What is Recursion?

Recursion is a way of solving a problem by a function calling itself.

Like a Russian doll.

Why do we need recursion?

1. Recursion thinking is important in programming, and it helps you break down big problems into smaller ones and easier to use.

When to choose recursion?

1. If you can divide the problem into similar sub problems
2. Design an algorithm to compute nth.
3. Implement a method to compute all.
4. Practice
5. The prominent usage in data structure like trees and graphs.
6. Interview
7. It is used in many algorithms (divide and conquer, greedy and dynamic programming)

The logic behind Recursion

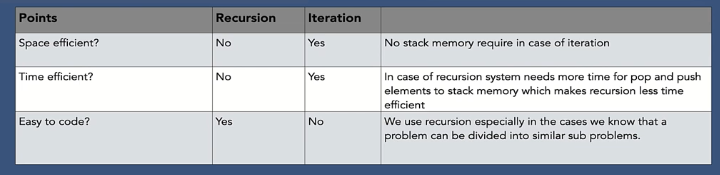
1. A method calls itself.
2. Exit from infinite loop.

Every time recursive method calls itself the system stores in the static memory for coming back because there is execution statements left after calling itself. This means that system somehow should remember the point it stops to call the recursive method.

Recursive vs iterative solutions

all recursive also can implement iteratively. a conditional statement decides the termination of recursion, but in iteration, values decide the termination of iteration. infinite recursion can lead to the system crash, whereas in iteration infinite iteration consumes CPU cycles.

The next point is recursion repeatedly invokes the mechanism consequently overhead of the method cost. This can be expensive in both processor time and memory space, while in iteration case we do not have such problem.



When to use/avoid recursion

Use:

1. When we can easily breakdown into similar subproblem
   1. When we are fine with extra overhead (both time and space) that comes with it
   2. When we need a quick working solution instead of efficient one.
   3. When we traverse a Tree.

Not used:

1. If time and space completely matter for us.
2. Recursion uses more memory. If we use embedded memory for example an application that takes more memory in the phone is not efficient.
3. Recursion can be slow. Because it needs the allocation of new stack frame. When our function calls itself recursively. It adds to the stack memory and when we are popping and pushing it takes time.

How to write Recursion in 3 steps:

1. Recursive case – the flow
2. Base case ß the stopping criterion
3. Unintentional case ß the constraint

