THE RECTANGLE INTERSECTION PROBLEM REVISITED

H.-W. SIX and D. WOOD

Abstract.
We take another look at the problem of intersecting rectangles with parallel sides. For this we derive a one-pass, time optimal algorithm which is easy to program, generalizes to \( d \) dimensions well, and illustrates a basic duality in its data structures and approach.

1. Introduction.
In [6] Bentley and Wood develop an algorithm to solve the rectangle intersection problem, namely: report all overlapping (intersecting) pairs of rectangles in a given set of \( n \) rectangles with parallel sides. The aim of the present paper is to re-examine this problem from the point of view of algorithm design. This re-examination leads to a single pass algorithm, which is a substantial descriptive and pedagogic improvement over the approach used in [6]. This unified and self-contained algorithm can easily be implemented in a language such as Pascal.

The complete algorithm, which is given in Section 2, uses the powerful scanning technique of Shamos and Hoey [11], used also in [5] and [6] for example. This technique enables the original problem to be reduced to two dual sub-problems, namely the point query and the interval query problems. These are discussed in Sections 3 and 4 and two dual "semi-dynamic" tree structures are used to solve them, the interval tree, see [2] or [6], and the range tree, see [4], respectively.

Motivations for studying this problem are discussed in [6].

2. Rectangle intersection problem.
The rectangle intersection problem is defined as: Given \( n \) named rectangles with parallel sides, report all pairs of rectangles which have at least one point in common.

We will assume that each named rectangle is given as a quadruple \( R = (x_a, x_b, y_a, y_b) \) where \( (x_a, y_a), (x_b, y_a), (x_a, y_b), (x_b, y_b) \) are the bottom-left, top-left,
bottom-right and top-right corner points of the rectangle, respectively.

Now two rectangles $R$ and $\bar{R} = (\bar{x}_q, \bar{x}_p, \bar{y}_q, \bar{y}_p)$ intersect (or overlap) iff

1. $[\bar{x}_q, \bar{x}_p]$ and $[x_q, x_p]$ overlap, and
2. $[\bar{y}_q, \bar{y}_p]$ and $[y_q, y_p]$ overlap,

where, for example $[x_q, x_p]$ denotes the closed interval given by the projection of $R$ onto the $x$-axis.

Our approach to a solution to the rectangle intersection problem is based on the scanning approach, see [11]. In Figure 1 the scanning line $SL$ is moving from left to right through the rectangles and at each instant of time it divides the rectangles into three disjoint subsets: the active rectangles which are currently cut by $SL$, the sleeping ones which will be cut, and the dead ones which have been cut. Observe that these sets only change when $SL$ passes through the left or right end of a rectangle.

Thus when the scanning line meets the left end of a rectangle $R = (x_q, x_p, y_q, y_p)$, that is at $x_q$, then $[x_q, x_p]$ overlaps $[\bar{x}_q, \bar{x}_p]$, for all active rectangles $\bar{R} = (\bar{x}_q, \bar{x}_p, \bar{y}_q, \bar{y}_p)$.

In other words, $R$ intersects an active rectangle $\bar{R}$ iff (2) is satisfied, and moreover two rectangles intersect iff at some point in the scanning process one of them is active when the left end of the other is met and (2) is satisfied at this time.

Therefore in the top-down design of an algorithm we have already progressed considerably. It only remains to find a data-structure which will hold the active rectangles (in fact, their $y$-projections $[\bar{y}_q, \bar{y}_p]$), allow for efficient insertion and