Comp Sci 418 HW1

Philip King

1-19-24

Question 1.3

So basically the Graham scan algorithm can be used.

Step 1: Take all points from every edge in E and put it into a list P

Step 2: From S pick the point with the lowest Y value, p_0

Step 3: With respect to the selected point, p_0,sort all other points based on the polar angle from the selected point.

In the event of a tie in polar values, use the point with the highest distance from p_0

Step 4: Scan every point in the list in acending order of polar value

In the event of a left turn, add the new point onto a stack

In the event of a right turn, pop the new point and compare the next point with the point on the top of the stack.

Step 5: repeat until back to p_0, the resulting stack will be in counterclockwise order.

Runtime: All steps other than the sorting will take $O(n)$, ske all points from every edge in E and put it into a list

Note: Since the polygon is already found, after step 3 you can just add all the points to the polygon list in acending polar angle and that will be the polygon as there will not be any right turns.

Question 1.8

a. Assume p_1 and p_2 are oriented horizontally from eachother. To combine the two,

Step 1: Find the right most point of p_1, p, and the left most point of p_2, q.

Step 2: take the line $\overline{pq}$ and rotate the point q clockwise to the next point such that the clockwise neighbor of q makes a left turn to the newly roated to point on p_2

Step 3: repeat step 3 until the clockwise neighbor of q makes a right turn, q will now be at the high tangent of p_2

Step 4: rotate p up in the same manner, checking that the counterclockwise neighbor makes a right turn and finishing when it turns left

$\overline{pq}$ is now the upper tangent line of the combined polygon

Step 5: find the lower tangent line in the same way as upper just reversing the turning to go down till arriving at the lower tanget line.

Step 6: p, q will be the left most and right most point of p_1 and p_2 respectively

q will rotate down if the counterclockwise neighbor makes a right turn, and p will rotate down if the clockwise neighbor makes a left turn.

Step 6: Using the two found tangents the new combined polygon will be formed.

Runetime: All points will be seen at most once in the worst case, so $O(n)$

b.

Step 1: First take the n points and find the vertical median line on the x axis,

Step 2: The points to the left of the line will be in one group and the points to the right will be in another

Step 3: Run this division recursivly on every resulting group untill the groups size is either 2 or 3 points.

Step 4: Time for mergin of the points,
Since the points were split on the median line, the two polygons to merge will always be to the left or right of eachother.
Take the left polygon p_0 and right polygon p_1 and run the algorithm found in part a.

Step 5: merge up all polygons until the result is the final fully merged polygon.

Runtime: The division and algo will take $O(n)$ and the merging will take $O(logn)$ resulting in $O(nlogn)$

Question 1.9

To show that S can be sorted in $O(n) + O(nlogn)$ we can do these steps.

Step 1: Take every point in S and create a new list S' with the constuction of $S(x) = S'(x, x^2)$

this results in a list that if everyone point was plotted, a exponential graph would be formed

Step 2: Then run this new list S', which is still unsorted, through the CONVEXHULL algorithm.

Step 3: The resulting polygon will be a sorted list of S', all left turns as well if starting from (0,0), or the bottom points.

Step 4: To get the sorted list of S simply take the x value of S'

Runetime: Converting the list S to S' will take $O(n)$ and the runtime of CONVEXHULL is $O(nlogn)$