Jordan Allard
CSCI 264-01
Homework 2

HW2 Problem 1 Writeup

a) Verbal Description

Given an input of (x, y) pairs, this algorithm calculates the greatest number of horizontal crossings possible at any point in the polygon that is created by connecting the pairs in a clockwise direction. It does this by sorting the points by y-value from least to greatest, then moving "up" the polygon and recalculating the sum each time it crosses a vertex vertically (the only plae where the number of crossigns can change).

b) Pseudocode

```
Let P be the set of points in the polygon
For each point p in P:
      Let y = the y value of the point
      Let leftY = the y value of the point immediately counterclockwise on the polygon
      Let rightY = the y value of the point immediately clockwise on the polygon
Use MergeSort to sort P by the values of Y from least to greatest
Let greatestSum = 0
Let currentSum = 0
For every point p = P[i] in P:
      If p.leftY > p.y:
            currentSum += 1
      If p.rightY > p.y:
            currentSum += 1
      If p.leftY < p.y:
            currentSum -= 1
      If p.rightY < p.y:
            currentSum -= 1
      If currentSum > greatestSum and P[i + 1].y != p.y:
            greatestSum = currentSum
Return greatestSum
```

c) Proof of Correctness

When moving a horizontal line upwards through a polygon, the only time the number of crossings can change is when the line passes a vertex. Thus, the algorithm can account for all possible numbers of crossings by checking the number of crossings after each set of vertices with unique heights. For each point at a unique height, the algorithm only needs to check the y-values of the immediate neighboring points to determine how the passing the new vertex affects the total number of crossings. If the y-value of a neighboring vertex is greater than the current vertex, the edge extends upward and is added to the crossings as the horizontal line moves upward. If the y-value is less than that of the current vertex, the edge extends downward and is removed from the crossing as the horizontal line moves above the vertex. If the edge leads to another vertex with the same y-value, nothing needs to be counted, since the horizontal edge is not included as a crossing and the other point will have its crossings checked and added as part of the set of points with the same y-value.

Since the algorithm checks every way the total number of crossings could change at every possible chance for the crossings to change, it is guaranteed to find the maximum number of crossings.

d) Running Time Estimate

$O(n \log n)$

e) Running Time Estimate Reasoning

The first and second for loops run at most n times each and contain only steps with constant time complexity. The function with the largest time complexity is the MergeSort function, which is O(n log n). Thus, the total running time for this algorithm is O(n) + O(n log n) + O(n), which is equal to O(n log n).