# Programming Assignment Geometric Algorithms
# Q2, 2024/2025

Mark de Berg

## 1 Problem description

Let $P$ be a set of $n$ points in the plane and let $R$ be a set of $m$ axis-aligned rectangles. The goal of this assignment is to develop and implement a plane-sweep algorithm that counts the number of pairs $(p, r) \in P \times R$ such that $p \in r$. The boundary $\partial r$ of a rectangle $r \in R$ is considered to be part of the rectangle—in other words, the rectangles are closed sets—so if point $p$ lies on $\partial r$ then $p \in r$.

The input file describing a problem instance has the following structure.

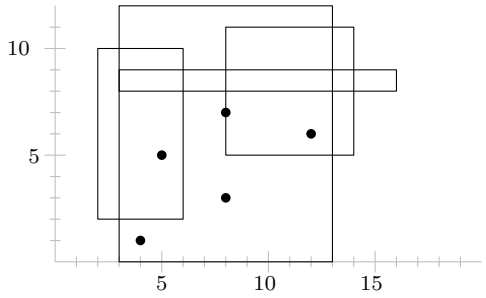
`points:` $n$
$x_1\ y_1$
$\vdots$
$x_n\ y_n$
`rectangles:` $m$
$a_1\ a'_1\ b_1\ b'_1$
$\vdots$
$a_m\ a'_m\ b_m\ b'_m$

The first line in the input contains the text `points:` and an integer $n \geq 1$ denoting the number of points in the input. This is followed by $n$ lines, each specifying a point $(x_i, y_i)$, where $x_i, y_i$ are integers in the range $[0, 2^{32} - 1]$. After all points are specified, the rectangles are specified in a similar way: there is a line containing the text `rectangles:` and an integer $m \geq 1$, followed by $m$ lines, each specifying a rectangle $[a_i, a'_i] \times [b_i, b'_i]$, where $a_i, a'_i, b_i, b'_i$ are integers in the range $[0, 2^{32} - 1]$ such that $a_i \leq a'_i$ and $b_i \leq b'_i$. Note that the rectangles can have zero width and/or zero height (namely when $a_i = a'_i$ and/or $b_i = b'_i$). Also not that the sets are multisets, so the same point or rectangle can appear multiple times. Figure 1 shows an example of an input file.

After reading the input, you algorithm must compute the number of pairs $(p, r) \in P \times R$ such that $p \in r$. The resulting count is the output of your program.

## 2 The assignment

The goal of this assignment is to develop, implement, and test a plane-sweep algorithm for the problem described above. You can do the implementation in Java, C++, or Python. Your program should read an input file that describes a problem instance, as described above, from standard input (System.in) and write the output to standard output (System.out). You should submit your program via Canvas. The way this is done is (hopefully) self-explanatory, but you should make sure your program is *a single file*. **To obtain en efficient plane-sweep algorithm, you can use a segment tree as a status structure; see Section 10.3 of the book [BKOS] used in the course.**




```
points: 5
5 5
4 1
12 6
8 7
8 3
rectangles: 4
2 6 2 10
3 13 0 12
3 16 8 9
8 14 5 11
```

Figure 1: A set of five points and four rectangles and the corresponding input file. The correct output for this input is 8.

- Assignment 1: Implement a brute-force algorithm that computes the correct answer by simply checking for each pair $(p, r) \in P \times R$ if $p \in r$. You must submit your program via Momotor; see Canvas for the deadline. This assignment is not graded, but it helps you to get started and get familiar with submitting via Canvas and seeing the results in Momotor. Moreover, you will need this implementation for the experimental study (Assignment 3).

- Assignment 2: Implement a plane-sweep algorithm algorithm that computes the correct count. Let $k$ be the correct answer for the instance, that is, $k = |\{(p, r) \in P \times R : p \in r\}|$. Note that $k$ can be as high as $nm$ Ideally your algorithm runs in $O((n + m) \log(n + m))$ time, independent of $k$.

  Your program will be evaluated by running it on a set of 10 test instances and the grade is simply the number of test instances for which your program gives a correct answer within the given time limit. (This grading scheme assumes that your your program indeed implements a plane-sweep algorithm; if not, the grade is upper bounded by 3.) The time limit will depend on the programming language used. Hence, the fact that C++ is faster than Python, for example, will not give C++ users an advantage. The time limit will be set such that most test cases can be solved within the allocated time even with an implementation running in $O((n + m) \log(n + m) + k)$ time.

- Assignment 3: Write a short report about your algorithm, including an experimental evaluation of it. This report should be at most four pages, and contain the following:

  - Section 1: Description of the plane-sweep algorithm, plus an analysis of its running time as a function of $n$ and $m$, and (if applicable) $k$. A proof of correctness is not needed.
  - Section 2: An experimental evaluation of the algorithm. This experimental evaluation should contain a comparison of the running time of your plane-sweep algorithm with the running time of your brute-force implementation. Try to come up with different types of input (inputs where the plane-sweep algorithm is clearly faster, and inputs where this is not the case) and include graphs that show the running times of your algorithms (for the various scenarios) as a function of $n$.
  - Try to come up with another algorithm, besides the brute-force algorithm and the plane-sweep algorithm running in $O((n + m) \log(n + m))$ time. If you have such an algorithm, describe and analyze it in Section 1 and include it in your experimental evaluation in Section 2.

Assignments 1 and 2 are individual, while Assignment 3 is done in pairs. The experiments and report for Assignment 3 can be based on the implementation of one of the two team members.