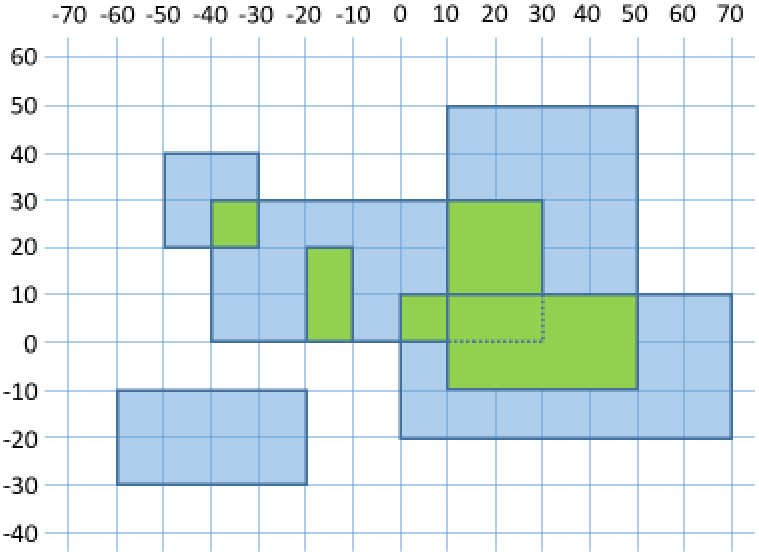
Homework: Problem Solving

This document defines the **homework assignments** for the "Algortihms" course @ Software University. Please submit a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems.

Problem 2. Rectangle Intersection

You are given **N rectangles** in the plane. The rectangles are parallel to the coordinate axes and each is defined by its coordinates: **{minX, maxX, minY, maxY}**. Write a program to find the **total area** of all areas that belong to more than one of the initial rectangles. All coordinates are integers in the **range [-1000, 1000]**. Example:



We have 6 rectangles. Their intersection areas are shown in green. The intersection area is 1600.

On the first line you'll receive the **number of rectangles N**. On the next N lines, you'll receive the coordinates of each rectangle in format **{minX} {maxX} {minY} {maxY}.** On the only output line, print the total area belonging to more than one rectangle. You can test your solution in the Judge system here.

Examples:

Input	Output

6
-60 -20 -30 -10
-50 -30 20 40
-40 30 0 30
10 50 -10 50
0 70 -20 10
-20 -10 0 20
	1600

3
40 80 -40 0
20 60 -20 30
50 100 -10 20
	800

9
-851 88 546 659
990 999 608 998
815 835 -517 734
157 623 994 996
	216777

947 956 529 925
561 688 -241 434
-966 530 -825 273
396 780 -705 590
110 202 713 891	

Hints

- **Solution #1 (slow)**
  - Create a **matrix of size 2001 x 2001**.
  - **Paint** all rectangles in the matrix.
  - **Count** the painted cells.
- *** Solution #2 (faster)**
  - Extract all **X coordinates `x[]`** from all rectangles (**`minX`** and **`maxX`**) and **sort them** in increasingly.
  - For each two coordinates **`x[i]`** and **`x[i+1]`** find all rectangles **`rects[]`** that overlap with this interval, sorted by **`minY`**. To implement this efficiently, first pre-calculate the list of rectangles for each interval **`x[i]`** … **`x[i+1]`** by a single scan through the initial list of rectangles.
  - Extract all **Y coordinates `y[]`** from all rectangles **`rect[]`** (**`minY`** and **`maxY`**) and **sort them** in increasing order.
  - For each two coordinates **`y[i]`** and **`y[i+1]`** find how many rectangles overlap with this interval, **calculate the area** where **`rect_count` ≥ 2** and **sum it**. To implement this efficiently, first pre-calculate the number of overlapping rectangles for each interval **`y[i]`** … **`y[i+1]`** by a single scan through **`rect[]`**.
- ***** Solution #3 (fastest)**
  - Implement a solution based on **interval trees** as described in http://www.oi.edu.pl/static/attachment/20110713/boi-2001.pdf (see problem "**Mars Maps**")