

CS1020E: DATA STRUCTURES AND ALGORITHMS I

Lab 2 – Rectangles

(Week 4, starting 29 August 2016)

Readme

You are given the Cartesian coordinates of $2 \leq N \leq 500,000$ rectangles. Create a **Rectangle class** with appropriate data and functionality, and use it to help you achieve 2 to 4 sub-tasks:

1. Find the region that is the intersection of all the rectangles 40%
2. Given a point **p**, find the rectangle whose center is nearest to **p** 40%
- Warning: No OOP** -40%
3. *Efficiently* combine all adjacent rectangles which can form another rectangle 15%
4. *Efficiently* find the two rectangles whose centers are the closest to each other 5%

Although you have to submit the four subtasks separately, the reading of input can be copied across all of them. You can also reuse and build on the Rectangle class from one task to the next.

In the **input**, each rectangle is given to you as a **pair** of (**x**, **y**) points representing any of its opposing corners. When you are asked to **output** a rectangle, you are to print the **bottom-left and top-right** points.

e.g. input data [(1, 13), (10, 2)], corresponding output [(1, 2), (10, 13)]

For simplicity, each ordinate given to you will be a positive integer and will fit within 30 bits, i.e. you can safely add two ordinates together without requiring a larger data type.

Problem 1 - Intersection



40%

Find the region that is the **intersection of ALL** the rectangles. The first line in the input contains only **N**. The next N lines each contain 4 integers describing a rectangle's opposing corners: **x₁ y₁ x₂ y₂**.

If the intersecting region is non-empty, it is itself a rectangle. Output its coordinates [(**x₁**, **y₁**), (**x₂**, **y₂**)]. Otherwise, output "No intersection"

Sample Input

```
3
1 11 17 4
5 3 20 15
3 13 12 2
```

Sample Output

```
[ (5, 4) , (12, 11) ]
```

Submission

Your source file should be named intersection.cpp

Tip: Draw out the sample input on paper! It helps you visualize what your task is, and may help you to "see" an algorithm that solves the problem easily

Problem 2 - Closest Rectangle



40%

Given each point p_i :

Find and output the rectangle whose **center is nearest** to p_i

If multiple rectangles are nearest, output the top-most one among them

If multiple rectangles are still nearest, output the left-most one among them

No two rectangles have the same center.

One test case may contain a few queries:

The first line in the input contains only **N**

The next N lines each contain 4 integers describing a rectangle's opposing corners: $x_1 y_1 x_2 y_2$

Each subsequent line, until the end of the file, contains two integers $x_i y_i$, which form p_i

Sample Input

```
5
2 7 8 3
5 15 3 17
1 1 9 29
24 27 6 3
17 2 13 8
4 16
10 10
15 5
11 9
6 3
```

Sample Output

```
[ (3, 15), (5, 17) ]
[ (1, 1), (9, 29) ]
[ (13, 2), (17, 8) ]
[ (13, 2), (17, 8) ]
[ (2, 3), (8, 7) ]
```

Submission

Your source file should be named closest.cpp

Tip: Identify what data is really needed to solve your problem. If that data can be viewed as a part of a rectangle, and can be easily derived from its member variables, then you can create a member function to compute that data easily.

Problem 3 - Combine Rectangles

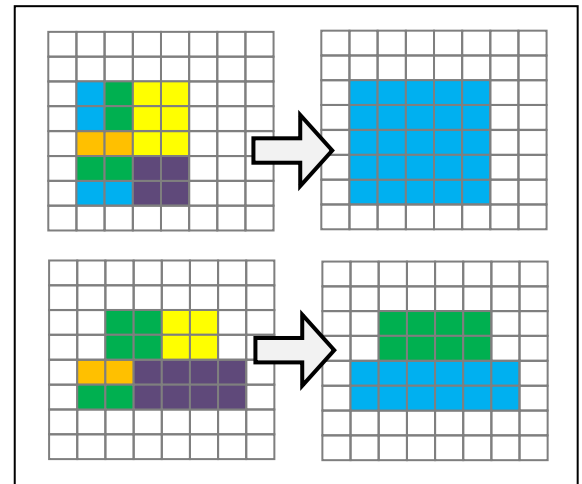


+15%

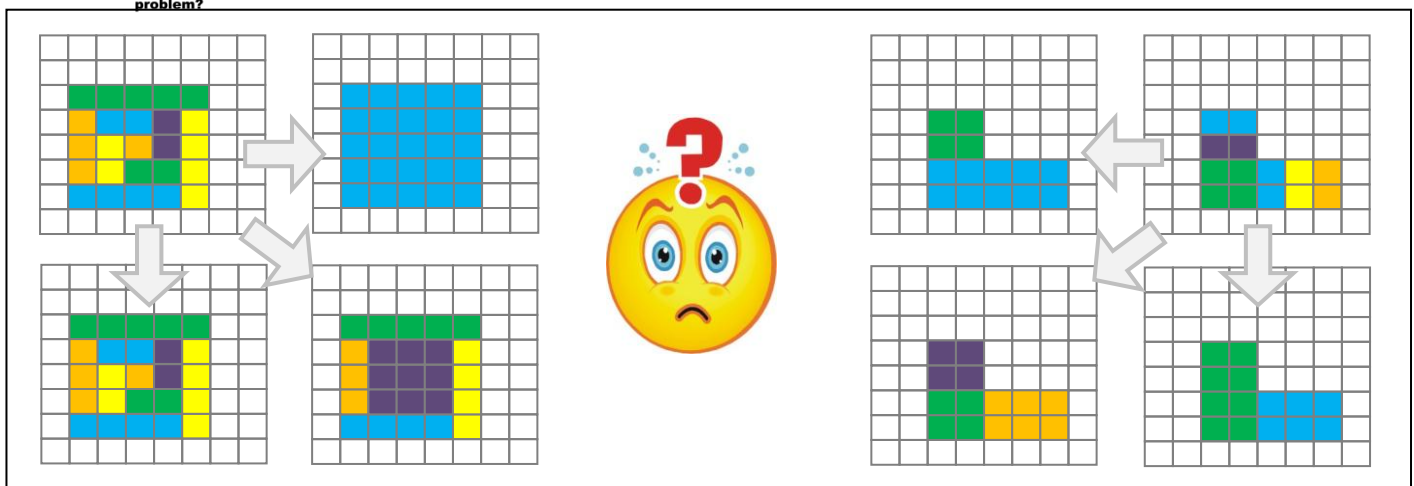
You observe that many adjacent rectangles *on the right* can be combined into one big rectangle. However, you are not sure how this can be implemented. Hmm... ..=(

This algorithm is too inefficient as proportional to N^3 rectangles are examined ($O(N^3)$ or cubic time):

```
keep trying to combine, N-1 passes
  for each rectangle
    for each other rectangle
      combine rectangles if possible
stop if nothing combined this pass
```



Just for this sub-task, $2 \leq N \leq 5,000$, and any of the N rectangles will also NOT overlap each other. You may also be thinking, "how do I handle the *cases below*?" Don't worry, such ambiguous cases will NOT be given....



As above, you are given $N+1$ lines of input describing the N rectangles. Output the number of rectangles left after combining them, such that no adjacent rectangle that can be combined is left behind.

Sample Input

```
8
5 1 7 6
3 6 2 4
1 6 2 4
5 1 3 3
1 3 3 4
3 6 5 3
1 1 3 2
3 3 1 2
```

Sample Output

```
1
```

Submission

Your source file should be named combine.cpp

Problem 4 - Nearest Rectangles



+5%

Find the pair of rectangles whose **centers are nearest** to each other, among all other pairs.

This algorithm is too inefficient as proportional to N^2 rectangles are examined ($O(N^2)$ or quadratic time):

```
initialize minDist
for each rectangle left
    for each other rectangle right
        if dist between left's, right's centers are closer than minDist
            update minDist
```

Problem 3 was tiring enough, now this... =X

As above, you are given $N+1$ lines of input describing the N rectangles. Output the distance between the centers of those two rectangles, always to 2 decimal places, even if exact.

Just for this subtask, $2 \leq N \leq 20,000$.

Sample Input

```
5
2 7 8 3
5 15 15 5
1 1 9 29
24 27 6 3
17 2 13 8
```

Sample Output

```
7.07
```

Submission

Your source file should be named nearest_rects.cpp

- End of Lab 2 -