Code Jam 2010 - Round 2

Analysis: Elegant Diamond

On this very difficult round, even the first problem was pretty challenging. The constraints were fairly small, but it might not be clear how to even get started. After all, there are a lot of ways you can enhance a diamond!

Let's start off with a slightly simpler question: given a position (c_x, c_y), is it *possible* to enhance (i.e, extend) the given diamond into an elegant diamond centered at (c_x, c_y)? (Note that this center might be either at a number, or at a space between numbers, depending on whether the elegant diamond has even or odd side length.) The resulting diamond would have to be symmetrical about the lines $x = c_x$ and $y = c_y$. Specifically, this means that for each (x, y), the values in positions (x, y), $(2c_x - x, y)$, $(x, 2c_y - y)$, and $(2c_x - x, 2c_y - y)$ all have to be equal. If the starting diamond already has two different values in one of these quartets, there's nothing we can do to change that.

Otherwise, it turns out that the diamond always can be extended to an elegant diamond with center at (c_x, c_y). For each of the quartets above where we have one value, we have to fill out the remaining values to be equal. After that, we can just enclose everything in a diamond shape and fill all remaining squares with 0. Here's an example:

```
  1
 2 3
5 6*6
 2 3
  1
```

Let's try to extend this to an elegant diamond centered at the *. First, we fill in all quartets, and then we extend to a proper diamond by adding 0's:

```
                           0
  1           1 1         1 1
 2 3         2 3 2       2 3 2
5 6*6  -->  5 6*6 5  --> 5 6*6 5
 2 3         2 3 2       2 3 2
  1           1 1         1 1
                           0
```

Done!

This is pretty clearly the smallest extension with the given center, so all we need to do is try each possible center, and choose the one that gives the smallest possible elegant diamond.

Of course, there are a lot of possible centers, but there is no need to consider one completely outside of the bounding box of the original diamond. We can always move such a center onto the edge of the bounding box without creating any inconsistencies, and that will lead to a smaller diamond in the end.

So in summary: we need to iterate over all possible centers contained within the original diamond, check if there is an elegant extension with that center, and then take the smallest of these. As always, please take a look at some of the correct submissions to see some full implementations!

By the way, the solution presented here is O(n^4), which is fast enough for this problem. Faster solutions do exist though, and you might enjoy looking for them.