COSC 262 Algorithms

Assignment Part 2

Max. Marks 25

Due: 5pm, 31 May 2013

1. Problem Description

In this assignment, you will implement methods for computing the convex hulls of two-dimensional points, and evaluate their performance using point data sets. For comparative analysis and testing of the methods, three programs are required to be developed as outlined below:

i. `giftwrap.py`: This program should contain the implementation of the "Simplified Gift-Wrap" algorithm based on the method given on Slide [1.1]-23

ii. `grahamscan.py`: This program should contain a stack-based implementation of the Graham-Scan algorithm as outlined on Slide [1.1]-28.

iii. `amethod.py`: This could be any other non-naïve method for computing the convex hull of a set of points (eg., insertion hull, quick hull), or a suggested improvement of the methods in (i) or (ii).

2. Data Sets

Each of the three programs must be able to read up to 12000 points from an input file, and output a single line containing the indices of the convex hull vertices. The programs should be able to read the input file name specified as the command line argument, eg.,

```
python giftwrap.py  A_500.dat
```

For this, you will need to use the `sys` module (`import sys`) in the program as shown below. The file name will then be available in `sys.argv[1]`:

```
import sys

f = open(sys.argv[1])
n = int(f.readline())
```

The following input data files are provided to help you with the testing and evaluation of your programs. You may also generate your own data sets.

```
A_10.dat,   A_50.dat,   A_500.dat, A_1000.dat, A_3000.dat,
A_6000.dat,  A_9000.dat,  A_12000.dat.

B_10.dat,   B_50.dat,   B_500.dat, B_1000.dat, B_3000.dat,
B_6000.dat,  B_9000.dat,  B_12000.dat.
```

The first line of a data file will contain an integer number n ($n \leq 12000$), specifying the number of points. This will be followed by n lines, each line containing the x and y coordinates of a point. The coordinates will have integer values in the range [1, 800]. The data sets are designed in such a way that there are no coincident points.

Files with names beginning with '`A_`' contain very few points on the convex hull - these point sets typically have triangles or quadrilaterals as their hull. Files with names beginning with '`B_`' will have relatively higher number of points on the convex hull, with the shape of the hull tending to a circle as the number of the points increases. For all data sets, the number in the file name indicates the total number of points contained in the file. All input files are editable text files.

The first point in the input file must be assigned an index 0, the second point index 1 and so on. All programs should produce the same output for the same input data file. The output should be a single line containing the indices of the convex hull vertices, starting from the lowermost point and ordered in the anticlockwise direction. If there are two points with minimum y-value, select the rightmost point as the starting point. The expected output for a data file can be found in a file with the same name and extension "`.out`". These are also plain text files, and you could use them to test the correctness of your programs.

If an edge of a convex hull contains a number of extra collinear points, it is not necessary to consider all those points as vertices of the convex hull. None of the supplied data contains extra points along hull edges.

3. Algorithm Analysis

After validating all three programs, you should perform an experimental analysis by estimating the variation of their complexity with respect to input size n. When estimating the time taken by a method, please run the program 3 or 4 times and take the average time. For each program, you should generate a graph containing two plots: one for the "A_" data set, and the other for the "B_" set. Use the graphs to compare the performance of each method 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? Did the growth rates seen in the graphs match with the theoretical measures of complexity?

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

4. 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 problems encountered while implementing and testing the algorithms, and how you could solve them. Give a brief outline of the algorithm used in the program "`amethod.py`".

2. Algorithm analysis: Provide the results of the comparative analysis in the form of graphs, 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.py`", describe how the suggested improvement could be seen in experimental results.
3. Acknowledgment and references: If you have used a code segment found in a book or on a website, please provide the details in "Acknowledgements" section. You may also include a list of references for the method implemented in "`amethod.py`".

5. Marking Scheme

The submitted programs will be tested with our input files which will be different from the ones provided. Even though marks are not assigned for design, style and readability of code, you are encouraged to give sufficient importance to these aspects while developing programs and to include comment lines where appropriate. The distribution of marks among different components of the assignment will be as follows:

Code `giftwrap.py` : 7 marks.

Code `grahamscan.py` : 7 marks.

Code `amethod.py` : 6 marks

Report: 5 marks.

6. Assignment Submission (Assignment due date: 31 May 2013; drop-dead date: 7th June 2013)

Your submission should include:

1. The source files `giftwrap.py`, `grahamscan.py` and `amethod.py`. Input data files should not be included.

2. A copy of the report in either WORD or PDF format.

Submit the source files and the report as a single .zip file, using the assignment submission link on the course page on Learn (`learn.canterbury.ac.nz`)

7. Miscellaneous

1. This is not a group project. Your report must represent your own individual work.

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