## Gift Wrapping

```
Gift-Wrap(P, n)
/* P = set of n points */
/* store vertices of CHull(P) in clockwise order in array A */
1.  A[0] ← leftmost (lowest) point in P;
2.  k ← 1;
3.  repeat
4.          A[k] ← first point in clockwise order around A[k-1];
5.          (Break ties by choosing point farthest from A[k-1].);
6.  until (A[k] = A[0])
```



## First clockwise point

```
First-Clockwise(P, n)
/* P = set of n points */
/* P[0] is a vertex of CHull(P) */
/* return first point in clockwise order around P[0] */
1.  j ← 1;
2.  for i ← 1 to n-1 do
3.          if (Orient(P[j], P[0], P[i]) = Right) then
4.                  j ← i;
5.          else if (Orient(P[j], P[0], P[i]) = 0) then
            /* Points P[j], P[0], P[i] are collinear */
6.                  if (  (P[0].x ≤ P[j].x < P[i].x) or
7.                        (P[0].x] ≥ P[j].x > P[i].x) or
8.                        (P[0].y ≤ P[j].y < P[i].y) or
9.                        (P[0].y ≥ P[j].y > P[i].y)) then
10.                             j ← i;
11. return(j);
```



## Graham Scan

```
Graham-Scan-Upper(P, n)
/* P = set of n points */
/* store upper vertices of CHull(P) in clockwise order in array A */
1.  Sort points by increasing x-coordinate.
    (Break ties by increasing y-coordinate.)
    Let p0, p1, ..., pn-1 be points in sorted order.
2.  A[0] ← p0;
3.  A[1] ← p1;
4.  k ← 1;
5.  for i ← 2 to n-1 do
6.          while (k > 0 and Orient(A[k-1], A[k], pi) ≠ Right) do
7.                  k ← k–1;
8.          endwhile
9.          k ← k+1;
10.         A[k] ← pi;
11. endfor
```



## Convex Hull Algorithms

n = # input points

h = # convex hull vertices

1. Graham scan. O(n log n)
2. Gift wrapping. O(n h)
3. Quickhull. O($n^2$)
4. Divide & conquer. O(n log n)
5. Kirk. & Seidel. - Prune & search. O(n log h)
6. Randomized incremental. O(n log n) expected.