Interview Cake

Your quirky boss collects rare, old coins...

They found out you're a programmer and asked you to solve something they've been wondering for a long time.

Write a function that, given:

1. an `amount` of money
2. a vector of coin `denominations`

computes the number of ways to make the `amount` of money with coins of the available `denominations`.

Example: for `amount=4` (4¢) and `denominations=`$[1, 2, 3]$ (1¢, 2¢ and 3¢), your program would output 4—the number of ways to make 4¢ with those `denominations`:

1. 1¢, 1¢, 1¢, 1¢
2. 1¢, 1¢, 2¢
3. 1¢, 3¢
4. 2¢, 2¢

Gotchas

What if there's *no way* to make the `amount` with the `denominations`? Does your function have reasonable behavior?

We can do this in $O(n * m)$ time and $O(n)$ space, where n is the `amount` of money and m is the number of `denominations`.

A simple recursive approach works, but you'll find that your function gets called more than once with the same inputs. We can do better.

We could avoid the duplicate function calls by memoizing, but there's a cleaner bottom-up approach.

Breakdown

Solution

Complexity

$O(n * m)$ time and $O(n)$ additional space, where n is the `amount` of money and m is the number of potential `denominations`.

What We Learned

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now ➡ (/upgrade)