Assignment 2 Problem 2: Upper Envelopes of Rectangles Touching the x-axis

a) Notice that the x-coordinates of endpoints of horizontal segments must be the x-coordinates of the corners of the rectangles.
And the number of vertical segments is the number of horizontal segments plus one, except the number of horizontal segments is 0, in this case, the number of vertical segments is also 0.
Therefore, the number of horizontal segments for n rectangles should be at most 2n, and the number of vertical segments should be at most 2n+1.
Thus, totally, the number of segments should be at most 4n+1 which is less than 6n for all positive n.
For n=0, the number of segments is 0, and which is less or equal to $\mathbf{6 \times 0 = 0}$
Therefore, the upper envelope of n such rectangles contains at most 6n segments.

b) We can set two pointers each point to the points in the two upper envelopes and do something that is similar to the method of merging two arrays in merge sort. We go over two upper envelopes in the same time, if one point is behind the upper envelopes that we are building (this can be obtained since we are traversing two envelopes), then we increase the pointer of that point in the corresponding upper envelope. If we decide to put a point into the new union upper envelope, then we also increase the pointer of that point in the corresponding upper envelope.
If one pointer reaches the end of its upper envelope, then we just simply calculate the next point in the union upper envelope and then append the rest of the upper envelope in another upper envelope to the existing union upper envelope.
By doing this, after both pointers reach the end of their upper envelope. We have the upper envelope of the union of the corresponding rectangles, and the time we use is O(n+n)=O(n) since we just traverse two upper envelope in once, and each step we do a constant operation.

c) The algorithm does the following.
Provided left[i], right[i], height[i], then we consider rectangle[i] to be a structure of {left[i], right[i], height[i]}. Then the function separate rectangle[ ] into two equal size parts. And recursively call itself within each part. Then the return values of both parts are upper envelops. Then we apply the method mentioned in part b) to get the union upper envelope in O(n) and return it. And the base case for the recursion is that if only one rectangle exists, then return the coordinates of the four corners in suitable order.
Note that since the upper envelop of n such rectangles contain at most 6n segments, therefore, at most 6n+1 points is in the upper envelope with n rectangles. Which means that the points in both parts which we separated in the recursion step should be in O(6n+1)=O(n).
Therefore, we have a recurrence relation of the running time of this algorithm:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \in O(n \log n)$$