

COMPUTATIONAL GEOMETRY

Homework Set # 4 – Solution Notes

(1). *O'Rourke, problem 5, section 3.2.3, page 68.*

The algorithm as given in the text assumes that no three points are collinear. If there are 3 or more collinear points on the boundary of the convex hull, then it will report as “extreme” every pair of such points that is properly oriented (forming an edge that has the other points on or to the left), whereas we really want only to report the pair of points that define the endpoints of the corresponding edge of the convex hull. Thus, we want to exclude a pair (p_i, p_j) of points from being extreme if either (1) there is a point p_k strictly to the right of the oriented line $p_i p_j$ or (2) there is a point p_k on the line $p_i p_j$ that does not lie between p_i and p_j . We give pseudocode below.

Algorithm: EXTREME EDGES

```

for each  $i$  do
  for each  $j \neq i$  do
    for each  $k \neq i \neq j$  do
      if Left( $p_j, p_i, p_k$ ) or (Collinear( $p_i, p_j, p_k$ ) and NOT Between( $p_i, p_j, p_k$ ))
        then  $(p_i, p_j)$  is not extreme
  
```

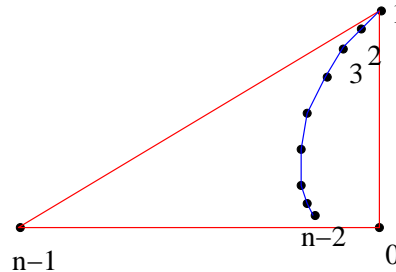
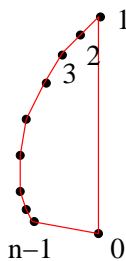
(2). *O'Rourke, problem 4, section 3.4.1, page 72.*

Consider the set of points S that consists of one point at $(1, 0)$, one at $(-1, 0)$, and $n-2$ points (p_2, p_3, \dots) on the unit circle centered at the origin, at angles $\pi/2, \pi/4, \pi/8, \dots$, respectively, with respect to the positive x -axis. These n points all lie on the unit circle. If QuickHull(a, b, S) is called with $a = (1, 0)$ and $b = (-1, 0)$, then we spend n steps to obtain that $c = p_2$, then $n-1$ steps in the call to QuickHull(a, p_2, A) to obtain that $c = p_3$, etc. The recursion is badly unbalanced, with all the remaining points always falling to the “right” of c ; this results in $n + (n-1) + (n-2) + \dots = \Theta(n^2)$ time in total.

(3). *O'Rourke, problem 2, section 3.5.7, page 86.*

In the figure on the left below, we illustrate the case in which all the points are in convex position. Then, Algorithm 3.6 will enter the **while** loop with $i = 2$ and push points 2, 3, \dots , $n-1$; no point is ever popped from the stack, since the strictly left condition always holds. Thus, the **while** loop is iterated a total of $n-2$ times.

While it was not asked of you, I also solve the next problem, *O'Rourke, problem 3, section 3.5.7, page 86*. In the figure on the right below, we illustrate the case in which the points indexed 0 through $n-2$ are in convex position, but the point indexed $n-1$ is positioned so that the convex hull of all n points is a triangle, $(0, 1, n-1)$. Then, Algorithm 3.6 will enter the **while** loop with $i = 2$ and push points 2, 3, \dots , $n-2$, each time incrementing i by 1. Once $i = n-1$, when we enter the **while** loop next, we pop $n-2$ (no change to i), then pop $n-3$ (no change to i), etc, until we pop point indexed by 2 (and $i = n-1$ still). In the final iteration of the **while** loop, we push point $n-1$, set $i = n$, and the **while** loop has ended (since $i < n$ is false). In total there were $n-3$ iterations of the **while** loop that do a push, then $n-2$ iterations that do a pop, then 1 iteration that pushes; in total, there are $2n-5$ iterations. Note that the same will happen in any case in which the convex hull has only 3 vertices ($h = 3$), even of the points that are interior to the hull are not in convex position (as in the figure below): all but 3 points will be both pushed and popped.



(4). *O'Rourke, problem 1, section 3.9.2, page 96.*

First, in order for an orthogonal polygon P to be orthogonally convex, it must have a unique vertical edge (L) that has smallest x -coordinate (otherwise, the vertical line through the edge would violate the definition of orthogonal convexity, since it would intersect the polygon in two or more segments). Similarly, there must be a unique right side edge (R), top edge (T), and bottom edge (B).

We claim that in going clockwise from L to T around P the edges of P must form a "staircase", with a rightwards edge leading out of the top endpoint of L , then an upwards edge, then a rightwards edge, then an upwards edge, etc, until the left endpoint of T is reached. (Note that this staircase of edges may be empty if the top endpoint of L equals the left endpoint of T .) To prove this, we can assume the contrary and arrive at a contradiction: If we ever follow a rightwards edge e by a downwards edge, then the horizontal line through e must intersect P in at least two segments (e and some other place, since the polygon must extend upwards to T). Similarly, if we ever follow an upwards edge e by a leftwards edge, then a vertical line just to the left of e must intersect P in at least two segments.

So the basic characterization is that the polygon P , in order to be orthogonally convex, must have unique leftmost, topmost, rightmost, bottommost edges, and these are linked together by staircase chains. (This can be made very precise.)

(5). *Let S be the set of points $\{(8,5), (8,1), (2,2), (-1,5), (2,6), (3,4), (10,4), (-3,2), (-3,4), (-3,1), (2,-2), (-1,-2), (2,0), (-2,2)\}$. When the Graham Scan code of Section 3.5 is run on this data, what is the sorting (as in Table 3.1) and what is the history of the stack (as in the example on page 86)? Plot the points and the resulting convex hull.*

By running the code, the sorting gives:

```
After sorting, ndelete = 2:
Points:
vnum= 10, x=  2, y= -2, delete=0
vnum=  1, x=  8, y=  1, delete=0
vnum=  6, x= 10, y=  4, delete=0
vnum=  0, x=  8, y=  5, delete=0
vnum=  5, x=  3, y=  4, delete=0
vnum= 12, x=  2, y=  0, delete=1
vnum=  2, x=  2, y=  2, delete=1
vnum=  4, x=  2, y=  6, delete=0
vnum=  3, x= -1, y=  5, delete=0
vnum=  8, x= -3, y=  4, delete=0
vnum= 13, x= -2, y=  2, delete=0
vnum=  7, x= -3, y=  2, delete=0
vnum=  9, x= -3, y=  1, delete=0
vnum= 11, x= -1, y= -2, delete=0
```

The stack history is:

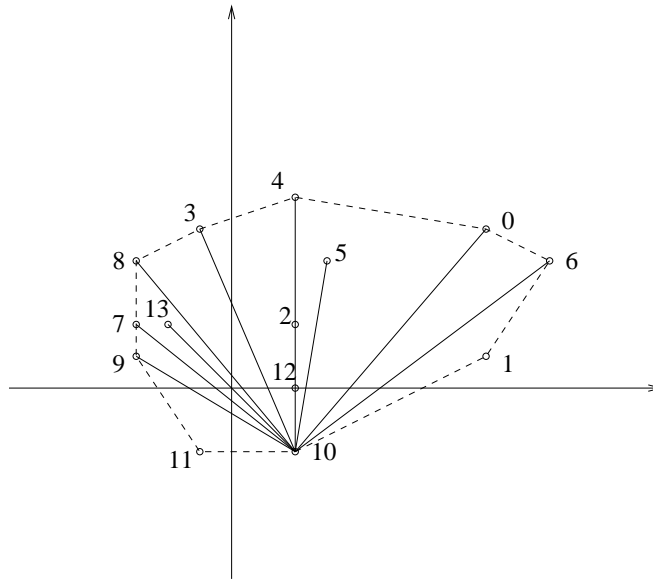
```
i=2: 1, 10
i=3: 6, 1, 10
i=4: 0, 6, 1, 10
i=5: 5, 0, 6, 1, 10
i=5: 0, 6, 1, 10
i=6: 4, 0, 6, 1, 10
i=7: 3, 4, 0, 6, 1, 10
i=8: 8, 3, 4, 0, 6, 1, 10
i=9: 13, 8, 3, 4, 0, 6, 1, 10
i=9: 8, 3, 4, 0, 6, 1, 10
i=10: 7, 8, 3, 4, 0, 6, 1, 10
i=10: 8, 3, 4, 0, 6, 1, 10
i=11: 9, 8, 3, 4, 0, 6, 1, 10
```

Finally, at the bottom of the while loop, the stack is

```
i=12: 11, 9, 8, 3, 4, 0, 6, 1, 10
```

The hull is given by:

Hull:
vnum=11 x=-1 y=-2
vnum=9 x=-3 y=1
vnum=8 x=-3 y=4
vnum=3 x=-1 y=5
vnum=4 x=2 y=6
vnum=0 x=8 y=5
vnum=6 x=10 y=4
vnum=1 x=8 y=1
vnum=10 x=2 y=-2



(6). Assume that we execute Melkman's convex hull algorithm on the vertices of P in the order $v_0, v_1, v_2, v_3, \dots$. (In the figure, I label v_i with "i".) Show the deque, indicating the "top" d_t and "bottom" d_b at the instant just after having computed the hull of the first 8 vertices (v_0-v_7) and also just after the first 9 vertices (v_0-v_8).

Using the convention of the handout on Melkman's algorithm, I give the deque as $\langle d_b, \dots, d_t \rangle$. The evolution of the deque during the algorithm is: $\langle 2, 1, 0, 2 \rangle, \langle 3, 2, 1, 0, 3 \rangle, \langle 3, 2, 1, 0, 3 \rangle, \langle 3, 2, 1, 0, 3 \rangle, \langle 3, 2, 1, 0, 3 \rangle, \langle 3, 2, 1, 0, 3 \rangle, \langle 8, 2, 1, 0, 8 \rangle, \langle 9, 8, 2, 1, 9 \rangle$. (A careful examination of the figure reveals that point v_9 lies to the right of the segment v_1v_0 .) See the figure below.

