Short Java solution with explanation (updated)

Last Edit: Mar 6, 2018, 10:04 PM

(/mylzsd) mylzsd (/mylzsd) ★ 137

If all rectangles can form an exact rectangular area, they should follow these conditions:

1. The sum of area of all small rectangles should equal to the area of large rectangle.
2. At any position except outer four corners, the amount of overlapping corners should be even (2, 4).
3. Corners that overlap at the same point should be different type (top-left, top-right, bottom-left, bottom-right).

So, I used

1. Four int variables to record the boundaries of large rectangle and then calculate the area.
2. A hashmap that maps corner with its type.
3. Four numbers (1, 2, 4, 8) to represent four types of corner. Then use bit manipulation to modify and check.

O(n) time complexity, O(n) space, 112 ms run time.
Special credit to @wu474purdue-edu

```
public class Solution {
    Map<String, Integer> map = new HashMap<String, Integer>();
    public boolean isRectangleCover(int[][] rectangles) {
        if (rectangles.length == 0 || rectangles[0].length == 0) return false;
        int lx = Integer.MAX_VALUE, ly = lx, rx = Integer.MIN_VALUE, ry = rx, sum =
        for (int[] rec : rectangles) {
            lx = Math.min(lx, rec[0]);
            ly = Math.min(ly, rec[1]);
            rx = Math.max(rx, rec[2]);
            ry = Math.max(ry, rec[3]);
            sum += (rec[2] - rec[0]) * (rec[3] - rec[1]);
            //bottom-left
            if (overlap(rec[0] + " " + rec[1], 1)) return false;
            //top-left
            if (overlap(rec[0] + " " + rec[3], 2)) return false;
            //bottom-right
            if (overlap(rec[2] + " " + rec[1], 4)) return false;
            //top-right
            if (overlap(rec[2] + " " + rec[3], 8)) return false;
        }
        int count = 0;
        Iterator<Integer> iter = map.values().iterator();
        while (iter.hasNext()) {
            Integer i = iter.next();
            if (i != 15 && i != 12 && i != 10 && i != 9 && i != 6 && i != 5 && i !=
        }
        return count == 4 && sum == (rx - lx) * (ry - ly);
    }

    private boolean overlap(String corner, Integer type) {
        Integer temp = map.get(corner);
        if (temp == null) temp = type;
        else if ((temp & type) != 0) return true;
        else temp |= type;
        map.put(corner, temp);
        return false;
    }
}
```

Notes

Comments: 12

Sort By

Type comment here... (Markdown is supported)

Preview | Post

xinyao (/xinyao) ★ 15 Aug 29, 2016, 7:39 AM

Nice solution! I read your code and found it also works if changing one line like the bold line below.

```
public class Solution {
```

Read More

4 Share Reply

xinyao (/xinyao) ★ 15 Aug 29, 2016, 7:58 AM

Sorry the bold code above didn't display and the Below is C++ implementation.

```
class Solution {
private:
    unordered_map<string, int> table;
```

Read More

3 Share Reply

StefanPochmann (/stefanpochmann) ★ 20750 Aug 28, 2016, 5:21 PM

~~It's wrong, fails for example `[[0,0,1,1],[0,0,2,1],[1,0,2,1],[0,2,2,3]]`.~~
Update: The code got changed and doesn't fail that case anymore. I guess the downvoter is incapable of reading the very next post confirming that this was indeed a working counterexample pointing out a flaw. (google's cache (https://webcache.googleusercontent.com/search?q=cache:Aqb4N3bBomEJ:https://discuss.leetcode.com/topic/55997/short-java-solution-with-explanation-updated+&cd=1&hl=en&ct=clnk&gl=de) currently still shows this page at least 21 minutes after @mylzsd's reply and before the downvote).

2 Share Reply

SHOW 2 REPLIES

juanren (/juanren) ★ 98 Aug 30, 2016, 11:17 AM

@mylzsd
best solution, really clever for overlapping solution!

1 Share Reply

Tanych (/tanych) ★ 14 Aug 29, 2016, 9:33 AM

I think this solution is very straight-forward, and we can build all process in our mind when we encounter in the interviews:
A python implementation but tweaks some position since the rectangles are marked as `[bottom,left,top,right]`

Read More

1 Share Reply

coldknight (/coldknight) ★ 30 Dec 10, 2016, 12:15 AM

I think your codes will fail on this case [[0,0,1,1],[0,1,1,2],[0,2,1,3],[0,3,1,4],[0,4,1,5],[1,0,2,2],[1,1,2,3],[1,4,2,5],[2,0,3,1],[2,1,3,2],[2,2,3,3],[2,3,3,4],[2,4,3,5]]. I hope leetcode can add this case.

0 Share Reply

SHOW 3 REPLIES

DonaldTrump (/donaldtrump) ★ 412 Sep 25, 2016, 6:33 PM

@mylzsd Nice solution. Would you mind sharing how did you come up with the algorithm? What characteristic of the problem gave you the hint of counting the corners? Thanks in advance.

0 Share Reply

flashpacker (/flashpacker) ★ 1 Sep 9, 2016, 3:07 PM

Thanks for sharing this splendid method! It changes this problem into a point counting task.

0 Share Reply

lingyy (/lingyy) ★ 0 Sep 7, 2016, 9:51 AM

Really concise and easy to understand, thanks for sharing!
A tiny optimization: we could check the sum of area right after the for loop before check the corner counts, and return false right away if we find the area sum is not right. So we save the corner check for some cases. After this optimization the runtime is 93ms.

0 Share Reply

saddays (/saddays) ★ 4 Sep 1, 2016, 1:37 AM

@mylzsd what if i change the condition
if (i != 15 && i != 12 && i != 10 && i != 9 && i != 6 && i != 5 && i != 3) count++;
TO
if (i==1 || i==2 || i==4 || i==8) count++;
does it same?

Read More

0 Share Reply

SHOW 2 REPLIES

< 1 2 >