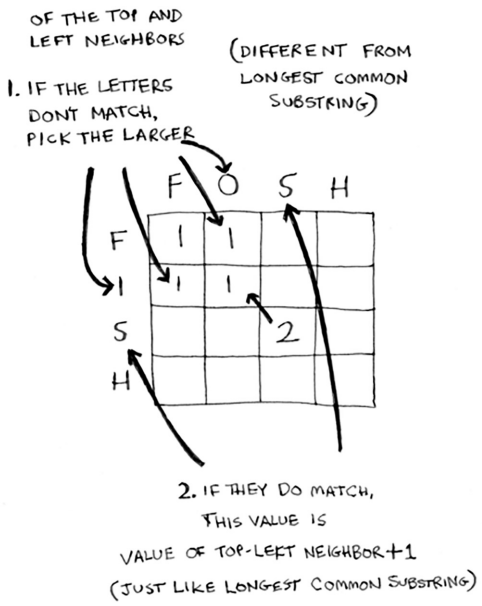
Here's my formula for filling in each cell.



And here it is in pseudocode:

```
if word_a[i] == word_b[j]:    <-- The letters match.
  cell[i][j] = cell[i-1][j-1] + 1
else:                         <-- The letters don't match.
  cell[i][j] = max(cell[i-1][j], cell[i][j-1])
```

Whew—you did it! This is definitely one of the toughest chapters in the book. So is dynamic programming ever really used? *Yes:*

- Biologists use the longest common subsequence to find similarities in DNA strands. They can use this to tell how similar two animals or two diseases are. The longest common subsequence is being used to find a cure for multiple sclerosis.
- Have you ever used diff (like `git diff`)? Diff tells you the differences between two files, and it uses dynamic programming to do so.
- We talked about string similarity. *Levenshtein distance* measures how similar two strings are, and it uses dynamic programming. Levenshtein distance is used for everything from spell-check to figuring out whether a user is uploading copyrighted data.