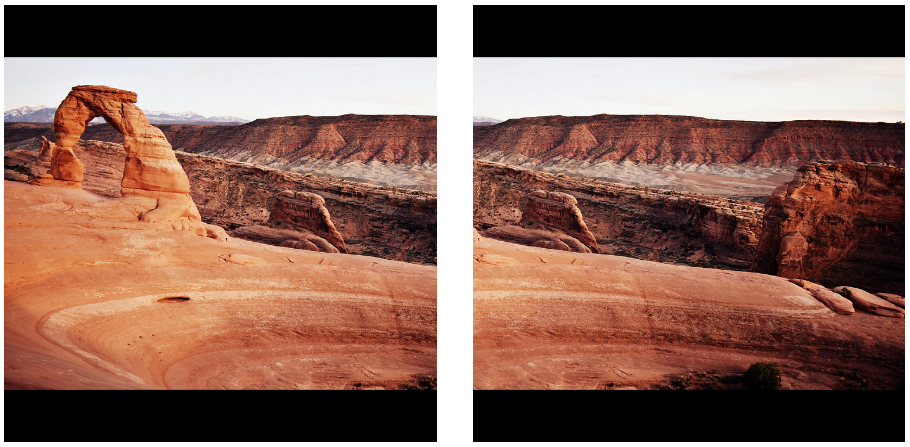

Figure 1: Two Images

Given a picture with width W and height H, let M the matrix that represents the image. Now, given a pixel $M[i, j]$, if $0 < j < W - 1$, its *importance* is defined as below:

$$importance(M[i, j]) = Dist(M[i, j - 1], M[i, j + 1]),$$

if $j = 0$, then

$$importance(M[i, j]) = Dist(M[i, j]M[i, j + 1]),$$

and if $j = W - 1$

$$XImportance(M[i, j]) = Dist(M[i, j]M[i, j - 1]).$$

Now, coming back to our problem of reducing the image width, we will apply the following procedure to reduce width from W to W-1. Given an image whose pixel matrix is M,

- Compute a Matrix named I, where $I[i, j] = Importance(M[i, j])$.
- Compute Min-Cost width cut of I. Let it be $\langle 0, y_0\rangle, \langle 1, y_1\rangle, \cdots, \langle H - 1, y_{H-1}\rangle$.
- For every i, remove the pixel $M[i, y_i]$ from the image. Now the width of the image is $W - 1$.

To reduce the width from W to $W - k$, repeat the above procedure k times.

2.2 Image Stitching

Given two images such as in Figure 1 where the right part of the first image overlaps with the left part of the second image.

The goal is to stitch these two images to create an image as in Figure 2. (This image is from `https://www.flickr.com/photos/iip-photo-archive/39020588482`).

The idea is to find a two paths: path1 in image 1 and path 2 in mage 2 such that the pixel values of images along these two paths are very close to each other. Then we can form a new image by taking all pixels that lie to the left of path 1 in image 1 and the all the pixels that lie to the right of path 2 in image 2. How do we find such paths? This can be reduced to finding stitch cuts in matrices. We will not further delve into details of this process as you will not implement this.

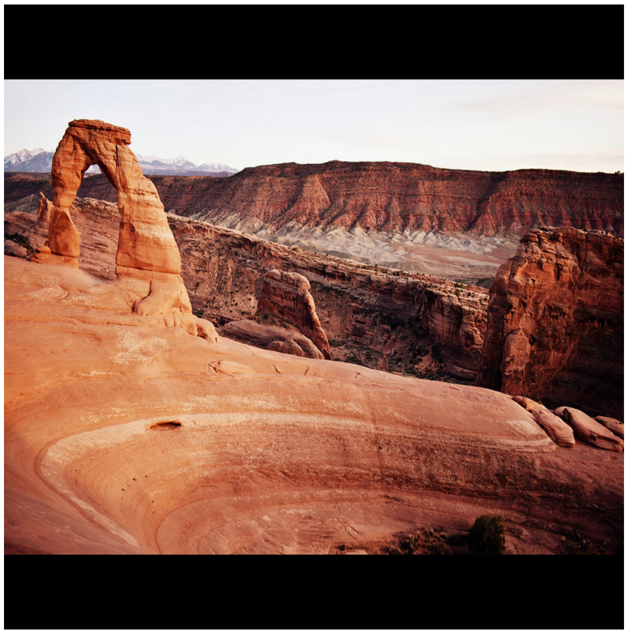

Figure 2: Stitched Image

2.3 Task2: ImageProcessor

You are provided a class named `Picture` that enables you to manipulate images. The `Picture` class is designed by Robert Sedgewick and Kevin Wayne. This class can manipulate `gif, jpg` and `png` images.

Remark. Please note that the addressing mechanism in the `Picture` class differs from the standard convention, $Pixel[x, y]$ refers to the pixel in xth column and yth row of the image.

You will use the notion of width cut to reduce width of images. Write a class named `ImageProcessor` with following `static method`.

`static Picture reduceWidth(int x, String inputImage)` Parameter `inputImage` is the name of image whose width will be reduced. Suppose that the width of this image is W. This method returns a new `Picture` whose width is $W - x$. Note that the type of this method must be `Picture`. Your method **must** use the algorithm described earlier to reduce the image width. And your method must use the static methods from the class **MatrixCuts** to compute the width cut and reduce the width. Otherwise, you will receive **zero** credit.

Remark. The visual quality of the reduced width images crucially depend on the notion of *importance* of a pixel. In this PA, we used a primitive notion of importance, more sophisticated methods will work better.

Image Stitching. You do not have to implement a method to stitch images. You are provided a class `ImageStitcher` that has methods to stitch images. Once you complete your implementation of stitch cut methods, you can play with this class to merge two images. Note that the implementation of this class is not optimized (it duplicates some computations) and is slow on large images. This also assume that the heights of the images are the same. You may use pictures from `http://web.cs.iastate.edu/ pavan/311/F19/images/` for testing.

3 Specifications

Your program must strictly adhere to specifications. The class, package, and method names must be exactly as specified. The return types of methods must be exactly as specified. Types of parameters to the methods must be exactly as specified. Failure to follow specifications will result in a lower grade even if your program works correctly.

You should not use any external libraries/packages, except `Picture.java` and `Tuple.java`. You are not allowed to modify `Picture.java` or `Tuple.java`.

Your grade will depend on : Correctness of the program, Report, and Efficiency of the program. If you do not use **Dynamic Programming** for the matrix cut methods, you will receive **zero** credit.

4 Report

For the methods `widthCutCost` and `stitchCutCost` write the recurrence relation and state the run time of your algorithms.

5 What to Submit

Your submission must be in the form of a zip archive containing

- Your report.pdf (please include names of team members).
- The package directory `pa2`, under which you have the following files:
  - MatrixCuts.java
  - ImageProcessor.java
  - Additional classes that you created.

You must include any additional helper classes that you created. Please include only source .java files, do not include any .class files. **All the code you have written or modified must be in the package `pa2`**. Everything else will be ignored.

Please include all team members names as `@author` tags in the class Javadoc for each of the Java files that you modify or create. **Only one submission per team please. It does not matter which team member does the submission.**