COSC 262 Algorithms

Assignment: Convex Hulls

Max. Marks 20

Due: 5pm, 5 June 2015

1. Problem Description

In this assignment, you will implement two methods for computing the convex hulls of two-dimensional points, and evaluate their performance using point data sets. The methods to be implemented are

- the simplified gift-wrap algorithm (Slides [1.1]: 35-38), and,
- the Graham-scan algorithm (Slides [1.1]: 44)

The assignment requires 4 programs to be developed and submitted:

`giftwrap_t.py` , `grahamscan_t.py` (Test versions)

`giftwrap_e.py` , `grahamscan_e.py` (Evaluation versions),

Also required is a report giving experimental analysis performed using the evaluation versions.

2. Algorithm Validation

The test versions `giftwrap_t.py`, `grahamscan_t.py` will be used to test the correctness of your implementations using small data sets consisting of up to 100 data points. Six files, (`T1.dat`,...,`T6.dat`) containing *x*, *y* coordinates of points are provided. The required outputs (indices of convex hull vertices) for each of these sets are given in separate files `T1.out`,...,`T6.out`. All files are editable text files. The file `pointPlots.pdf` shows a graphical representation of the distribution of points stored in the input files, and the corresponding convex hulls.

Your test programs must be able to read up to 100 points from an input file (`T*.dat`), compute the convex hull of the points and print out a single line exactly as given in the output file (`T*.out`). The programs should be able to read the input file name specified as the command line argument, eg.,

```
python giftwrap_t.py  T3.dat
```

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

```
import sys

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

The first line of a data file will contain an integer number n ($3 \leq n \leq 100$), specifying the number of points stored in that file. This will be followed by n lines, each containing the *x* and *y* coordinates of a point. You may make the following assumptions regarding the input points:

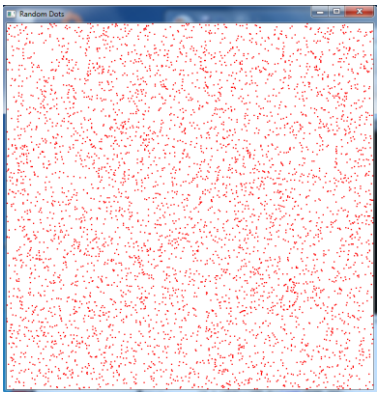
- Coordinates are always integers in the range (0, 1000).
- There are no coincident points.
- None of the edges of a convex hull has extra points that are collinear with the end-points of the edge.

The first point in the input file must be assigned an index 0, the second point index 1 and so on. 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 vertex of the convex hull. Both test programs should produce the same output for the same input data file.

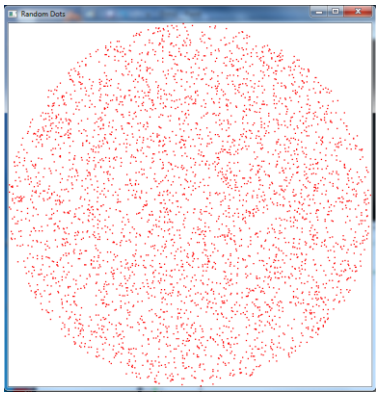
3. Algorithm Analysis

After validating your implementations using test data, you will create slightly modified versions of the programs suitable for performance evaluation of the algorithms. These evaluation versions `giftwrap_e.py`, `grahamscan_e.py` should generate the input data within the program using random numbers, instead of reading the values from an external file. The number of points must be varied from 1000 through to a large value such as 20,000 in steps of 1000 points, and the programs must output the time taken to compute the convex hull for each case. The time taken for generating input data must not be counted in the performance analysis. You will use the time measurements to discuss the complexity of the algorithms in your report. The evaluation versions need not output vertices of the convex hull.

Input points must have integer coordinates in the range (0, 1000). You must generate two types of point distributions as shown below, and obtain performance graphs for each type.



Rectangular



Circular

1. In the rectangular distribution, points are generated within the square ($0 \le x < 1000$, $0 \le y < 1000$). With such a distribution of points, the convex hull tends to have the shape of a square with only a few points as its vertices when the number of input points is large.

2. For a circular distribution, we impose an additional condition $(x-500)^2 + (y-500)^2 < 250000$. A circular region will tend to have many points on the convex hull.

Please note that when the number of points becomes large, more points will start to fall on the edges of the convex hull. You should include necessary conditions to find such collinear points on an edge and to select the farthest point as the next vertex of the convex hull. Some of the points may also coincide.

4. Report (2-4 pages)

Please 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 in your implementation. You may also discuss problems encountered while implementing or testing the algorithms, and how you could solve them.

2. Algorithm analysis: Here you will include two graphs: one corresponding to the rectangular distribution and the other circular distribution of input points. Each graph will contain the plots of the time taken by the two convex hull algorithms with respect to variations in the input size. Please provide your interpretation of the results, including 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.

3. Further improvement: Suggest a method to further improve the implementation or performance of either the gift-wrap or the Graham-scan algorithm. This could be a pre-processing step to reduce data, special data structures or Python functions, or data specific improvements (eg. methods suitable only for circular distributions of points). You are not required to implement the method or show the performance improvement through experiments.

4. References Required only if you found any source of information or resource particularly useful for your understanding of the algorithms, program development, or analysis.

5. Marking Scheme

The test versions of the programs (`giftwrap_t.py, grahamscan_t.py`) will be tested with a set of input files that 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_t.py, giftwrap_e.py`:	8 marks.
Code `grahamscan_t.py, grahamscan_e.py`:	8 marks.
Report:	4 marks.

6. Assignment Submission (Assignment due date: 5 June 2015; drop-dead date: 12 June 2015)

Your submission should include:

1. The source files `giftwrap_t.py`, `grahamscan_t.py`, `giftwrap_e.py`, `grahamscan_e.py`. Please do not include input data files.
2. A copy of your 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. 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.