CS218 - Programming Assignment 1

Saksham Rathi
22B1003

1 Problem Description

We are given n trapeziums in the form of (a_i, b_i) and (c_i, d_i). These are right angled trapeziums: for any trapezium, its base is on the x-axis, two of its edges are parallel to y-axis. The goal is to compute the total area covered by the union of these trapeziums and the total length covered on the x-axis.

2 Length Calculation

For the calculation of length covered on the x-axis, I sort the trapeziums based on the x-coordinates of their first points. Then I iterate over these trapeziums. Three cases arise, one in which the trapezium's left x-coordinate is to the right of the end point of the length covered till now, else it's right x-coordinate is to the right of the end point of length covered or the trapezium's base has already been covered. In the first case, we add difference of trapezium's right and left x-coordinates to the total length covered. In the second case, I add the difference of the trapezium's right x-coordinate and the previous end point to the length. In both of these cases, I also update the end point to be the current trapezium's right x-coordinate. In the third case, I do nothing because the trapezium's length has already been covered. Then I return the final length to the main function.

3 Area Calculation

I calculated the area of the union of the trapeziums using divide and conquer approach. The trapeziums were already sorted, so this makes our task a bit easier. I pass the vector of trapeziums to the divide function. It recursively calls the function on the first half and the second half of the vector with the base case, when the size of the vector is 1, I return the vector itself. Then I call merge function on these two merged left half and right half. The merge function is similar to that of merge sort. I maintain left and right index as iterators over the left and right half vectors which we need to merge. I call the two-trapezium-merger function on $left\_half[left\_index]$ and $right\_half[right\_index]$. I push the merged trapeziums to a new vector which stores all the modified trapeziums. When either the left index becomes greater than the size of the left half vector or the right index becomes greater than the size of the right half vector, I no longer call this two-trapezium-merger function, and push the trapeziums directly to the modified trapeziums' vector. This vector is then returned to the divide function.

4 How to merge two trapeziums?

This is the most important function of my code to merge two trapeziums and return a vector of distinct trapeziums, which no longer overlap/intersect. Once again, three cases arise:

- The two trapeziums are already separated. This means the left x-coordinate of the right trapezium is larger than the right x-coordinate of the left trapeziums. In this case, we do nothing and return them as such.

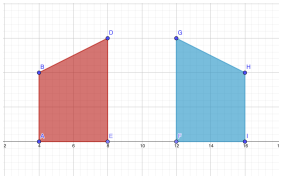

Figure 1: Case 1

- The slanting tops of the trapeziums intersect with each other as follows: As shown in the figure,

Figure 2: Case 2

  the lines i, j, k, l, m help us in dividing the union of these two trapeziums into four smaller trapeziums. First we calculate the x-coordinates and the highest y-coordinates where these lines intersect the union figure. Then we sort them (to handle different cases). And then return the vector of 4 trapeziums.

- The slanting tops of the trapeziums do not intersect with each other as follows: As shown in

Figure 3: Case 3

  the figure, the lines i, j, k, l help us in dividing the union of these two trapeziums into three smaller trapeziums. First we calculate the x-coordinates and the highest y-coordinates where these lines intersect the union figure. Then we sort them (to handle different cases). And then return the vector of 3 trapeziums.

5 References

I used the following references:

- https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/
- https://www.geeksforgeeks.org/program-for-point-of-intersection-of-two-lines/