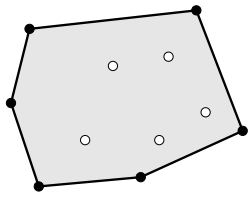
14 Convex Hulls (March 20)

14.1 Definitions

We are given a set P of n points in the plane. We want to compute something called the *convex hull* of P. Intuitively, the convex hull is what you get by driving a nail into the plane at each point and then wrapping a piece of string around the nails. More formally, the convex hull is the smallest convex polygon containing the points:

- **polygon:** A region of the plane bounded by a cycle of line segments, called *edges*, joined end-to-end in a cycle. Points where two successive edges meet are called *vertices*.
- **convex:** For any two points p, q inside the polygon, the line segment $\overline{pq}$ is completely inside the polygon.
- **smallest:** Any convex proper subset of the convex hull excludes at least one point in P. This implies that every vertex of the convex hull is a point in P.

We can also define the convex hull as the *largest* convex polygon whose vertices are all points in P, or the *unique* convex polygon that contains P and whose vertices are all points in P. Notice that P might have *interior* points that are not vertices of the convex hull.



A set of points and its convex hull.
Convex hull vertices are black; interior points are white.

Just to make things concrete, we will represent the points in P by their Cartesian coordinates, in two arrays $X[1..n]$ and $Y[1..n]$. We will represent the convex hull as a circular linked list of vertices in counterclockwise order. if the ith point is a vertex of the convex hull, $next[i]$ is index of the next vertex counterclockwise and $pred[i]$ is the index of the next vertex clockwise; otherwise, $next[i] = pred[i] = 0$. It doesn't matter which vertex we choose as the 'head' of the list. The decision to list vertices counterclockwise instead of clockwise is arbitrary.

To simplify the presentation of the convex hull algorithms, I will assume that the points are in *general position*, meaning (in this context) that *no three points lie on a common line*. This is just like assuming that no two elements are equal when we talk about sorting algorithms. If we wanted to really implement these algorithms, we would have to handle colinear triples correctly, or at least consistently. This is fairly easy, but definitely not trivial.

14.2 Simple Cases

Computing the convex hull of a single point is trivial; we just return that point. Computing the convex hull of two points is also trivial.

For three points, we have two different possibilities — either the points are listed in the array in clockwise order or counterclockwise order. Suppose our three points are (a, b), (c, d), and (e, f),