

2001 IC/CAD Contest

Problem 1: Rectangle Merging

Source: Avant!, Taiwan.

December 20, 2000

I. Introduction and Problem Definition

Mask geometries are often represented using a list of polygons. Each polygon can be represented by a sequence of its vertices, a collection of its edges, or a collection of trapezoids. Here, we consider the trapezoid-based representation. A horizontal trapezoidalization of a polygon is obtained by drawing a horizontal line through every vertex of the polygon. The detailed definition can be found in [1]. A trapezoid has two horizontal edges. A trapezoid might be a rectangle or a triangle. In a real mask layout, most of the trapezoids are rectangles. In order to avoid rounding differences in computation and to concentrate on the core algorithm design, *we consider rectangles only*. Suppose that we have a large amount of rectangles. The first task of your program is to group them into disjoint polygons, where each polygon consists of connected rectangles from the input. Two rectangles are *connected* if they are overlapping or edge-touching. The case of touching at only one point is not considered as connected. After grouping rectangles into different connected polygons, the second task of your program is to clean up the overlapping among the rectangles, and then to obtain a horizontal trapezoidalization for each polygon.

II. Input/Output Specification

Input is a sequence of integers. The first line of the input is the number 1 or 2. If the number is 1, only the output of the 1st task is needed. Otherwise, both of the 1st and the 2nd tasks are necessary for the output. Each of the remaining lines represents a rectangle. A rectangle is defined by two points, the lower-leftmost and the upper-rightmost points. Each point is defined by its X and Y coordinates. Each X or Y coordinate is a 32-bit signed integer. Consider the input example below. The first line is the number 2. Hence, we need to perform both of the 1st and the 2nd tasks. There are four rectangles in the example. The first two rectangles are connected, and so are the second and the third rectangles. Therefore, the first three rectangles form a polygon. The last rectangle, however, is disjoint from the polygon formed by the first three rectangles.

Input format	Sample Input
<The number 1 or 2>	2
$X_1^l Y_1^l X_1^u Y_1^u$	-20 -20 20 40
$X_2^l Y_2^l X_2^u Y_2^u$	0 -10 40 20
... ..	40 -10 60 20
$X_n^l Y_n^l X_n^u Y_n^u$	30 30 60 60

Output is a list of numbers. The output of the 1st task is just the number of connected polygons of the input. For our previous example, it is 2 since the first three rectangle forms a polygon and the last rectangle itself gives another polygon. For the output of the 2nd task, from the second line, output

connected polygons one by one. For each polygon, first print out the number of rectangles in it and then print out each rectangle per line. The listing order of rectangles does not matter. Below give the output format and a sample output for the aforementioned sample input:

Output Format	Sample Output
<Number of polygons>	2
<Number of rectangles in the 1 st polygon>	3
$X_{1,1}^l Y_{1,1}^l X_{1,1}^u Y_{1,1}^u$	-20 -10 60 20
.....	-20 -20 20 -10
<Number of rectangles in the 2 nd polygon>	-20 20 20 40
$X_{2,1}^l Y_{2,1}^l X_{2,1}^u Y_{2,1}^u$	1
.....	30 30 60 60
.....	
<Number of rectangles in the n^{th} polygon>	
$X_{m,1}^l Y_{m,1}^l X_{m,1}^u Y_{m,1}^u$	
.....	

III. Language/Platform

1. Language: C or C++.
2. Platform: SUN OS/Solaris.

IV. Evaluation

The score will be given based on correctness, time efficiency and memory requirement. When given a very large input, how can you verify if your answer is the same as the standard one for the 2nd task? Extra bonus will be given if you provide such a method and a useful visualization tool to view the input and output when developing and debugging your program.

VII. Questions

Please report any question regarding this problem to cad@cis.nctu.edu.tw with the email subject "CAD Contest: Problem 1." Your question(s) will be answered in two weeks, and the Q&A's will be posted at the contest web site

References

- [1] Joseph O'Rourke. *Computational Geometry in C*. 2nd ed., Cambridge University Press, Cambridge, UK, 1998.
- [2] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf. *Computational Geometry, Algorithm and Applications*, 2nd ed., Springer-Verlag, Berlin, 1998.
- [3] F. P. Preparata and M. I. Shamos. *Computational Geometry, An Introduction*. Springer-Verlag, NY, 1985.
- [4] S. S. Skiena. *Algorithm Design Manual*. Springer-Verlag, NY, 1998.