Project 7 (Seam Carving)

Clarifications and Hints

Prologue

Project goal: create a data type that resizes a W-by-H image using a content-aware image resizing technique called *seam carving*, where the image is reduced in size by one pixel of height (or width) at a time

The zip file (`http://www.swamiiyer.net/cs210/seam_carving.zip`) for the project contains

- project specification (`seam_carving.pdf`)
- starter files
  - `SeamCarver.java`
- test script (`run_tests.py`)
- test data (`data/`)
- visualization clients (`ShowEnergy`, `ShowSeams`, and `ResizeDemo`)
- report template (`report.txt`)

This checklist will help only if you have read the writeup for the project and have a good understanding of the problems involved. So, please read the project writeup before you continue with this checklist.

Problems

Your task in this assignment is to implement a mutable data type `SeamCarver`, with the following API:

method	description
`SeamCarver(Picture picture)`	create a `SeamCarver` object from the given picture
`Picture picture()`	current picture
`int width()`	width of current picture
`int height()`	height of current picture
`double energy(int x, int y)`	energy of pixel at column x and row y
`int[] findHorizontalSeam()`	sequence of indices for horizontal seam
`int[] findVerticalSeam()`	sequence of indices for vertical seam
`void removeHorizontalSeam(int[] seam)`	remove horizontal seam from current picture
`void removeVerticalSeam(int[] seam)`	remove vertical seam from current picture

Hints

- Instance variable
    - The picture to carve, `Picture picture`
- `SeamCarver(Picture picture)`
    - Initialize instance variable appropriately

Problems

- `Picture picture()`
    - Return the picture
- `int width()`
    - Return the width W of the picture
- `int height()`
    - Return the height H of the picture

Seam carving involves three parts: energy calculation; seam identification; and seam removal

Problems

Problem 1 (*Energy Calculation*) Implement `energy()` to calculate the energy of a pixel, which is a measure of its importance.

Hints

- Return the energy of the pixel (x, y), calculated as $\Delta_x^2(x, y) + \Delta_y^2(x, y)$, where $\Delta_x^2(x, y) = R_x^2(x, y) + G_x^2(x, y) + B_x^2(x, y)$, and where $R_x(x, y), G_x(x, y)$, and $B_x(x, y)$ are the absolute value in differences of red, green, and blue components between pixel $(x + 1, y)$ and pixel $(x - 1, y)$; analogous definition holds for $\Delta_y^2(x, y)$
- To handle pixels on the borders of the image, calculate energy by defining the leftmost and rightmost columns as adjacent and the topmost and bottommost rows as adjacent (use the helper method `int wrap(int x, int y)` for this); for example, to compute the energy of a pixel $(0, y)$ in the leftmost column, use its right neighbor $(1, y)$ and its "left" neighbor $(W - 1, y)$

Problems

Problem 2 (*Seam Identification*) Implement `findVerticalSeam()`/`findHorizontalSeam()` to find a vertical/horizontal seam of minimum total energy.

Hints

- `int[] findVerticalSeam()`
  - This is similar to the shortest path problem in an edge-weighted digraph, but there are three important differences:
    - The weights (energies) are on the vertices instead of the edges
    - The goal is to find the shortest path (vertical seam) from any of the W pixels in the top row to any of the W pixels in the bottom row
    - The digraph is acyclic, where there is a downward edge from pixel (x, y) to pixels $(x-1, y+1)$, $(x, y+1)$, and $(x+1, y+1)$, assuming that the coordinates are in the prescribed ranges — seams cannot wrap around the image
    - Since the digraph is acyclic, a shortest path can be found efficiently by relaxing the vertices in topological order
  - Create a 2D array `int[][] distTo` with dimensions H-by-W, and initialize the first row to the energies of the corresponding pixels from `picture`
  - Relax the vertices in row-major order of pixels in `picture`, starting from the second row
  - Identify the value of `i` such that `distTo[H - 1][i]` is minimum
  - Backtrack starting from `(H - 1, i)` and ending on the first row to retrieve the shortest path (ie, a vertical seam) in an array `int[] seam` and return the array
- `int[] findHorizontalSeam()`
  - Construct a `SeamCarver` object `sc` from the transpose of `picture` (use the helper method `Picture transpose(Picture picture)` for this)
  - Return a vertical seam obtained from `sc`

Problems

Problem 3 (*Seam Removal*) Implement `removeVerticalSeam()`/`removeHorizontalSeam()` to remove from the image all of the pixels along the vertical/horizontal seam.

Hints

- `void removeVerticalSeam(int[] seam)`
  - Create a `Picture` object `temp` with dimensions $(W - 1, H)$
  - For $0 <= i < W$ and $0 <= j < H$, copy pixel `(i < seam[j], j)` from `picture` into pixel `(i, j)` in `temp`, and any other pixel `(i, j)` from `picture` to pixel `(i - 1, j)` in `temp`
  - Set `picture` to `temp`
- `void removeHorizontalSeam(int[] seam)`
  - Construct a `SeamCarver` object `sc` from the transpose of `picture`
  - Remove the vertical seam `seam` on `sc`
  - Set `picture` to the transpose of the picture obtained from `sc`

Epilogue

The `data` directory contains several sample input files for testing, along with some reference solutions

We provide three visualization clients that we highly recommend using for testing and debugging your solutions

1. `ShowEnergy` takes the name of an image as a command-line argument and displays the pixel energies on the screen

   ```
   $ java ShowEnergy data/mandrill.jpg
   ```

2. `ShowSeams` takes the name of an image as a command-line argument and displays the minimum-energy horizontal and vertical seams on the screen

   ```
   $ java ShowSeams data/mandrill.jpg
   ```

3. `ResizeDemo` takes the name of an image and number of horizontal and vertical seams as command-line arguments, carves out those many seams from the image, and renders the resized image along with the original image

   ```
   $ java ResizeDemo data/mandrill.jpg 50 50
   ```

Epilogue

Your project report (use the given template, `report.txt`) must include

- time (in hours) spent on the project
- short description of how you approached each problem, issues you encountered, and how you resolved those issues
- acknowledgement of any help you received
- other comments (what you learned from the project, whether or not you enjoyed working on it, etc.)

Before you submit your files

- make sure your programs meet the input and output specifications by running the following command on the terminal

  ```
  $ python run_tests.py -v [<problems>]
  ```

- make sure your programs meet the style requirements by running the following command on the terminal

  ```
  $ check_style <program>
  ```

- make sure your report isn't too verbose, doesn't contain lines that exceed 80 characters, and doesn't contain spelling/grammatical mistakes