Step 3: Generalizing Values

Now that we know what we did for this particular instance, we need to generalize to all instances of the problem. We will note that this step is often the most difficult (you have to think about *why* you did what you did, recognize patterns, and figure out how to deal with any possible inputs) and the most mistake prone (which is why we test the algorithm in Step 4 before we proceed).

Generalizing Values

One aspect of generalizing your algorithm is to scrutinize each value you used, and contemplate what it is in the general case. Is it a constant that does not change depending on the inputs? Does it depend on one (or more) of the parameters? If it does depend on some of the parameters, what is the relationship between them?

Going back to the rectangle example on which we did Step 2, we came up with **-1** for the left value of the answer rectangle. We can quickly rule out the idea that this is a constant—surely not all rectangles have -1 as the left side of their intersection (counterexamples would be easy to come by if we needed to convince ourselves).

Now we are left figuring out how -1 relates to the input parameters. It could be that the left value of the answer rectangle matches one of the values of the input rectangles—both the left and the bottom of the second rectangle are -1. It could be the case that it has some mathematical relationship to another value—maybe the left of the first rectangle divided by 2, or plus 1, or maybe the negative of the bottom of the first rectangle. Any of these would yield -1, and work in this case, but we need to think about *why* the answer is -1 to figure out the correct generalization.

Sometimes this analysis is quite difficult. Whenever you get stuck on generalization, it can help to repeat Steps 1 and 2, to give us more information to work with and more insight. For example, looking back at the other example we worked first in Step 1, we can rule out some of the ideas we pondered in the prior paragraph. From these two examples, we might draw the conclusion that the left value of the intersection is the left value of the second rectangle. We might proceed similarly to generate the following generalized algorithm (as with Step 2, notational specifics do not matter as long as you are precise enough that each step has a clear meaning):

To find the intersection of two rectangles, r1 and r2:

Your answer is a rectangle with

left: r2's left

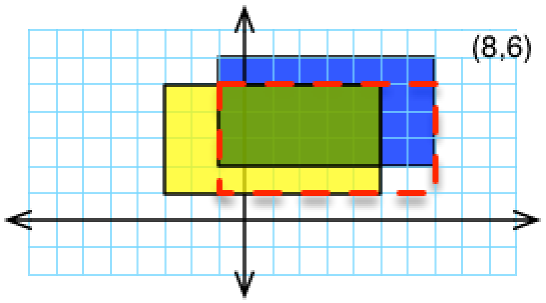
bottom: r1's bottom

right: r2's right

top: r1's top

While these generalized steps accurately describe the two examples we did, they are in fact not a correct generalization.

This figure shows a pair of rectangles where our algorithm gives the wrong answer—shown a red dashed rectangle. If we make an incorrect generalization such as this, we *should* catch it in Step 4 (or if not, then when we test the code at the end of Step 5). In such a case, we must return to Step 3 before proceeding, and fix our algorithm.



When you detect a mis-generalization of your algorithm, you have the advantage that you have already worked at least one example which highlights a case you need to analyze carefully. In this case, we can see that we want r1's right (*not* r2's right) for the right side of the answer, and r2's bottom (*not* r1's bottom) for the bottom side of the answer. Note that r1's right and r2's bottom did not work for the earlier cases, so we cannot simply change our algorithm to use those in all cases. Instead, we must think carefully about when we need which one and why.

Careful scrutiny will lead us to conclude that we need the minimum of r1's right and r2's right, and the maximum of r1's bottom and r2's bottom. We may also realize that we should do something similar for the left and top (if not, we should find that out when repeating Step 4). We could then come up with the following correctly generalized steps:

To find the intersection of two rectangles, r1 and r2:

Make a rectangle (called ans) with

left: maximum of r1's left and r2's left

bottom: maximum or r1's bottom and r2's bottom

right: minimum of r1's right and r2's right

top: minimum of r1's top and r2's top

That rectangle called ans is your answer.

We will note that in the case of rectangles that do not intersect, this algorithm will produce an illogical rectangle as the answer (its top will be less than its bottom and/or its left will be greater than its right). For the purpose of this problem, we will say that giving such an invalid rectangle in these cases is the intended behavior of the algorithm—in part because we have not learned how to represent "no such thing" easily.