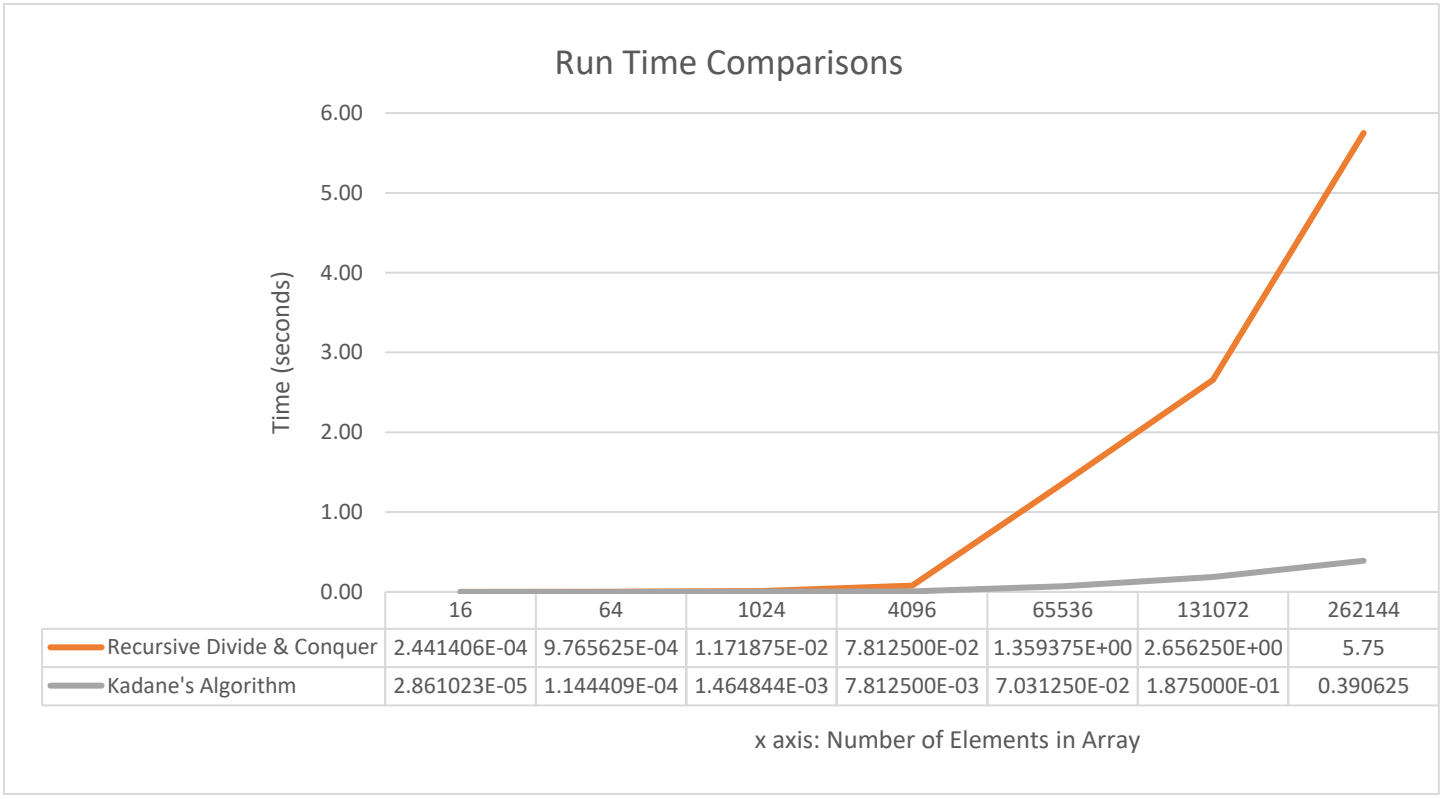
Rachel Pierstorff
COMP 2370
October 4, 2018

Assignment 3 Analysis

The graph below compares the run times between each of the two algorithms for finding the maximum subarrays. As you can see, the recursive algorithm is slower as the size of the input array grows. This is because the recursive algorithm has a time complexity of $\Theta(n * \log(n))$ while Kadane's algorithm has a time complexity of $\Theta(n)$.

For Kadane's algorithm, each element in the array only needs to be looked at once; that loop keeps track of both the maximum sum to k and max sum so far, as well as the minimum sum to k and minimum sum so far, which is used to identify the maximum subarray of a circular array.

For the recursive divide and conquer algorithm, on the other hand, the input array is divided in several subarrays and calls itself with different low, mid, and high indices. This process identifies the maximum subarray in each left and right subproblem, as well as a potential maximum subarray that crosses the boundary between the left and right. In order to account for a potentially circular array, I ran the same algorithm again after modifying the array so that the original midpoint acted as i=0, and so the first half of the array was tacked on to the end. In other words, the modified array allowed the algorithm to check for maximum subarrays that cross the boundary between the original first and last elements of the array. This adds a constant to the running time, but does not change the overall time complexity.



	16	64	1024	4096	65536	131072	262144
Recursive Divide & Conquer	2.441406E-04	9.765625E-04	1.171875E-02	7.812500E-02	1.359375E+00	2.656250E+00	5.75
Kadane's Algorithm	2.861023E-05	1.144409E-04	1.464844E-03	7.812500E-03	7.031250E-02	1.875000E-01	0.390625