Convex Hulls in 2D

Srinivas Suresh Kumar – ssures11@jhu.edu

The task for this assignment was to

- Implement Quick Hull
- Implement Graham's scan
- Capture the output size and running time for various input sizes
- Plot output size as a function of input size
- Plot running time as a function of input size
- Plot running time as a function of output size

Caveats, Claims and Assumptions

- The **most blaring and obvious anomaly** is that Graham's scan and Quick Hull take significantly ***longer*** than Naive Hull construction. Initially I was also puzzled by this. So I went ahead and tried to implement Native Hull Construction myself. Then it became clear
  - Heavy use of dynamic STL data structures and convenience methods and several levels of redirection negatively impact performance
  - Though the Naive implementation you provided outperforms Graham's scan and Quick Hull, both of them outperform my implementation of Naive Hull significantly when I wrote it in a manner that uses a lot of STL and redirection
  - However the version of Naive Hull that was plotted and analyzed is your implementation as the assignment explicitly asks us to analyze it
- I have implemented a script that runs the Convex hull program and captures the results, it is in **Ruby,** a scripting language similar to Python. Therefore if you wished to test that script out I recommend testing it on a Linux box or Mac as Ruby is notoriously difficult to get to work on Windows
- Once the Ruby script provided me with table of input vs output vs running time I plotted the graphs using This Link . That url plots points as a connected line

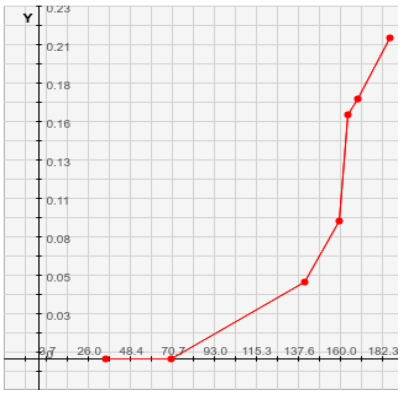
Discerning Running Time and Output Dependence

In the same folder as this document there are 9 graphs each plotting input size vs output size, input size vs running time and output size vs running time for all 3 algorithms

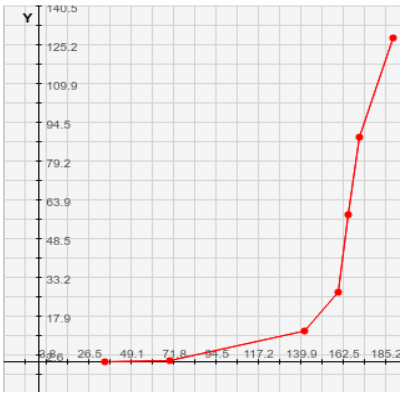
Before we proceed let us establish 3 things

- input size and output size are obviously positively correlated, as the number of points in space increase so will the points on the hull, in an almost proportional manner
- the same holds for input size and running time. More points to process means more time for execution
- Using the above 2 claims we can see that output size and running time have at least a causal positive correlation for all 3 algorithms in addition to any other latent relation they might have

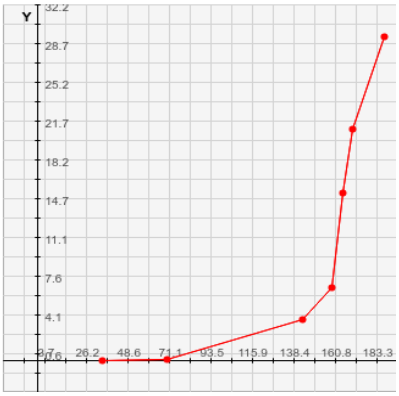
However to discern output dependence we only need the output size vs running time for all 3 algorithms. Below are the 3 graphs for Naive Hull, Quick Hull and Graham's scan respectively



Naive



Quick hull



Graham's Scan

Look closely at the y-axis for Quick Hull, it grows rather rapidly compared to those of Naive hull and Graham's Scan. If we were to plot the curves of Graham's scan and Naive Hull with the y-axis scaling of Quick Hull their respective curves would be a lot less sloping and a lot more flat.

They would not be horizontal definitely but would not be increasing as fast as they seem to be without the y-axis scaling. This indicates that the running time of quick hull is dependent on a scalar multiple of the size of the hull.

This is in line with out algorithmic knowledge that quick hull has a running time of $\Theta(n*h)$ where n is the size of input and h is the size of the hull