# COSC262 Assignment: Convex Hulls

## Algorithm Implementation

### Gift Wrap algorithm

This implementation of the giftwrap algorithm uses two helper functions, one Theta() which gives an approximation of the angle between two points, and another to find the point of minimum y coordinate in the input points. Theta() takes two input points - A and B, and returns an approximation of an angle. The approximation calculated is the angle between the line AB and a horizontal line through A. The other helper function takes a list of points and returns the point with minimum y coordinate. A nested for-if structure is used to loop over all points and then check if the current point has a smaller y-coordinate than the current minimum point. If there are multiple points of minimum y coordinate, the rightmost point is returned. A non-recursive nested while-for structure along with a python list is used to construct the convex hull. A variable holds the angle of the previously added convex hull point which is minimum angle allowed for the next comparison. Two index variables are also used to hold the current point being compared and the point in the hull with minimum angle. ("Convex Hull | Set 1 (Jarvis's Algorithm or Wrapping) - GeeksforGeeks", n.d.)

### Graham-Scan algorithm

This implementation of the Graham-Scan algorithm uses three helper functions to compute a convex hull. The two functions used in the Gift Wrap algorithm to calculate an angle approximation and the minimum y coordinate are used as well as another to check if a turn is counter clockwise. The counter clockwise function uses the line function described in lectures to return a boolean value holding if the line makes a counter clockwise turn or not. The implementation also makes use of lambda functions to sort the points by their angle. A non-recursive nested for-while structure along with a stack represented as a python list is used to construct the convex hull. ("The Convex Hull of a Planar Point Set", n.d.)

### Monotone Chain algorithm

The Monotone chain algorithm uses a non-recursive structure to construct a convex hull. The input points are first sorted by their x-coordinates. If there are points of equal x-coordinate they are then ordered by their y-coordinate. The lower and upper convex hulls are then constructed using a for-while stack based structure very similar to the Graham-Scan algorithm, where the points are reversed when constructing the upper convex hull. The two convex hulls are then concatenated into the final convex hull which is returned.

This implementation makes use of one helper function. The function is used calculate the cross product of given points. The cross-product function is used as a comparator for the condition inside the loop in which the hulls are constructed. Both the upper and lower convex hulls are constructed using a nested for-while loop structure, which performs operations on a stack that holds the convex hull after construction. The implementation also has a base case catch which checks if the input points are of length 0 or 1. If this condition is met, the input points are returned as the convex hull. ("Monotone Chain Convex Hull - Algorithmist", n.d.)

## Algorithm Analysis Implementation

A testing suite was implemented to help with validating and analysing the algorithms. The testing suite comprises of two functions. The first is average_tests(), this function was used to get the average time value of the algorithms over all data sets. It uses the modified convex hull file convexhull_time.py as this file has the additional code which returns the time taken to construct the convex hull. The average_tests() function takes two inputs. One which holds the number of times each algorithm must be run, and another which is a boolean flag where True indicates that data set A will be tested, and False is for data set B. Each algorithm is run with the selected data sets, the selected amount of times. A list for each algorithm is output to the command line, each list holds the average time taken for each file. The other function output_tests() takes no inputs and uses the original convexhull.py file. It validates that each algorithm gives the expected result for each data file in all data sets. The function works through the data sets file by file running each algorithm on each file. When a file is run, the file name and an indication of whether each algorithm gave the expected results or not is output to the command line. Another file Graphs.py was produced, this file used the matplotlib library to graph the results of the analysis.

# Algorithm Analysis

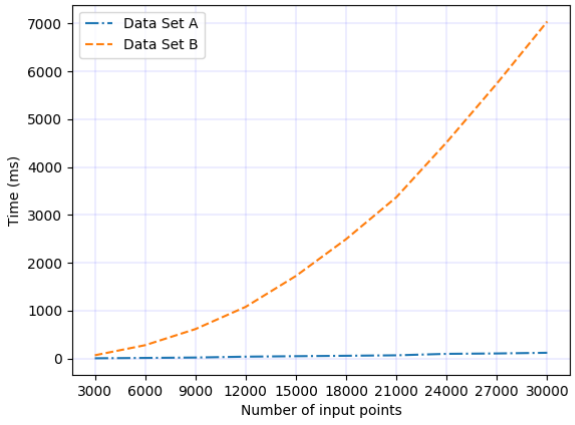


*Figure 1 Experimental analysis of the Gift Wrap algorithm*

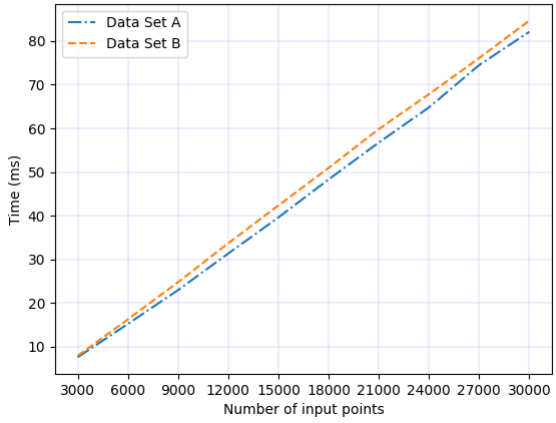


*Figure 2 Experimental analysis of the Graham-Scan algorithm*

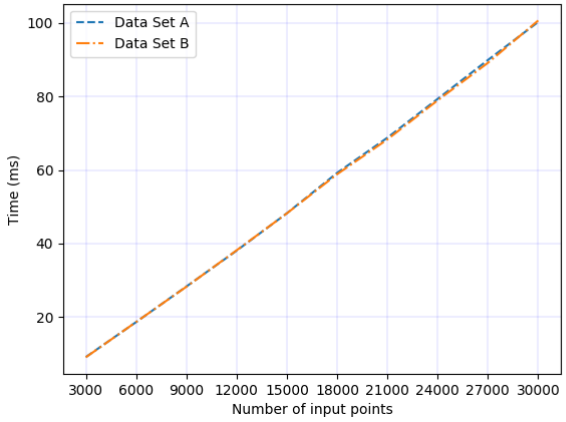


*Figure 3 Experimental analysis of the Monotone chain algorithm*

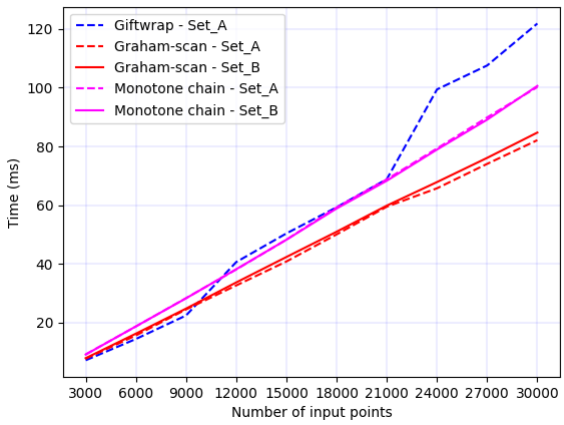


*Figure 4 Experimental analysis of all algorithms with respect to input size*

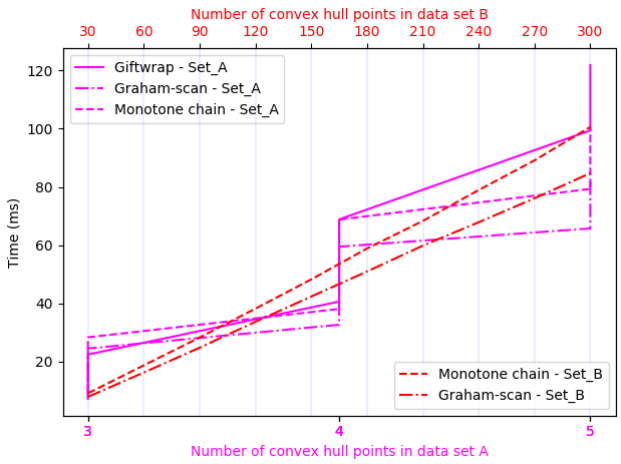


*Figure 5 Experimental analysis of all algorithms with respect to convex hull size*

The above graphs show the results of the experimental analysis of the three algorithms over both data set A and B. The results of the Gift Wrap algorithm over data set B have been omitted in figure 4 and figure 5, as the results make comparison of the other algorithms too hard. For both the Graham-Scan and Monotone chain algorithm as shown in figures 2 and 3 respectively, the times taken over data set A and B vary very slightly. This shows that the size of the convex hull constructed is not a major factor in the time complexity of these two algorithms. The time complexity of these two algorithms comes mainly from the length of the input points. This is because both algorithms require sorting of the input points as a preprocessing step. This then restricts the algorithms to a best case of O(nlogn) as that is the time complexity of sorting the points. After the sorting step both the Graham-Scan and Monotone chain algorithms use very similar stack based methods which take only O(n) time to construct the convex hull. The Graham-Scan algorithm performs at most, 2n push or pop stack operations during this step. ("The Convex Hull of a Planar Point Set", n.d.) As the Monotone chain constructs two separate hulls, it will use up to 4n push or pop operations. The Graham-Scan algorithm requires extra pre-processing steps, the minimum y-coordinate point must be found and the angle of every point, with respect to the point of minimum y-coordinate is calculated. The Monotone chain algorithm has slightly worse performance than the Graham-Scan algorithm. This is due to two main differences in the algorithms. It constructs two convex hulls, an upper and a lower convex hull, therefore it must loop over all the input points twice. The second is that it must concatenate the two convex hulls. The time complexity of concatenating two lists in python is O(n + m), n and m are the sizes of the two lists. ("Runtime of merging two lists in Python", n.d.) Where n and m are both equal to h, h is the size of the convex hull being constructed and this then gives a time complexity of O(h). These differences in the algorithm outweigh the extra pre-processing that Graham-Scan requires, and causes the Monotone chain algorithm to have slightly decreased performance than Graham-Scan.

When the Gift Wrap algorithm is analysed over data set A, it can be seen again that the main factor of the time complexity is the size of the input and not the size of convex hull. However, the size of the convex hull does play a bigger part in the time complexity of the Gift Wrap algorithm than it does for the Graham-Scan and Monotone chain algorithms. This can be seen when data set B is considered. When data set B is run through the Gift Wrap algorithm, a massive increase in time is seen, whereas for file A_30000.dat it takes 0.122 seconds to construct the convex hull, and for file B_30000.dat it takes 7.039 seconds. These files have equal input sizes, but the file from set A has only 5 points in its convex hull and the file from set B has 300 points in its convex hull. This change is due to the time complexity of the algorithm which is O(nh), where n is the size of the input and h is the size of the convex hull, hence the size of the convex hull has a massive impact on the time complexity. Gift Wrap algorithm is therefore output sensitive. It also requires an O(n) pre-processing step in which the point of minimum y-coordinate is found. These differences

along with the less efficient while-for structure for constructing the convex hull is what drives up the time complexity of the algorithm and causes it to be a less efficient algorithm.

The time complexities of both Graham-Scan and Monotone chain come from the size input which is due to their pre-processing steps giving O(nlogn). The Gift Wrap time complexity is derived from both the input size and the amount of points in the convex hull as it is output sensitive which gives O(nh). Therefore, the Graham-Scan and Monotone chain algorithms have a more efficient approach into constructing a convex hull than the Gift Wrap algorithm. Furthermore, the Graham-Scan algorithm takes a slightly more efficient approach in the construction of a convex hull than the Monotone chain algorithm does, and is the most efficient of the three implementations.

# References


The Convex Hull of a Planar Point Set. Retrieved from http://geomalgorithms.com/a10-_hull-1.html

Runtime of merging two lists in Python. Retrieved from https://stackoverflow.com/questions/29197811/runtime-of-merging-two-lists-in-python

Convex Hull | Set 1 (Jarvis's Algorithm or Wrapping) - GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/

Monotone Chain Convex Hull - Algorithmist. Retrieved from http://www.algorithmist.com/index.php/Monotone_Chain_Convex_Hull