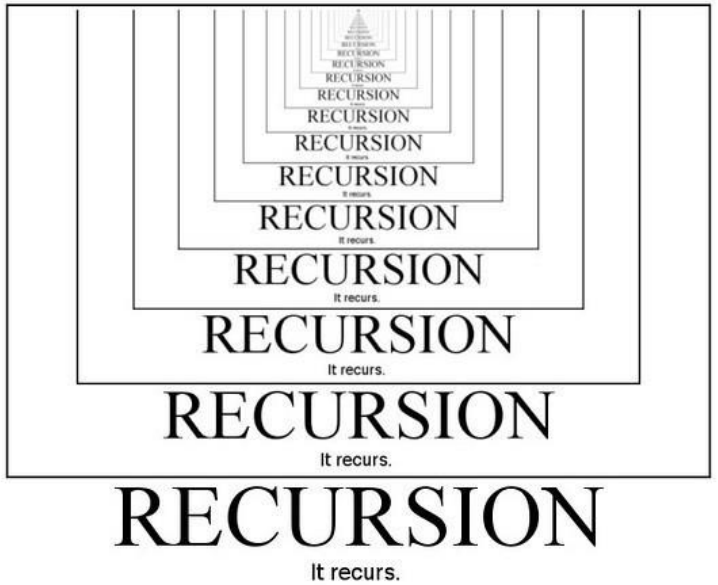
Recursion



By
Dr. Kawal Jeet

Introduction to Recursion

- Recursion is the process of defining a problem (or the solution to a problem) in terms of (a simpler version of) itself.

```
Way-to-Home
{
    If you are at home
            Stop moving
    Else
    {
            Take one step toward home
            Way-to-home
    }
}
```

Recursion in Programming

- Recursion happens when a function calls itself directly or indirectly.
- All recursive algorithms must o impor

```
A()
{
   A()
}
```

```
A()
{
   B()
}

B()
{
   A()
}
```

  - A recursive algorithm must have a base case.
  - A recursive algorithm must change its state and move base case.
  - A recursive algorithm must call itself, recursively.

Recursion in Java

n! = 1*2*3*4*…..*n

Factorial without recursion

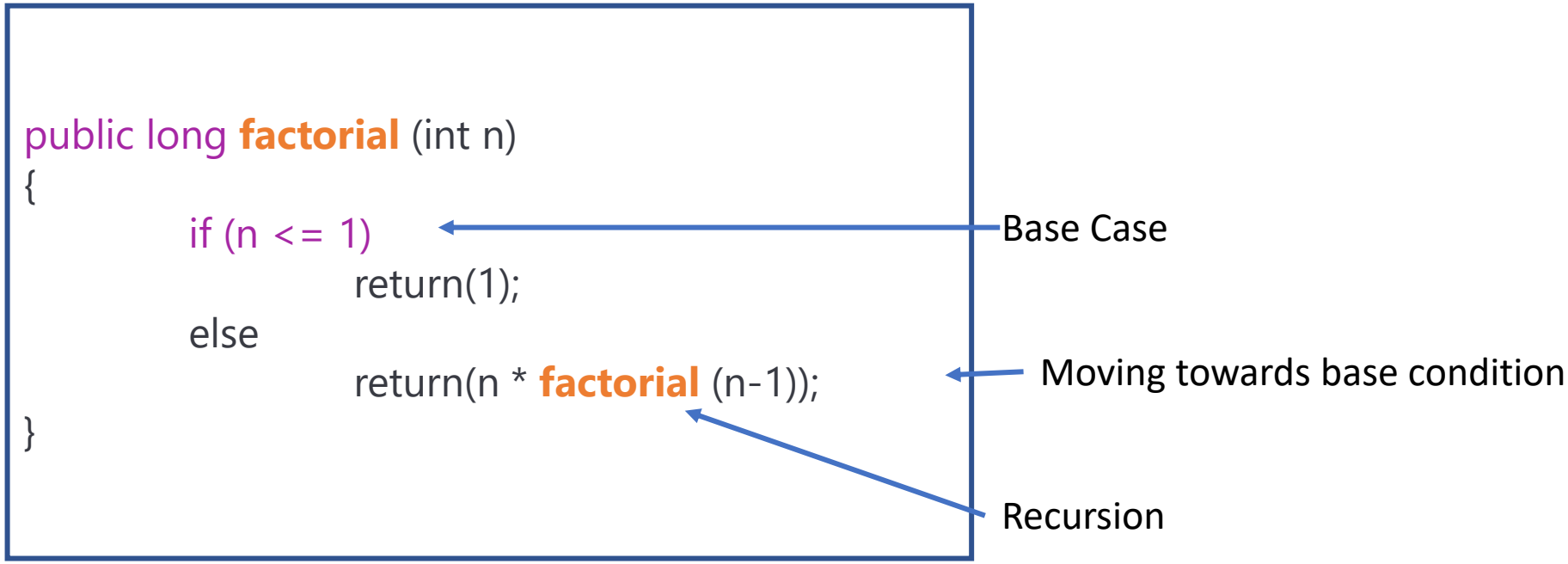
```
public long factorial (int n)
{
        long fact = 1;

        for(int i = 1; i <= n; ++i)
                fact = fact * i;

        System.out.println ("Factorial=“, fact);

        return(fact);
}
```

Space Complexity : Constant (for variables only)
Time Complexity : (n) for loop

Recursion in Java

n! = n * (n-1)! 4! = 4 * 3! 3! = 3 * 2! 2! = 2 * 1!

Factorial with recursion



Factorial with Recursion

Stack

Stack Overflow

```
Public static void main (Strings args[])
{
        long x=factorial (4);
        System.out.println(x);
}
```

```
public long factorial (4)
{
  if (num <= 1)      return(num);
  Else               return(4 * factorial (3));
}
```

4*3*2

```
public long factorial (3)
{
  if (num <= 1)      return(num);
  Else               return(3 * factorial (2));
}
```

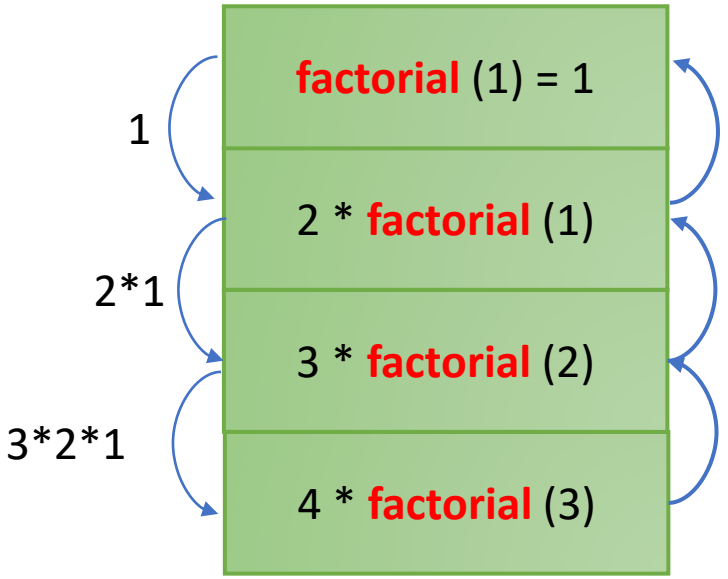
3*2

```
public long factorial (2)
{
  if (num <= 1)      return(num);
  Else               return(2* factorial (1));
}
```

2*1

1

```
public long factorial (1)
{
  if (num <= 1)
                     return(1);
}
```



4 * 3 * 2 * 1

Space Complexity : n (stack)
Time Complexity : 2*n
(One direct call and one return)

Factorial with Recursion

Stack

Stack Overflow

```
Public static void main (Strings args[])
{
        long x=factorial (4);
        System.out.println(x);
}
```

```
public long factorial (4)
{
  if (num <= 1)      return(num);
  Else               return(4 * factorial (3));
}
```

4*3*2*1

```
public long factorial (3)
{
  if (num <= 1)      return(num);
  Else               return(3 * factorial (2));
}
```

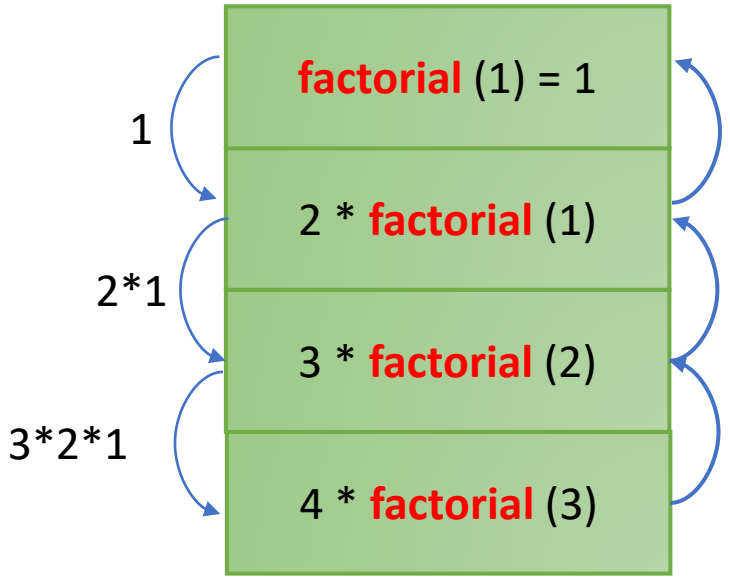
3*2*1

```
public long factorial (2)
{
  if (num <= 1)      return(num);
  Else               return(2* factorial (1));
}
```

2*1

1

```
public long factorial (1)
{
  if (num <= 1)
                     return(1);
}
```



4 * 3 * 2 * 1

Space Complexity : n (stack)
Time Complexity : 2*n
(One direct call and one return)

Disadvantages of Recursion

- Greater space requirements than iterative program
  - All functions will remain in the stack until the base case is reached.

- It also has greater time requirements
  - Function calls and returns overhead.

Advantages of Recursion

- Recursion provides a clean and simple way to write code.

  Some problems are inherently recursive like Tree Traversals, Tower of Hanoi, etc. For such problems, it is preferred to write recursive code.

- In many cases, recursive algorithms resemble more closely the logical approach we'd take to solve a problem.

Maximum of an array using recursion

- Int [] a= {6, 9, 2, 4};
- Math.max(4, rec{6,9,2})
- Math.max(2, rec{6,9})
- Math.max(9, rec{6})
- Math.max{6}=6
- Max(9,6)=9
- Max(2,9)=9
- Max(4,9)=9

Reverse a string using recursion

- Recursion(soap)
- p+Recursion(soa)
- a+Recursion(so)
- o+Recursion(s)
- Recursion(s)--→s
- os
- aos
- paos

References

- Eckel, Bruce. *Thinking in JAVA*. Prentice Hall Professional, 2003.
- https://web.mit.edu/6.005/www/fa15/classes/10-recursion/#reading_10_recursion (Accessed on October 17, 2020)
- Lafore, Robert. *Data structures and algorithms in Java*. Sams publishing, 2017.


Thank you for the opportunity