CLOSEST-PAIR AND CONVEX-HULLPROBLEMS

We consider a straight forward approach (Brute Force) to two well-known problems dealing with a finite set of points in the plane. These problems are very useful in important applied areas like computational geometry and operations research.

Closest-Pair Problem

The closest-pair problem finds the two closest points in a set of n points. It is the simplest of a variety of problems in computational geometry that deals with proximity of points in the plane or higher-dimensional spaces.

Consider the two-dimensional case of the closest-pair problem. The points are specified in a standard fashion by their (x, y) Cartesian coordinates and that the distance between two points pi(x_i, y_i) and pj(x_j, y_j) is the standard Euclidean distance.

$$d(e_i, e_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

The following algorithm computes the distance between each pair of distinct points and finds a pair with the smallest distance.

ALGORITHM:

Brute Force Closest Pair(P)

//Finds distance between two closest points in the plane by brute force

//Input: A list *P* of *n* $(n \geq 2)$ points $p1(x1, y1), \ldots, pn(xn, yn)$

//Output: The distance between the closest pair of points

$d \leftarrow \infty$

for $i \leftarrow 1$ **to** $n - 1$ **do**

  for $j \leftarrow i + 1$ **to** n **do**

    $d \leftarrow \min(d, sqrt((xi - xj)^2 + (yi - yj)^2))$ //*sqrt* is square root

return *d*

The basic operation of the algorithm will be squaring a number. The number of times it will be executed can be computed as follows:

$$(n)=\sum_{i=1}^{n-1}\sum_{j=(i+1)}^{n} 2$$

$$=2\sum_{i=1}^{n-1}(n-i)$$

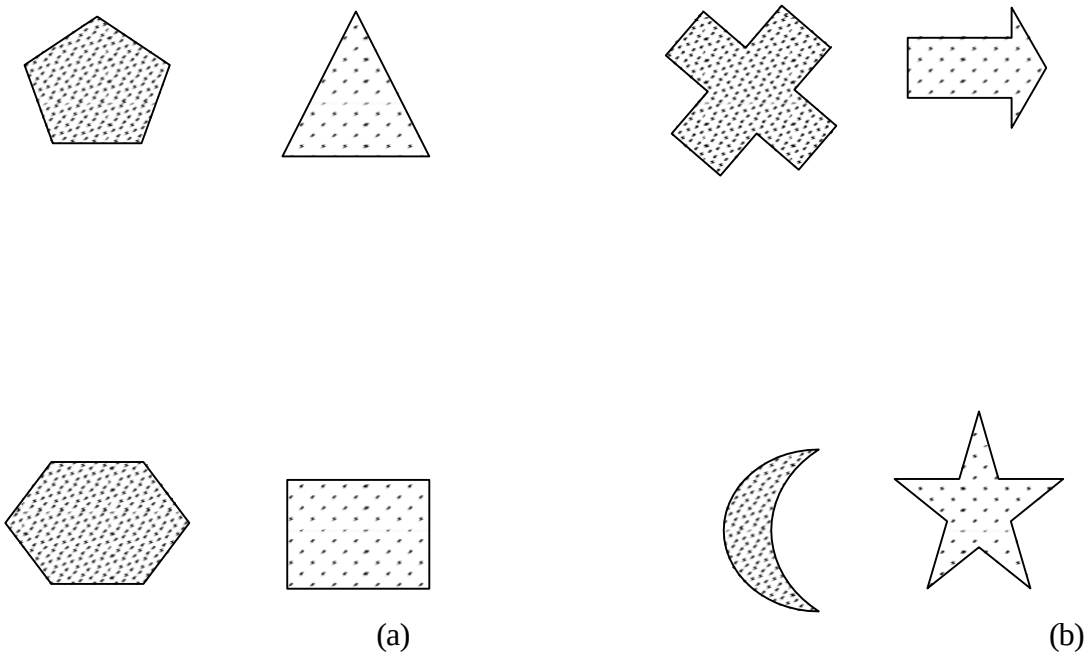
$$= 2[(n-1)+(n-2)+\ldots+1]$$

$$= (n-1)n\in \Theta(n^2).$$

Of course, speeding up the innermost loop of the algorithm could only decrease the algorithm's running time by a constant factor, but it cannot improve its asymptotic efficiency class.

Convex-Hull Problem Convex Set

A set of points (finite or infinite) in the plane is called ***convex*** if for any two points *p* and *q* in the set, the entire line segment with the endpoints at *p* and *q* belongs to the set.



(a) (b)

FIGURE 2.1 (a) Convex sets. (b) Sets that are not convex.

All the sets depicted in Figure 2.1 (a) are convex, and so are a straight line, a triangle, a rectangle, and, more generally, any convex polygon, a circle, and the entire plane.

On the other hand, the sets depicted in Figure 2.1 (b), any finite set of two or more

distinct points, the boundary of any convex polygon, and a circumference are examples of sets that are not convex.

Take a rubber band and stretch it to include all the nails, then let it snap into place. The convex hull is the area bounded by the snapped rubber band as shown in Figure 2.2

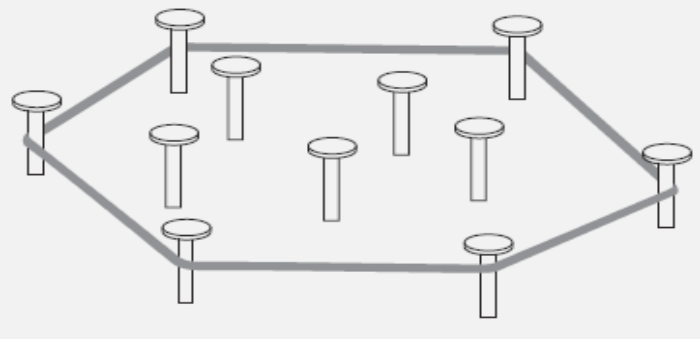


FIGURE 2.2 Rubber-band interpretation of the convex hull.

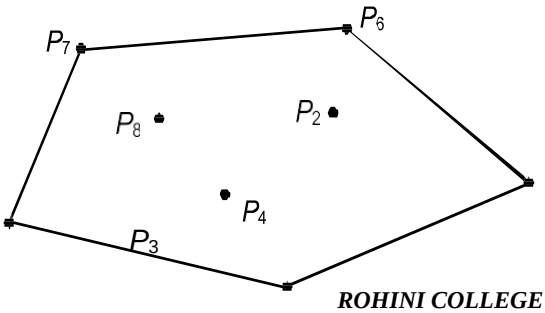
Convex hull

The ***convex hull*** of a set *S* of points is the smallest convex set containing *S*. (The smallest convex hull of *S* must be a subset of any convex set containing *S*.)

If *S* is convex, its convex hull is obviously *S* itself. If *S* is a set of two points, its convex hull is the line segment connecting these points. If *S* is a set of three points not on the same line, its convex hull is the triangle with the vertices at the three points given; if the three points do lie on the same line, the convex hull is the line segment with its endpoints at the two points that are farthest apart. For an example of the convex hull for a larger set, see Figure 2.3.

THEOREM:

The convex hull of any set *S* of $n>2$ points not all on the same line is a convex polygon with the vertices at some of the points of *S*. (If all the points do lie on the same line, the polygon degenerates to a line segment but still with the endpoints at two points of *S*.)



P5

P1

FIGURE 2.3 The convex hull for this set of eight points is the convex polygon with vertices at p1, p5, p6, p7, and p3.

The ***convex-hull problem*** is the problem of constructing the convex hull for a given set *S* of *n* points. To solve it, we need to find the points that will serve as the vertices of the polygon in question. Mathematicians call the vertices of such a polygon "extreme points." By definition, an ***extreme point*** of a convex set is a point of this set that is *not a middle point of any line segment with endpoints in the set*. For example, the extreme points of a triangle are its three vertices, the extreme points of a circle are all the points of its circumference, and the extreme points of the convex hull of the set of eight points in Figure 2.3 are p1, p5, p6, p7, andp3.

Application

Extreme points have several special properties other points of a convex set do not have. One of them is exploited by the ***simplex method***, this algorithm solves ***linear programming*** **Problems.**

We are interested in extreme points because their identification solves the convex-hull problem. Actually, to solve this problem completely, we need to know a bit more than just which of *n* points of a given set are extreme points of the set's convex hull. we need to know which pairs of points need to be connected to form the boundary of the convex hull. Note that this issue can also be addressed by listing the extreme points in a clockwise or a counter clock wise order.

We can solve the convex-hull problem by brute-force manner. The convex hull problem is one with no obvious algorithmic solution. there is a simple but inefficient algorithm that is based on the following observation about line segments making up the boundary of a convex hull: a line segment connecting two points *pi* and *pj* of a set of *n* points is a part of the convex hull's boundary if and only if all the other points of the set lie on the same side of the straight line through these two points. Repeating this test for every pair of points yields a list of line segments that make up the convex hull's boundary.

Facts

A few elementary facts from analytical geometry are needed to implement the above algorithm.

- First, the straight line through two points *(x1, y1), (x2, y2)* in the coordinate plane can be defined by the equation $ax + by = c$, where $a = y2 - y1$, $b = x1 - x2$, $c = x1y2 - y1x2$.
- Second, such a line divides the plane into two half-planes: for all the points in one of them, $ax + by > c$, while for all the points in the other, $ax + by < c$. (For the points on the line itself, of course, $ax + by = c$.) Thus, to check whether certain points lie on the same side of the line, we can simply check whether the expression $ax + by - c$ has the same sign for each of these points.

Time efficiency of this algorithm.

Time efficiency of this algorithm is in $O(n^3)$: for each of $n(n-1)/2$ pairs of distinct points, we may need to find the sign of $ax + by - c$ for each of the other $n - 2$ points.