**Seam carving**

**Problem Statement**

Seam carving is a calculation for content-mindful picture re sizing. Right now diminished in size by each pixel of tallness or width in turn. In the event that any of the picture measurements are one, at that point re sizing doesn't happens. The reason for this calculation is picture re focusing on, which is the issue of showing pictures without contortion on media of different sizes.

**Vertical Seam**

A vertical seam in a picture is a path of one pixel in each row connected from top to bottom.

**Horizontal Seam**

A horizontal seam in a picture is a path of one pixel in each row connected from left to right.

**Computing seams**

Computing the seam comprises of finding the way of least vitality cost starting with one finish of the picture then onto the next. This should be possible by means of Dijkstra's calculation, dynamic programming, greedy calculations or diagram cuts and so on

In this program following functionalities were implemented:

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

**Related Concepts**

Utilize Dynamic programming strategy which is a programming technique that stores the results of sub-computations so as to streamline ascertaining an increasingly mind boggling result. Right now is identical to Topological Sort Algorithm.

**Test Cases**

There is some glitch in energy calculation and difference of 1 and less than 1 was occurring when compared to actual energy except this all the test cases were passed.

**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 :** 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)

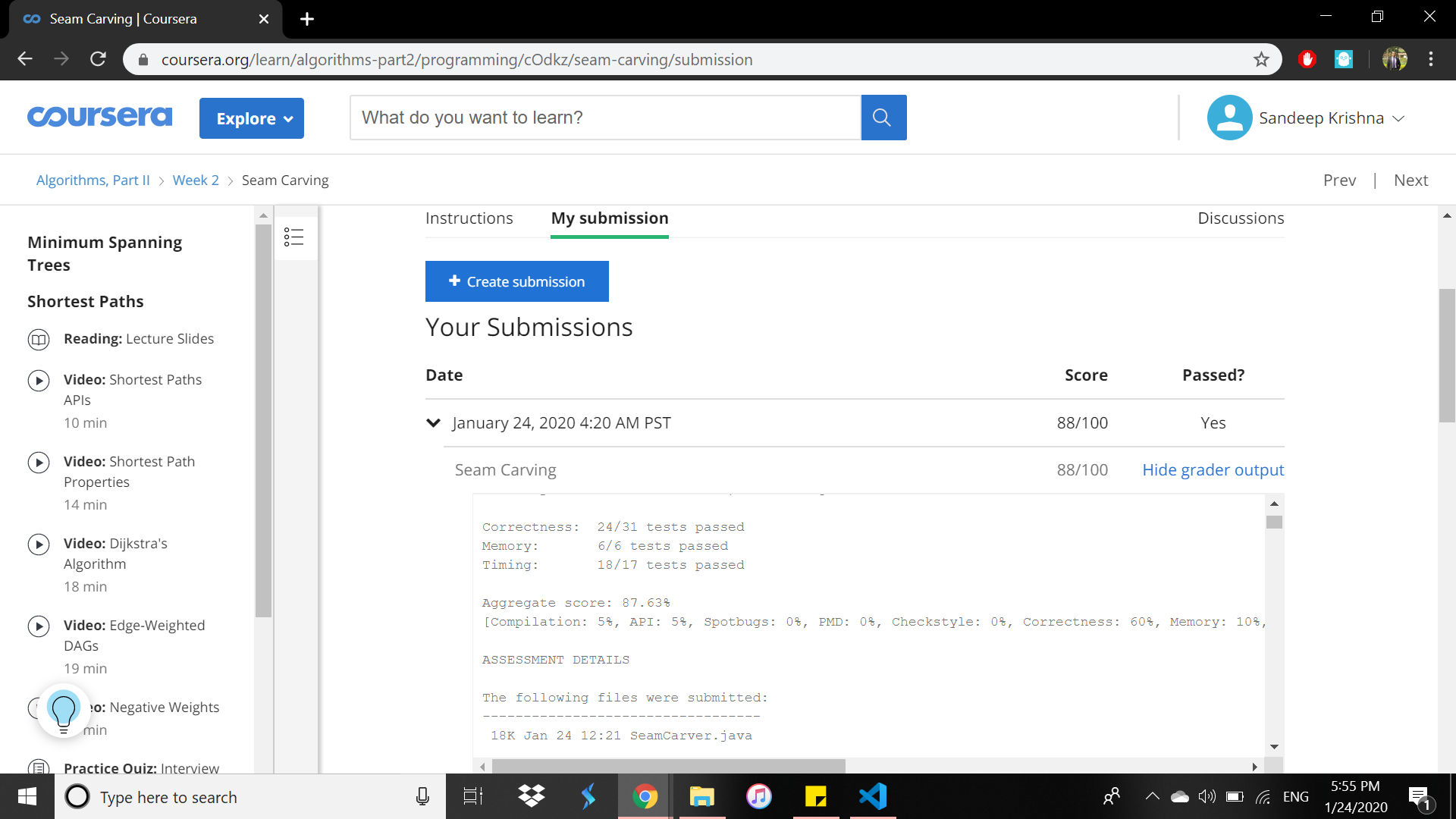
// remove vertical seam from current picture

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

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

public void removeVerticalSeam(int[] seam)

}



**2019501035**

**Sandeep Krishna.**