Problem Title Description. Problem Name: Escape from the Mines Author's Name: Rectangles Hierarchy Problem Code: RNEST Alphabet: J

Problem: Given a set of nested rectangles, find for each rectangle, which one encloses it.

Solution: Very often, in order to get an intuition of what you have to do in two dimensions, it helps if you first visualize the equivalent problem in 1 dimension. In this case, the 1d version would ask you given a set of *intervals* (nested), find which one is nested in which. And indeed, for this a simple solution would be to just scan the intervals from left to right, and whenever you encounter a new interval, its "parent" is just the previously encountered interval. This can be easily done using a stack. (on encounter, mark parent as top and push on stack; on removal, pop from stack).

In 2 dimensions, how would this generalize? Well, lets still scan rectangles from left to right. Now, when we encounter a new rectangle, lets say we have the set of rectangles that have not been "popped" from our set. We now need to decide which of these is actually the current rectangle's parent.

In essence, we are doing a **line-sweep** of the plane. Imagine a vertical cross-section of the rectangles at any point of time. Unless you're at the vertical edge of some rectangle, this cross-section will consist of points where the (active) rectangles' top/bottom edges meet this line.

Now, given a rectangle whose vertical endpoints are (y1, y2), what should we do? We ask what is the area of the plane which is about this (y1, y2) segment. For this, pick one end-point (say y1), and find which y-value from the active set is just below it. Let that be y0. We need to know, the area between y0 and y1 belongs to which rectangle, given that y0 belongs to rectangle R'.

The answer to this is also simple. Note that if y0 is the *lower* side of R', then between y0 and y1 would belong to the area of R' itself. If y0 is the upper edge of R', then since there is no y-value between y0 and y1, this area would be in fact the area in the parent of R'.

If we compute parents of rectangles on encountering them, then we already know the parent of R' when we are computing the current rectangle's, and hence this part is O(1) once we have found R.

Algorithm:

Sort the rectangles by x-coordinate, and traverse in this sorted order.
Maintain sorted set (say, STL set) of intersection-points with vertical cross-section sweep-line, and corresponding rectangles.

When you encounter a rectangle R, (i.e. sweep-line's x = R's x1, and y-values are y1, y2)
Find y-value 'y0' just below y1, and hence rectangle R' (Include a "dummy"

rectangle that stretches from -inf, -inf to inf, inf for this to be always defined).
If y0 == R'.y1, then R.parent = R'; else R.parent = R'.parent;
Insert y1 and y2 associated with rectangle R into your cross-section set.

When you reach the end of a rectangle R, (i.e. sweep-line's x = R's x2, and y-values are y1, y2)
Remove y1 and y2 from the cross-section set.

Time complexity: O(N log N).