Design and Analysis of Algorithms — Lab

B.E. CSE, SSN College of Engineering

Session: CAT 4

11 June 2020

We are given a set of rectangles, possibly overlapping, drawn in the xy-plane, with their bottom sides placed on the x-axis, as shown in Figure 1. Our objective is to construct an outline of the rectangles, as shown in Figure 2.

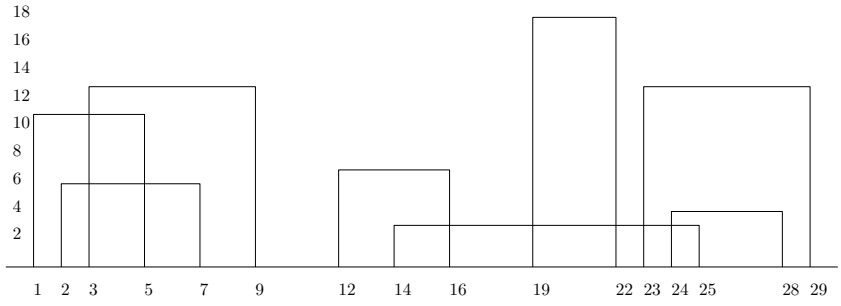

Figure 1: Set of (overlapping) rectangles

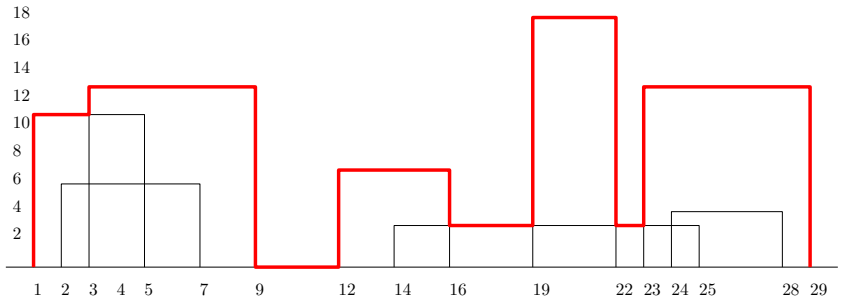

Figure 2: Set of rectangles and their outline

Represent a rectangle by its bottom-left and top-right corners (x_1, y_1) and (x_2, y_2). In your program, represent a corner by a tuple of two integers, a rectangle by a list two corners. Then the set of rectangles shown in Figure 1 will be represented as a list of rectangles

```
[
 [(1,0),(5,11)],[(2,0),(7,6)],[(3,0),(9,13)],[(12,0),(16,7)],
```

```
 [(14,0),(25,3)],[(19,0),(22,18)],[(23,0),(29,13)],[(24,0),(28,4)]
]
```

and the outline of this set of rectangles shown in Figure 2 will be represented as a list of corners

```
[(1, 0),(3, 11),(9, 13),(12, 0),(16, 7),(19, 3),(22, 18),(23, 3),(29, 13)]
```

Note that in an outline, the corners are arranged from left to right.

1. An outline is list of corners arranged from left to right. You will be developing three versions of a function `merge_outlines(u, v)` that takes two outlines `u` and `v` and combines them into a single outline.

   ```
   >>> u = [(1, 0), (3, 11), (9, 13), (12, 0), (16, 7)]
   >>> v = [(14, 0), (19, 3), (22, 18), (23, 3), (29, 13)]
   >>> merge_outlines(u,v)
   [(1, 0),(3, 11),(9, 13),(12, 0),(16, 7),(19, 3),(22, 18),(23, 3),(29, 13)]
   ```

   (a) First, implement `merge1_outlines(u, v)` to combine `u` and `v` from left to right.

   (b) Then create `merge2_outlines(u, v)` so that each corner in the combined outline has the right height. Suppose `u[0]` is to the left of `v[0]`. Then, height of `u[0]` is the maximum of the heights of `u[0]` and `v[0]`.

   (c) If two consecutive corners have the same height, the first corner is redundant. Write an improved version `merge3_outlines(u, v)` so that there are no redundant corners in the combined outline.

2. Implement a divide-and-conquer algorithm `construct_outline(rs)` that takes `rs`, a list of rectangles, and returns an outline of `rs`. Divide `rs` into almost two equal parts, construct outline for each part, and then combine the two outlines into the outline of `rs` using `merge3_outlines(u, v)`.