Recursion 101

Recursion

Recursion is a technique where a `function calls` itself to solve a problem by **breaking** it down into **smaller sub-problems**.

Base Condition:

Every **function call** in `recursion` is stored in the `call stack`. If the recursion is too deep or has no base condition, the call stack keeps growing until memory is exhausted, causing a stack overflow error.

A **base condition** is essential in recursion. It stops the recursion when a certain condition is met. Without it, recursion goes infinite and causes a stack overflow. **`if (num === 0) return;`.**

Approach:

`Problem:` Print numbers from n to 1 using `recursion`.

Print the `number`.

Recurse with `num - 1`.

Stop when `num === 0`.

Time Complexity: `O(n)`

-

one function call per number from n to 1.

Space Complexity: `O(n)`

-

Due to recursive call stack frames.

JavaScript | Python | Java | C++ | C | C#

```
function printDescending(num) {
    if (num === 0) return;
    console.log(num);
    printDescending(num - 1);
}
printDescending(5);
```

Video | Course | Discuss doubts | Contribute | Certificate

Print numbers from n to 1 using recursion - DSA Notes

Print numbers from n to 1 using recursion - DSA Notes