Algorithm implementation:

The main built in python functions and modules I found useful for developing these algorithms were; math, sorted and TKinter. The other method I chose to implement was Monotone Chain.

Math

I found the python math module to be very useful in the Gift-Wrap and Graham Scale algorithms for it includes the tangent function, making it very easy to calculate angles between a point and a line. In Gift-Wrap this made getting the angles of all the different points with the previous point and comparing it to the previous angle very easy to implement and check. In Graham Scale this made getting the angles of the points with a horizontal reference line easier and therefore sorting the points by that line much simpler. On top of this the math module also contains a method to convert radians to degrees which I found useful for I am far more confident with degrees than radians, so made error checking far easier.

Sorted

The sorted method was very useful in sorting the points into any order that I needed for the particular algorithms. The fact that you can easily change the key in which the method will sort the collections by became especially useful in Graham Scale for I created a dictionary of all the points and assigned their value to the angle between them and the horizontal reference line. Then I used the sorted method to sort the points in the dictionary into a list by its respective angle value. Another useful feature of this method is being able to add the parameter reverse, to easily reverse the order of the resulting list.

TKinter:

I found TKinter especially useful in verifying if the results I had were correct or if they were incorrect, then how. I used TKinter to plot all the points in the input file onto a canvas as a red dot, then to draw lines between all the points that my program returned as part of the convex hull. When the output was correct, this made it very quick to check. When the output was incorrect, this gave me valuable information into where the problem may be based of where the lines were drawn. This helped me may times to find and fix bugs.

Monotone Chain

I found Monotone Chain an interesting algorithm and a good one to implement as my final method. The basis of how it works is it creates two, half convex hulls. A top and a bottom, which when combined will create a full convex hull. This works by first ordering all points by their x value (Then y value in equal x values), then creates the upper and lower "half hulls".

For the lower half it iterates through all points and for each point, firstly checks if there are at least two points in the current calculated points of the lower half. If it does, it will check if the last two points in the current calculated points of the lower half and the current point in the iteration make an anti-clockwise turn. If they do not, then the last point in the currently calculated lower half will be removed for it is not part of the convex hull. These statements will continue to loop until one of the above conditions are not met. When that occurs the current point in the iteration is added to currently calculated points for the lower half and the process loops for the next point.

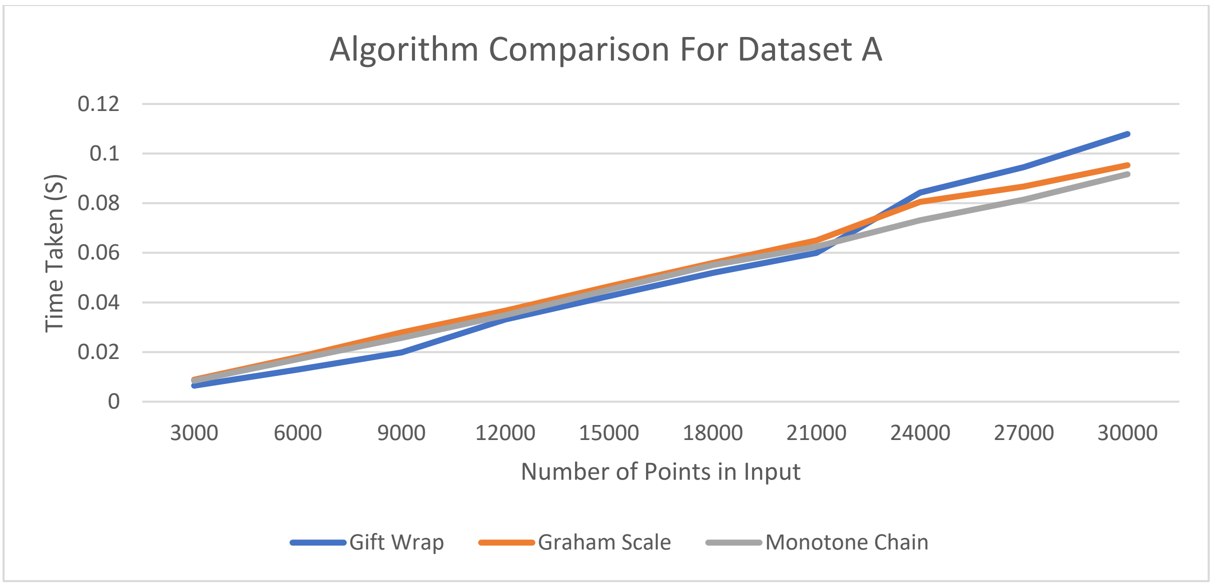
Since all the points are ordered by their x value, we know that the first point in the iteration for the lower half is the left most point and since it iterates through all points, it must end at the right most point.

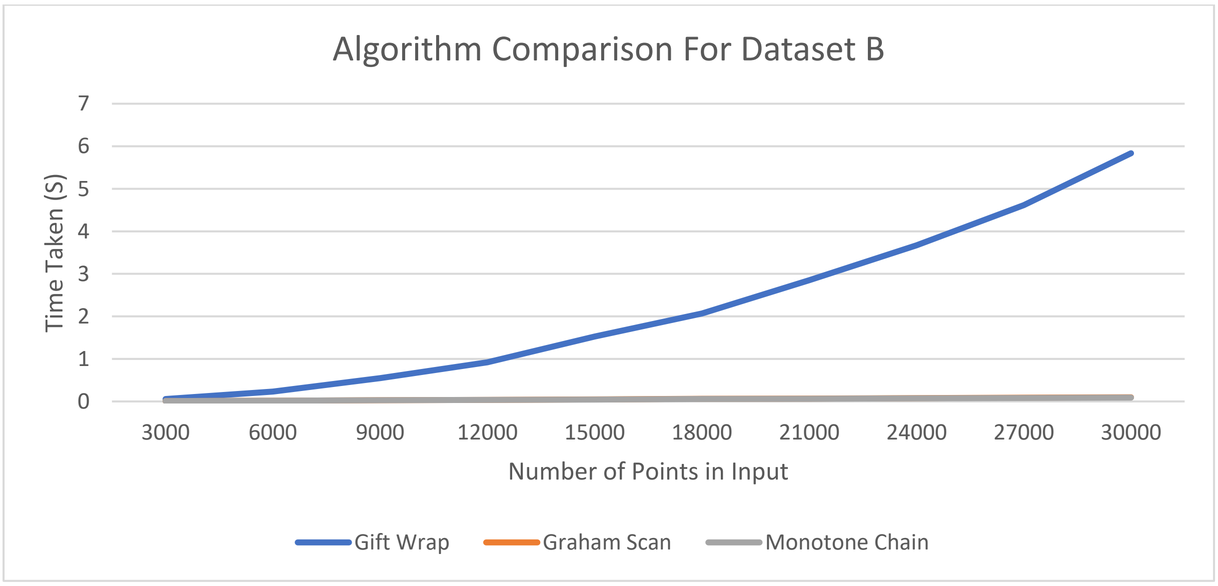
For the upper half the exact same method is used but this time starting at the right most points, rather than the left most, and iterating through the points in reverse. This means that instead of starting on the left most point and ending on the right most, the upper half will start on the right most point and end on the left most.

The lower and upper halves will now be a complete half of the total convex full, however we still need to remove the last point from each of the half. This is because both halves contain the two points where they connect, or in other words the last point in each half is the first in the other. Then concatenate the lower and the upper halves to obtain the full convex full with the points visited in anti-clockwise order, starting at the left most point.

Algorithm analysis:

Examining the datasets, A and B we can tell a clear difference between them. Dataset A appears to keep a reasonably constant amount of points that are part of the convex hull, with only three for the three thousand data points file and five for the thirty thousand data points file. Dataset B has a significant growth in the amount of points that are part of the convex hull, sixty more than the previous size datafile, or 2% of the number of points in the datafile. This goes from 60 (in the three thousand data points file) to 600 (in the thirty thousand data points file) data points that are part of the convex hull.

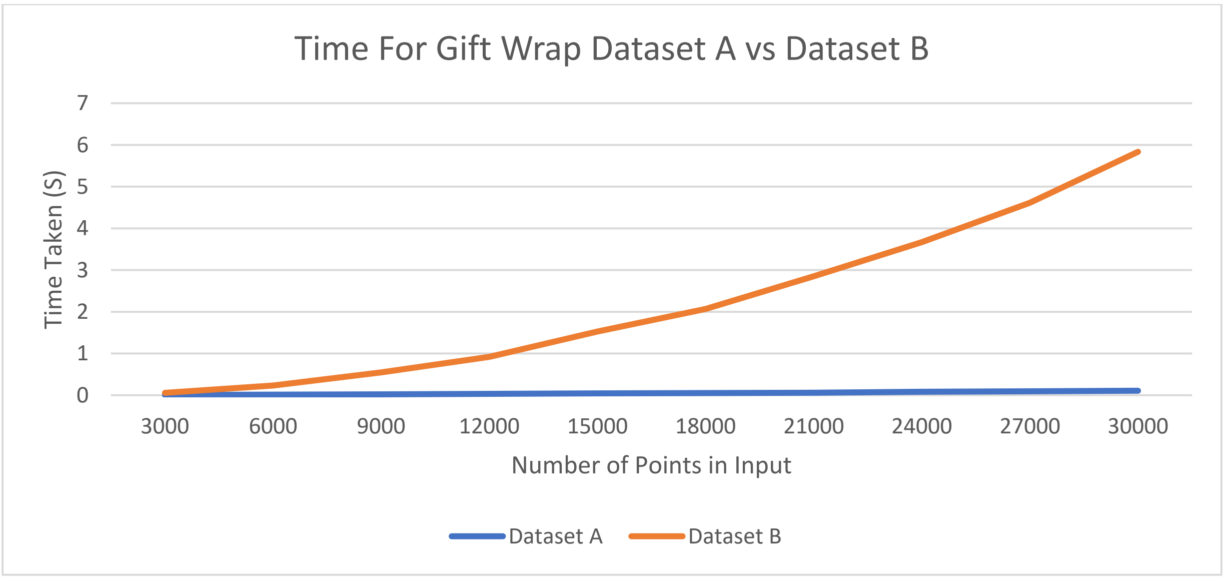




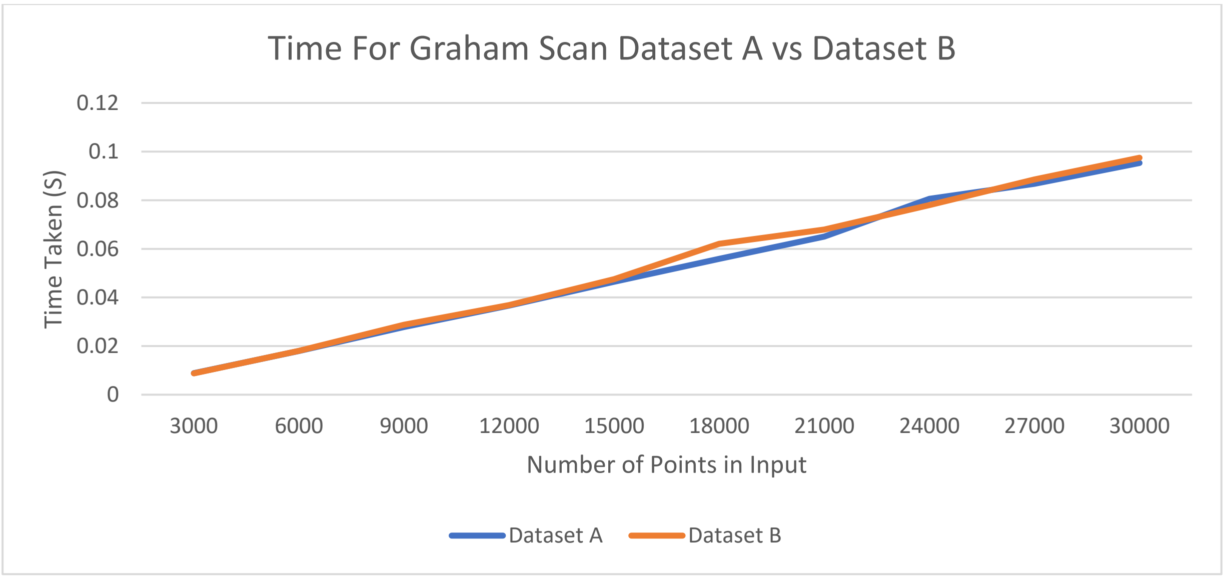
Looking at the above graphs, regarding dataset A all the algorithms are relatively similar. Gift Wrap appears to be slightly faster for total number of data points less than twelve thousand and slightly slower for total number of data points more than twenty-four thousand. Graham Scan and Monotone Chain are very similar in this dataset with Monotone Chain being negligibly faster with datasets of size twenty-one thousand or lower. Higher than twenty-one thousand, Monotone Chain appears to be slightly faster than Graham Scan.

Regarding dataset B in the above graph no difference can be seen between Graham Scan and Monotone Chain. However, Gift Wrap takes exponentially more time to complete as the total number of data points raise.

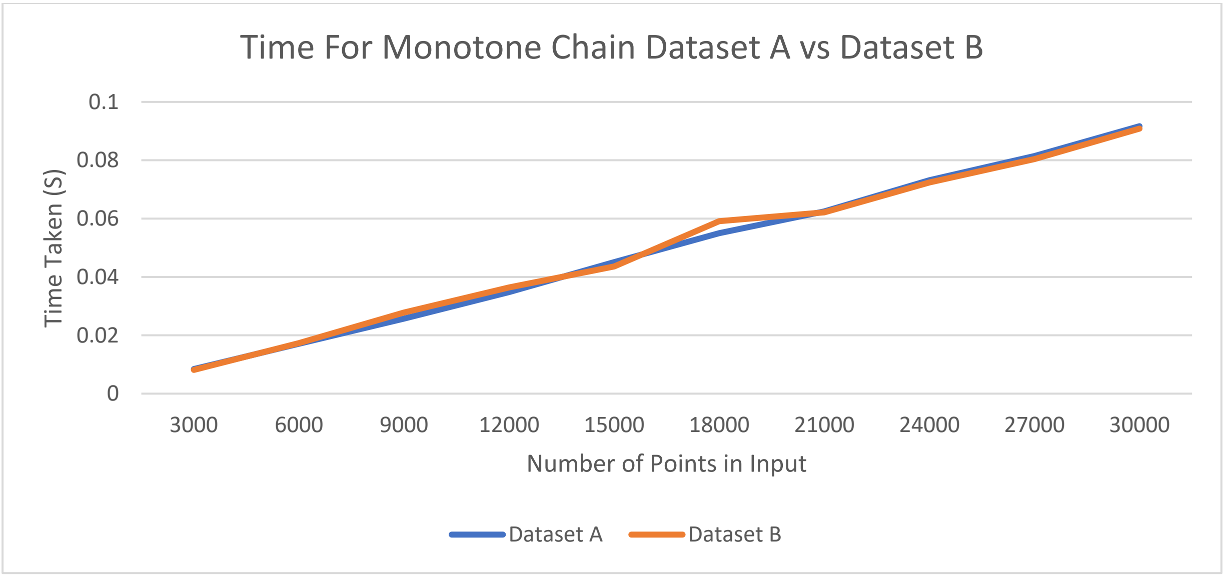
Overall the algorithm with the best average performance is Monotone Chain. However, if the dataset is small and it is known that there will not be very many points that are part of the convex hull then Gift Wrap may be more suitable.



Looking at the above graph dataset A appears to be a reasonably linear growth for time taken based on the total number of data points. However, for dataset B Gift Wrap appears to take an exponentially large amount of time for the increasing total number of data points. This suggests a correlation between the time taken for the Gift Wrap algorithm and the number of data points that are part of the convex hull for that is the main difference between the two data sets.



Looking at the above graph, the time taken by Graham Scan for both datasets A and B very similar for the same number of data points and both follow an almost linear growth with the total number of data points. This suggests that Graham Scan is not affected by the number of data points in the convex hull and rather only affected by the total number of data points.



Looking at the above graph, the time taken by Monotone Chain for both datasets A and B extremely similar for the same number of data points and both follow an almost linear growth with the total number of data points. This suggests that Monotone Chain is not affected by the number of data points in the convex hull and rather only affected by the total number of data points.

References:

1) http://www.algorithmist.com/index.php/Monotone_Chain_Convex_Hull
This was very useful into completely understanding the Monotone Chain algorithm and how I should go about implementing it.

2) https://docs.python.org/3/library/timeit.html
This was the in-built module I used to time the different algorithms. I used the official docs to figure out how to use it best for my situation.

3) https://www.saltycrane.com/blog/2007/09/how-to-sort-python-dictionary-by-keys/

I used this site to figure out how to sort a dictionary into a list of its keys, with the sorting based off the value associated to that key.