**Dynamic Programming**

WHAT IS DP?

Before we move to discuss a 1D DP problem, let's define dp -- dynamic programming. Dynamic programming (DP) is a technique used by programmers to solve problems wherein the problems are **broken down into smaller sub-problems** (that have the same nature as that of the main problem) which are **solved just once,** and the results of the computation are saved for the future so that we need not compute the saved results again

IDENTIFICATION OF DP PROBLEMS

1. Maximise or minimize a certain quantity.
2. Count arrangements
3. Problems involving probability
4. Try all possible ways(recurison)

HOW TO RECOGNIZE 1D, 2D, OR ND DP?

When you analyze a problem, what you **search for is a function** with some number of **parameters that represent a state**. Then you try to relate the current state to some previous states of the function. If you can do this, and you can compute some state of the function by hand (in other words, if there is a state of the function that you know the value for) then the function is completely defined and you can compute it for any state you want. The **number of parameters doesn't matter** and doesn't tell you anything about the difficulty of the problem. The number of parameters depends on the problem. Now, if you have one parameter that describes the state(eg. Fibonacci numbers) you use a 1-D array. Why? Because arr[0] represents the value of the function for state 0, arr[1] represents the value of the function for state 1, and so on. **If you have k parameters that represent the state of the function then you have a k-dimensional array** and arr[x1][x2][x3][..][xk] represents the value of the function for the state (x1,x2,x3,..,xk). The only things that matter are these:

Recursion computes the values of the function at some state on the fly and in DP you store them because you want to avoid recomputation and that's all.

TABULATION AND MEMOISATION

In tabulation, we basically reduce the recursion stack space that we are using

SHORTCUT

1. Try to represent the problem in terms of index
2. Do possible stuffs on that index
3. Count all ways: Sum up all the stuff
4. Min or max = min or max(all stuffs)

QS Striver

1. Fibonacci

<https://www.codingninjas.com/codestudio/problems/nth-fibonacci-number_1115780?leftPanelTab=0>

1. Count ways to reach nth stair

<https://www.codingninjas.com/codestudio/problems/count-ways-to-reach-nth-stairs_798650>

1. Frog jump

<https://www.codingninjas.com/codestudio/problems/frog-jump_3621012?leftPanelTab=3>