The most general problem under study is to compute the area of complex polygons.

# Area of complex polygons

Use sweep line algorithm, sum the areas of trapezoids. Areas could be negative. A complex polygon means it is neither concave nor convex.

# Area of overlapping rectangles

An efficient way of computing this area is to use a sweep algorithm. Let us assume that we sweep a vertical line through the union of rectangles :

First of all, you need to build an event queue , which is, in this case, the ordered list of all x-coordinates (left and right) of the rectangles.

during the sweep, you should maintain a 1D data structure, which should give you the total length of the intersection of L(x) and U. The important thing is that this length is constant between two consecutive events q and q' of Q. So, if l(q) denotes the total length of L(q+) (i.e. L just on the rightside of q) intersected with U, the area swept by L between events q and q' is exactly l(q)\*(q' - q).

you just have to sum up all these swept areas to get the total one.

We still have to solve the 1D problem. You want a 1D structure, which computes dynamically a union of (vertical) segments. By dynamically, I mean that you sometimes add a new segment, and sometimes remove one.

I already detailed in my answer to this collapsing ranges question how to do it in a static way (which is in fact a 1D sweep). So if you want something simple, you can directly apply that (by recomputing the union for each event). If you want something more efficient, you just need to adapt it a bit:

assuming that you know the union of segments S1...Sn consists of disjoints segments D1...Dk. Adding Sn+1 is very easy, you just have to locate both ends of Sn+1 amongs the ends of D1...Dk.

assuming that you know the union of segments S1...Sn consists of disjoints segments D1...Dk, removing segment Si (assuming that Si was included in Dj) means recomputing the union of segments that Dj consisted of, except Si (using the static algorithm).

This is your dynamic algorithm. Assuming that you will use sorted sets with log-time location queries to represent D1...Dk, this is probably the most efficient non-specialized method you can get.

This is somehow related to skyline problem