CS218 Design and Analysis of Algorithms

Programming Assignment 1

Report

Aditya Singh - 22B1844

Programming Task:

In this Assignment we had to solve the skyline problem for trapezium posters and to compute the total area covered by the union of these posters and the total length covered on the x-axis in $\mathcal{O}(n \log n)$ time.

Approach:

Finding the Length:

For finding the length covered by the posters on the x-axis we first sort the given posters on the basis of the x-coordinate of the left vertex of the base of the trapezium. Now we traverse this array of trapeziums. In each iteration we go to the next trapezium in the array and assume that we have calculated the length covered on x-axis for the set of trapeziums before this step. In each iteration we also maintain the maximum x coordinate of the right vertex of the base among all trapeziums seen till now and we call this curr_x_max. We need to consider 3 cases in each iteration. Let the x-coordinates of the base of the trapezium in the current iteration be x_1 and x_2.

1. Case 1: $x_1 \geq$ curr_x_max. In this case we add the length covered by the new trapezium which is $x_2 - x_1$ into the current result. We also update curr_x_max to x_2.
2. Case 2: $x_1 <$ curr_x_max $< x_2$. In this case we add $x_2 -$ curr_x_max to the result. We also update curr_x_max to x_2.
3. Case 3: $x_1 < x_2 \leq$ curr_x_max. We don't need to update anything in this case.

In this manner we recursively find the length covered by the trapeziums on the x-axis.

Finding the Area:

For this part we use the **Divide and Conquer** approach. First we sort the given trapeziums as mentioned above. Our first task is to find the outline after which we can easily find the area covered. We store the outline as an array of pairs of points because we want to store the lines which make the outline and the pair of points we store are the end points of these lines.

We divide the set of trapeziums into two parts and solve for each part recursively and now we need to merge the two outlines obtained in linear time. For this we systematically traverse both the outlines by keeping a pointer for each one. We move along both the outlines incrementing the pointers and merging the outlines along the way. For each step we consider the 2 lines to which the pointers point and consider the various possibilities in which they could be arranged with respect to each other, for example they could intersect or not intersect or one line may have its right end point with greater x-coordinate than that of the other line. We make cases for each of these arrangements and separately write the code for merging in each of these cases. After covering all cases we get the correct merged outline and now we can proceed to find the area covered. For simplifying the code we also create many functions like intersection_finder, checkIntersection, y_coord_finder etc. which make the task more manageable.

Now that we have got the final outline we can easily calculate the total area covered. The array of pairs of points which we have stored as the outline can be used for this. We take all such pairs of points and treat them as trapeziums with these two points as it two vertices and the other two vertices are the projections of these points on the x-axis. For each such pair of points we add the area of the trapezium this pair of points represents, which is 0.5 $*$ (sum of y-coordinates of the points) $*$ (difference of the x-coordinates of the points).

This summation gives the total area covered by the trapezium posters as required.