$dp[n] = \sum dp[n - c_i]$

[1, 2, 5], 5

0 1 2 3 4 5

```
const fn = (amount, coins) => {
    const dp = new Array(amount+1).fill(0);
    dp[0] = 1;
    for(const c of coins) {
        for( let i = 1; i <= amount; i++) {
            if (i - c >= 0) {
                dp[i] += dp[i-c];
            }
        }
    }
    return dp[amount];
}
```

By moving coins to outer loop, we are able to consider combinations ending at each coins only.

```
dp[n] = (() => {
    let count = 0;
    for (const c of coins) {
        if (n-c >= 0) {
            count += dp[n-c];
        }
    }
    return count;
})

const fn = (amount, coins) => {
    const dp = new Array(amount+1).fill(0);
    dp[0] = 1;
    for(let n = 1; n <= amount; n++) {
        dp[n] = (() => {
            let count = 0;
            for(const c of coins) {
                if(n-c >= 0) {
                    count += dp[n-c];
                }
            }
            return count;
```

[2, 3]

dp	0	1	2	3	4	5	6	7	8
	1	0	1	1	1	2	2	3	

[1, 2, 5]

0	1	2	3	4	5
1	1	2	3	5	9

wrong answer !

```
        }
    }
    return dp[amount];
}
```