**Seam carving**

**Problem Statement**

Seam carving is an algorithm for content-aware image resizing. In this image is reduced in size by one pixel of height or width at a time. If any one of the image dimensions are one then resizing does not takes place. The purpose of this algorithm is image retargeting, which is the problem of displaying images without distortion on media of various sizes.

**Vertical Seam**

A vertical seam in an image is a path of pixels connected from the top to the bottom with one pixel in each row.

**Horizontal Seam**

A horizontal seam is a path of pixels connected from the left to the right with one pixel in each column.

**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**

1. Use Dynamic programming method which is a programming method that stores the results of sub-calculations in order to simplify calculating a more complex result. In this case it is equivalent to Topological Sort Algorithm.

**Test Cases**

1. All test cases are passed and scored 100/100

**API**

public class SeamCarver {

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

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

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

public SeamCarver(Picture picture)

// current picture

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

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

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)

}