Data structures project, Testing document

Heikki Haapala and Aleksi Markkanen
Student numbers 014090190 and 013126382
6 pages

February 25, 2013

Contents

1 Introduction **2**

2 Inputs Used in Tests **3**

3 Graphical Presentation of the Results of the Empirical Testing of the Correctness of the Program **4**

1 Introduction

The program was tested using *JUnit* tests during developement. The tests include cases for degenerate hulls and also contain normally distributed 2D point sets.

The correct output was verified using *Octave*, and correct results were saved so that the *JUnit* tests could check the output of the program.

2 Inputs Used in Tests

Different distributions were used as inputs during testing; these include normal distribution, exponential distribution and uniform distribution. We also used a combination of different distributions for x and y coordinates, respectively.

Testing was conducted with and without the *Akl-Toussaint heuristic*.

We also implemented a simple iteration functionality to the program and this allowed us to capture average running times. This eliminates interferance from other running processes.

3 Graphical Presentation of the Results of the Empirical Testing of the Correctness of the Program

In the following pictures the base point set are be coloured in green if Akl-Toussaint was used and red otherwise. Points left after Akl-Toussaint heuristic are coloured in red. Points of the convex hull are coloured in blue with blue lines connecting them.

Figure 1: A basic test with 10000 points, Gift wrapping and Akl-Toussaint heuristic used.

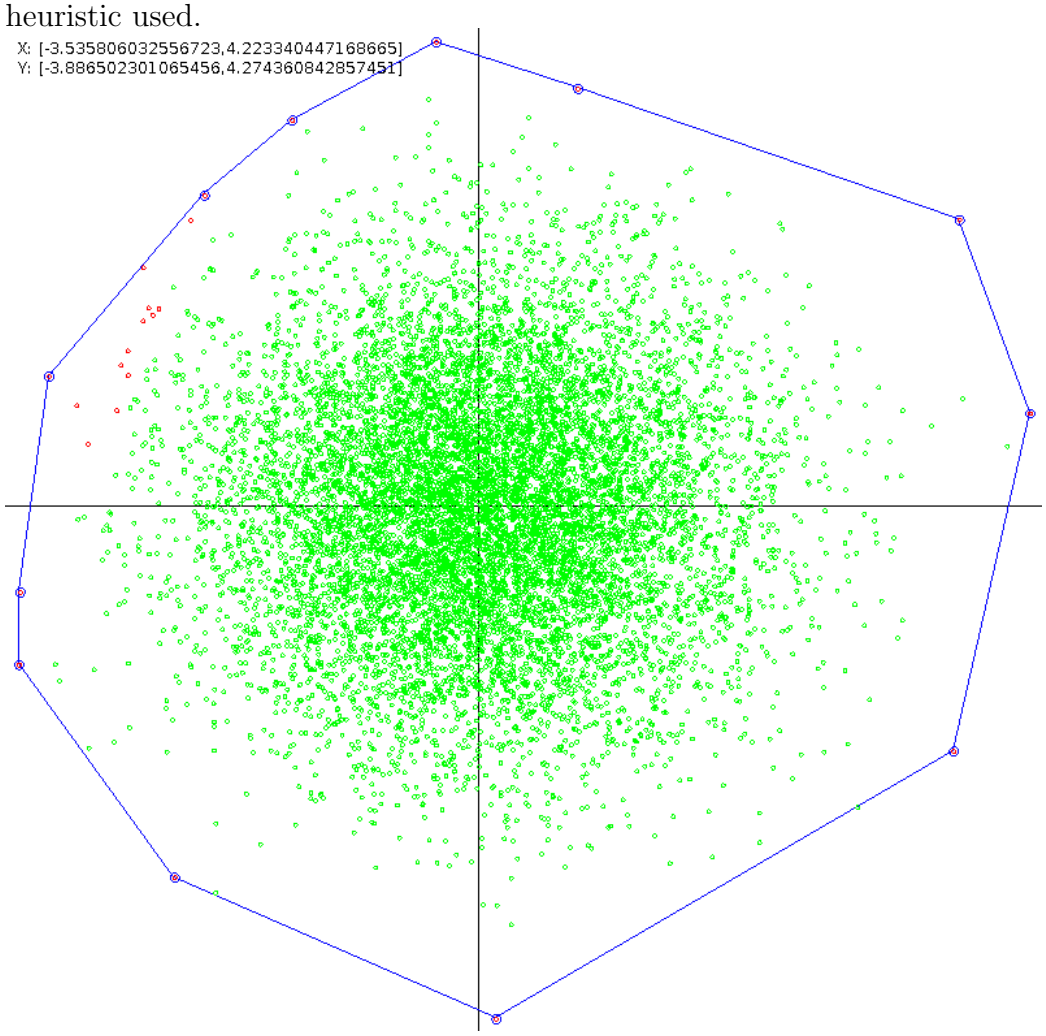


Figure 2: Gift wrapping used on a set of 10000 collinear points, all points included in hull.

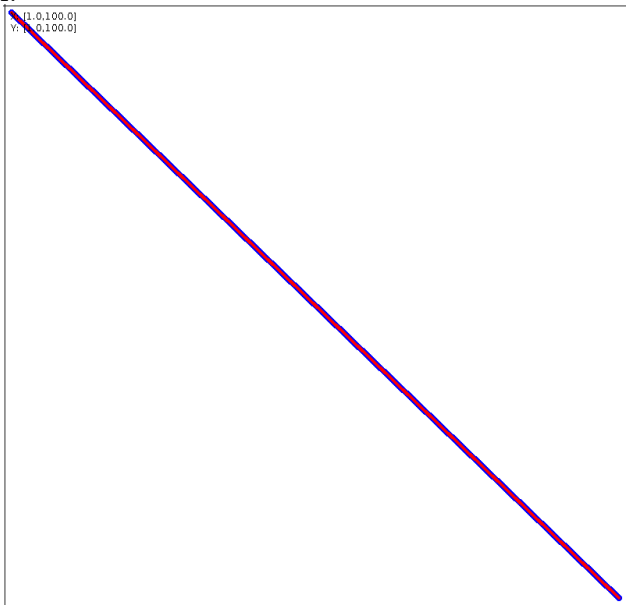
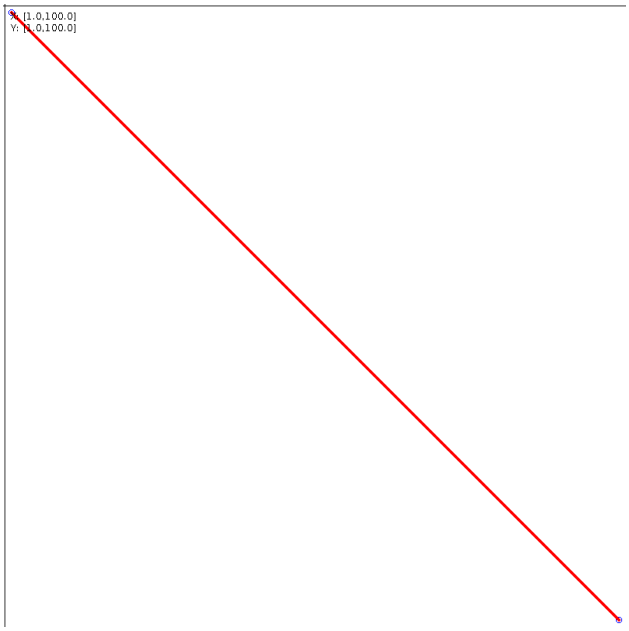
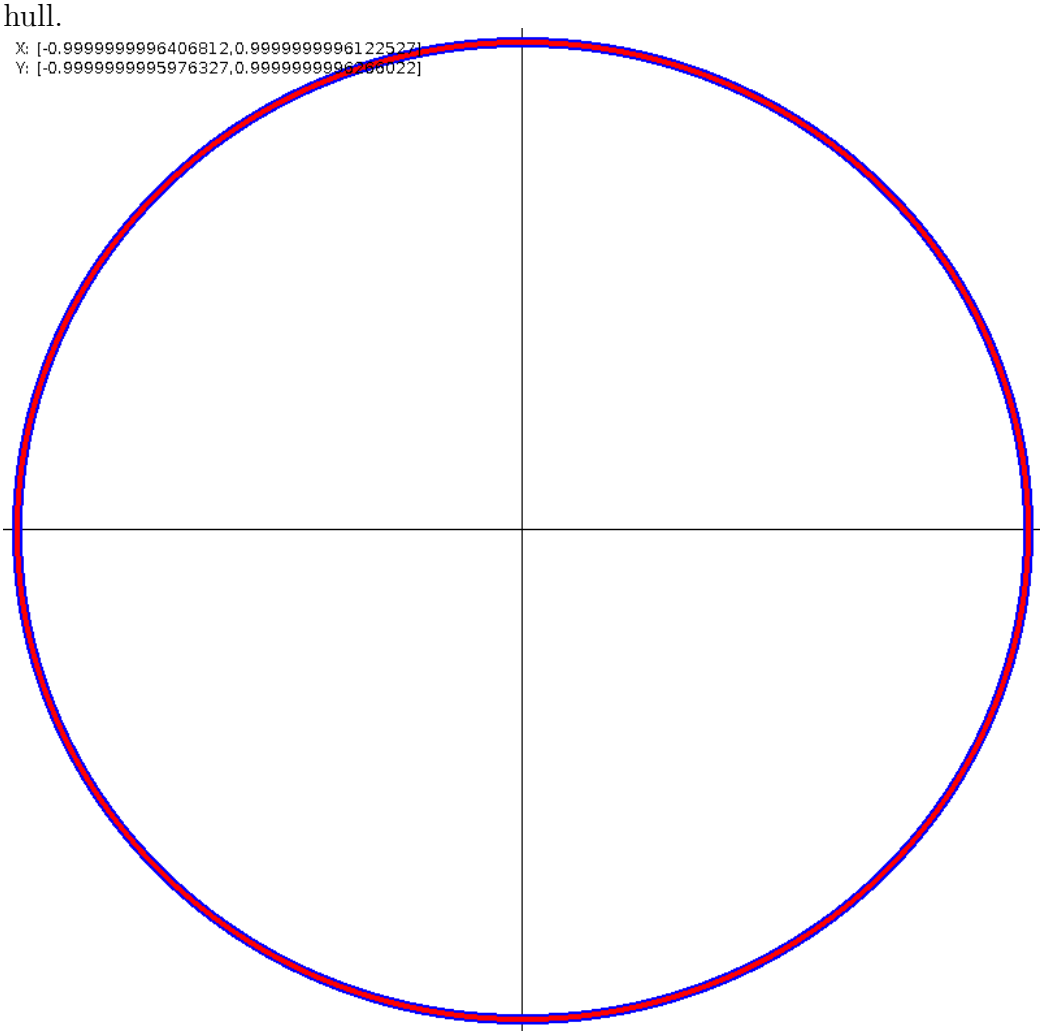


Figure 3: QuickHull used on the same set of 10000 collinear points, only end points included in hull.



As seen from figures (2) and (3) we allowed differences in output with collinear point sets. Only condition was that the algorithm had to include both end points of the set.

Figure 4: A test with 100000 points in a circle, all points included in the hull.



We also tested algorithm performance with worst case scenarios like seen in figure (4).