William Scott

SN: 11876177

COSC 262 Algorithms

Assignment: Convex Hulls

Algorithm Implementation

The goal of this assignment was implementing two methods for computing the convex hulls of two-dimensional points and to evaluate their performance. The two methods which were implemented were: the simplified gift-wrap algorithm (also known as Package wrap, or Jarvis March) and Graham-scan.

Beginning with what was in common; one requirement for both algorithms was to return the original index locations of the points along the convex hull. This meant I had to keep track of, or at least be able to lookup where the points were originally. For this, I found the use of dictionaries incredibly useful. Dictionaries were created using the point as a key, and the original index location as the value. At the end of these switch heavy algorithms, dictionaries allowed for easy look-up of the original index of the point in the data set. As space efficiency was not a concern, this made the algorithm significantly easier to both write and follow.

Another problem I ran into with both algorithms was properly finding and removing collinear points from the convex hull. In gift-wrap I solved this problem by adding a comparison of the distance from the current point, to the currently being checked point, and the previous point to the current point. If the distance from the current point, to the currently being checked point was less than the distance from the previous point to the current point, then you can determine that the currently checked point was both between the other two points, as well as collinear. Within Graham-scan I had to add a check for an edge-case where the last two points in the convex hull were collinear, this was easy to add when finally discovered.

Python's functions made it easy to read the data in from the file and convert the points to tuples. 'Try', 'except', and 'del' all proved to be useful within my graham-scan ensuring the first three points were not collinear unless absolutely necessary.

Algorithm Analysis

Convex Hull - Performance Data (Average of 5 trials, in seconds)				
Number of Points	Graham-scan Square	Gift-wrap Square	Graham-scan Circle	Gift-wrap Circle
1000	0.006	0.023401403	0.0044	0.038202
2000	0.008601	0.046002293	0.008201	0.101806
3000	0.013001	0.0648036	0.013401	0.17621
4000	0.017001	0.089204979	0.017601	0.245014
5000	0.022401	0.111406469	0.022001	0.365421
6000	0.026202	0.132207727	0.026601	0.462026
7000	0.030001	0.157208824	0.030401	0.569633
8000	0.036802	0.162209463	0.035002	0.649037
9000	0.040202	0.203811789	0.038602	0.793445
10000	0.044603	0.201411438	0.043003	0.886051
11000	0.048403	0.209011793	0.048203	1.004658
12000	0.055003	0.259414673	0.053403	1.084262
13000	0.058603	0.283415794	0.060404	1.22027
14000	0.062603	0.253014421	0.063604	1.344477
15000	0.068804	0.302617216	0.068404	1.466684
16000	0.073004	0.359020472	0.073604	1.639694
17000	0.077204	0.322618866	0.074805	1.757501
18000	0.084605	0.343419409	0.081205	1.866107
19000	0.090005	0.390222025	0.085605	2.081119
20000	0.094205	0.375421667	0.090405	2.161724

Fig 1.Data of Algorithm Performance – Average of 5 Trials

Algorithm Analysis (Continued)

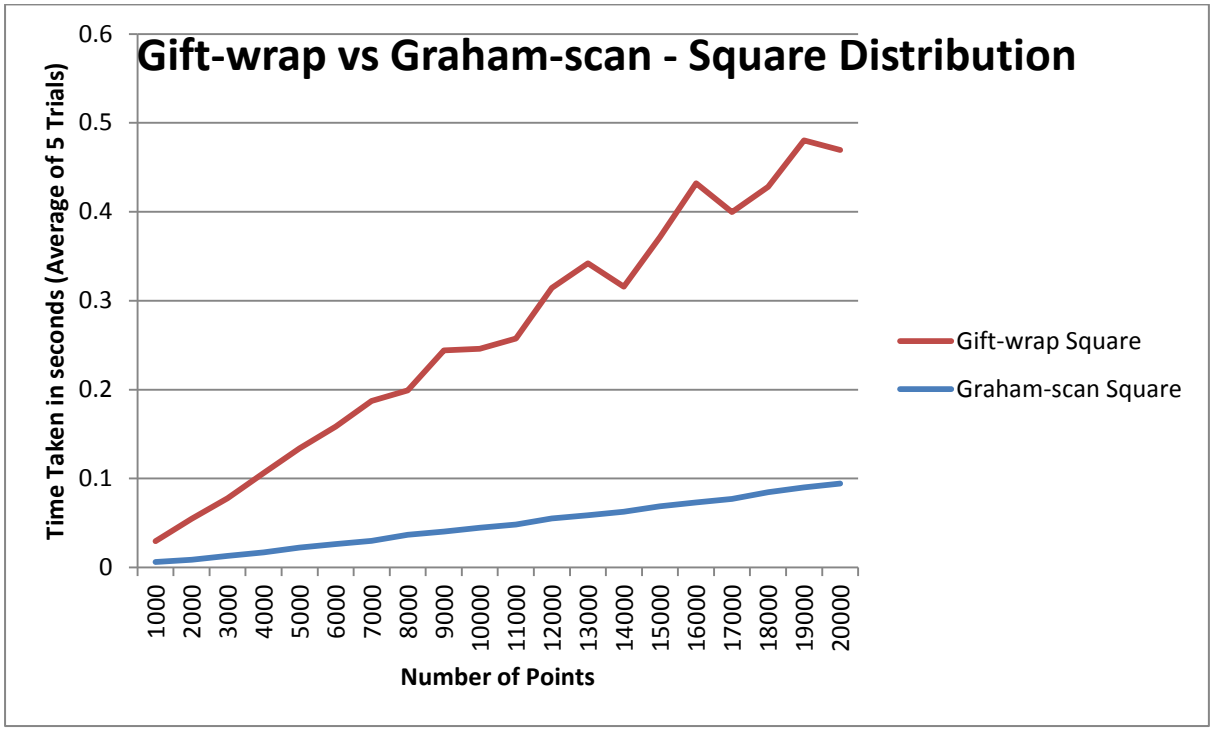

Fig 2. 'Gift-wrap vs. Graham-scan - Square Distribution' Graph

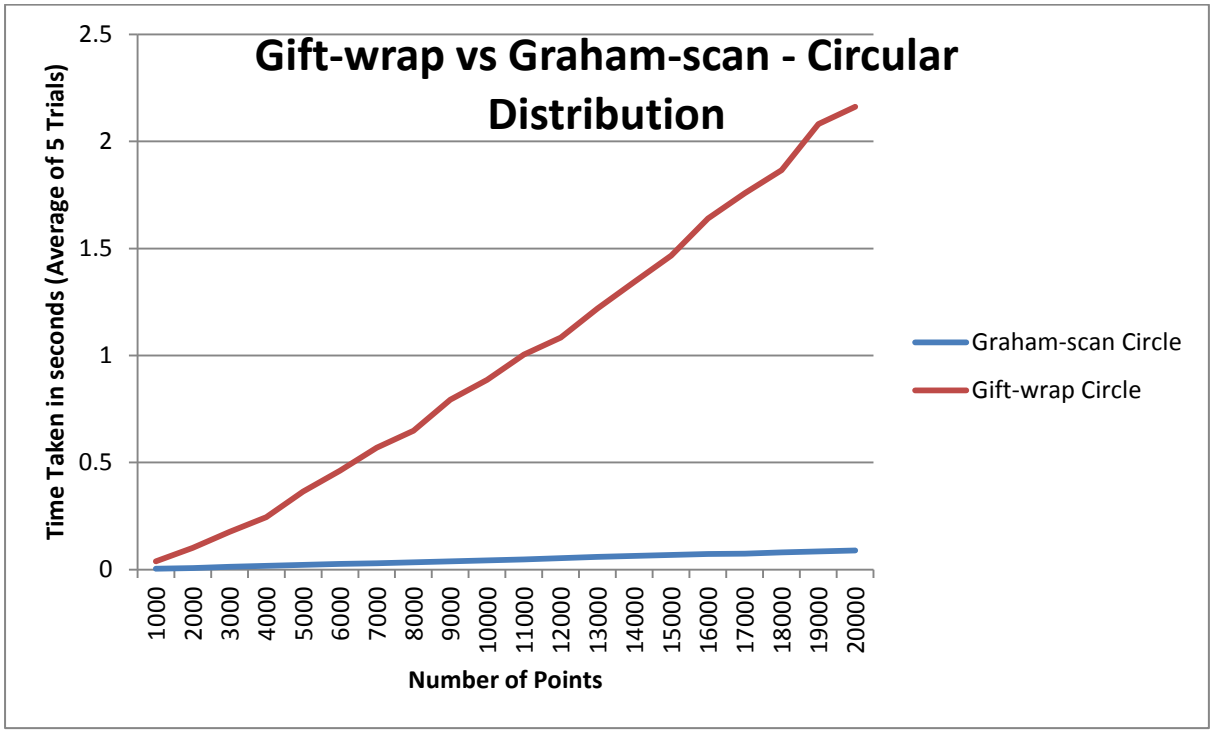

Fig 3. 'Gift-wrap vs. Graham-scan - Circular Distribution' Graph

Algorithm Analysis (Continued)

Looking at the performance graphs it is immediately apparent that graham-scan significantly outperforms gift-wrap in both circular and rectangular distributions. The jaggedness of gift-wrap's square performance can be explained by its O(n*m) time complexity; where n is the number of points, and m is the number of points that lay on the convex hull. Since only 5 trials were averaged getting a slightly more circular square in any trial meant a much longer overall runtime, if more trials were run the slope would likely be much smoother. Graham-scan had a consistently fast runtime as expected, and my results match the predicted O(n log n) time complexity – this is seen as the shape of the distribution had little to no effect on the speed it took to complete the convex hull.

Further Improvement

Since both methods generate convex hulls, a pre-processing of the data to efficiently eliminate unnecessary points would be immensely useful. I believe you could use four points: the lowest-leftmost, lowest-rightmost, highest-leftmost, and highest-rightmost; to find an area containing a significant number of points which would not be on the convex hull.

Another improvement I could make to both my gift-wrap and graham-scan algorithms is use of memory/space. I use a dictionary to keep track of all the points' original index location from the data to return at the end. I could improve both algorithms' use of space by keeping track of the original index another way.

I would be interested in adding support for creating 3D convex hulls. I think the challenge of trying to adapt either of these algorithms to 3D space would be both challenging and interesting with many applications.