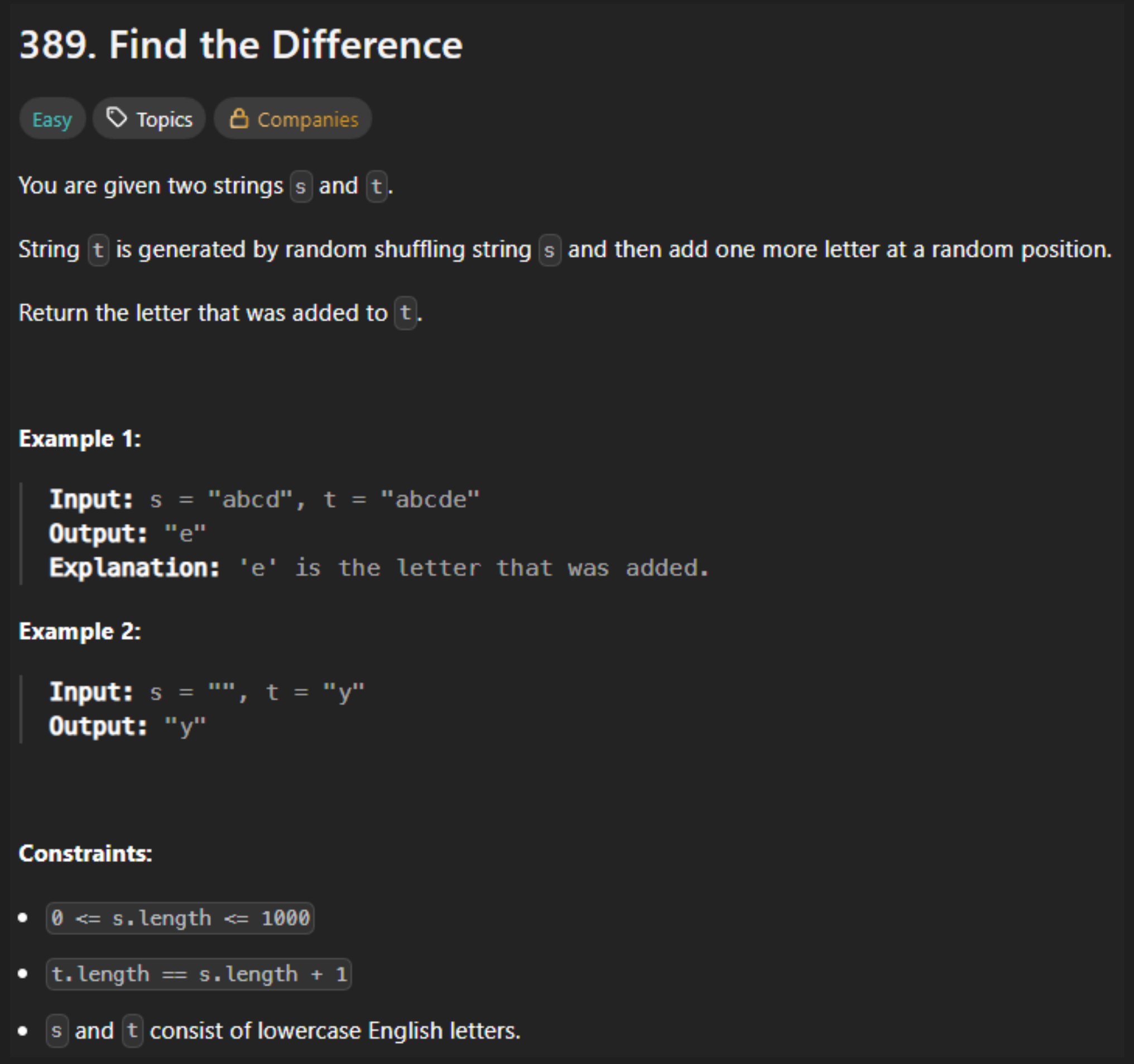
Weekly Contest -2



389. Find the Difference

Easy · Topics · Companies

You are given two strings `s` and `t`.

String `t` is generated by random shuffling string `s` and then add one more letter at a random position.

Return the letter that was added to `t`.

Example 1:

```
Input: s = "abcd", t = "abcde"
Output: "e"
Explanation: 'e' is the letter that was added.
```

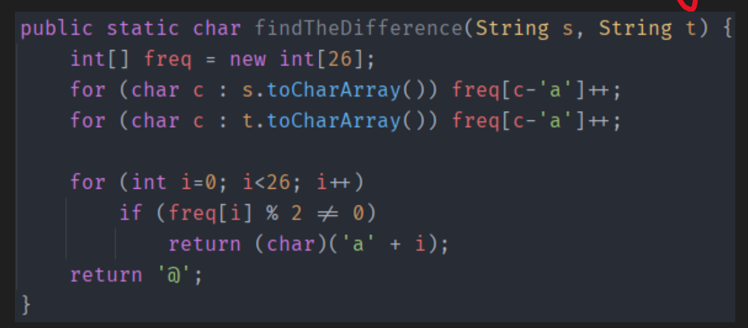
Example 2:

```
Input: s = "", t = "y"
Output: "y"
```

Constraints:

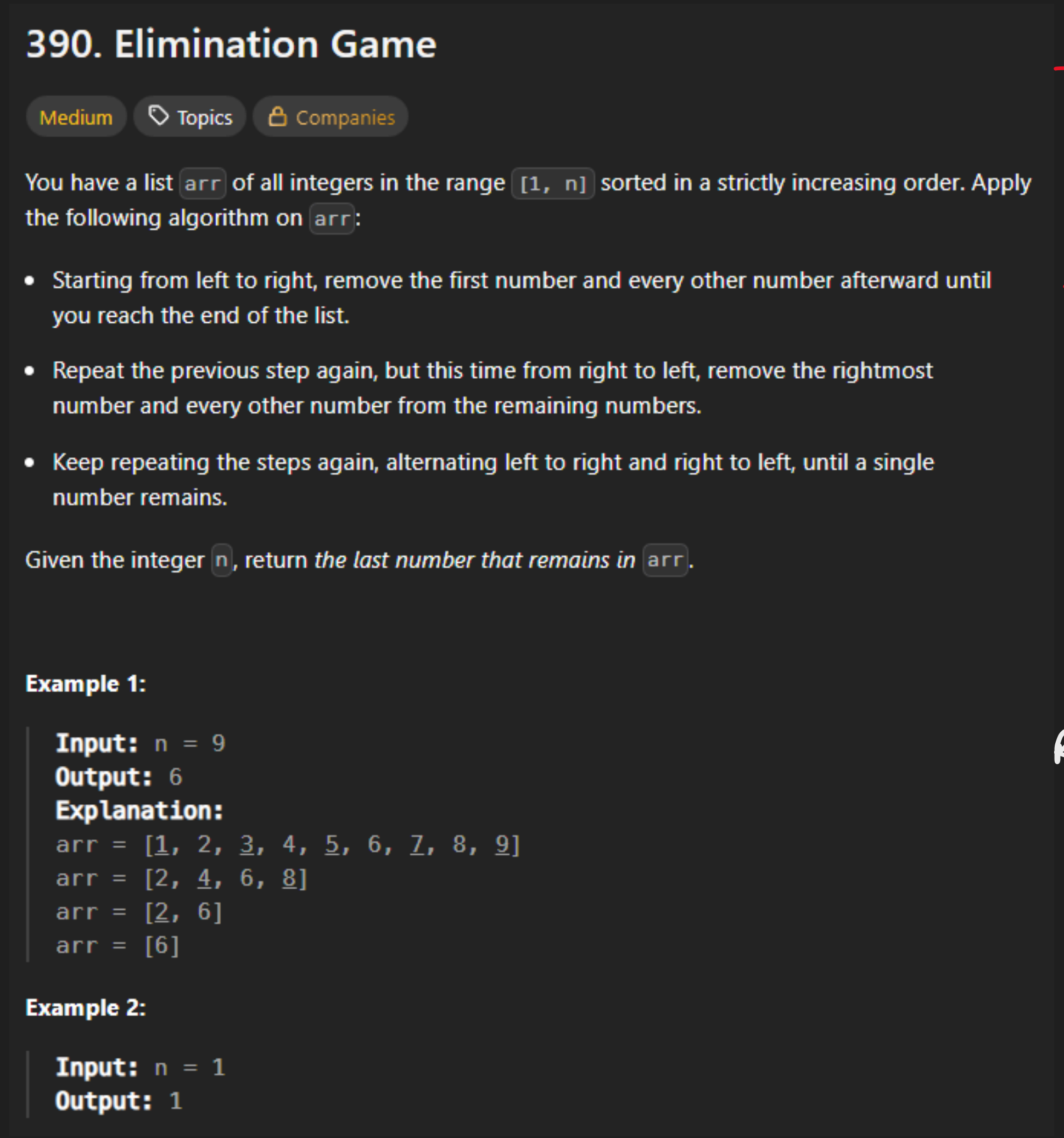
- `0 <= s.length <= 1000`
- `t.length == s.length + 1`
- `s` and `t` consist of lowercase English letters.

Store the frequency of each character
find which one has odd frequency



```java
public static char findTheDifference(String s, String t) {
    int[] freq = new int[26];
    for (char c : s.toCharArray()) freq[c-'a']++;
    for (char c : t.toCharArray()) freq[c-'a']++;

    for (int i=0; i<26; i++)
        if (freq[i] % 2 != 0)
            return (char)('a' + i);
    return '@';
}
```



390. Elimination Game

Medium · Topics · Companies

You have a list `arr` of all integers in the range `[1, n]` sorted in a strictly increasing order. Apply the following algorithm on `arr`:

- Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.
- Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.
- Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer `n`, return *the last number that remains in* `arr`.

Example 1:

```
Input: n = 9
Output: 6
Explanation:
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
arr = [2, 4, 6, 8]
arr = [2, 6]
arr = [6]
```

Example 2:

```
Input: n = 1
Output: 1
```

len - 9

→ ~~1~~ 2 ~~3~~ 4 ~~5~~ 6 ~~7~~ 8 ~~9~~

2 ~~4~~ 6 ~~8~~ ←

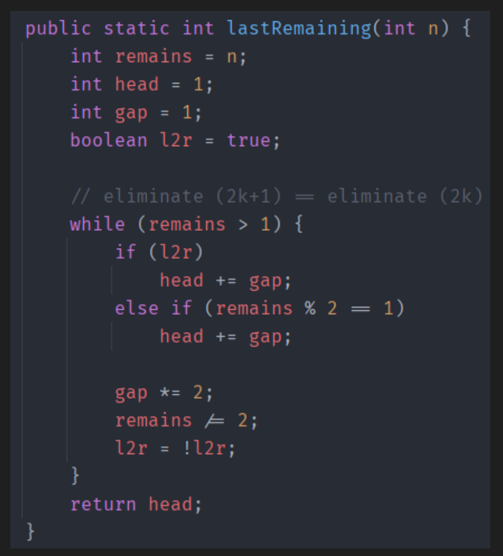
→ ~~2~~ 6

len - 10

→ ~~1~~ 2 ~~3~~ 4 ~~5~~ 6 ~~7~~ 8 ~~9~~ 10

~~2~~ 4 ~~6~~ 8 ~~10~~ ←

→ ~~4~~ 8



```java
public static int lastRemaining(int n) {
    int remains = n;
    int head = 1;
    int gap = 1;
    boolean l2r = true;

    // eliminate (2k+1) == eliminate (2k)
    while (remains > 1) {
        if (l2r)
            head += gap;
        else if (remains % 2 == 1)
            head += gap;

        gap *= 2;
        remains /= 2;
        l2r = !l2r;
    }
    return head;
}
```

Head 1
Gap 1

Even

For odd we would have the same values

L → R ~~1~~ 2 ~~3~~ 4 ~~5~~ 6 ~~7~~ 8 ~~9~~ 10 ~~11~~ 12 ~~13~~ 14 ~~15~~ 16 ~~17~~ 18 ~~19~~ 20 ~~21~~ 22

~~2~~ ↔ 4 ~~6~~ 8 ~~10~~ 12 ~~14~~ 16 ~~18~~ 20 ~~22~~

Head 2 Gap 2

R → L ~~4~~ 8 ~~12~~ 16 ~~20~~

Head 4
Gap 4

L → R Head 8
Gap 8

8 ~~16~~

R → L ~~Head~~ 16
Gap 16

R → L But the length is even, Keep head as it is

Head = 8 Gap = 16

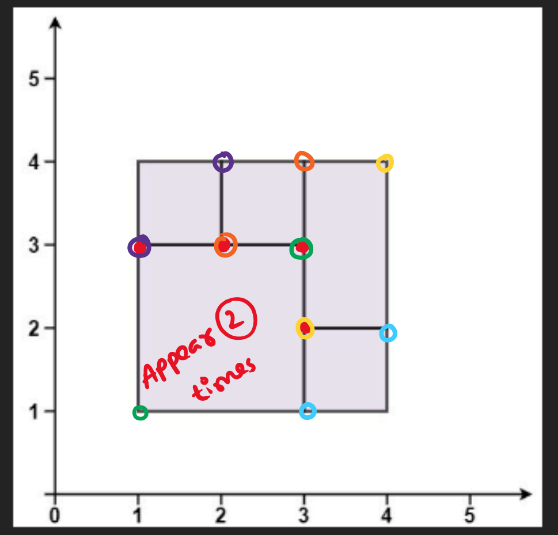
391. Perfect Rectangle

Hard · Topics · Companies

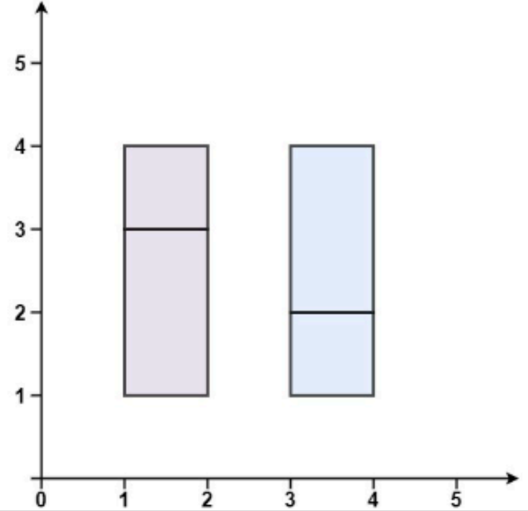
Given an array `rectangles` where `rectangles[i] = [x_i, y_i, a_i, b_i]` represents an axis-aligned rectangle. The bottom-left point of the rectangle is `(x_i, y_i)` and the top-right point of it is `(a_i, b_i)`.

Return `true` *if all the rectangles together form an exact cover of a rectangular region.*

Example 1:

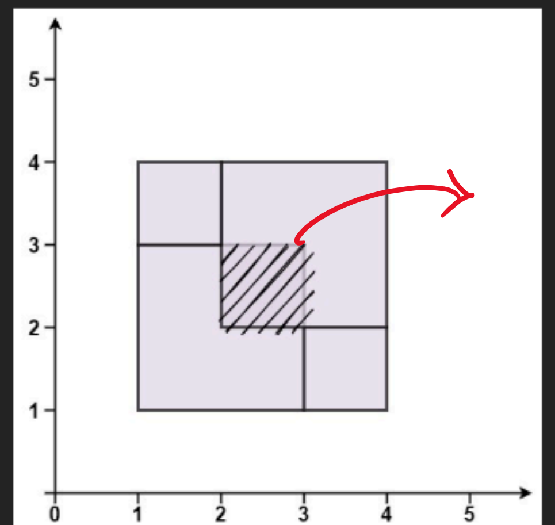


```
Input: rectangles = [[1,1,3,3],[3,1,4,2],[3,2,4,4],[1,3,2,4],[2,3,3,4]]
Output: true
Explanation: All 5 rectangles together form an exact cover of a rectangular region.
```



```
Input: rectangles = [[1,1,2,3],[1,3,2,4],[3,1,4,2],[3,2,4,4]]
Output: false
Explanation: Because there is a gap between the two rectangular regions.
```

Example 3:



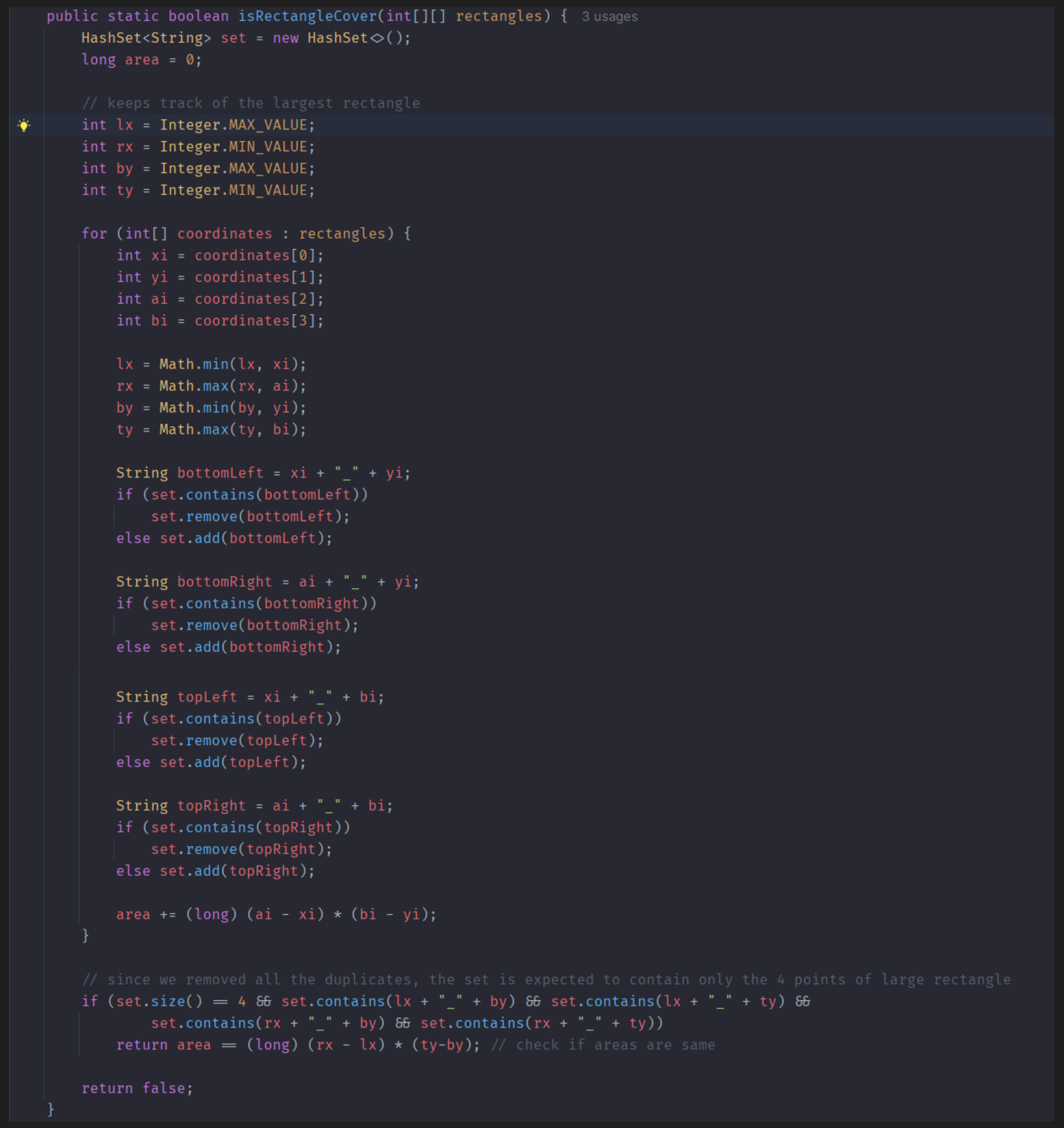
Overlap

Picks from example →

1. Area of smaller rectangles = Area of Large Rectangle
2. Except the edge points all appear twice (at least)
3. xy coordinates of the Large rectangle =

left lx → Min of xi
right rx → Max of ai
bottom by → Min of yi
top ty → Max of bi

Area = (rx - lx) × (ty - by)



```java
public static boolean isRectangleCover(int[][] rectangles) {  3 usages
    HashSet<String> set = new HashSet<>();
    long area = 0;

    // keeps track of the largest rectangle
    int lx = Integer.MAX_VALUE;
    int rx = Integer.MIN_VALUE;
    int by = Integer.MAX_VALUE;
    int ty = Integer.MIN_VALUE;

    for (int[] coordinates : rectangles) {
        int xi = coordinates[0];
        int yi = coordinates[1];
        int ai = coordinates[2];
        int bi = coordinates[3];

        lx = Math.min(lx, xi);
        rx = Math.max(rx, ai);
        by = Math.min(by, yi);
        ty = Math.max(ty, bi);

        String bottomLeft = xi + "_" + yi;
        if (set.contains(bottomLeft))
            set.remove(bottomLeft);
        else set.add(bottomLeft);

        String bottomRight = ai + "_" + yi;
        if (set.contains(bottomRight))
            set.remove(bottomRight);
        else set.add(bottomRight);

        String topLeft = xi + "_" + bi;
        if (set.contains(topLeft))
            set.remove(topLeft);
        else set.add(topLeft);

        String topRight = ai + "_" + bi;
        if (set.contains(topRight))
            set.remove(topRight);
        else set.add(topRight);

        area += (long) (ai - xi) * (bi - yi);
    }

    // since we removed all the duplicates, the set is expected to contain only the 4 points of large rectangle
    if (set.size() == 4 && set.contains(lx + "_" + by) && set.contains(lx + "_" + ty) &&
            set.contains(rx + "_" + by) && set.contains(rx + "_" + ty))
        return area == (long) (rx - lx) * (ty-by); // check if areas are same

    return false;
}
```