PREV < Day Day Up > NEXT

Chapter 33: Computational Geometry

Overview

Computational geometry is the branch of computer science that studies algorithms for solving geometric problems. In modern engineering and mathematics, computational geometry has applications in, among other fields, computer graphics, robotics, VLSI design, computer-aided design, and statistics. The input to a computational-geometry problem is typically a description of a set of geometric objects, such as a set of points, a set of line segments, or the vertices of a polygon in counterclockwise order. The output is often a response to a query about the objects, such as whether any of the lines intersect, or perhaps a new geometric object, such as the convex hull (smallest enclosing convex polygon) of the set of points.

In this chapter, we look at a few computational-geometry algorithms in two dimensions, that is, in the plane. Each input object is represented as a set of points $\{p_1, p_2, p_3, ...\}$, where each $p_i = (x_i, y_i)$ and $x_i, y_i \in \mathbf{R}$. For example, an n-vertex polygon P is represented by a sequence $\langle p_0, p_1, p_2, ..., p_{n-1} \rangle$ of its vertices in order of their appearance on the boundary of P. Computational geometry can also be performed in three dimensions, and even in higher-dimensional spaces, but such problems and their solutions can be very difficult to visualize. Even in two dimensions, however, we can see a good sample of computational-geometry techniques.

Section 33.1 shows how to answer basic questions about line segments efficiently and accurately: whether one segment is clockwise or counterclockwise from another that shares an endpoint, which way we turn when traversing two adjoining line segments, and whether two line segments intersect. Section 33.2 presents a technique called "sweeping" that we use to develop an $O(n \lg n)$-time algorithm for determining whether there are any intersections among a set of n line segments. Section 33.3 gives two "rotational-sweep" algorithms that compute the convex hull (smallest enclosing convex polygon) of a set of n points: Graham's scan, which runs in time $O(n \lg n)$, and Jarvis's march, which takes $O(nh)$ time, where h is the number of vertices of the convex hull. Finally, Section 33.4 gives an $O(n \lg n)$-time divide-and-conquer algorithm for finding the closest pair of points in a set of n points in the plane.

PREV < Day Day Up > NEXT