Seam Carver

To find vertical seams I create an array of Vertices (a data type I created) and initialize all of them. Setting their total energies, the amount of energy required to get to that vertex, to max double. Except for the first row, for which I set their total energies to 0. I then go through every pixel. If it is not on the bottom row I check all three of its children (the pixel below and to the left, the pixel below, and the pixel below and to the right). If any of the children’s total energy is greater than the current pixels total energy plus its own energy then set that child’s total energy to the current pixel’s total energy plus its energy. If the current pixel is on the bottom row I compare it to a global variable the represents the smallest energy. I also have a variable that stores the index of the pixel with the smallest energy. This finds the pixel with the smallest path. Once I have gone through every pixel I have the index of the shortest path. With that I add all of the parents to an int array that represents the seam. This stores the cheapest seam. I then return this seam.

To find horizontal seams I do the same thing, but rather than checking the bottom left, bottom, and bottom right; I check the right top, right, and right bottom. And I sort the right most pixels in each row.

To remove vertical seams I create a new Picture that is one column smaller than the original, and then I iterate through every pixel of the original image. If the current pixel’s x is greater than or equal to the path’s x at the current index I set the new image’s xth and yth pixel to the original image’s next pixel (x + 1 and y). This moves all of the pixels to the right of the path left one pixel.

I do the same thing for removing horizontal seams, except I make the new image’s height one less rather than the width. And I check to see if the current y is greater than the path’s y at the current index. If it is then I move every pixel below it up one pixel. Removing the bottom row and pushing every pixel below the path up one pixel.

My algorithm for finding seams operates in Θ(WH) because if iterates through every pixel. It does so twice. Once to initialize the Vertices, and once to interpret the pixels data and generate the topo graph. It then iterates through an entire path (of length W or length H) to create the seam. Meaning that in all my algorithm for finding seams is Θ(2WH + W) for horizontal seams and Θ(2WH + H) for vertical seams. But because WH is the largest power, the overall Θ() is Θ(WH).

My algorithm for removing seams is also Θ(WH) because I iterate through every pixel to create the new image.