

Re: Covering rectangles in 2D

Source: <http://sci.tech-archive.net/Archive/sci.math/2008-03/msg03203.html>

- *From:* quasi <quasi@xxxxxxxx>
 - *Date:* Sat, 22 Mar 2008 04:57:27 -0500
-

On Sat, 22 Mar 2008 05:41:42 -0000, Tim Little
<tim@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

On 2008-03-22, quasi <quasi@xxxxxxxx> wrote:

But as far as I can see, you need to check every cell in a row (or column), not just the one that generated the grid lines for that row (or column).

Sort the x and y bounds of each rectangle, so that every rectangle is associated with exactly two grid lines.

Why 2, why not 4?

Let's start with the concept of grid lines.

Presumably, a grid line is the extension of a side of a rectangle -- yes?

Also, since it seems you only care about grid lines that can be `_crossed_`, there's no need to consider the sides of the outer rectangle as grid lines -- right?

Now the grid lines partition the big rectangle into cells, at most

$(2n + 1)^2$ of them, I think,

although if there are that many, then there will be empty cells, for sure.

In any case, $O(n)$ cells. Correct so far?

So when you talk about "crossing grid lines", I think you mean crossing from a cell to an adjoining cell, moving either horizontally or vertically -- right?

Re: Covering rectangles in 2D

It appears you want each grid line to be "owned" by one if its bordering rectangles — yes? Of course, there can be more than one such rectangle for each grid line, but it seems like you only want one of them to be designated as the owner. One way to do that is to simply examine the coordinates of each rectangles, one at a time, giving it ownership of any unowned grid line that it has — and in fact, simultaneously defining the grid lines. Am I'm still following your method?

(In the case of coinciding grid lines, treat the grid cells between them as a region of zero area which is not eligible for "uncovered" status)

This is not clear. I can see that you don't want shared ownership, but if the grid lines for some rectangle would coincide with an already owned grid line, why not simply refuse to assign that particular grid line to the new rectangle. I don't see the need to create a duplicate grid line with empty cells between them.

Start at some cell. I chose the top left.

No problem.

Do an $O(N)$ check for how many rectangles cover that starting cell.

I don't see why the initial cell gets special treatment. All the other cells just get their counts incremented or decremented. Why does the top cell get fully counted? If you did a full count for the other cells as well, that would give $O(n^3)$, not $O(n^2)$.

Keep track of this in a variable C that records the number of rectangles covering the current cell.

Presumably, you have a separate C for each cell — right?

Choose any path that covers the grid, crossing one grid line per step. I chose a sweep back and forth: along each row, then down one cell, then back along the next row in the opposite direction.

No problem.

Re: Covering rectangles in 2D

(Crossing any grid line gives the possibility of entering or leaving at most the one, known, rectangle associated with that grid line.)

Check whether that rectangle boundary was actually crossed.
Increment, decrement, or leave C unchanged accordingly. This is an $O(1)$ operation.

So in other words, when you cross a grid line, you only care about the owner. When you enter a cell, you check to see whether you are entering or leaving a rectangle that owns that grid line, and if so, you increment or decrement the count for that cell. The only thing you check for is whether you are entering or leaving the owning rectangle for the grid line just crossed -- right?

If $C = 0$ for a cell with nonzero area, the area is not covered.

Now I'm lost. Suppose we start at cell (1,1) -- row 1, column 1, in the top left hand corner, and cross into cell (1,2) -- row 1, column 2. Suppose the two vertical grid lines for column 2 are owned by a 1×1 rectangle which lives right there at cell (1,2). As you enter cell (1,2), the count for that cell gets incremented. If you now proceed horizontally to cell (1,3), the count for cell (1,2) gets decremented, so is now 0. Are you saying that we should stop? Obviously, I'm now off course.

If instead the grid is exhausted, the area is covered.

I'm definitely confused about the C variable. Is there just one for the whole grid, or one for each grid line, or one for each cell?

If this is still too obscure, I could probably post code in some programming language of your choice.

I think it would actually be clearer to just hash out the points of confusion. It would suffice to indicate the `_first_` place where I've seriously gone off course. I can then retry, starting from that point. I'd like to understand this.

Thanks.

quasi

.