Exercises

Problem 1. Prove: every polygon (not necessarily convex) of \( n \) vertices can be triangulated into \( n - 2 \) triangles. (Hint: induction.)

Problem 2. Let \( G \) be a polygon (not necessarily convex); denote by \(|G|\) the number of vertices in \( G \). Suppose that we divide \( G \) into smaller polygons \( G_1, G_2, \ldots, G_t \) for some \( t \geq 1 \) using non-intersecting diagonals. Prove: \( \sum_{i=1}^{t} |G_i| = O(|G|) \).

Problem 3. Consider the following algorithm for triangulating a polygon \( G \):

1. add diagonals to break \( G \) into non-overlapping polygons \( G_1, G_2, \ldots, G_t \) without split vertices
2. for \( i = 1 \) to \( t \) do
3. add diagonals to break \( G_i \) into non-overlapping polygons without merge vertices
4. for every polygon \( G' \) obtained at Line 3 do
5. triangulate \( G' \) using a monotone algorithm

Prove: the above algorithm runs in \( O(n \log n) \) time where \( n \) is the number of vertices in \( G \).

Problem 4. Let \( G \) be an x-monotone polygon whose \( n \) edges are given in clockwise order. Describe an algorithm to sort the vertices of \( G \) by x-coordinate in \( O(n) \) time.

Problem 5 (Polygon Intersection). Let \( G_1 \) and \( G_2 \) be two convex polygons, whose edges are given in clockwise order. Describe an algorithm to compute the intersection of \( G_1 \) and \( G_2 \) in \( O(n) \) time, where \( n \) is the total number of edges in \( G_1 \) and \( G_2 \). Note: the intersection is a polygon and you need to output its edges in clockwise order. (Hint: planesweep.)

Problem 6* (Point in Polygon) Let \( G \) be a convex polygon of \( n \) vertices, which are given in clockwise order. Given an arbitrary point \( q \), describe an algorithm to decide whether \( q \) is inside or outside \( G \) in \( O(\log n) \) time. (Hint: general binary search; see an earlier exercise.)