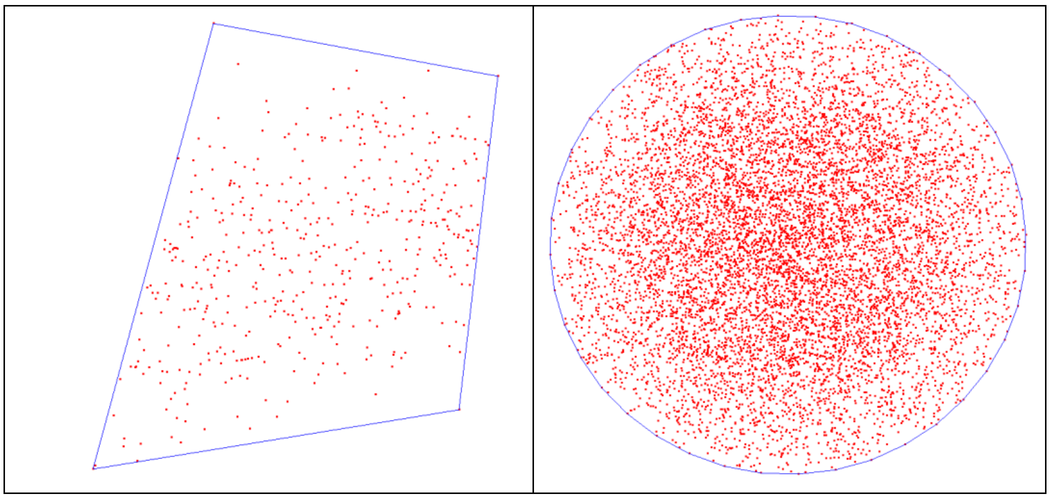
COSC262 Algorithms
Assignment: Convex Hulls

Max. Marks 20

Due: 5pm, 1 June 2018



1. Outline

In this assignment, you will implement algorithms for computing convex hulls of two-dimensional points, and validate your implementations using the datasets provided. You will then perform an experimental analysis of time complexities of the algorithms, and discuss your findings in a report.

2. Program for computing convex hulls

You are required develop a Python program containing the implementations of the following convex hull algorithms:

i. The **Gift Wrap** algorithm (ref: lecture slides [1.1]: 38-43)

ii. The **Graham-Scan** algorithm (ref: lecture slides [1.1]: 44-49)

iii. A third algorithm for computing convex hulls: This could be any other non-naïve method for computing the convex hull of a set of points (eg., insertion hull, quick hull, monotone chain etc.), or a suggested improvement of the method in (i) or (ii).

A template for your Python program is provided in the file `convexhull.py`. Please do not change the file name or the names of functions included in the file. You may import modules and define additional functions, variables as necessary. Please also use short comment lines to annotate important steps in the program. Please do not use Python 2.x.

3. Datasets

You are given two data sets, "`Set_A.zip`" and "`Set_B.zip`". Each dataset contains 10 files. All files are editable text files containing several lines of data, where each line contains two values representing the x and y coordinates of a two-dimensional point. The data in the files satisfy the following properties

- The x and y coordinates are stored as floating point values with at most two decimal places, and satisfy the conditions $0 < x < 1000$, $0 < y < 1000$.
- There are no duplicate (coincident) points
- All convex hull edges will contain exactly two points, i.e., none of the edges will contain three or more collinear points.

The data size N is also included as the numeric value in the file name. The files in "`Set_A.zip`" contain points that generate convex hulls with only a few vertices, whereas the files in "`Set_B.zip`" contain points that generate convex hulls with a much larger number of vertices, as given in the table below.

Set A			**Set B**		
File Name	Number of points	Number of hull vertices	File Name	Number of points	Number of hull vertices
A_3000.dat	3000	3	B_3000.dat	3000	30
A_6000.dat	6000	3	B_6000.dat	6000	60
A_9000.dat	9000	3	B_9000.dat	9000	90
A_12000.dat	12000	4	B_12000.dat	12000	120
A_15000.dat	15000	4	B_15000.dat	15000	150
A_18000.dat	18000	4	B_18000.dat	18000	180
A_21000.dat	21000	4	B_21000.dat	21000	210
A_24000.dat	24000	5	B_24000.dat	24000	240
A_27000.dat	27000	5	B_27000.dat	27000	270
A_30000.dat	30000	5	B_30000.dat	30000	300

4. Algorithm Validation

For validating the algorithms, you will read a set of points from the given data files, and compare the outputs generated by the giftwrap and Graham-scan methods with the expected outputs provided in the file "`Outputs.pdf`". Both the giftwrap and Graham-scan algorithms should produce the same list of points in exactly the same order. The third algorithm should also generate a list containing all the required hull vertices, but the order of vertices need not be the same as that produced by the other two methods.

To help you visualize the distribution of points in each data file and the shapes of the corresponding convex hulls, the plots of the points and the convex hulls are provided in the file `Plots.pdf`.

5. Algorithm Analysis

After validating the algorithms for generating convex hulls, you should perform an experimental analysis of their time complexity with respect to input size *N,* using datasets Set_A and Set_B.

For each of the three algorithms, you should generate a graph (line chart), comparing the time measurements for Set_A and Set_B, with input size *N* varying from 3000 to 30,000 in steps of 3000 along the *x*-axis. Use the graphs to compare the performance of each algorithm for the two types of data, and also to compare the performance between the three methods. Provide the graphs and a brief outline of your analysis in your report (see next section). Please also try to answer the following questions in your report.

a). How did the results vary for each algorithm? Please give a brief explanation of the trends seen in each graph, unexpected variations if any, and similarities to (or deviations from) the theoretical measures of complexity.

b). Which one of the three algorithms gave the best result in performance analysis? Why did that algorithm perform better than the others?

You may create a modified version of the file `convexhull.py` for performing the analysis of time complexity. Please name this file `convexhull_time.py`

6. Report (3-5 pages)

Prepare a report detailing your work. The report should include the following sections:

1. Algorithm implementation: In this section, you may discuss any specific data structures or Python functions that you found useful for your implementation. Please also give a description of the third algorithm ("`amethod`").
2. Algorithm analysis: Provide the results of the comparative analysis in the form of graphs as described in the previous section, and give your interpretation of the results. If you have proposed an improvement of either the giftwrap algorithm or the Graham-scan algorithm in "`amethod`", describe how the suggested improvement could be seen in experimental results.
3. References Required only if you found any source of information or resource particularly useful for your understanding of the algorithms, program development, or analysis.

7. Marking Scheme

The submitted programs will be tested with our input files which will be different from the ones provided. The points in the data files used for testing will also satisfy all the conditions outlined in Section 3. The distribution of marks among different components of the assignment will be as follows:

Report : 10 marks.
Code - design and correctness of algorithms: 10 marks.

8. Assignment Submission

Assignment due date and time: **1 June 2018, 5pm.**

Drop-dead date with 15% penalty: 8 June 2018, 5pm.

Please submit the source files (`convexhull.py, convexhull_time.py` and any other supplementary files) and the report (in WORD or PDF format), using the assignment submission link on Learn (`learn.canterbury.ac.nz`).

9. Miscellaneous

1. This is not a group project. Your report must represent your own individual work. In particular, students are not permitted to share program source code in any way.

2. Please check regularly on Learn system forums for spec updates and clarifications.

3. Standard departmental regulations regarding dishonest practices and late submissions (1 week "drop dead" with 15% penalty) apply.