ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Prof. Dr. B. Gärtner, A. Gundert,
Dr. M. Hoffmann, Prof. Dr. E. Welzl
URL: `http://www.ti.inf.ethz.ch/ew/courses/CG13/`

Computational Geometry — Homework 1 — HS13

Exercise 1 (10 Points)

Show that $\lceil n/2 \rceil$ vertices always suffice and are sometimes necessary to guard the complement of any simple polygon with n vertices.

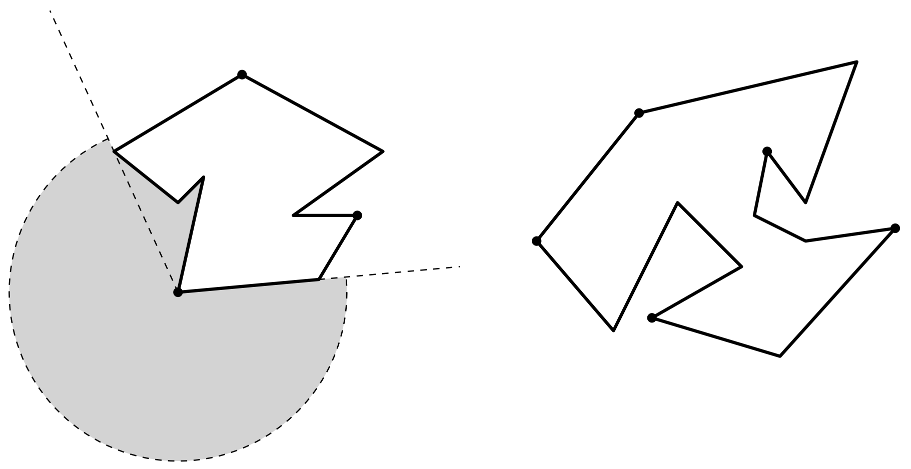

Figure 1: Two simple polygons with vertices guarding their complement

Exercise 2 (10 Points)

Usually, computers do not represent real numbers exactly, but in a limited precision floating point representation. Round-off errors may cause unexpected results, especially when comparing two values that are almost equal. Such behaviour can cause trouble in geometric algorithms.

We focus on the following setting: For three points a, b, c, the predicate `rightturn(a,b,c)` should indicate whether c lies to the right of the (oriented) line ab. For points that are far enough apart, the predicate works correctly. However, if the points a, b, c are close to collinear, `rightturn(a,b,c)` might not give the correct answer, because of round-off errors. For a fixed ordered triple, it will always give the same answer (correct or not correct), but circular permutations of the same three near-collinear points a, b, c do not necessarily yield the same result, e.g., `rightturn(a,b,c)` does not have to be consistent with `rightturn(b,c,a)`. This can make the Jarvis wrap, which you have seen in the lecture, fail completely. (As a reminder, a pseudocode formulation is given on the last page of this homework.)

Figures 2 and 3 show two sets of input points, together with the respective outcomes of the Jarvis wrap. Both sets consist of points lying close to a triangle (so that each point that is not one of the triangle's vertices "belongs" to exactly one of its sides). In the case depicted in Figure 2, the algorithm fails to find the convex hull and instead returns a polygon which does not even contain all the points inside. In the second case, depicted in Figure 3, it gets into an infinite loop, adding the lower thick point followed by the upper thick point again and again.

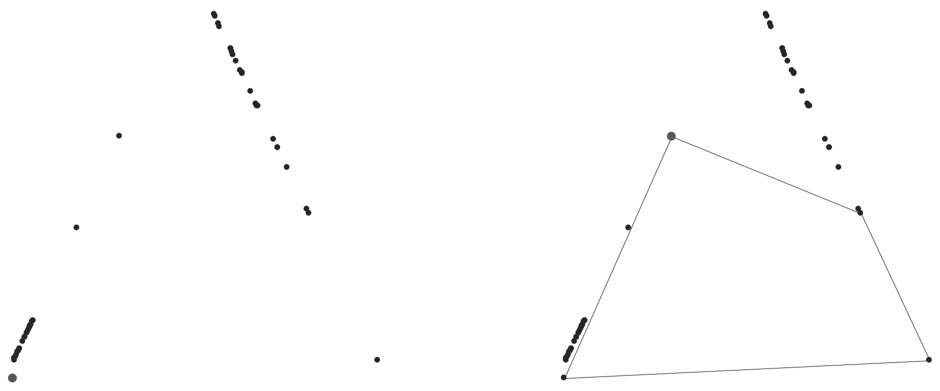

Figure 2: The first point set and its "convex hull"

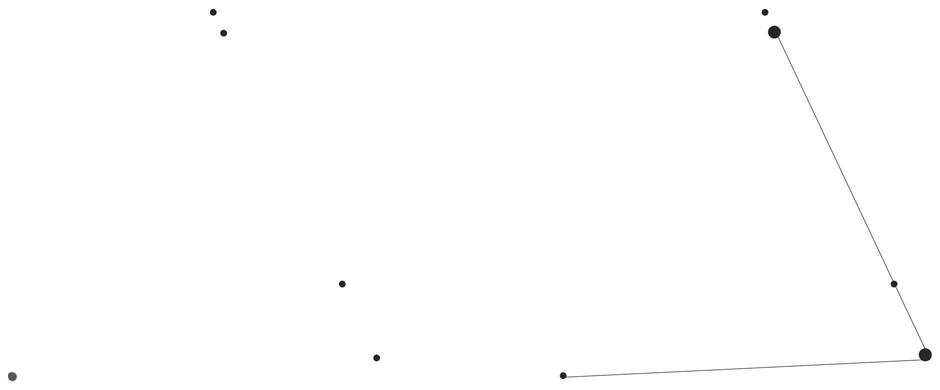

Figure 3: The second point set and its "convex hulls"

Question: How does it happen that the algorithm fails in these two examples? For each of the two inputs depicted in the figures, describe a run of the algorithm (step by step) that leads to the result described on the right side of the figure. You are allowed to determine the order of (the relevant) points in the input (in the `points` array) and values of the predicate `rightturn` obeying the following rules:

$$\texttt{rightturn(a,b,c)} := \begin{cases} \texttt{!rightturn(a,c,b)} & \text{if all three points are distinct} \\ \texttt{false} & \text{if any two points are the same} \\ \texttt{false} & \text{if } a, b, c \text{ are all distinct and not almost collinear} \\ & \text{(i.e., do not all lie on the same side of the triangle)} \\ & \text{and } c \text{ is to the left of the oriented line } ab \\ \texttt{true} & \text{if } a, b, c \text{ are all distinct and not almost collinear} \\ & \text{and } c \text{ is to the right of the oriented line } ab \\ \texttt{any} & \text{if } a, b, c \text{ are distinct and almost collinear} \\ & \text{(i.e., on the same side of the triangle)} \end{cases}$$

Exercise 3 (30 Points)

Choose one of the topics below to investigate. Find the relevant research papers, surveys or textbooks that deal with this problem and find out what is known about it. What are the main results? What are the open questions related to this problem? You are not supposed to read papers in detail, but rather to try to gain an overview. Hand in a short report (of between 1 and 2 pages) about the problem and what you have found out. Your report should contain:

- an informal description of the problem using your own words (as you would explain it to a friend),
- a precise definition of the problem (in which even the most nitpicking reader is not able to find anything unclear),
- the important results regarding this problem (providing enough explanations to make the difference between the results apparent, but without going into unnecessary details),
- the current state of the problem (how much has already been solved, what remains open),
- a complete list of references (every theorem or result you state should have an appropriate citation so that it is easy to find [and check]; if you have fewer than 3 or 4 references, you might have searched not thoroughly enough).

Note that you can access many journals from the ETH network only. If you want to search at home, you might need to connect to the ETH intranet using VPN.

(a) Minimum weight triangulation.

(b) Counting planar triangulations.

(c) Additively weighted Voronoi diagrams.

(d) Higher order Delaunay triangulations.

(e) Halfplane range searching.

(f) Order types of point sets.

(g) Pseudo-triangulations.

(h) Delaunay refinement meshing.

(i) Kinetic data structures in geometry.

(j) Linear programming with few violated constraints.

(k) Support vector machines.

(l) Geometric in-place-algorithms.

(m) Well-separated pair decomposition.

(n) Core sets.

The choice of your topic should be briefly discussed with the assistant, so that no topic is assigned twice.

Due date: 10.10.2013, 13h15

Jarvis Wrap

This is a pseudocode description for the Jarvis wrap:

```
INPUT: points[1..n];
OUTPUT: convex_hull;

pt_start := pt_current := lexicographically_smallest_element(points);
pt_next := first_element_of_points_different_from(pt_start);

do {
  convex_hull.add(pt_current);
  for (i := 0; i < n; i++) {
    p := points[i];
    if (rightturn(pt_current, pt_next, p)) then pt_next := p;
  }
  pt_current := pt_next;
  pt_next := pt_start;
} while (pt_current != pt_start);
```

The predicate `rightturn(a,b,c)` is calculated as follows:

```
rightturn(a,b,c) := [(b.x - a.x)*(c.y - a.y) < (c.x - a.x)*(b.y - a.y)];
```