Recursion 1

AGENDA:

- What is recursion ?
- How to write recursion code ?
- How it works ?

Time and Space Complexity - Recursion 2

Advanced Batch

- Merge, Quicksort
- Trees, Heaps, Tries
- Backtracking
- DP
- Graphs

What is Recursion ?

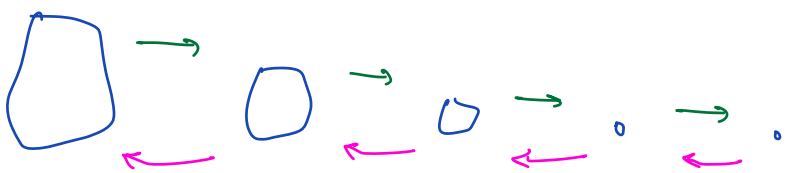
Function calling itself

Observations

1) Similar dolls

2) Size keeps decreasing

3) End doll



Solving a problem using a smaller instance of the same problem

subproblem

3 Steps to solve Recursion problems

1) Make an assumption

↳ Decide what your recursive function does & trust that it will do it.

2) Main Logic

↳ Solve the big problem using a subproblem

3) Base Condition

↳ When your recursion stops

Example: Sum of first N natural numbers

$$sum(N) \rightarrow \underbrace{1 + 2 + 3 + 4 + 5 + \ldots\ldots (N-2) + (N-1)}_{\text{Sum of all no.s from 1 to } (N-1)} + N$$

sum (N-1)

$$\Rightarrow \quad sum(N) = \underbrace{sum(N-1)}_{\text{subproblem}} + N$$

Assumption - sum(N) gives sum of all natural no.s from 1 to N

```
sum ( int N) {
    if (N==1)
        return 1
```

← Base Condition

sum(1) = 1 ✓

```
    return sum(N-1) + N
}
```

← Main Logic

Example: Factorial of N

$$fact(N) = \underbrace{1 \times 2 \times 3 \times \ldots\ldots (N-1)}_{(N-1)!} \times N$$

$$fact(N) = \underbrace{fact(N-1)}_{\text{Subproblem}} \times N$$

Quiz

Assumption : fact(N) gives me N!

```
fact (int N) {
    if ( N==0)
        return 1
```

← Base Condition

```
    return fact(N-1) x N
}
```

← Main Logic

$0! = 1$

$1! = 1$

if N==1

$1! = fact(0) \times 1$

$= 1 \times 1 = 1$

if N==0

$0! = fact(-1) \times 1$

↑ Trouble

Dry Run - sum(N)

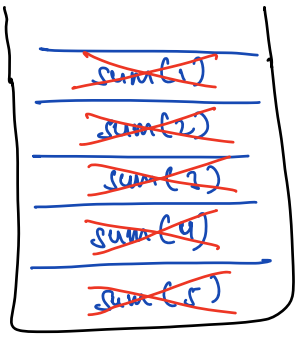
```
sum ( int N) {
    if (N==1)
        return 1

    return sum(N-1) + N
}
```

N=5

↓

Ans = 15



Function Call Stack

LIFO - Last In First Out

```
15
sum(5) {
    // N=5
    ret sum(4) + 5
}
```

```
10
sum(4) {
    // N=4
    ret sum(3) + 4
}
```

```
6
sum(3) {
    // N=3
    ret sum(2) + 3
}
```

```
3
sum(2) {
    // N=2
    ret sum(1) + 2
}
```

```
1
sum(1) {
    ret 1
}
```

Dry Run - Factorial

```
fact (int N) {
    if ( N==0)
        return 1

    return fact(N-1) x N
}
```

N = 3
↓
Ans = 6

```
6
fact (3) {
    // N=3
    ret fact (2) x 3
}
```

```
2
fact (2) {
    // N=2
    ret fact (1) x 2
}
```

```
1
fact (1) {
    // N=1
    ret fact (0) x 1
}
```

```
1
fact (0) {
    // N=0
    ret 1
}
```

Break till
10:10 PM

Example: Fibonacci Series

Golden Ratio

N = 0 1 2 3 4 5 6 7 8 9 10

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 .....

Given N, compute N^{th} fibonacci term

$$fib(N) = fib(N-1) + fib(N-2)$$

Assumption : fib(N) will give N^{th} fib number

```
fib( int N) {
    if ( N==0 or N==1)          ← Base Condition
        return 1

    return fib(N-1) + fib(N-2)  ← Main Logic
}
```

if N==0

Trouble

fib(0) = fib(-1) + fib(-2)

→ ans = 1

if N==1

fib(1) = fib(0) + fib(-1)

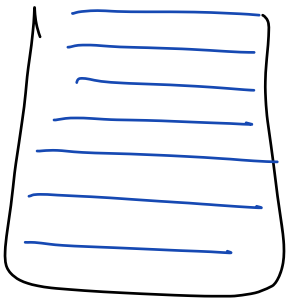
→ ans = 1

If you write wrong base case and your code keeps on running, what error will you get?

Stack overflow error

```
sum( int N) {
    if (N == 100)        ← wrong base case
        return 1

    return sum(N-1) + N
}
```



sum(5)

Infinite Recursion

Increasing print

Given a number N, print all numbers from 1 to N in increasing order using recursion.

incPrint(5) → 1, 2, 3, 4, 5

incPrint(4)

print(5)

incPrint(N)

↳ incPrint(N-1)

print(N)

Assumption : incPrint(N) will print all nums from 1 to N

```
incPrint (int N) {
    if (N==0)
        return;

    incPrint (N-1)
    print (N)
}
```

← Base Condition

← Main Logic

```
incPrint (3) {
    // N=3

    incPrint ( 2 )
    print (3)
}

incPrint (2) {
    // N=2

    incPrint ( 1 )
    print (2)
}

incPrint (1) {
    // N=1

    incPrint (0)
    print (1)
}

incPrint (0) {
    // N= 0

    return ;
}
```

Output

1

2

3

Decreasing print

Given a number N, print all numbers from N to 1 in decreasing order using recursion.

decPrint(5) → 5, 4, 3, 2, 1

TODO

One or two line change from the previous problem

Check Palindrome

Given a string, check if it is palindrome using a recursive function.

aba	malayalam	anna
madam	dad	appa
racecar	mom	bob
	sir	

---

a b c d d c b a

j i

0 N-1

if (s[i] == s[j])

i → i+1

j → j-1

else

return false

Assumption : isPalindrome (s, i, j) will check if s[i j] is palindromic or not

```
                                  0        N-1
bool    isPalindrome ( string s , int  i ,  int j ) {
        if ( i >= j)
            return true                                  ← Base Case

        if (s[i] == s[j])
            return isPalindrome (s, i+1, j-1)            ← Main Logic
        else
            return false
}
```

Power function

Implement power function using recursion.
Given a, n compute a^n . n >= 0.

Quiz

$a = 3$
$n = 4$ $\Rightarrow$ $a^n = ?$

$3^4 \Rightarrow 3 \times 3 \times 3 \times 3 = 81$

$a, n \Rightarrow a^n$

$$a^n = \underbrace{a \times a \times a \times a \ldots \times a}_{a^{n-1}} \times a \quad \text{(n times)}$$

$$a^n = a^{n-1} \times a$$

$$pow(a, n) = pow(a, n-1) \times a$$

Assumption : pow(a, n) will return a^n

```
pow( int a, int n) {
    if (n==0)
        return 1                    ← Base Case

    return pow(a, n-1) x a          ← Main Logic
}
```

$a^0 = 1$

Doubts

Thank You

Loops vs Recursion

→ Faster

→ No stack

→ Takes extra space for call stack

→ Simpler

Tower of Hanoi

↳ Loops - 300 - 400 lines

↳ Recursion - 5 line code

Integer

Hashset <Integer> hs = new Hashset <> ( );

---

pfsum → range based queries

Carry forward → Repetition in iterations

Hashing → Order does not matter

Sliding Window → k sized

Binary Search → Data is sorted

Good Night

Thank You

Friday