Computational Geometry (CS60064)
Spring 2024-25

Instructions

(a) The submission deadline is hard. There may be unforeseen glitches during submission. So, for safety, submit your files well ahead.

(b) All submissions should be on moodle only. No email submission will be accepted, excepting medical reasons.

(c) Do not forget to typeset your solutions. In particular, every mathematical expression must be properly typeset, e.g., the square of n must appear as n^2 and not as `n^2`. Improper typesetting may incur up to 25% deduction in marks.

You can use LaTeX for writing (that is what we recommend); else, typeset in Word and convert to pdf.

Handwritten text—converted to images or to pdf—will not be evaluated.

(d) You must submit all the source files and the final pdf as a single zip file. The name of the zip should be your roll number, followed by a hyphen, followed by the assignment number. For example, if your roll number is `XY190047`, then the zip file for the 1st assignment should be named as `XY190047-a1.zip`. For subsequent assignments, your zip files should be named as `XY190047-a2.zip`, `XY190047-a3.zip`, ...

If you typeset in LaTeX, then the zip should contain one tex file, image files if any, and the final pdf.

If you typeset in Word, then the zip should contain one odt/doc/docx file and the final pdf. Image files get embedded in a Word file, so image files are not needed.

Failing this, your assignment will not be evaluated.

Assignment 1

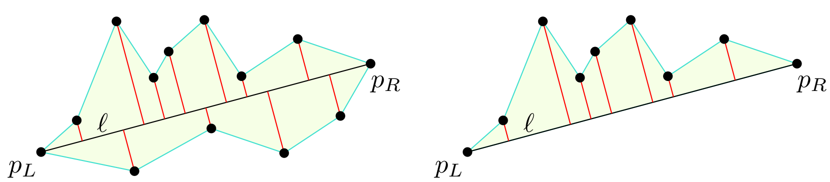
Submission deadline: 19-Jan-2025, 11:55 PM

1.1 Polygon Construction

Given n points on the xy-plane, design an algorithm to construct a simple polygon P such that all the given points serve as vertices of P, and no other points are included as vertices. Provide a proof of correctness for your algorithm and deduce its time complexity. (A *simple polygon* is defined as one in which no two edges intersect, except possibly at their endpoints.) $\boxed{4+3+3=10 \text{ marks}}$

Solution key:

Find the leftmost point p_L and the rightmost point p_R, and join them with a straight-line segment ℓ. Project all points that are above ℓ, on ℓ. Sort these footprints and connect the original points serially in the sorted order to form the upper chain of P; do the same for the lower chain. If one side of ℓ is empty, use ℓ as an edge of the polygon P (as shown in the figure on the right). This can be done in $O(n \log n)$ time.



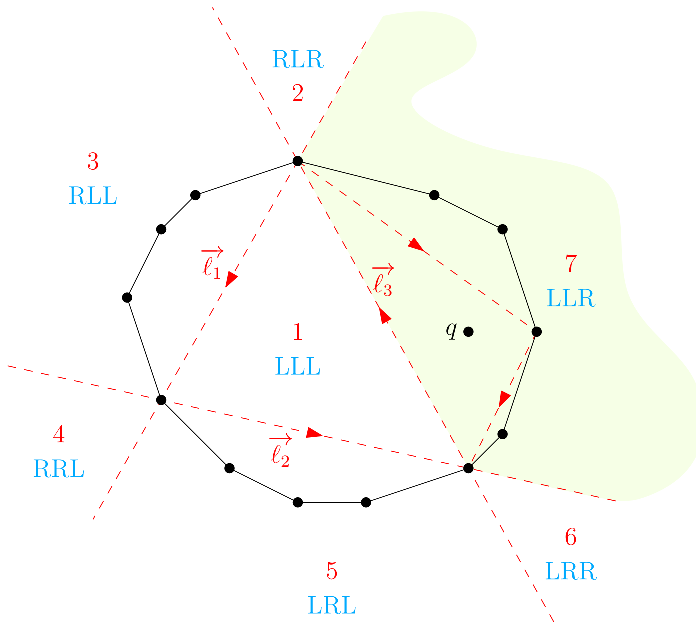
1.2 Point Location

A convex polygon P is provided as a counter-clockwise ordered sequence of n vertices, with their locations specified as (x, y) coordinates. Given a query point q, develop an algorithm to determine whether q lies inside P in $O(\log n)$ time, using $O(n)$ space, including any necessary preprocessing. Justify the time and space complexities of your algorithm. $\boxed{6+2+2=10 \text{ marks}}$

Solution key:

Choose three points on the boundary of P, which are almost equispaced—can be done in $O(1)$ time—via indexing. Construct three directed rays (cut-lines) $\overrightarrow{\ell_1}, \overrightarrow{\ell_2}, \overrightarrow{\ell_3}$ through these points—they partition the 2D-space into seven disjoint regions as shown. The location of the query point q w.r.t. these regions can be determined in $O(1)$ time via three orientation tests. Further refined partitioning can be done through $O(\log_3 n)$ steps, thus giving the precise location of q in $O(\log n)$ time, and in $O(n)$ space.

In the following example, q is initially identified to lie in Region 7, characterized by the unique 3-bit label LLR—indicating that q lies left of $\overrightarrow{\ell_1}$, left of $\overrightarrow{\ell_2}$, and right of $\overrightarrow{\ell_3}$. Subsequently, its position is evaluated with respect to $\overrightarrow{\ell_1}$ and two other rays, resulting in the label LLL, thereby confirming that q lies inside P.



Assignment 2

Submission deadline: 26-Jan-2025, 11:55 PM

2.1 Point location w.r.t. line

For some algorithm, we have to test whether a point r lies to the left or right of the directed line $\overrightarrow{pq}$ through two points p and q. Let $p = (p_x, p_y)$, $q = (q_x, q_y)$, and $r = (r_x, r_y)$.

(a) Show that the sign of the determinant

$$D = \begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix}$$

determines whether r lies to the left or right of the line.

(b) Show that $|D|$ is in fact twice the area of the triangle determined by p, q, and r.

(c) Why is this an attractive way to implement the basic test in any algorithm where the location of a point is determined w.r.t. a directed line? Provide arguments for both integer and floating-point coordinates. $\boxed{6 + 2 + 2 = 10 \text{ marks}}$

Solution key:

(a) Expanding the determinant:

$$D = \begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix} = (q_x r_y - q_y r_x) + p_x(q_y - r_y) + p_y(r_x - q_x).$$

Let $\overrightarrow{u_x}$, $\overrightarrow{u_y}$, and $\overrightarrow{u_z}$ denote the respective unit vectors along the x-, y-, and z-axes. Then,

$$\begin{aligned}
\overrightarrow{pq} \times \overrightarrow{pr} &= \Big((q_x - p_x)\overrightarrow{u_x} + (q_y - p_y)\overrightarrow{u_y}\Big) \times \Big((r_x - p_x)\overrightarrow{u_x} + (r_y - p_y)\overrightarrow{u_y}\Big) \\
&= (q_x - p_x)(r_y - p_y)\overrightarrow{u_z} - (q_y - p_y)(r_x - p_x)\overrightarrow{u_z} \quad [\text{since } \overrightarrow{u_x} \times \overrightarrow{u_y} = \overrightarrow{u_z} \text{ and } \overrightarrow{u_y} \times \overrightarrow{u_x} = -\overrightarrow{u_z}] \\
&= \Big((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)\Big)\overrightarrow{u_z} \\
&= D\overrightarrow{u_z}.
\end{aligned}$$

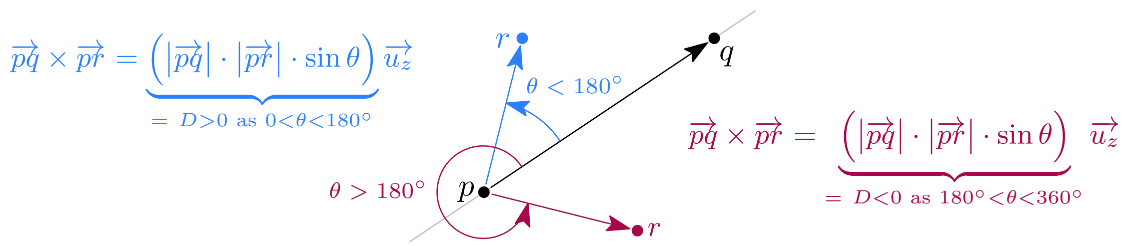


Figure 2.1: Determining the location of a point r w.r.t. the directed line p-to-q using the sign of D, which is basically the scalar value $|\overrightarrow{pq}| \cdot |\overrightarrow{pr}| \cdot \sin\theta$.

Now, consider an alternative way of evaluating the vector $\overrightarrow{pq} \times \overrightarrow{pr}$, as illustrated in Figure 2.1. Recall the cross-product formula that the vector $\overrightarrow{pq} \times \overrightarrow{pr}$ is given as $\left(|\overrightarrow{pq}| \cdot |\overrightarrow{pr}| \cdot \sin\theta\right)\overrightarrow{u_z}$, where, θ is the angle measured counterclockwise from $\overrightarrow{pq}$ to $\overrightarrow{pr}$. Since either of the above two ways gives the same vector, we have

$$D = |\overrightarrow{pq}| \cdot |\overrightarrow{pr}| \cdot \sin\theta.$$

Hence, D is positive, negative, or zero depending on the signed value of $\sin\theta$. In other words, the sign of D determines the position of r, as follows:

(i) $D > 0 \implies 0^\circ < \theta < 180^\circ \implies r \in \text{left}(\overrightarrow{pq})$.

(ii) $D < 0 \implies 180^\circ < \theta < 360^\circ \implies r \in \text{right}(\overrightarrow{pq})$.

(iii) $D = 0 \implies \theta = 0^\circ \implies r$ lies on $\overrightarrow{pq}$.

(b) The area of a triangle formed by points p, q, and r is given by

$$\frac{1}{2}\Big||\overrightarrow{pr}|\sin\theta \cdot |\overrightarrow{pq}|\Big| = \frac{|D|}{2}.$$

(c) **(i)** **Integer coordinates:** The determinant D is computed using only integer arithmetic, ensuring correct output.

(ii) **Floating-point coordinates:** When dealing with high-precision values (e.g., small numbers with numerous decimal places), the determinant computation may yield inaccurate results due to floating-point rounding errors.

2.2 Point location in strip

Let S be a set of n disjoint line segments whose upper endpoints lie on the line $y = 1$ and whose lower endpoints lie on the line $y = 0$. These segments partition the horizontal strip $[-\infty : \infty] \times [0 : 1]$ into $n + 1$ regions: $R_1, \ldots, R_{n+1}$. Give an $O(n \log n)$-time algorithm to build a binary search tree on the segments in S such that the region containing a query point can be determined in $O(\log n)$ time. Also, describe the query algorithm in full detail. $\boxed{5 + 5 = 10 \text{ marks}}$

Solution key:

See Figure 2.2. Any horizontal line within the strip intersects the segments of S in the same order as the order of their lower (or upper) endpoints. A query point q lies in R_i if and only if to its

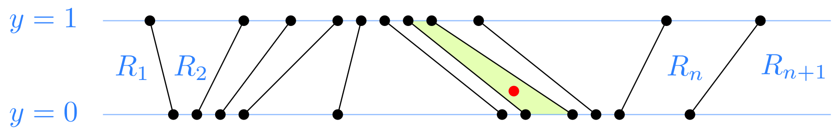


Figure 2.2: Determining the region of a query point in a strip.

immediate left lies the $(i-1)$st segment. The sole exception is when q lies in R_1, in which case there is no segment to the left of q.

Using this observation, we construct a height-balanced binary search tree ordered by the x-coordinates of the lower endpoints of the n segments. (Write the steps in detail.) This construction can be completed in $O(n \log n)$ time.

To determine the region of $q = (q_x, q_y)$, use q_x as the search key to find the segment immediately to its left. (Write the steps in detail.) Since the height of the tree is $O(\log n)$, the time complexity for this search is $O(\log n)$.

Assignment 3

Submission deadline: 03-Feb-2025, 11:55 PM

3.1 Diameter of convex polygon

Diameter of a convex polygon is a/the longest line segment contained within it. Given a convex polygon P with n vertices in counterclockwise order, design an $O(n)$-time algorithm to find a diameter of P. Justify its correctness and why its time complexity is $O(n)$. $\boxed{5 + 3 + 2 = 10 \text{ marks}}$

To design the algorithm, you may use Observation 1 based on the notion of a tangent. Recall that a ***tangent*** to a convex polygon P is a straight line passing through a vertex or edge of P such that the interior of P lies entirely on one side of it. Two vertices u and v comprise an ***antipodal pair*** if there exist two tangents, say T_u and T_v, passing through u and v, such that T_u and T_v are parallel to each other and P is sandwiched between them.

Observation 1. *The endpoints of a diameter always comprise an antipodal pair. However, an antipodal pair may not form a diameter, as shown in Figure 3.1.*

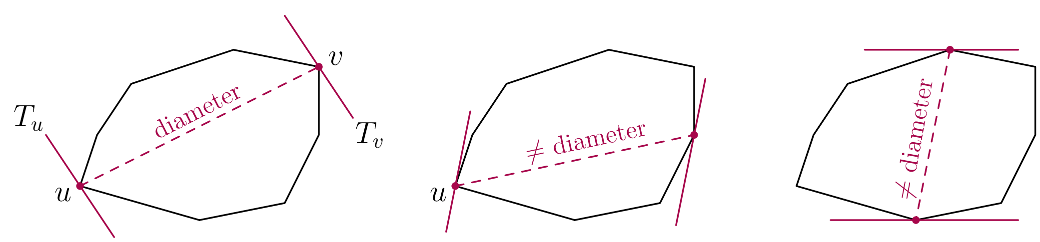


Figure 3.1: Left: $\overline{uv}$ is a diameter of a convex polygon. Middle and right: Not diameters because they are not longest.

3.2 Convex polygon containment

Let P and Q be two convex polygons with m and n vertices, respectively, given in counterclockwise order. Suggest an $O(m \log n)$-time algorithm to check whether P contains Q. Suggest another algorithm that will take time $O(m + n)$.

Derive the time complexities of both. $\boxed{(2 + 2) + (3 + 3) = 10 \text{ marks}}$