Lec-17
DATE:
TIME:

D.S.A
Section 1

Function calling is vital

A()
B()
C()

A() → B() → C()

Function call is vital

Function that calls itself

Stopping base case!

Basic introduction

Divide and Conquer Algorithm
Sort Algorithm

Recursion

Function call itself

$P(n) \rightarrow$ Performance

Space

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Time

$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Asymptotic notation

$O(n) \rightarrow$ Worst case

$\Omega(n) \rightarrow$ Best case

$\Theta(n) \rightarrow$ Average case

$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$

$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$

$\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$

$1+2+3+\cdots+n$ $(n+1)$

$\frac{n(n+1)}{2}$

$\frac{n(n+1)}{2} + \frac{n(n+1)}{2} = \frac{(n+1)(n+1)}{2}$

$= \frac{(n+1)(n+1)}{2}$

$= \frac{n(n+1)(n+1)}{2} - \frac{(n+1)(1+1)}{2}$

$= \frac{n(n+1)(n+1)}{2} - \frac{(n+1)(2)}{2}$

$= \frac{n(n+1)(n+1)}{2} - (n+1)$

$\theta \rightarrow N! = (n-1)! \rightarrow$ Recursion (कोई भूल जाता है!)

$\theta \rightarrow \theta$ Stackoverflow $\rightarrow \theta$

① $N! = \boxed{(n-1)!} \rightarrow$ ✗ (कभी नहीं)

$0! = 1$
$1! = 1$

$5! = (4)! \times 5 = 120$

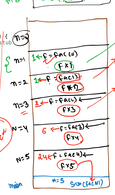
$f(3) \times 4$
$2 - 2 \times 3$
-1×2
1×1
$(0)! \times 1$

$5! = 1 \times 2 \times 3 \times 4 \times 5$

$1!$

Stack Fall

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    System.out.println(Fac(n));
}

public static int Fac(int n) {
    int f = Fac(n - 1);
    return f * n;
}
```

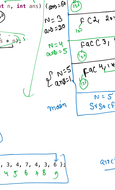



```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    System.out.println(Fac(n));
}

public static int Fac(int n) {
    // Base Case
    if (n == 0) {
        return 1;
    }
    int f = Fac(n - 1);
    return f * n;
}
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int a = 3;
    int n = 4;
    System.out.println(Pow(a, n));
}

public static int Pow(int a, int n) {
    if (n == 0) {
        return 1;
    }

    int curr = Pow(a, n - 1);
    return curr * a;
}
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    PD(n);
}

public static void PD(int n) {
    // TODO Auto-generated method stub
    if (n == 0) {
        return;
    }

    System.out.print(n);
    PD(n - 1);
}
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    PD(n);
}

public static void PD(int n) {
    // TODO Auto-generated method stub
    if (n == 0) {
        return;
    }

    PD(n - 1);
    System.out.print(n);
}
```



```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    System.out.println(Fac(n));
}

public static int Fac(int n) {
    // Base Case
    if (n == 0) {
        return 1;
    }
    // int f = Fac(n - 1);// ip recursion dega 2nd step hai PMI
    return n * Fac(n - 1);// BP 3rd step hai PMI
}
```



```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int n = 5;
    System.out.println(Fac(n, 1));
}

public static int Fac(int n, int ans) {
    // Base Case
    if (n == 0) {
        return ans;
    }

    return Fac(n - 1, ans * n);
}
```



```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int[] arr = {5, 7, 8, 10, 11, 15};
}

public static boolean isSorted() {
}
```

arr[0] > arr[1]
arr[1] > arr[2]
arr[2] > arr[3]
arr[3] > arr[4]
arr[4] > arr[5]

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int[] arr = {5, 7, 8, 11, 15, 15};
}

public static boolean isSorted(int[] arr, int i) {
    if (arr[i] > arr[i + 1]) {
        return false;
    }
    return isSorted(arr, i + 1);
}
```


n=5

1+2+3+4+5