In all the problems we have looked at so far, the recurrence relation is a static equation - it never changes. Recall [Min Cost Climbing Stairs](https://leetcode.com/problems/min-cost-climbing-stairs/). The recurrence relation was:

dp(i)=min(dp(i - 1) + cost[i - 1], dp(i - 2) + cost[i - 2])

because we are only allowed to climb 1 or 2 steps at a time. What if the question was rephrased so that we could take up to \text{k}k steps at a time? The recurrence relation would become dynamic - it would be:

dp(i)=min(dp(j) + cost[j]) for all (i - k)≤j<i

We would need iteration in our recurrence relation.

This is a common pattern in DP problems, and in this chapter, we're going to take a look at some problems using the framework where this pattern is applicable. While iteration usually increases the difficulty of a DP problem, particularly with bottom-up implementations, the idea isn't too complicated. Instead of choosing from a static number of options, we usually add a for-loop to iterate through a dynamic number of options and choose the best one.