COSC262 Algorithms
Assignment: Convex Hulls

Max. Marks 20

Due: 5pm, 2 June 2017

1. Outline

In this assignment, you will implement methods for computing the convex hulls of two-dimensional points, and validate the algorithms using the dataset provided. You will then perform an experimental analysis of their time complexity, 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.

3. Datasets

You are given two data files, "Set_A.dat" and "Set_B.dat". Each file contains the x and y coordinates of 30,000 points, one point per line.

- The x and y coordinates are stored as floating point values, and satisfy the conditions $0 \le x \le 1000$, $0 \le y \le 1000$.
- There are no duplicate (coincident) points
- Both files are plain text files.

Your program should be able to read a specified number N of points from the data files and generate a list of points belonging to the convex hull of the N points. The output list generated by the giftwrap and Graham-scan algorithms should begin with the lowermost point of the convex hull, with subsequent points ordered in the anticlockwise direction along the convex hull. If there are two points with the same minimum y-value, select the rightmost point as the starting point.

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. You are given a file "Validation.pdf" containing the recommended values *N* (the number of points you should read from the input files), and the expected outputs. 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 giftwrap method.

Please note that the points in the data set are generated using random numbers, and therefore for certain values of *N*, the convex hull may have edges with three or more collinear points. Only the end points of such edges must be included in the hull.

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,* for both datasets Set_A and Set_B. You should record the time taken by each of the three methods for 15 values of *N*: 2000, 4000, 6000,…30,000 (in steps of 2000). The supplied data sets are designed in such a way that for the above values of *N*,

- the convex hulls generated using points in Set_A contain only 4 vertices.
- the number of vertices of convex hulls generated using points in Set_B, steadily increase with *N*.

It is recommended to take the average of three time measurements for each of the above values of *N*. 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`

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 2000 to 30,000 in steps of 2000 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?

6. Report (2-4 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 2. The distribution of marks among different components of the assignment will be as follows:

Report : 6 marks.
Code - design and correctness of algorithms: 14 marks.

8. Assignment Submission

Assignment due date: **2 June 2017**; Drop-dead date with 15% penalty: 9th June 2017.

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.