**COSC262 Assignment: Convex Hulls**

**Algorithm Implementation**

Gift-wrap algorithm – Problem encountered

When there is more than one point with minimum y value, the error ‘index out of range’ occurs. To resolve this problem, an extra ‘if’ condition is added to theta() function. This ‘if’ is used to check for the case when t = 0 and return 360 when it does.

When the points are collinear, the function either gives error 'index out of range' or returns the wrong points for convex hull. To avoid this problem, the ‘elif’ condition is added to the code to check whether the calculated angle is the same with the previous angle. If they are the same, the point with the maximum distance from the current point of the convex hull is chosen.

Graham-scan algorithm - Useful algorithm and problem encountered

The sorting step of the Graham-scan algorithm takes O(nlogn) time. Merge sort has the worst case performance of O(nlogn). It would not get any worse and therefore it is useful structure to use for the sorting step of the code which requires O(nlogn) time.

When the first three points are collinear, the function gives error 'list index out of range'. To prevent this error, the sorted list is used to construct a new list which has only unique angle. When two points with the same angle are found, they are compared to each other. The point with more distance between the point and the current point of the convex hull is chosen. The point with maximum distance is appended to the new list.

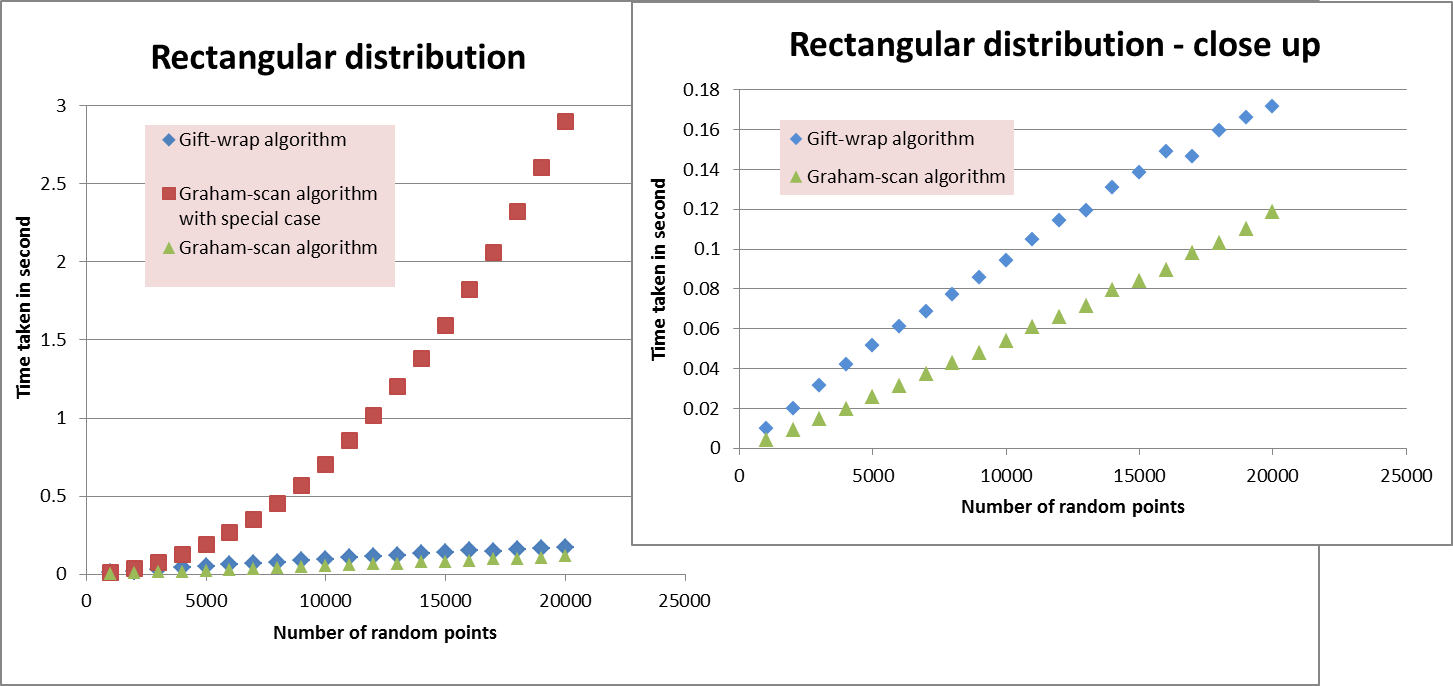
In the sorting step of the Graham-scan algorithm, the points are sorted according to their angle. The angle can be sorted but the points are not sorted according to their angle. The resolution is to store the angle to the list of that point as the third argument.

Both algorithms – Problem encountered

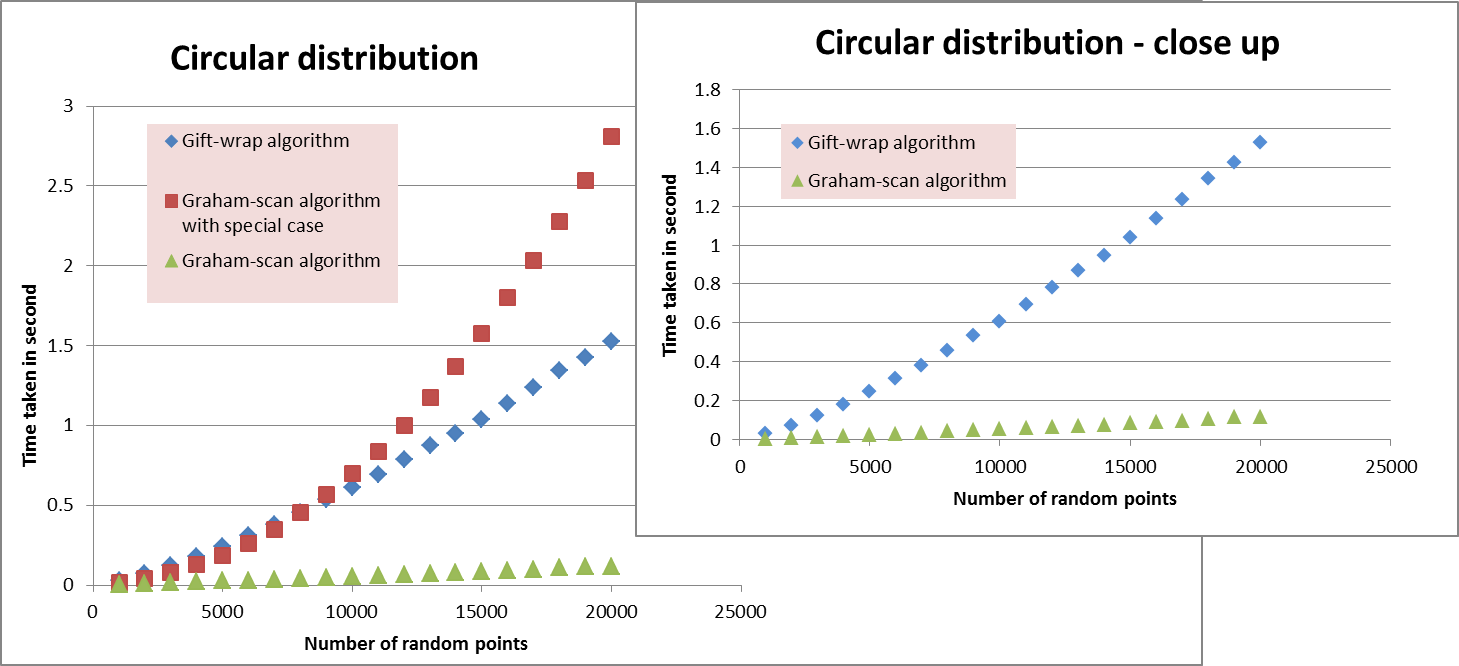
In theta() function, division (/) gives integer value. This leads to inaccurate t value and potentially leads to ‘list index out of range’ error. To prevent this from happening, the numerator is multiplied by 1.0 to change its type to float instead of int.

Randrange() function from random module is used to generate random values for points within the specific range. However sometimes, the function generates duplicated points. This happens frequently especially when number of points is large. To avoid this, the while loop is used to ensure the list has a unique set of values before appending it to the tested list, and regenerate the point if it is not.

**Algorithm analysis**

****

*Graph1 : Time taken to produce the convex hull with rectangular distribution condition for given size of points*

**

*Graph2 : Time taken to produce the convex hull with circular distribution condition for given size of points*

In theory, the time complexity of Graham-scan algorithm is O(nlogn) where m is number of vertices on convex hull and n is number of set of points in the list. In Graph1 and Graph2, the graphs from using Graham-scan algorithm(red) rise like graph for y = x2 as the number of random points increases. This is likely due to the extra code added when resolving the collinear point problem. As when that part of code is removed, the graphs(green) becomes shallow and look like the graph for y = xlogx as predicted.

In theory, the time complexity of Gift-wrap algorithm is O(mn) where m is number of vertices on convex hull and n is number of set of points in the list. In rectangular distribution, most points are not on the convex hull. This means the time complexity is expected to be above or equal to O(n) and less than O(mn). Example, O(n) or O(nlogn). In Graph1, the graph using Gift-wrap algorithm(blue) does not look like a graph for linear function. It looks steeper. It also does not look like graph for y = xlogx, it is shallower. This is consistent with the prediction. In circular distribution, most points are on convex hull. This means the time complexity is expected to be O(nlogn) or O(n2). This is consistent with the graph using Gift-wrap algorithm in Graph2 as the graph(blue) looks like the graph for y = xlogx.

**Further Improvement**

Graham-scan algorithm – using Python sort function and preprocessing data points, and use different approach for special case

The sorting step in Graham-scan algorithm can improve if it uses python sort function on preprocessing data points. The best case performance of Python sort function is O(n). This is better than merge sort best case performance which is O(nlogn). So if the input data points are already sorted according to their angle, the Python sort function can perform its best case performance of O(n) instead of O(nlogn). This affects the whole algorithm complexity to reduce to O(n).

The graphs of Graham-scan algorithm code for special case(collinear)and normal Graham-scan algorithm in Graph1 and Graph2 show dramatic change in the performance. If the program only throws the error ‘list index out of range’ when the first three points are collinear, the program can be improved by checking only the first three element in the stack instead of the whole list.

**References**

- <http://en.wikipedia.org/wiki/Timsort>

- http://interactivepython.org/courselib/static/pythonds/SortSearch/TheMergeSort.html