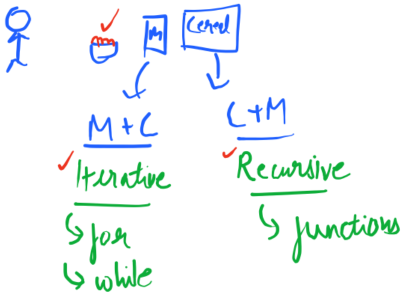
Thursday, 8 February 2024 8:26 PM

Recursion → Problem Solving Technique

↳ where a function will call itself

↳ Re-occur

↳ functions will be called again & again



Recursion

↳ a function will call itself until a specific condition is met.

↳ We will try to break problems into smaller problem, we assume the smaller task works and we just do the selfwork.

Steps to solve Recursion Problems

1) Identify the bigger problem

2) Identify the smaller problem (multiple)

↳ lesser effort

$$\underbrace{1+2}_{SP1} + \underbrace{3+4}_{SP2}$$

$$\underline{3} \underset{SW}{+} \underline{7}$$

$$\downarrow$$

$$\underline{10}$$

→ [illegible]

3) Trust me and assume that recursion works for
smaller problem

4) From the answer of smaller problems, do self work
and create the answer of bigger problem

5) Base condition → Smallest Problem
↳ Very important
Stack Overflow Exceptions

x+y ✓
↓
z

clr

dbt

1) Print Acciojob N times
BP → Print Acciojob N times
SP → Print Acciojob N-1 times
SW → Print Acciojob 1 time
BC → N==0

✓ ACCIO JOB
3 ACCIO JOB
ACCIO JOB
ACCIO JOB

$\underbrace{1,2,3,4},5$

2) Print 1 to N

BP → Print, 1,2,3, ... N

SP → Print 1 to N-1

SW → Print N

BC → N>0

3) Print N to 1

BP → N, N-1, N-2, ..., 3,2,1

SP → Print N-1 to 1

Print N

SW → Print N

BC → N > 0

4) Decrease Print Increase Print

BP : N, N-1, N-2, ..., 3, 2, 1, 1, 2, 3, ..., N-2, N-1, N

SP : ✓N-1, N-2, ... 3, 2, 1, 1, 2, 3, ... N-2, N-1

SW : Print N
SP
Print N

BC : N <= 0 return;

5) Calculate Factorial Recursively

BP : Find N!

$\hookrightarrow N \times \underbrace{(N-1 \times N-2 \times N-3 \ldots \times 4 \times 3 \times 2 \times 1)}_{sp \leftarrow fact(n-1)}$

SP : (N-1)! ✓

SW : N × SP

BC : 0! = 1

→ Recursion on Arrays

↳