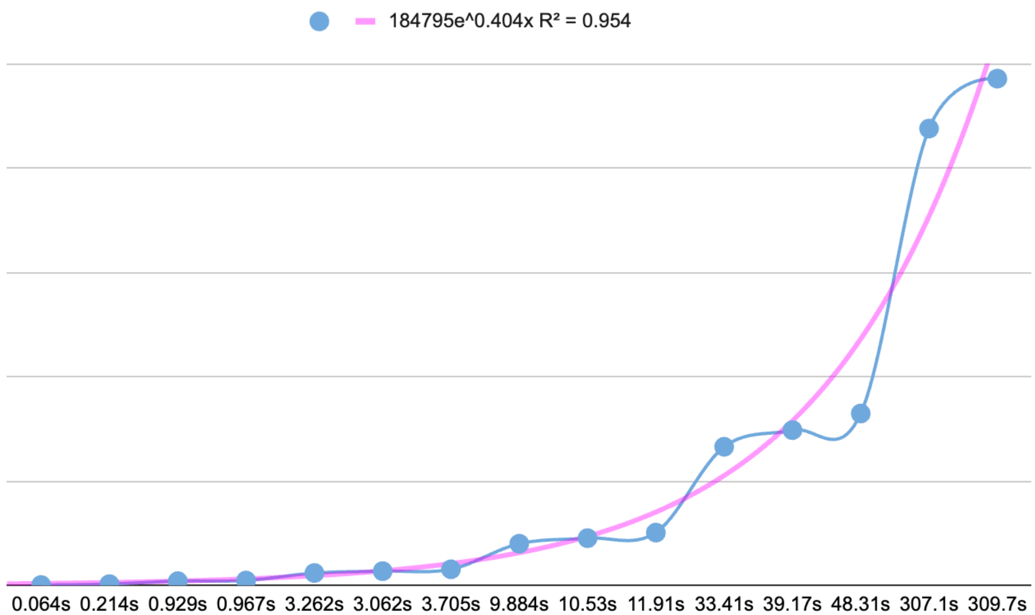
CSC 440 Assignment 5

Greedy algorithm benchmark

time	# of seams
0.064s	35112
0.214s	120556
0.929s	408085
0.967s	467134
3.262s	1188275
3.062s	1365421
3.705s	1542568
9.884s	3993259
10.53s	4524699
11.91s	5056140
33.41s	13278007
39.17s	14872329
48.31s	16466652
307.1s	43762948
309.7s	48545916



Limits include: A high **time** complexity: it takes way too long to find the best seam. My program never stops running for images as small as 20 by 20.

- Sometimes, the "best" seam might not be the one with the lowest energy. it could be the one with the highest when the main element is low energy in a high energy background

We can infer an exponential asymptotic complexity from how fast the running time increments which is closely related to the total number of possible seams.

Analysis: The number of possible seams grows exponentially as the number of heights increases. For each new row added, for each possible seam, there are about 3 possible new seams. The total number of seams at height 1, is proportional to the width of the image.

Based on our implementation and benchmark we know that the number of iteration or running time is related to the number of seams so **O(w . 3^h)**

The number of possible seams for a pixel in an image is a recurrence relation T(h) with h the height of the pixel and k <=3 the number of adjacent lower pixels. **T(h) = k . T(h - 1)**

That relation gives us a ternary tree of with a depth $\log_3$ (h), so the number of possible seams (leaves) is 3^h for k always equal to 3 ⇔ **T(h) = 3^h**

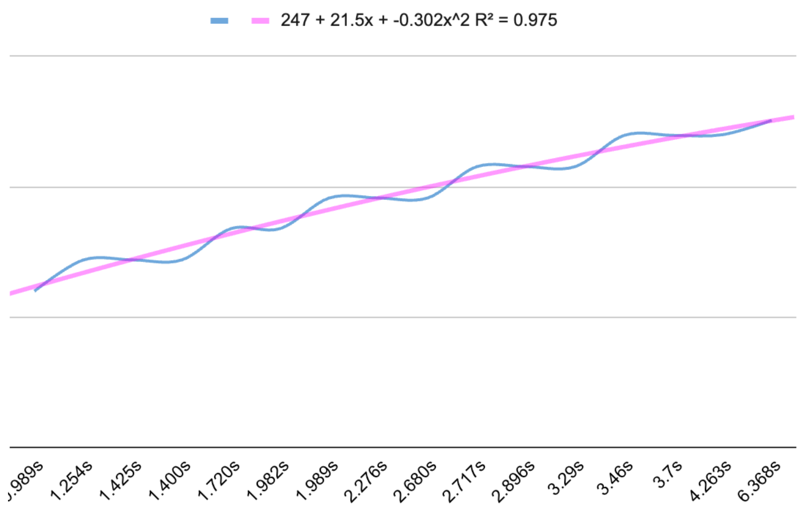
For an image of width w we have about ⇔**w . T(h) = w. 3 ^ h.** (big approximate due to edges)

This is confirmed by a highest r squared value for exponential functions of 0.954

Yann Youbi

Dynamic programming benchmark and analysis

time	# of seams	height	width
1.254s	10^288	600	500
1.425s	10^288	600	600
1.400s	10^288	700	600
1.720s	10^336	600	700
1.982s	10^336	700	700
1.989s	10^383	800	700
2.276s	10^383	700	800
2.680s	10^383	800	800
2.717s	10^431	900	800
2.896s	10^431	800	900
3.29s	10^431	900	900
3.46s	10^479	1000	900
3.7s	10^479	900	1000
4.263s	10^480	1000	1000
6.368s	10^502	1500	1000
8.801s	10^593	2000	1200



Despite being faster, the dynamic approach has a higher **space** complexity as we have to store a matrix with minimum cumulative energy for each seams starting at each pixel. It also takes more lines of code to implement, also the program only does vertical seams.

We notice that the running time is no longer related to the exponentially increasing number of possible seams. Instead it increases in a linear way, proportionally to the total number of pixels in the image w * h, with the height h having more impact than the width w.

This is due to the fact that after iterating through each pixel for the lowest cumulative energy from bottom to top. I iterate through all the pixels in the best seam of height h.
So, we infer a polynomial time **O(w . h + h)**

Which can be confirmed with a r squared value for polynomial of 0.975.

Yann Youbi