125 36

391. Perfect Rectangle

Notes

Description (/problems/perfect-rectangle/description/) Hints (/problems/perfect-rectangle/hints/) Submissions (/problems/perfec

(/problems/perfect-rectangle/discuss) > O(n log n) sweep line solution

Share Subscribe Report

O(n log n) sweep line solution

8.3K
VIEWS

41

Last Edit: Mar 27, 2018, 8:12 PM (/wddd) wddd (/wddd) ★ 47

Standard sweep line solution.

Basic idea:

Sort by x-coordinate.

Insert y-interval into TreeSet, and check if there are intersections.

Delete y-interval.

```
public class Event implements Comparable<Event> {
        int time;
        int[] rect;

        public Event(int time, int[] rect) {
                this.time = time;
                this.rect = rect;
        }

        public int compareTo(Event that) {
                if (this.time != that.time) return this.time - that.time;
                else return this.rect[0] - that.rect[0];
        }
}

public boolean isRectangleCover(int[][] rectangles) {
        PriorityQueue<Event> pq = new PriorityQueue<Event> ();
        // border of y-intervals
        int[] border= {Integer.MAX_VALUE, Integer.MIN_VALUE};
        for (int[] rect : rectangles) {
                Event e1 = new Event(rect[0], rect);
                Event e2 = new Event(rect[2], rect);
                pq.add(e1);
                pq.add(e2);
                if (rect[1] < border[0]) border[0] = rect[1];
                if (rect[3] > border[1]) border[1] = rect[3];
        }
        TreeSet<int[]> set = new TreeSet<int[]> (new Comparator<int[]> () {
                @Override
                // if two y-intervals intersects, return 0
                public int compare (int[] rect1, int[] rect2) {
                        if (rect1[3] <= rect2[1]) return -1;
                        else if (rect2[3] <= rect1[1]) return 1;
                        else return 0;
                }
        });
        int yRange = 0;
        while (!pq.isEmpty()) {
                int time = pq.peek().time;
                while (!pq.isEmpty() && pq.peek().time == time) {
                        Event e = pq.poll();
                        int[] rect = e.rect;
                        if (time == rect[2]) {
                                set.remove(rect);
                                yRange -= rect[3] - rect[1];
                        } else {
                                if (!set.add(rect)) return false;
                                yRange += rect[3] - rect[1];
                        }
                }
                // check intervals' range
                if (!pq.isEmpty() && yRange != border[1] - border[0]) {
                        return false;
```

```
                                        //if (set.isEmpty()) return false;
                                        //if (yRange != border[1] - border[0]) return false;
                                }
                        }
                        return true;
                }
```

Comments: 13

Sort By

Notes

Type comment here... (Markdown is supported)

Preview Post

zhangCabbage (/zhangcabbage) ★ 122 Aug 29, 2016, 8:22 AM

said in O(n log n) sweep line solution (https://discuss.leetcode.com/post/null):
Good idea to solve this problem, I read the code very careful, so I write some to make the code easy to understand the logic.
this way first to sort x-coordinate in order to handle rectangle from left to right
when the time is same then sort by the rect[0], why do like this?

Read More

6 Share Reply

SHOW 1 REPLY

wangxinbo (/wangxinbo) ★ 85 Dec 29, 2016, 8:19 PM

@wddd Really nice solution! Here is my C++ version.

Read More

3 Share Reply

SHOW 1 REPLY

LifeIsSimple (/lifeissimple) ★ 54 Oct 8, 2016, 2:36 PM

The comparator of the tree set is really awesome to detect intersection!
I hope my explanation below will help. For example, so basically we have three rects: Rect 1 (1, 1, 3, 3); Rect2 (1, 3, 3, 4); and Rect3 (2, 3, 3, 4). Rect3 intersects with Rect2 obviously.
So what the tree set does is that, **when the sweep line goes to the time 2 (x = 2)**, the tree set still contains Rect2(1, 3, 3, 4) because the right time event (time=3) has not been polled from the priority queue, which means

Read More

3 Share Reply

StefanPochmann (/stefanpochmann) ★ 20750 Aug 28, 2016, 11:03 AM

Very nice way of detecting and handling interval overlaps… I think few people even know/use that `set.add` returns something (at least I often see people do `if (!set.contains(x)) { set.add(x); foo(); } else { bar(); }`).
The `if (set.isEmpty()) return false;` seems unnecessary, isn't it?

3 Share Reply

SHOW 1 REPLY

Notes

jiongxiaobu (/jiongxiaobu) ★ 8 Jun 18, 2017, 7:13 AM

The idea of using TreeSet to determine overlaps is so brilliant!
Here is my Java implementation:

Read More

0 Share Reply

pinkfloyda (/pinkfloyda) ★ 358 May 6, 2017, 5:37 PM

Smart idea, I find such sweep line method can be used to check if given rectangles overlap or not.

0 Share Reply

ofLucas (/oflucas) ★ 744 Nov 30, 2016, 6:56 AM

I'm surprised by your excellent solution.
Here I share my understanding of your implementation.

Read More

0 Share Reply

hot13399 (/hot13399) ★ 915 Oct 31, 2016, 2:26 AM

My Java solution of idea of sweepline:

Read More

0 Share Reply

xiong6 (/xiong6) ★ 43 Oct 21, 2016, 8:41 AM

I cannot see the role of the TreeSet here. Elements are added to and removed from this set, but it is not used for judgement in the proces. The only place is `if (!set.add(rect)) return false;` , but the purpose here is to detect duplication, which is not essential. Right ?
PS: Sorry. I am wrong. The set here is used to detect overlap, which is very important.

0 Share Reply

InvictusNeo (/invictusneo) ★ 0 Oct 10, 2016, 12:28 AM

What if we allow overlap for this question? Meaning that example 4 is considered as a legal perfect rectangle, how can we tackle this then?

0 Share Reply

Notes

‹ 1 2 ›