L3: Framework for solving Dynamic Programming

# Memoization

* In the previous [Nsum problem](https://github.com/Mcdaddy-2pointoh/Dynamic-Programming/blob/main/Session%20notes/Session%202%20(Identify%20and%20N%20sum%20Problem).docx), we observed that f(N) = f(N+1) + 1 was the general formula and created an array of size N+1 that we used as cache table to access the result of a previous subproblems.
* We don’t want to recalculate things we want to rely on previously calculated solutions. Hence in our case in the cache table we just utilize the final most element of the cache table and add the latest number to it. No need to recalculate.
* **Memoization** is when we cache the result of a computation so that we can reuse it later to optimize the time complexity by giving up space.

# Problem 2 - A: Climbing Stairs Problem. There is a staircase and your goal is to answer the question, in how many distinct ways can you climb to the top. Assuming that we have two ways to climb a step i.e.

# One step at a time (-)

# Two steps at one time (-)

# To further illustrate the problem look at the diagrammatic representation below.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Way I (1-2) | Way II (2-1) | Way III (1-1-1) |

* Hence in total we can reach the top in 3 different ways
* Let us express the goal as an objective function (it is the function we want to minimize or maximize).
* In our case it is a distinct number of ways to reach the goal. Hence f(i) = number of steps to the Ith stair.
* Breaking down the problem into simpler sub problems and identify boundary and base cases.
  + f(0) => we are at goal and there is only one distinct way i.e. do nothing
  + f(1) => we can only use one step to reach goal, making our distinct steps as 1
  + f(2) => we can either use 2 one steps or 1 two steps making our distinct number of steps as 2
* Write a recurrence relation for the optimization equation. In our case we can only get to F(N) = F(N-2) + F(N-1)
* Combinatorics calls this as the rule of sum, it states that there are A ways to do something and B ways to do something else and we cannot do them at the same time then there are A+B ways of choosing one of the two actions.

# **Steps to Solve DP problems**

1. Define objective function. (What to achieve)
2. Identifying base cases (Understand the non-computable cases that must be fed)
3. Recurrence relation (Pattern that reoccurs over increasing steps)
4. Order of Computation (Order in which sub problems are solved)
5. Location of the Answer (Where we are looking for the answer)

# Problem 2 - B: Climbing Stairs Problem. There is a staircase and your goal is to answer the question, in how many distinct ways can you climb to the top. Assuming that we have two or more ways to climb a step i.e.

# One step at a time (-)

# Two steps at one time (-)