

# Recursion



1) Dhyaan jyada lagana

2) 2 baar

Topics :

Backtracking

DP

Trees

Merge & Quick Sort

# Recursion Aasaan Hai.

Recursion ke magic pe agar aapko bharosa ho gaya , to Samjhao  
maya aa raha hai.

# Contents

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Questions on Sorting

# What and Why?

[  
function calling itself

# Method Calling **Itself**

## Ques: Factorial of a number

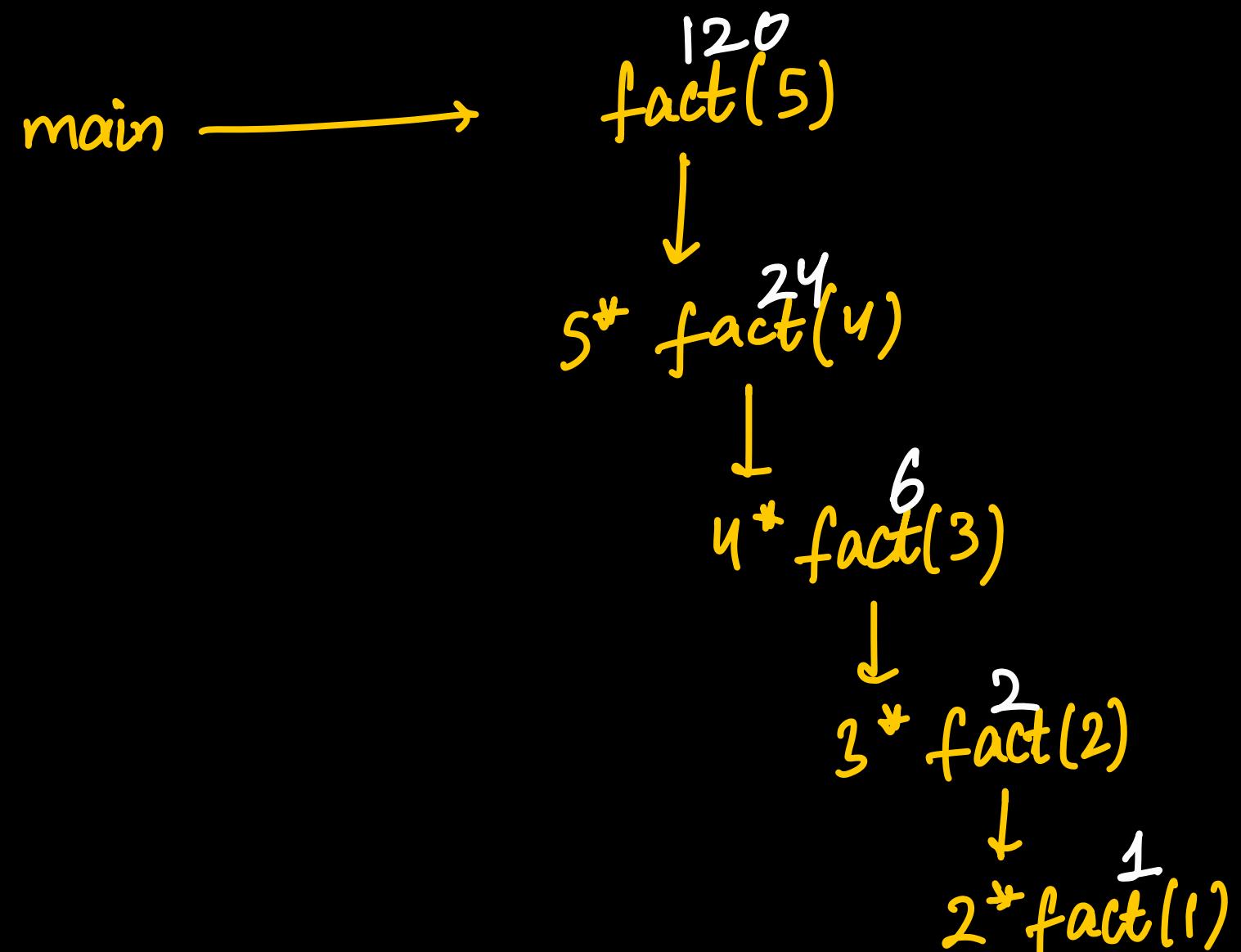
$$5! \text{ or } 15 = 5 \times 4 \times 3 \times 2 \times 1$$

$$n! = n \times (n-1)!$$

$$\text{fact}(n) = n * \text{fact}(n-1);$$

```
public static int fact(int n){  
    if(n==0 || n==1) return 1;  
    int ans = n * fact(n-1);  
    return ans;  
}
```

# Ques: Factorial of a number



# Ques: Print n to 1

```
public static void print(int 5 n){  
    ✓if(n==0) return;  
    ✓System.out.println(n);  
    ✓print(n1);  
}
```

```
public static void print(int 4 n){  
    ✓if(n==0) return;  
    ✓System.out.println(n);  
    ✓print(n3);  
}
```

```
public static void print(int 3 n){  
    ✓if(n==0) return;  
    ✓System.out.println(n);  
    ✓print(n2);  
}
```

```
public static void print(int 0 n){  
    ✓if(n==0) return;  
    System.out.println(n);  
    print(n-1);  
}
```

```
public static void print(int 1 n){  
    ✓if(n==0) return;  
    ✓System.out.println(n);  
    ✓print(n0);  
}
```

```
public static void print(int 2 n){  
    ✓if(n==0) return;  
    ✓System.out.println(n);  
    ✓print(n1);  
}
```

Output

- 5
- 4
- 3
- 2
- 1
- .

## Ques: Print n to 1

```
public static void print(int n){  
    if(n==0) return; → base case  
    System.out.println(n); → work  
    print(n-1); → call  
}
```

print(n) → prints 'n' & print(n-1) will take care of rest

# Ques: Print 1 to n (2 parameters)

```
public static void print(int n){  
    if(n==0) return; base case  
    print(n-1); call  
    System.out.print(n+" "); work  
}
```

*faith*

*print(n) → 1 to n print Karna*

*print(n-1) → 1 to n-1 print Karna*

# Ques: Print 1 to n (1 parameter)

```
public static void print(intn n){  
    ✓ if(n==0) return;  
    ✓ print(n-1);  
    ✓ System.out.print(n+" ");  
}
```

```
↓ ↑  
public static void print(intn n){  
    ✓ if(n==0) return;  
    ✓ print(n-1);  
    ✓ System.out.print(n+" ");  
}
```

```
↓ ↑  
public static void print(intn n){  
    ✓ if(n==0) return;  
    ✓ print(n-1);  
    ✓ System.out.print(n+" ");  
}
```

```
public static void print(intn n){  
    ✓ if(n==0) return;  
    ✓ print(n-1);  
    ✓ System.out.print(n+" ");  
}
```

```
↑ ↓  
public static void print(intn n){  
    ✓ if(n==0) return;  
    ✓ print(n-1);  
    ✓ System.out.print(n+" ");  
}
```

Output

• 1 2 3 4

# Standard Recursive Code

```
fun( ) {  
    |  
    base case  
    work  
    fun()  
    work  
}  
3
```

Time Complexity  $\rightarrow$  Max no. of calls in call stack

Space  $\rightarrow$  no. of calls

# HW: Print Decreasing-Increasing

↓  
 $n = 5$

5  
4  
3  
2  
|  
|  
2  
3  
4  
5

K.W. Find the reverse of a number

**Ques:** 'a' raised to the power 'b'

$$a^b = a * a^{b-1}$$

$$\text{pow}(a, b) = a * \text{pow}(a, b-1);$$

**Ques:** 'a' raised to the power 'b'

(Logarithmic Time Complexity)

$$2^{64}$$

$$\downarrow \\ 2 * 2^{63}$$

$$\downarrow \\ 2 * 2^{62}$$

⋮

$$2 * 2^0$$

64 calls

$$2^{64} = 2^{32} \times 2^{32}$$

$$2^{32} = 2^{16} \times 2^{16}$$

$$2^{16} = 2^8 \times 2^8$$

$$2^8 = 2^4 \times 2^4$$

$$2^4 = 2^2 \times 2^2$$

$$2^2 = 2^1 \times 2^1$$

$$6 = \log_2 64 \text{ calls}$$

$$a^b = a^{\frac{b}{2}} * a^{\frac{b}{2}} \quad (\text{if } b \text{ is even})$$

$$2^5 = 2^2 \times 2^2 \times 2$$

$$a^b = a^{\frac{b-1}{2}} * a^{\frac{b-1}{2}} * a \quad \text{if } (b \text{ is odd})$$

$$2^4 = (2^2)^2$$