Most of the problems we have looked at in earlier chapters ask for either the maximum, minimum, or longest of something. However, it is also very common for a DP problem to ask for the number of distinct ways to do something. In fact, one of the first examples we looked at did this - recall that [Climbing Stairs](https://leetcode.com/problems/climbing-stairs/) asked us to find the number of ways to climb to the top of the stairs.

Another term used to describe this class of problems is "counting DP".

What are the differences with counting DP? With the maximum/minimum problems, the recurrence relation typically involves a max() or min() function. This is true for all types of problems we have looked at - iteration, multi-dimensional, etc. With counting DP, the recurrence relation typically just sums the results of multiple states together. For example, in Climbing Stairs, the recurrence relation was dp(i) = dp(i - 1) + dp(i - 2). There is no max() or min(), just addition.

Another difference is in the base cases. In most of the problems we have looked at, if the state goes out of bounds, the base case equals 0. For example, in the Best Time to Buy and Sell Stock questions, when we ran out of transactions or ran out of days to trade, we returned 0 because we can't make any more profit. In Longest Common Subsequence, when we run out of characters for either string, we return 0 because the longest common subsequence of any string and an empty string is 0. With counting DP, the base cases are often not set to 0. This is because the recurrence relation usually only involves addition terms with other states, so if the base case was set to 0 then you would only ever add 0 to itself. Finding these base cases involves some logical thinking - for example, when we looked at Climbing Stairs - we reasoned that there is 1 way to climb to the first step and 2 ways to climb to the second step.

These next 3 problems are very good practice problems that ask for the number of distinct ways to do something. Try them on your own, but if you get stuck, come back here for hints:

**276. Paint Fence**

Click here to show hint regarding state variables and dp

Let dp(i) represent the number of ways to paint i posts.

Click here to show hint regarding recurrence relation

At each new fence post, we can either paint it the same color as the previous post, or a different color. If we choose a different color, then there are k - 1 colors to choose from. If we choose the same color, we need to make sure that the post at i - 1 and i - 2 are not the same color.

Click here to show hint regarding base cases

There are k ways to paint one fence post. Since we are allowed to paint 2 consecutive fence posts the same color, there are k⋅k ways to paint two fence posts.

**518. Coin Change 2**

Click here to show hint regarding state variables and dp

Let dp[i] represent the number of ways to make i money.

Click here to show hint regarding recurrence relation

For each coin, for each amount of money i, you can make i money by adding the coin to i - coin money. Make sure to stay in bounds.

Click here to show hint regarding base cases

Since our recurrence relation only involves addition with other states, we need dp[0] to be a nonzero value, otherwise we will only ever be adding 0 to itself.

**91. Decode Ways**

Click here to show hint regarding state variables and dp

Let dp(i) return the number of ways the string starting from index i can be decoded. Return dp(0).

Click here to show hint regarding recurrence relation

You can always treat the current number as a one digit number. In addition, if the current number and next number combined is between 10 and 26, you have an additional option.

Click here to show hint regarding base cases

If s[i] ==  "0", then there is no way for this string to ever be decoded to anything. If i goes out of bounds, we need to return a nonzero value.