

Assignment A3

Due Date: November 10

Purpose

There are several objectives for this project. One is to continue practicing your programming skills in C. Another is to write a program using an algorithm covered in class; that is, figuring out the details of an algorithm described in more general terms. Finally, you will do some (minimal) analysis of your finished project.

Problem

Many applications require finding the convex hull of a set of points. These include applications from basic paint programs to pattern recognition, image processing, and statistics problems. We have looked at the problem and have found two solutions, one using a brute-force algorithm that results in an efficiency of only $O(N^3)$, and a second that results in an efficiency of at least $O(N^2)$.¹ You will implement the better algorithm, Quickhull, for an arbitrary set of planar points.

Input

The program should first prompt for a file name (up to 20 characters) and then read that file. The first item in the file is an integer n that indicates the number of points in the file. This is then followed by n pairs of x and y integer coordinates.

For example, a valid file would look like this:

```
5
3 7
4 4
2 2
8 6
3 9
```

Output

The program should display the set of points that make up the convex hull of the initial given points. The convex hull should be printed in **clock-wise** order starting from the **least** x value. For the above, the output might look like:

The convex hull is made up of the following points:

```
2, 2
3, 9
8, 6
```

The program should also display the time required to complete the sorting step, the time required to determine the convex hull independent of the sorting, and the total time, all in seconds.

Using \LaTeX or a word processor, type up a short (one page) report describing the time efficiency of the algorithm relative to the size of the input. Most importantly, write an analysis comparing the efficiency of your implementation of Quickhull to the stated efficiency in the text. Are they the same? Why or why not?

¹Why did I write “at least”? Could it be better?

Specifics

- Implement the Quickhull algorithm as described in class and in the text. Other implementations exist (such as Graham's scan), but part of this exercise is for you to code the given algorithm.
- Use a sorting routine appropriate for the data. Copy sort code from the text or any reputable source. Whichever sort you use, however, it should be the fastest available for this application, and you must understand its performance to write your analysis. Note that this would be much easier using STL in C++!
- You do not have to do any error checking of the data contained in the file.
- Timing of the code segments can be done as described in class, using the `clock()` function in `<time.h>`.
- As before, follow good programming practices.

Notes

- Be sure you are reading the data file correctly before continuing! Note that all of the data in the file will be integers, so please read the file using only integer types.
- Start with small files to make sure your algorithm(s) work properly. Then create some large files. You can do this with random numbers. Use Excel or other tool to graph your points to make sure you know what the convex hull should be!
- The name of your source code file should be reflect your last name and project number; e.g., `gousieA3.cpp`. You are welcome to use multiple files, though. In that case, bundle them together; e.g., `gousieA3.zip`.
- Turn in the project via email to `mgousie@wheatonma.edu` before midnight on the due date.
- A printed version of your source code and your one-page report is due in class on November 11. Write/print and sign the Wheaton Honor Code Pledge on what you turn in: "I have abided by the Wheaton College Honor Code in this work."
- Remember to save all of your work until your project is returned.

*There are some minds like either convex or concave mirrors,
which represent objects such as they receive them,
but never receive them as they are.*

– Joseph Joubert