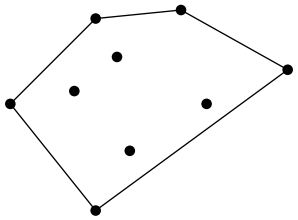
Chapter A1: Convex Hulls: An Example

A polygon is ***convex*** if any line segment joining two points on the boundary stays within the polygon. Equivalently, if you walk around the boundary of the polygon in counterclockwise direction you always take left turns.

The ***convex hull*** of a set of points in the plane is the smallest convex polygon for which each point is either on the boundary or in the interior of the polygon. One might think of the points as being nails sticking out of a wooden board: then the convex hull is the shape formed by a tight rubber band that surrounds all the nails. A ***vertex*** is a corner of a polygon. For example, the highest, lowest, leftmost and rightmost points are all vertices of the convex hull. Some other characterizations are given in the exercises.



We discuss three algorithms for finding a convex hull: Graham Scan, Jarvis March and Divide & Conquer. We present the algorithms under the ***assumption*** that:

- *no 3 points are collinear (on a straight line)*

A1.1 Graham Scan

The idea is to identify one vertex of the convex hull and sort the other points as viewed from that vertex. Then the points are scanned in order.

Let x_0 be the leftmost point (which is guaranteed to be in the convex hull) and number the remaining points by angle from x_0 going counterclockwise: $x_1, x_2, \ldots, x_{n-1}$. Let $x_n = x_0$, the chosen point. (We're assuming no two points have the same angle from x_0.)

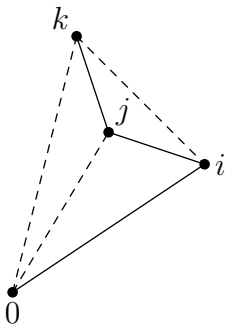
The algorithm is simple to state with a single stack:

Graham Scan

1. Sort points by angle from x_0
2. Push x_0 and x_1. Set i=2
3. While $i \leq n$ do:
   If x_i makes left turn w.r.t. top 2 items on stack
   then { push x_i; i++ }
   else { pop and discard }

To prove that the algorithm works, it suffices to argue that:

- *A discarded point is not in the convex hull.* If x_j is discarded, then for some $i < j < k$ the points $x_i \to x_j \to x_k$ form a right turn. So, x_j is inside the triangle x_0, x_i, x_k and hence is not on the convex hull.



- *What remains is convex.* This is immediate as every turn is a left turn.

The running time: Each time the while loop is executed, a point is either stacked or discarded. Since a point is looked at only once, the loop is executed at most $2n$ times. There is a constant-time subroutine for checking, given three points in order, whether the angle is a left or a right turn (Exercise). This gives an $O(n)$ time algorithm, apart from the initial sort which takes time $O(n \log n)$. (Recall that the notation $O(f(n))$, pronounced "order $f(n)$", means "asymptotically at most a constant times $f(n)$".)

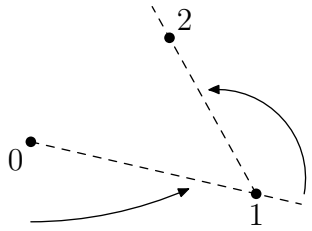
A1.2 Jarvis March

This is also called the ***wrapping algorithm***. This algorithm finds the points on the convex hull ***in the order*** in which they appear. It is quick if there are only a few points on the convex hull, but slow if there are many.

Let x_0 be the leftmost point. Let x_1 be the first point counterclockwise when viewed from x_0. Then x_2 is the first point counterclockwise when viewed from x_1, and so on.

Jarvis March

$i = 0$
while not done do
    x_{i+1} = first point counterclockwise from x_i



Finding x_{i+1} takes linear time. The while loop is executed at most n times. More specifically, the while loop is executed h times where h is the number of vertices on the convex hull. So Jarvis March takes time $O(nh)$.

The best case is $h = 3$. The worst case is $h = n$, when the points are, for example, arranged on the circumference of a circle.

A1.3 Divide and Conquer

Divide and Conquer is a popular technique for algorithm design. We use it here to find the convex hull. The first step is a Divide step, the second step is a Conquer step, and the third step is a Combine step.

The idea is to:

Divide and conquer

1. Divide the n points into two halves.
2. Find convex hull of each subset.
3. Combine the two hulls into overall convex hull.

Part 2 is simply two ***recursive*** calls. Note that, if a point is in the overall convex hull, then it is in the convex hull of any subset of points that contain it. (Use characterization in exercise.) So the task is: given two convex hulls, find the convex hull of their union.

◊ *Combining two hulls*

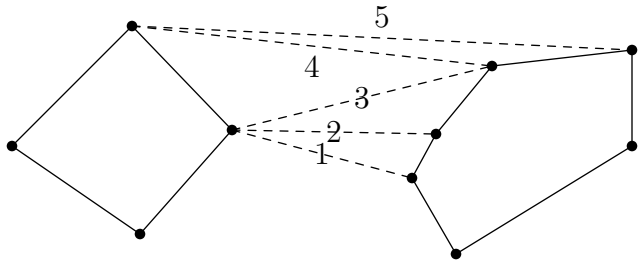
It helps to work with convex hulls that do not overlap. To ensure this, all the points are ***presorted*** from left to right. So we have a left and right half, and hence a left and right convex hull.

Define a ***bridge*** as any line segment joining a vertex on the left and a vertex on the right that does not cross the side of either polygon. What we need are the ***upper*** and ***lower*** bridges. The following produces the upper bridge.

1. Start with any bridge. For example, a bridge is guaranteed if you join the rightmost vertex on the left to the leftmost vertex on the right.

2. Keeping the left end of the bridge fixed, see if the right end can be raised. That is, look at the next vertex on the right polygon going clockwise, and see whether that would be a (better) bridge. Otherwise, see if the left end can be raised while the right end remains fixed.

3. If made no progress in Step 2 (cannot raise either side), then stop else repeat Step 2.



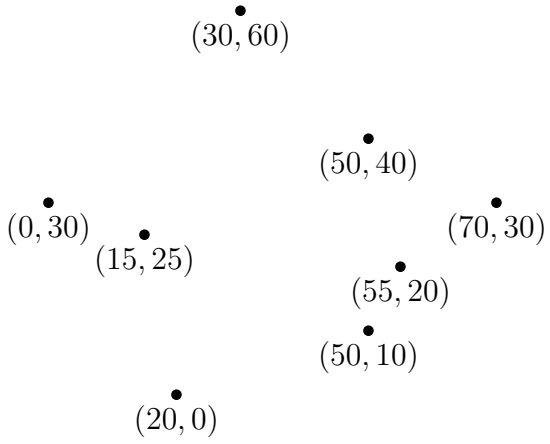
We need to be sure that one will eventually stop. Is this obvious?

Now, we need to determine the running time of the algorithm. The key is to perform Step 2 in constant time. For this it is sufficient that each vertex has a pointer to the next vertex going clockwise and going counterclockwise. Hence the choice of data structure: we store each hull using a ***doubly linked circular linked list***.

It follows that the total work done in a merge is proportional to the number of vertices. And as we shall see from a later chapter, this means that the overall algorithm takes time $O(n \log n)$.

Exercises

1. Find the convex hulls for the following list of points using the three algorithms presented.



2. Give a quick calculation which tells one whether three points make a left or a right turn.

3. Discuss how one might deal with collinear points in the algorithms.

4. Show that a point D is on the convex hull if and only if there do not exist points A, B, C such that D is inside the triangle formed by A, B, C.

5. Give a good algorithm for the ***convex layers*** problem. The convex layers are the convex hull, the convex hull of what remains, etc.

6. ⓔ Assume one has a fast convex hull subroutine that returns the convex hull ***in order***. Show how to use the subroutine to sort numbers.