**Seam carving**

**Problem Statement**

Seam-carving is a content-aware image resizing technique where the image is reduced in size by one pixel of height (or width) at a time. A vertical seam in an image is a path of pixels connected from the top to the bottom with one pixel in each row; a horizontal seam is a path of pixels connected from the left to the right with one pixel in each column.

**Vertical Seam**

The findVerticalSeam() method returns an array of length H such that entry y is the column number of the pixel to be removed from row y of the image.

**Horizontal Seam**

The behavior of findHorizontalSeam() is analogous to that of findVerticalSeam() except that it returns an array of length width such that entry x is the row number of the pixel to be removed from column x of the image.

**Computing seams**

Computing the seam consists of finding the path of minimum energy cost from one end of the image to another. This can be done via Dijkstra's algorithm, dynamic programming, greedy algorithm or graph cuts etc.

**This program expects us to**

* Find the Vertical Seam
* Find the Horizontal Seam
* Removing Vertical Seam
* Removing Horizontal Seam

**Related Concepts**

Use Dynamic programming method which stores the results of sub-problems in order to simplify the calculation of a complex result. In this case, it is equivalent to Topological Sort Algorithm.

**Test Cases**

* One Timing test cases is failed due to excessive calls to getRGB() method in Picture Class.
* Few test cases for findVerticalSeam(), findHorizontalSeam() are failed.
* The Score for the Project is 85 / 100.

**API**

public class SeamCarver {

**Time Complexity** : Proportional to Picture Width and Height

**Space Complexity :** Proportional to Picture Width and Height

// create a seam carver object based on the given picture

public SeamCarver(Picture picture)

// current picture

**Time Complexity** : Constant

**Space Complexity** : Constant

public Picture picture()

// width of current picture

**Time Complexity** : Constant

**Space Complexity** : Constant

public int width()

// height of current picture

**Time Complexity** : Constant

**Space Complexity** : Constant

public int height()

// energy of pixel at column x and row y

**Time Complexity** : Constant

**Space Complexity** : Constant

public double energy(int x, int y)

// sequence of indices for horizontal seam

**Time Complexity** : Proportional to Picture Width and Height

**Space Complexity** : Proportional to Picture Width and Height

public int[] findHorizontalSeam()

// sequence of indices for vertical seam

**Time Complexity** : Proportional to Picture Width and Height

**Space Complexity** : Proportional to Picture Width and Height

public int[] findVerticalSeam()

// remove horizontal seam from current picture

**Time Complexity** : Proportional to Picture Width and Height

**Space Complexity** : Proportional to Picture Width and Height

public void removeHorizontalSeam(int[] seam)

// remove vertical seam from current picture

**Time Complexity** : Proportional to Picture Width and Height

**Space Complexity** : Proportional to Picture Width and Height

public void removeVerticalSeam(int[] seam)

}