Summary of my project:

I use Min-heap to store the input segments by its Y value. So, my algorithm will loop over the y-coordinator instead of x-coordinator. I also create a Boolean array to store Boolean variable for (0,800). So, each time I generate the lower envelope, the Boolean variable for that part will become true to represent this part has been used. Since that, each time I'm looking for a lower envelope for a new segment. It will go to that Boolean array to check does any past of this lower envelope has been used, if so, find the available places for lower envelope.

I have three arrays of segments when I find the lower envelope. One is to store lower envelopes at the left of the lowest segment and it self. One is to store the right parts. The last one is used to combine these two arrays. So, this is array will be send into "segHL" to draw.

The whole running time will be O(nlogn)+O(3n)+O(800) which is O(nlogn).

Merge:

'total = merge(subList1, subList2);'

This is where I used my merge function. I stored subList1 and subList2 both into a min heap and let min heap sort them. After sorting, I pull them out into a ArrayList by calling extractMin(). So, this ArrayList will contain all the lower envelope by increasing order.