Name: Thomas Kearsley ID:98112573

COSC 262 Report

Algorithm Implementation

Gift-Wrap Algorithm

For this algorithm, I initially decided to use the math module to compute angles using . However, I realized it was easier to simply take the absolute values of *dx* and *dy* and use a reference circle. This seemed like a better method as it made it easier to deal with negative angles. I also decided to divide the algorithm into two functions *gift wrap* and *theta*, this improved the overall readability of the algorithm.

Graham Scan Algorithm

This algorithm took me the longest as I had to deal with issues that I had no considered with the previous algorithms such as collinearity of points. I also had to implement a specific theta function, *theta\_graham* as if *t = 0* like in the gift wrap this would append vertices that

were not in the convex hull. I used a stack for the convex hull, this allowed me to pop the most recent element when a turn was made that was not counter clockwise. I also used a lambda function in order to initially sort the *listPts* by lowest y value and another to sort the angle, point tuple by the angle. This was an easy way to sort tuple values which would have been more tedious without the lambda.

Monotone Chaining Algorithm

I decided to use a monotone chaining algorithm for my *amethod* algorithm, this algorithm constructs the convex hull by forming a lower and upper hull and merging them. First, the points are sorted lexicographically by their x co-ordinate. The upper hull runs from the right most point to the left most point in a counter clockwise order. These two hulls are then merged omitting the last point of the list as it is appended to both lists, this algorithm runs in . I chose this algorithm because it was a different approach from the graham scan and the gift wrap algorithm as it forms two convex hulls and then merges them.

Algorithm Analysis

Data Analysis for Gift Wrap Algorithm

As shown in the graph below, we see the gift wrap algorithm’s performance appears linear as the number of points increase. This is due the calculation of the *p0* point taking longer as it must search for more points.

Fig 1. Gift Wrap Performance Set A Data

Set B is a better representation of the Gift-wrap algorithm, as the time increases proportional to the number of points. This is because in Set B, as the number of points increase, the number of hull points increase unlike Set A where the number of hull points varies significantly. From the graph, we see the nature where h is the number of hull points and how it tends to towards a quadratic with a time complexity.

Fig 2. Gift Wrap Performance Set B Data

A possible improvement for Gift Wrap could be to use a lambda function to sort the *listPts* this would mean the first element would always be *p0* and it would not be necessary to iterate through the entire list of points to find the lowest y-value point.

Data Analysis for Graham Scan Algorithm

From this Fig 3, the difference between the Graham Scan and Gift Wrap performance appears to be minimal. In fact, Graham Scan takes longer in the case of 30,000 test points averaging 0.14 seconds while Gift Wrap averages 0.12 seconds.

Fig 3. Graham Scan Performance Set A

In Fig 4, we see the real difference between Gift wrap and Graham Scan and the vs time complexity becomes far more prominent. We see Graham Scan form the convex hull for 30,000 points in, on average, less than 0.15 seconds while the Gift wrap algorithm takes nearly 4.5 seconds.

Fig 4. Graham Scan Performance Set B

Data Analysis for Monotone Chaining Algorithm

As we can see from Fig 5 below, the monotone chaining algorithm has very similar performance to the Graham Scan. This is expected as both algorithms have the same time complexity of .

Fig 5. Monotone Chaining Performance Set A

Like the Graham scan we see the average time performance in the Set B data shown in Fig 6 below. However, we see the slight advantage of Monotone Chaining over the Graham Scan especially around on Fig 6 where the data point lies under the 0.08 second line while on Fig 4 it lies on the line.

Fig 6. Monotone Chaining Performance Set B

This result is expected because the way the monotone chain performs the operation from the rightmost position to leftmost in a counter clockwise and subsequently down for the lower hull gives the algorithm a slight advantage over graham scan.

References:

• All data & graphs were created by me. Raw data can be viewed in ‘Algorithm Performance.’

• Algorithmist (2011) Retrieved May 25th, 2017

<http://www.algorithmist.com/index.php/Monotone_Chain_Convex_Hull>

• Pseudo Code from 1.1 Convex Hulls by R. Mukundan