Lehrstuhl für Informatik I
Effiziente Algorithmen und
wissensbasierte Systeme
Universität Würzburg

Würzburg, October 19, 2016

Dr. Joachim Spoerhase
M.Eng. Dongliang Peng

Homework Assignment #1

Computational Geometry (Winter Term 2016/17)

– mit Lösungen –

1a	b	c	2a	b	c	Summe
2	4	4	3	5	2	20

Exercise 1

Recall that a subset A of the plane is called *convex*, if, for all points $u, v \in A$, the line segment $\overline{uv}$ is also contained in A. Let S be a finite set of points. Then we define the set

$$\mathcal{H}(S) := \{ h \mid h \text{ is a closed half-plane and } S \subseteq h \}$$

of all closed half-planes that contain S. (A *closed* half-plane h contains its own boundary, the line ∂h). The *convex hull* $\mathrm{CH}(S)$ of S can be defined as the intersection $\bigcap \mathcal{H}(S)$ of all these half-planes. Prove that the convex hull fulfills the following properties. (For properties b) and c), we assume that S contains at least three non-collinear points. You may use the property that half-planes are convex.)

a) $\mathrm{CH}(S)$ is convex. **[2 points]**

b) Let $\mathcal{H}_2(S) := \{ h \mid h \in \mathcal{H}(S) \text{ and } |\partial h \cap S| \geq 2 \}$ be the set of all closed half-planes that contain S and that have at least two points of S on their own boundary. Then

$$\mathrm{CH}(S) = \bigcap \mathcal{H}_2(S) \,.$$

[4 points]

c) $\mathrm{CH}(S)$ is a (convex) polygon of which all corners are points of S. **[4 points]**

Lösung zu Aufgabe 1

a) Proof by contradiction. Suppose there is a point $w \in \overline{uv}$ with $u, v \in \mathrm{CH}(S)$ and $w \notin \mathrm{CH}(S)$. Then there must be a half-plane h with $u, v \in h$ and $w \notin h$. This is impossible.

b) $\mathcal{H}_2(S) \subseteq \mathcal{H}(S) \Rightarrow \mathrm{CH}(S) = \bigcap \mathcal{H}(S) \subseteq \bigcap \mathcal{H}_2(S)$. It remains to show $\bigcap \mathcal{H}_2(S) \subseteq \mathrm{CH}(S)$. Suppose that $p \in \bigcap \mathcal{H}_2(S) \setminus \mathrm{CH}(S)$. Rotate the plane so that p is below all points in S. Let q be the point in S with the smallest y-coordinate. Then all the points of S lie in the half-plane which is bounded by the horizontal line ℓ going through q, and opened to the top. We now rotate ℓ to the left (right) around q, until ℓ meets another point from S. We call the resulting half-plane h_1 (h_2).

Obviously S is in both half-planes and both half-planes contain at least two points of S on their boundary, that is, $h_1, h_2 \in \mathcal{H}_2(S)$. However, the intersection of both, $h_1 \cap h_2$, does not contain any point below q, thus also not p. Contradiction.

c) Assume there are no 3 collinear points.

Observation: For every point $a \in S$, there exists either no half-plane or exactly two half-planes in $\mathcal{H}_2(S)$ such that they have a on their boundary.

Proof: If only one half-plane uses a as a point of its boundary, then we can rotate this half-plane around a until its boundary meets another point of S, so that there are two half-planes.

For polygons: it is known that $\partial(A \cap B) = (\partial A \cap B) \cup (\partial B \cap A)$. Thus $\partial CH(S)$ is the non-overlapping union of a finite number of line segments, that is, a polygon. Assume that a corner p of $CH(S)$ is not a point from S. Since p is on the boundary, there exists a half-plane in $h \in \mathcal{H}_2(S)$, which has the p on its boundary. Then there are two points $a, b \in S$, which lie on the boundary of h. Since p is a vertex, $p \notin \overline{ab}$. Without loss of generality, assume $a \in \overline{pb}$. Then there exists another half-plane $h' \in \mathcal{H}_2(S)$ which has a on the boundary. But this h' cannot contain both b and p, so p cannot be a corner of $CH(S)$. If there are collinear points, the convex hull remains the same after removing the point in the middle. For this, the property holds, so it is valid even after reinserting the points.

Exercise 2

In Exercise 1 we have shown that the convex hull of a finite set of points in the plane is a polygon. We require the output of a convex-hull algorithm to be a list of the vertices of the corresponding polygon in clock-wise order.

a) Prove that every algorithm that computes the convex hull of n points needs a running time of $\Omega(n \log n)$. That means that the algorithm of the lecture is optimal in the sense of the asymptotic running time.

Hint: Use the property that *sorting* n keys (in certain computer models) requires running time $\Omega(n \log n)$. **[3 points]**

b) Let P be a simple, not necessarily convex polygon in the common list representation. (In a *simple* polygon the edges are crossing-free.) Develop an algorithm that computes the convex hull of the vertices of this polygon in $O(n)$ time. Explain why this is not a contradiction to the claim in subexercise a). **[5 points]**

c) Is there also a linear-time algorithm if we drop the requirement of simplicity in subexercise b)? **[2 points]**

Lösung zu Aufgabe 2

a) Given: a (unordered) set of numbers $a_1, \ldots, a_n$. We map each number a_i to the point (a_i, a_i^2). Then all the points will be on a convex parabola. Thus the convex hull immediately yields a sorting of the points. As the mapping runs in linear time, this algorithm would sort in $o(n \log n)$ time, if there was a $o(n \log n)$ time algorithm for calculating the convex hull. Contradiction.

b) From the book *Computational Geometry:Algorithms and Applications–Third Edition*: Use a variant of algorithm CONVEXHULL where the vertices are not treated in lexicographical order, but in some other order.

An instance of polygon can be found in Figure 1.

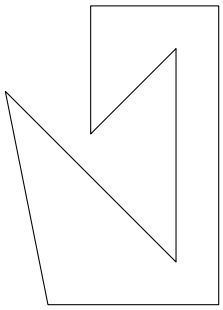

FIGURE 1: An instance of polygon.

A possible algorithm would be Algorithm 1.

The correctness of Algorithm 1 can be shown by proving that the following invariant holds for every step i of the algorithm. That is, there is a vertex p_j ($j \geq i$), such that $p_1, \ldots, p_j \subseteq \bigcap_{l=1}^{t-1} h^+(q_l, q_{l+1})$, where $h^+(q_l, q_{l+1})$ is the half-plane with its boundary going through edge (q_l, q_{l+1}), and $\langle q_1, \ldots, q_t \rangle = \mathcal{L}$ at step j.

Note that as the calculation of a non-crossing polygon already requires $O(n \log n)$ time, this is not a contradiction to a).

c) No, because a non-simple polygon can be obtained from a set of points with arbitrary order in linear time. So if one could compute the convex hull for a non-simple polygon in linear time, then one could compute the convex hull for an arbitrary set of points. That would be a contradiction to a).

This assignment is due at the beginning of the next lecture, that is, on October 26 at 10:15. Solutions will be discussed in the tutorial on Friday, October 28, 14:15–15:45 in room SE I.

Algorithm 1 Compute conver hull for a polygon

Input. A simple, not necessarily convex polygon P
Output. A list containing the vertices of $CH(P)$ in clockwise order.

1: **function** CHPolygon(P)
2: Find the left most vertex p_{left}.
3: Number the vertices on P clockwise, starting at p_{left}, as $p_1, \ldots, p_n$.
4: Put the vertices p_1 and p_2 in a List $\mathcal{L}$, with p_1 as the first vertex. Let v_l represent the last vertex in $\mathcal{L}$, and v_{l-1} represent the second last vertex in $\mathcal{L}$.
5: $i \leftarrow 3$.
6: **while** $i \leq n$ **do**
7: Append p_i to $\mathcal{L}$.
8: ExsitNonRightTurn $\leftarrow$ false.
9: **while** $\mathcal{L}$ contains more than two vertices **and**
the last three vertices from $\mathcal{L}$ do not make a right turn **do**
10: Delete the middle of the last three vertices from $\mathcal{L}$.
11: ExsitNonRightTurn $\leftarrow$ true.
12: **if** ExsitNonRightTurn == true **then**
//v_l and p_i indicate the same vertex.
13: **if** the direction of $\overrightarrow{p_i p_{i+1}}$ is in the direction range of the wedge consisting of $\overrightarrow{v_l v_{l-1}}$ and $\overrightarrow{p_i p_{i-1}}$ **then**
14: $i \leftarrow$ IgnoreVerticesInBay(v_l, v_{l-1}, P, $i+1$).
15: $i \leftarrow i+1$.
16: **return** $\mathcal{L}$

17: **function** IgnoreVerticesInBay(v_l, v_{l-1}, P, i)
//W.l.o.g., no p_i is on $\overrightarrow{v_l v_{l-1}}$.
18: **while** $\overrightarrow{p_i p_{i+1}}$ does not cross $\overrightarrow{v_l v_{l-1}}$ **do**
19: $i \leftarrow i+1$.
20: **return** i