"The convex hull of a configuration of nails in a board is bounded by the rubber band let loose around the nails."

5 Convex hulls and paradigms

Computing the convex hull of finitely many points in the plane is the most prototypical problem studied in the early days of geometric algorithms, see [5] which started off a sequence of papers on the subject. Its role in two-dimensional computational geometry is similar to that of sorting in the classic area of one-dimensional algorithms: it serves as an example that permits the introduction of a wide variety of algorithmic paradigms.

Definitions. We already know what it means a set is convex, see section 4. Observe that the intersection of two convex sets is again convex. More generally, the common intersection of any family of convex sets is convex. Given a set S, it thus makes sense to define

$$\operatorname{conv} S = \bigcap_{S \subseteq C, \text{convex}} C,$$

and to call it the *convex hull* of S. Thus, $\operatorname{conv} S$ is the smallest convex set that contains S. A more analytical definition is often useful. A point x is an *affine combination* of S if there are finitely many points p_i in S and corresponding real numbers ξ_i so that $x = \sum \xi_i p_i$ and $\sum \xi_i = 1$. x is a *convex combination* if furthermore $\xi_i \geq 0$ for all i. The *affine hull*, $\operatorname{aff} S$, is the set of affine combinations, and the *convex hull*, $\operatorname{conv} S$, is the set of convex combinations of S, see figure 5.1.

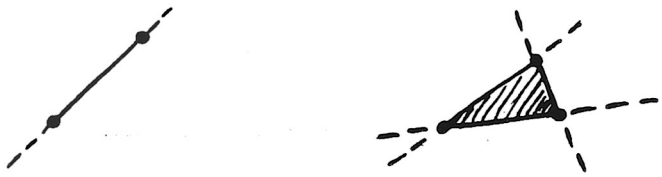

Figure 5.1: The affine hull of two points is a line, and the convex hull is a line segment. The affine hull of three non-collinear points is a plane, and the convex hull is a triangle.

Representation. Note that the convex hull of a finite set $S \subseteq \mathbf{R}^2$ is a convex polygon. More precisely, the boundary of the convex hull, $\operatorname{bd} \operatorname{conv} S$, is a polygon, and the convex hull itself, $\operatorname{conv} S$, is a polygon together with its inside. In any case, the cyclic sequence of vertices of this polygon is a suitable representation of $\operatorname{conv} S$, see figure 5.2. For notation, use $\mathtt{next}(p)$ to denote the counterclockwise (ccw) next vertex of p and use $\mathtt{prev}(p)$ to denote the clockwise (cw) next vertex.

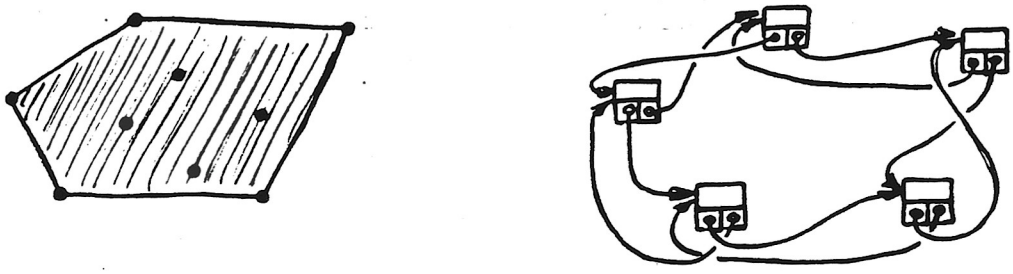

Figure 5.2: The vertices of $\operatorname{conv} S$ are arranged as a doubly-linked cyclic list.

Observe that the requirement to compute the points in sorted order around the convex hull immediately implies that $\Omega(n \log n)$ time is necessary in the worst case. This is because any such convex hull algorithm can be used to sort numbers. It turns out that $\Omega(n \log n)$ is also a lower bound just for identifying the convex hull vertices among the input points, but this is not so easy to prove, see [11].

Incremental construction. Constructing conv S incrementally means to add a point at a time and to update the data structure each time. This is particularly easy if the points are presorted, say from left to right. When the ith point is added, we find its lower and upper tangents to the convex hull of the first $i-1$ points. This method can be generalized to 3 and higher dimensions where it is sometimes referred to as the beneath-beyond method, see [10] or [4, chapter 8]. The time-complexity for n points in $\mathbf{R}^d$ is $O(n^{\lceil d/2 \rceil})$ in the worst case.

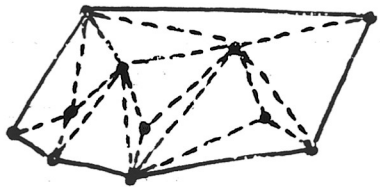

Figure 5.3: The solid edges show the boundary of the convex hull, and the broken edges show temporary edges constructed by the incremental algorithm.

initial step. store points p_1, p_2, p_3 in a cyclic list of length 3.
general step. **for** $i := 1$ **to** n **do** compute endpoints t and b of the top and bottom tangents from p_i:
1. set $t := b := p_{i-1}$;
2. **while** $p_i, t,$ **next**(t) form a right-turn **do** $t :=$ **next**(t) **endwhile**;
3. **while** $p_i, b,$ **prev**(b) form a left-turn **do** $b :=$ **prev**(b) **endwhile**;
4. add p_i as a new vertex right after b and right before t

endfor.

ANALYSIS. The time for an iteration of the **for**-loop varies between constant and proportional to n. Using an amortization argument, we can show that the time per point is only constant. One iteration of the **for**-loop adds one point to the cyclic list. The point pays for its own addition and for the first iteration of the two **while**-loops. Every iteration of the **while**-loops after the first removes a point from the list. The point pays for its own removal. Each point pays at most 3 times, which implies the running time of the algorithm is $O(n)$. Recall, however, that the points are assumed to be sorted before the algorithm starts, and this takes time $O(n \log n)$.

Divide-and-conquer. The general idea of the divide-and-conquer paradigm is to divide a problem into subproblems, to solve each subproblem recursively, and to eventually combine the solutions. It is usually advantageous to divide into roughly equally large subproblems. For the convex hull problem we use a vertical line to divide into two subproblems. The two resulting convex polygons are combined by adding the top and bottom common tangents and removing two chains of edges between them. This method has been generalized to $\mathbf{R}^3$ [8], where it takes time $O(n \log n)$. In 4 and higher dimensions the merge step seems too complicated to justify the effort.

Again assume the points are sorted from left to right. The recursive procedure takes an index interval and returns ℓ and r, the leftmost and rightmost vertices in the list representing the convex hull.

convex_hull$(i, j; \ell, r)$;
if $j - i \leq 2$ **then** build the cyclic list from scratch;
else convex_hull$(i, \lfloor \frac{i+j}{2} \rfloor, \ell_L, r_L)$; convex_hull$(\lfloor \frac{i+j+2}{2} \rfloor, j, \ell_R, r_R)$;
compute the endpoints t_L, t_R and b_L, b_R of the top and bottom common tangents:
1. set $t_L := b_L := r_L$ and $t_R := b_R := \ell_R$;
2. **loop while** $t_L, t_R,$ **prev**(t_R) form a left-turn **do** $t_R :=$ **prev**(t_R) **endwhile**;
if $t_R, t_L,$ **next**(t_L) form a right-turn **then** $t_L :=$ **next**(t_L) **else exit endif**
forever;
3. symmetric to 2 the bottom common tangent

endif.

ANALYSIS. Each recursive call pays for two iterations of the combined **forever**- and **while**-loop. Every other iteration is paied for by the point removed from the collection of cyclic lists representing the convex hulls of the various not yet combined sets. Again the running time is $O(n)$ after sorting.

Gift-wrapping. Suppose we know one of the vertices of conv S, say the leftmost point $p_0 \in S$. We can then rotate a half-line anchored at $p = p_0$, say in a ccw order, until it encounters another point, $q \in S$. pq is an edge of conv S and q is the ccw next vertex after p. The rest of the convex hull can be completed by iterating this step until the polygon is closed. To generalize this approach to 3 and higher dimensions, the sequential search needs to be replaced by a graph search method, such as depth-first or breadth-first search, see [2].

initial step. find $p_0 \in S$ with minimum x_1-coordinate, and initialize $v := (0, -1)$ and $p := p_0$;
general step. **repeat** find point $q \in S$ that minimizes the angle from v to $q - p$;
set $v := q - p$ and $p := q$;
until $q = p_0$.

ANALYSIS. The two-dimensional gift-wrapping algorithm, published in [6], takes time $O(n)$ per point, and thus time $O(nh)$ altogether, if h of the n points end up being vertices of the convex hull. In the worst case, $h = n$ and the time-complexity is $O(n^2)$. It should be noted, however, that in many cases h is considerably smaller than n. For example, if n points are chosen randomly and independently from a uniform distribution in the unit-square then the expected value of h is proportional to $\log_2 n$, see e.g. [9].

Homework exercises

5.1 The *diameter* of a set S is the maximum distance between any two points in S. Let S be a finite set of points in $\mathbf{R}^2$ and assume the cyclic sequence of vertices of conv S has already been computed. Give an algorithm that finds the diameter of S in time $O(h)$, where h is the length of the sequence.
(Hint. The diameter of S is defined by two "opposite" vertices of conv *S.)*

5.2 The *width* of S is the smallest distance between two parallel lines that sandwich S. Assume the sequence of convex hull vertices is available, as in exercise 5.1. Give an algorithm that computes the width of S in time $O(h)$.

5.3 Let S be a set of n points in $\mathbf{R}^2$. Call bd conv S the *1st convex layer* of S, and for $i \geq 2$ define the *ith convex layer* as the boundary of conv $(S - T_{i-1})$, where T_{i-1} contains the vertices of the first $i - 1$ convex layers. Give an algorithm that computes all convex layers of S in time $O(n^2)$.

5.4 Consider the following recursive algorithm for constructing the convex hull of a set S of n points in $\mathbf{R}^2$. To get started, find points p and q with smallest and largest x_1-coordinates. Treat the points on the two sides of the line, ℓ_{pq}, through p and q separately. For the points on one side, find point r furthest from ℓ_{pq}, and recurse for p and r and for r and q. Does this algorithm run in worst-case time $O(n \log n)$?
(Remark. This algorithm has first been described in [3] and later independently in [1]. Only one of the two papers contains the right answer.)

References

[1] A. BYKAT. Convex hull of a finite set of points in two dimensions. *Inform. Process. Lett.* **7** (1978), 296–298.

[2] C. R. CHAND AND S. S. KAPUR. An algorithm for convex polytopes. *J. Assoc. Comput. Mach.* **17** (1970), 78–86.

[3] W. EDDY. A new convex hull algorithm for planar sets. *ACM Trans. Math. Software* **3** (1977), 398–403.

[4] H. EDELSBRUNNER. *Algorithms in Combinatorial Geometry* Springer-Verlag, Heidelberg, 1987.

[5] R. L. GRAHAM. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.* **1** (1972), 132–133.

[6] R. A. JARVIS. On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.* **2** (1973), 18–21.

[7] D. G. KIRKPATRICK AND R. SEIDEL. The ultimate planar convex hull algorithm? *SIAM J. Comput.* **15** (1986), 287–299.

[8] F. P. PREPARATA AND S. J. HONG. Convex hulls of finite sets of points in two and three dimensions. *Comm. ACM* **20** (1977), 87–93.

[9] A. RÉNYI AND R. SULANKE. Über die konvexe Hülle von n zufällig gewählten Punkten. *Z. Wahrscheinlichkeitstheorie* **2** (1963), 75–84.

[10] R. SEIDEL. A convex hull algorithm optimal for point sets in even dimensions. Rept. 81-14, Dept. Comput. Sci., Univ. British Columbia, Vancouver, 1981.

[11] A. C. YAO. A lower bound for finding convex hulls. *J. Assoc. Comput. Mach.* **28** (1981), 780–787.