CS 2110: Computer Programming Lab (Odd 2012)
Lab 7

The Convex Hull Problem

Instructor - John Augustine

October 1, 2012

Notes

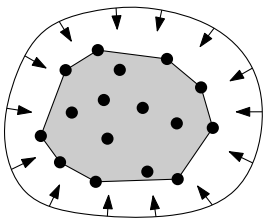
- Your code must be written as a single .c file so that you can upload it to moodle.
- Please follow the input/output format. This is very important from the automated grading point of view.

Problem Definition

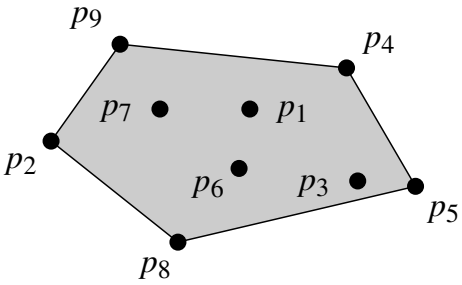
Consider the following scenario. You are managing a set of oil wells, each located at some point (x, y) on a 2D plane. You need to put an expensive fence around your wells. Therefore, you want to:

- minimize the total length of the fence and
- ensure that every oil well is within your fence.

More precisely, let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points — they represent the location of the oil wells. Each p_i is given as a pair (x_i, y_i). Your program must be able to output a "fence" polygon F such that every $p_i \in P$ is inside or on the boundary of F. Furthermore, you must ensure that the perimeter of F is minimized.
What does F look like? Consider (as shown below) a rubber band that is stretched around your points and the released. The rubber band will assume the shape of a polygon that is convex and encloses every point in P. That polygon is precisely the required output.



Notice that the vertices of F will always be some point in P. (Think about why this is the case.) Also notice that the leftmost point[1] is always a vertex in F. (Again, think about why this is the case.) Therefore, the polygon F is fully described by a sequence of points (that are all elements of P) starting from the leftmost point[2] and tracing the vertices of F in a clockwise manner.



In the above figure, the required fence polygon $F = (p_2, p_9, p_4, p_5, p_8)$. (Note that we don't have to repeat p_2 at the end as it is assumed that the last point p_8 must be connected to the first point p_2 in order to complete the polygon.)

0.1 Command Line Arguments

Your command line argument for this assignment is a single input file name.

0.2 The Input File Format

The input file format includes multiple test cases. Each new test case starts with the keyword "NewCase" which must be placed in a line without anything else in that line. After the NewCase line, the input must be a set of points (one per line). Each point must be specified as two integers with a space between them. When all the points in your test case have been given, you can either

- start a new test case with the keyword "NewCase" or
- mark the end of all inputs by the keyword "TheEnd" in a line by itself.

The following is a simple example.

```
NewCase
0 0
5 0
5 5
4 4
NewCase
10 20
15 30
5 40
30 35
TheEnd
```

[1]For the purpose of this assignment, assume that there is a unique leftmost point.

[2]Although you can assume that there is a unique leftmost point, ponder about what might be a good starting point if there is more than one leftmost point.

0.3 Output Format

Your final output must be printed on the screen. The output pertaining to each test case must be printed each in one line. Thus, with 2 test cases, your output must be of the form:

```
output line pertaining to test case 1
output line pertaining to test case 2
```

The format of each line is a clockwise sequence of points (starting from the leftmost) that form the fence polygon. Since it is well understood that each point is a pair of integers (x, y), we can simply represent each output line by a sequence of integers. For example, the first test case in the sample input should give us the following output.

```
0 0 5 5 5 0
```

Thus the corresponding output fence polygon $F = ((0, 0), (5, 5), (5, 0))$ taken in clockwise order.

The above sample input (taking both test cases into consideration) should give you the following output.

```
0 0 5 5 5 0
5 40 30 35 10 20
```

Uploading into MOODLE

Your code should be written as a single .c file. This file must be uploaded into moodle. A link will be set up for this purpose in moodle.

To Ponder About ...

What is the running time of your algorithm? Can you think of a faster algorithm?

Your TA for this lab

CS08B031, CS10B052, CS11B001 — CS11B009	Shrikant Polawar
CS11B011 — CS11B021	Sai Sreennivas
CS11B022 — CS11B032	Nishaanth
CS11B033 — CS11B042	Saurav Kant Jha
CS11B043 — CS11B053	Tejas Kulkarni
CS11B054 — CS11B063	Paresh Nakhe