Benchmarks:
I have benchmarked both my naive implementation of the seam carving algorithm on 7 different image sizes (5, 10, 11, 12, 13, 14, and 15 seconds) and my dynamic programming implementation of the seam carving algorithm on 9 different image sizes (100, 150, 200, 250, 300, 500, 750, 1,000, and 1920x1080), and then have plotted the obtained run times onto a graph with a best fit curve and R^2 value, the individual run times are listed in the tables below:

Naive Algorithm:

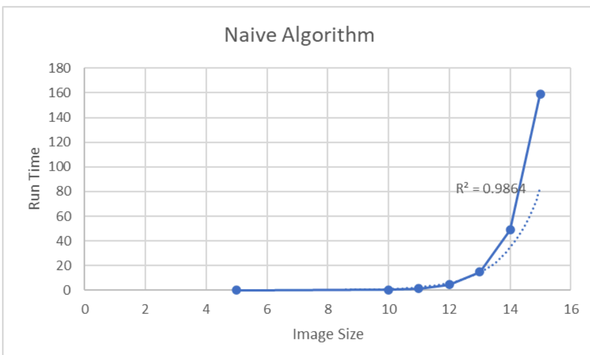
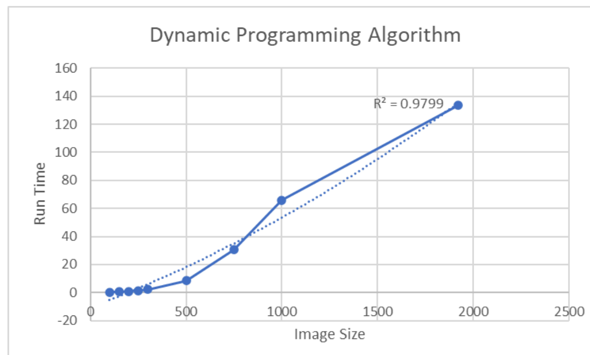
Input Size	Run Time
5x5 Pixel Image	0.028 seconds
10x10 Pixel Image	0.435 seconds
11x11 Pixel Image	1.391 seconds
12x12 Pixel Image	4.563 seconds
13x13 Pixel Image	15.041 seconds
14x14 Pixel Image	48.970 seconds
15x15 Pixel Image	158.983 seconds

Dynamic Programming Algorithm:

Input Size	Run Time
100x100 Pixel Image	0.201 seconds
150x150 Pixel Image	0.466 seconds
200x200 Pixel Image	0.852 seconds
250x250 Pixel Image	1.426 seconds
300x300 Pixel Image	2.138 seconds

500x500 Pixel Image	8.44 seconds
750x750 Pixel Image	30.497 seconds
1,000x1,000 Pixel Image	65.945 seconds
1920x1080 Pixel Image	133.372 seconds

Graphs:

Naive Algorithm (Exponential Trend Line)	Dynamic Programming Algorithm (Order 2 Polynomial Trend Line)
	

As per the graphs above, my findings are that the naive version of this algorithm runs in exponential time and that the dynamic programming version of the algorithm runs in $O(n^2)$ time.

Analysis:

Without Dynamic Programming:

As per the graphs above, my findings are that the naive version of this algorithm runs in exponential time. This is because the algorithm utilizes recursion to function, and as the input size increases, so does the amount of recursive calls which will scale exponentially.

Subproblems: For each pixel (i, j) in the image, we need to find the lowest-energy seam starting from the top row and ending at (i, j).

Recurrence Relation: $N(i, j) = N(i - 1, j) + N(i - 1, j - 1) + N(i - 1, j + 1) + O(1)]$

Recurrence Relation Solved: $O(i * 3^j)$

With Dynamic Programming:

As per the graphs above, my findings are that the dynamic programming version of the algorithm runs in $O(n^2)$ time. For the dynamic programming algorithm, we do a constant amount of work per subproblem, thus assuming an image of i width and j height, we get a time complexity of $O(i * j + i)$. This will average out to $\sim O(n^2)$ time complexity over the course of using the algorithm on a wide variety of images of different sizes.

Subproblems: We still need to find the lowest-energy seam for each pixel (i, j).

Recurrence Relation: $dp[i, j] = \min(dp[i - 1, j - 1], dp[i, j - 1], dp[i + 1, j - 1]) + energy(i, j)$

Recurrence Relation Solved: $O(i * j + i)$

I believe these results make sense given the results of my benchmarking tests. The solved recurrence relations equate to an exponential equation and an order 2 polynomial respectively. These answers match up accurately with the two graphs I obtained from my benchmarking results.