Convex Hull

Andre Kessler

December 18, 2009

1 Convex Hull

Given a collection of points in the plane, we want to find the convex polygon with smallest area such that each point is contained within (or on the boundary of) the polygon. This polygon is known as the convex hull of the set of points. If each point is a fixed peg on a board, the problem is the same as finding the polygon formed when we stretch a rubber band around that set of points. We will describe the Graham Scan algorithm below, which has complexity $O(N \log N)$. We find the average of all the points as the center point that we are certain is inside the convex hull. Then, we set this point as the "origin" and find the angles all the potential vertices make to the positive x axis. This lends itself to using the atan2 function or arg (z) if we're using complex numbers. We sort the angles to get the points in counterclockwise order. Now we traverse the points one by one. For every new point, we check the previous two points and see if the angle from $P_i P_{i-1} P_{i-2}$ is concave, using cross products. If it isn't concave, we delete point P_{i-1} and check until it is concave. After we go through all the points, we have to check points P_n , P_0 , and P_1 , to make the polygon closed and still ensure concavity. (Pseudocode below taken from the USACO training pages)

```
# x(i), y(i) is the x,y position
    #    of the i-th point
    # zcrossprod(v1,v2) -> z component
    #             of the vectors v1, v2
    #  if zcrossprod(v1,v2) < 0,
    #     then v2 is "right" of v1
    #   since we add counter-clockwise
    #     <0 ->  angle > 180 deg
1   (midx, midy) = (0, 0)
2   For all points i
3     (midx, midy) = (midx, midy) +
           (x(i)/npoints, y(i)/npoints)
4   For all points i
5     angle(i) = atan2(y(i) - midy,
                       x(i) - midx)
6     perm(i) = i

7   sort perm based on the angle() values
    # i.e., angle(perm(0)) <=
            angle(perm(i)) for all i

    # start making hull
8   hull(0) = perm(0)
9   hull(1) = perm(1)
10  hullpos = 2
11  for all points p, perm() order,
```

```
                except perm(npoints - 1)
12     while (hullpos > 1 and
           zcrossprod(hull(hullpos-2) -
13         hull(hullpos-1),
           hull(hullpos-1) - p) < 0)
14            hullpos = hullpos - 1
15        hull(hullpos) = p
16        hullpos = hullpos + 1

     # add last point
17   p = perm(npoints - 1)
18   while (hullpos > 1 and
           zcrossprod(hull(hullpos-2) -
19         hull(hullpos-1),
           hull(hullpos-1) - p) < 0)
20     hullpos = hullpos - 1

21   hullstart = 0
22   do
23     flag = false
24     if (hullpos - hullstart >= 2 and
            zcrossprod(p -
25          hull(hullpos-1),
            hull(hullstart) - p) < 0)
26       p = hull(hullpos-1)
27       hullpos = hullpos - 1
28       flag = true
29     if (hullpos - hullstart >= 2 and
           zcrossprod(hull(hullstart) - p,
           hull(hullstart+1) -
           hull(hullstart)) < 0)
30         hullstart = hullstart + 1
31         flag = true
32   while flag
33   hull(hullpos) = p
34   hullpos = hullpos + 1
// Convex hull is now stored in hull [] from hull [hullstart]
// to hull [hullpos - 1]
```

One possibly nicer algorithm to code (although the one above is pretty easy) is Andrew's algorithm. This algorithm splits the convex hull into two parts: the upper and lower hull. These usually meet at the ends, although it's possible for them to be joined by a line segment. We sort the points by x-coordinate, breaking ties by taking the largest y-coordinate. Afterwards, points are added in order of x-coordinate. This will sometimes still make the hull concave instead of convex, so we need to delete the second to last, third to last, etc, points until a convex triangle is found. At first glance, it may appear that this algorithm (excluding the sort) is $O(N^2)$. However, excluding the sort, this algorithm is actually $O(N)$. Why?

2 Line Sweep Algorithms

Farmer John's N ($1 \leq N \leq 100,000$) cows are scattered across his pasture at unique (x, y) locations, grazing happily. However, Farmer John would like to know the location of the two closest grazing cows. Can you

help him?
This is an example of a line sweep problem, where we consider a line "sweeping" across the plane, gathering the information that we need (although we can't consider all locations, we must only consider special ones).

3 Problems

1. Given a set of N horizontal or vertical line segments in the plane, report all intersection points among the segments.

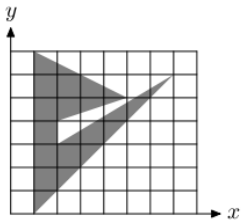
2. Given a set of N line segments in the plane, report all intersection points among the segments.

3. Farmer John's cows eat grass in rectangular areas. Given a list of these rectangular areas, compute the total area of grass that cows have eaten from in $O(N \log(N))$. Note that parts of areas where the cows eat may intersect other cows' grazing areas.

4. (IOI 98) Do the problem above, replacing "area" with "perimeter."

5. Given N ($1 <= N <= 100,000$) lines in the plane, find the x-coordinate of the leftmost intersection.

6. (ACM ICPC Nov06 NEERC) You are given lattice points in a $W \times H$ grid in an order that forms a closed polygon ($1 \leq W, H \leq 100$). Print out an ASCII representation of the polygon. Do this by printing a '.' for each grid square filled in from 0% inclusive to 25% exclusive, a '+' for 25% to 50%, an 'o' for 50% to 75%, a '$' for 75% to 100%, and a '#' for exactly 100%. For example, given points (7,6), (1,0), (1,7), (5,5), (2,4), and (2,3), in that order, we get the picture



The correct output for these points would be:

```
.$+.....
.##$+...
.#$oo+..
.#+$o...
.##o....
.#o.....
.o......
```

7. (USACO DEC08) Given N ($1 \leq N \leq 250$) fence posts in the plane, what is the largest number of posts that Farmer John can select to form a convex fence?