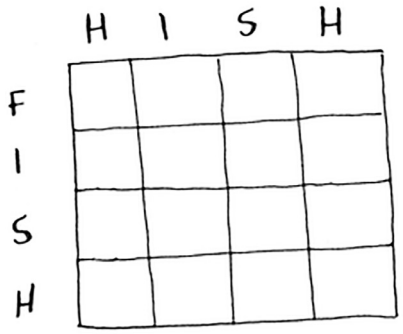
Remember, the values for the cells are usually what you're trying to optimize. In this case, the values will probably be a number: the length of the longest substring that the two strings have in common.

How do you divide this problem into subproblems? You could compare substrings. Instead of comparing *hish* and *fish*, you could compare *his* and *fis* first. Each cell will contain the length of the longest substring that two substrings have in common. This also gives you a clue that the axes will probably be the two words. So the grid probably looks like this.



If this seems like black magic to you, don't worry. This is hard stuff—that's why I'm teaching it so late in the book! Later, I'll give you an exercise so you can practice dynamic programming yourself.

Filling in the grid

Now you have a good idea of what the grid should look like. What's the formula for filling in each cell of the grid? You can cheat a little, because you already know what the solution should be—*hish* and *fish* have a substring of length 3 in common: *ish*.

But that still doesn't tell you the formula to use. Computer scientists sometimes joke about using the Feynman algorithm. The *Feynman algorithm* is named after the famous physicist Richard Feynman, and it works like this:

1. Write down the problem.
2. Think real hard.
3. Write down the solution.