Notes on Computational Geometry
Chapter 1: Intro

Andy Pavlosky

Computational geometry = study of algorithms and data structures for geometric objects. Has applications in computer graphics, CAD, robotics, geographic information systems.

1 Convex Hulls

Definition 1. A subset S of the plane is called **convex** if for all points $p, q \in S$, the line segment $\overline{pq}$ is contained entirely in S.

Definition 2. The **convex hull** $\mathcal{CH}(S)$ of S is the smallest convex set containing S. Specifically, it is the intersection of all convex sets containing S.

- Intuitively, if we have a set of points P in the plane, we can get its convex hull by stretching a rubber band around all of the points, then letting go. So the convex hull is the (unique) polygon with points from P that contains all points in P.
- We can begin writing an algorithm to compute the convex hull of such a set. Let's represent a polygon by listing its vertices in clockwise order. Observe that given two points p, q of P, the segment $\overline{pq}$ is an edge of $\mathcal{CH}(S)$ iff all other points of P lie to its right. This leads to the following algorithm:

input : A set P of points in the plane.
output: A list $\mathcal{L}$ of the vertices of $\mathcal{CH}(S)$ in clockwise order.

$\mathsf{E} \leftarrow \emptyset$.
for all ordered pairs $(p, q) \in P \times P$ with $p \neq q$ **do**
    $\mathsf{valid} \leftarrow$ **true**
    for all points $r \in P$ not equal to p or q **do**
        if r is to the left of the directed line from p to q **then**
            $\mathsf{valid} \leftarrow$ **false**.
        end
    end
    if valid **then**
        Add the directed edge $\overrightarrow{pq}$ to E.
    end
end
Construct $\mathcal{L}$ by sorting the elements of E in clockwise order.
return $\mathcal{L}$

Algorithm 1: SlowConvexHull(P)

- This algorithm has some issues, though. First is that it doesn't handle three points on one line. To fix this, we should require that r be strictly to the right of $\overrightarrow{pq}$.
- Second, floating point computations can cause inaccuracies in the computation of the convex hull. For this algorithm, it may cause the result not to be a closed polygon at all!
- Lastly, it is slow. We check $n^2 - n$ pairs of points, and $n - 2$ points for each of those, making it run in $O(n^3)$ time.

- The following algorithm rectifies many of the issues in the previous one.
- It works by sorting the points by x-coordinate (then by y-coordinate, if an x-coordinate is repeated), then adding the points in this order and updating the solution after each addition.

input : A set P of points in the plane.
output: A list $\mathcal{L}$ of the vertices of $\mathcal{CH}(S)$ in clockwise order.

Sort P by x- (then y-) coordinate, giving us a sequence $p_1, \ldots, p_n$.
Add p_1 then p_2 to $\mathcal{L}_{upper}$.
for i $\leftarrow 3$ ***to*** n **do**
    Append p_i to $\mathcal{L}_{upper}$.
    while $\mathcal{L}_{upper}$ contains more than two points and the last three points in $\mathcal{L}_{upper}$ do not make a right turn **do**
        Delete the middle of the last three points in $\mathcal{L}_{upper}$.
    end
end
Similarly compute the lower hull $\mathcal{L}_{lower}$.
Delete the first and last points in $\mathcal{L}_{lower}$ to avoid duplication.
Append $\mathcal{L}_{lower}$ to $\mathcal{L}_{upper}$.
return $\mathcal{L}$

Algorithm 2: ConvexHull(P)

- This rectifies most of the problems with the last algorithm.
- Floating-point arithmetic may still cause errors, but not as badly as the last algorithm, which could have output something that wasn't even a closed polygon.
- The runtime for this is also dominated by the sorting, which can be done in $O(n \log n)$ time.

2 Algorithm Strategies

Development of a geometric algorithm often goes through three phases:

- First, develop an algorithm ignoring degenerate cases that clutter our understanding of the problem.
- Second, adjust the algorithm to deal with degenerate cases, ideally by handling special cases with the general case.
- Third, actual implementation, where we must implement or find libraries to handle primitive operations like testing whether a point is to the left or right of a line, as well as consider the consequences of floating-point arithmetic.