Algo- Recursion

1. What is recursion? How does it work?
2. How to solve a problem recursively?
3. How to analyze the time and space complexity of a recursive algorithm?
4. How can we apply recursion in a better way?

A recursive function should have the following properties so that it does not result in an infinite loop:

1. A simple base case (or cases) — a terminating scenario that does not use recursion to produce an answer.
2. A set of rules, also known as recurrence relation that reduces all other cases towards the base case.

Recurrence Relation

There are two important things that one needs to figure out before implementing a recursive function:

* recurrence relation: the relationship between the result of a problem and the result of its subproblems.
* base case: the case where one can compute the answer directly without any further recursion calls. Sometimes, the base cases are also called *bottom cases*, since they are often the cases where the problem has been reduced to the minimal scale, *i.e.* the bottom, if we consider that dividing the problem into subproblems is in a top-down manner.

Once we figure out the above two elements, to implement a recursive function we simply call the function itself according to the recurrence relation until we reach the base case.

Duplicate Calculation in Recursion

Memoization- is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again. (store the intermediate results in the cache so that we could reuse them later without re-calculation.)

-demonstrates how one can reduce compute time in exchange for some additional space.

Problem: Duplicate in Fibonacci numbers F(4) = F(3) + F(2) = (F(2) + F(1)) + F(2)

use Hash table-

-use ‘decorator’ to make memo more general and non-intrusive

Time complexity:

<https://leetcode.com/explore/learn/card/recursion-i/256/complexity-analysis/1669/>

Space Complexity:

<https://leetcode.com/explore/learn/card/recursion-i/256/complexity-analysis/1671/>