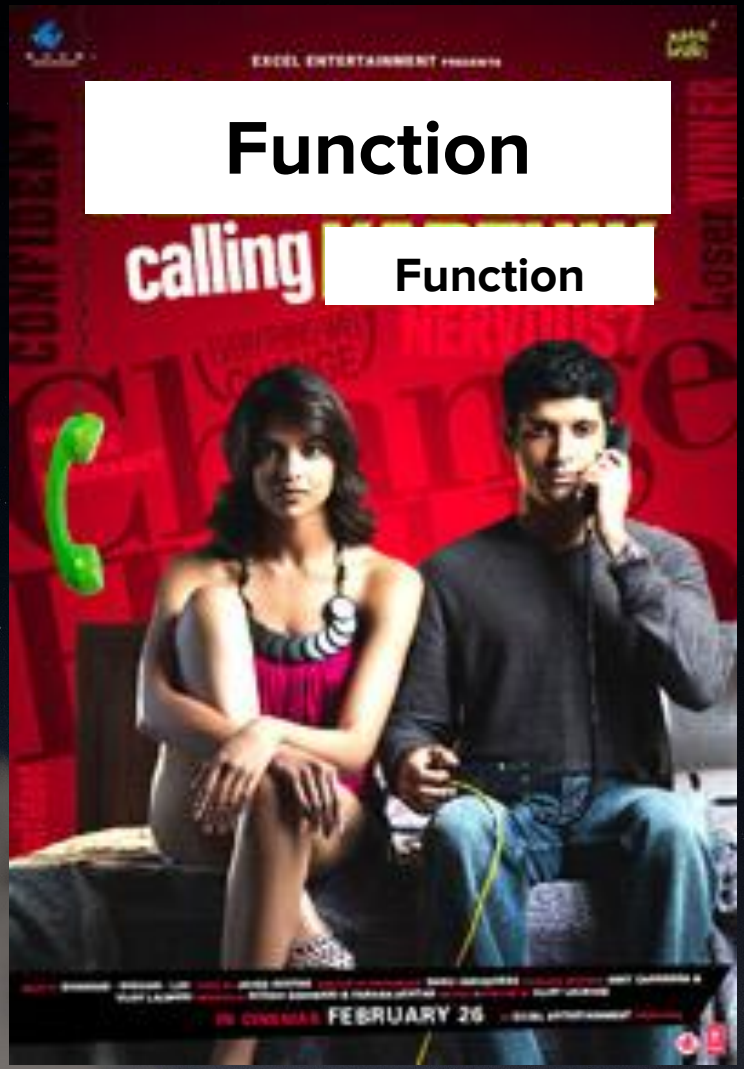
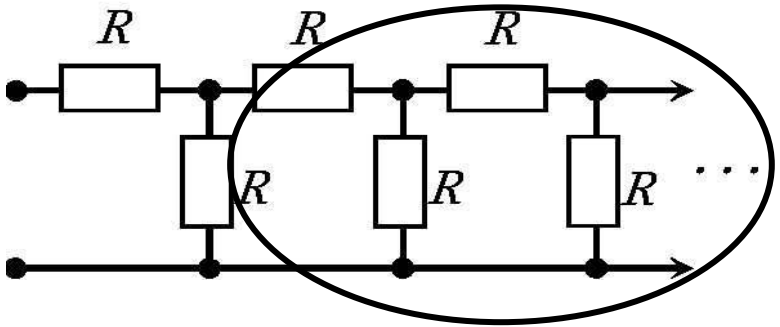
Recursion

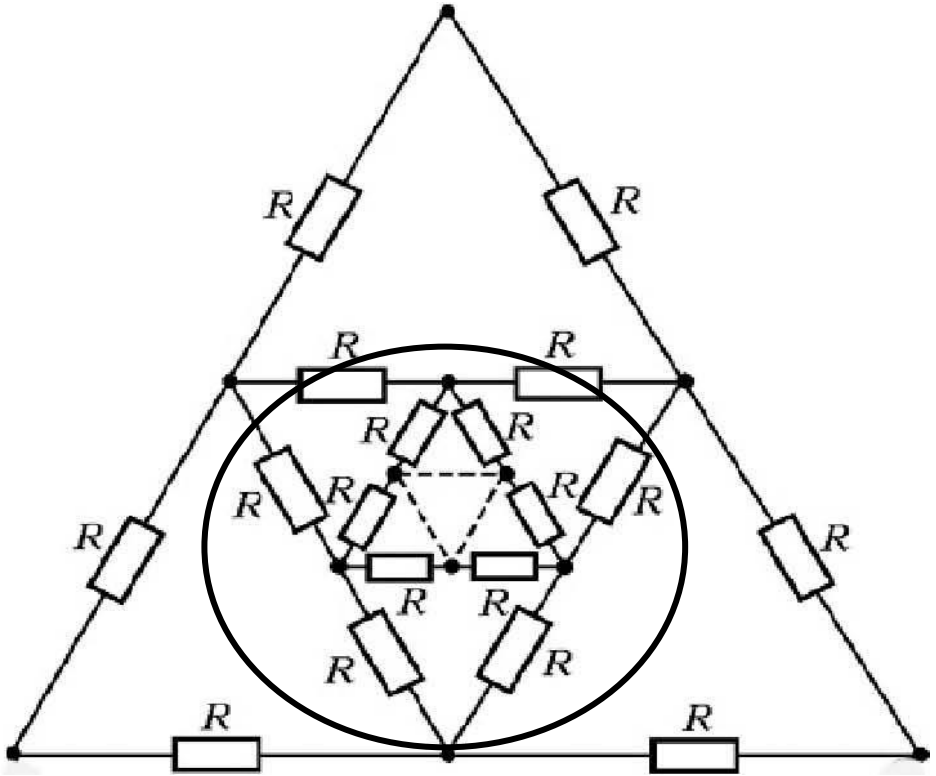
Abhimanyu Sethia

Academic Mentor Interview- ESC101



Recursion from JEE Days





Example 1: Factorial

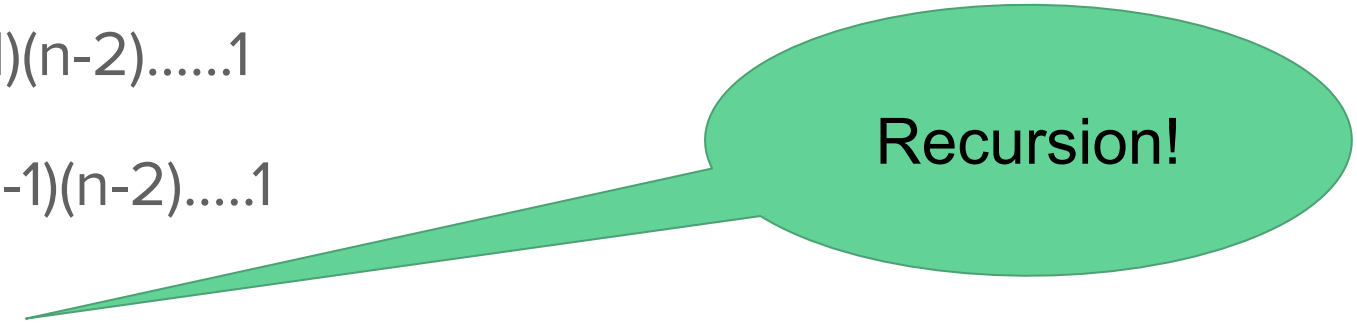
Definition:

Defined for n>=0

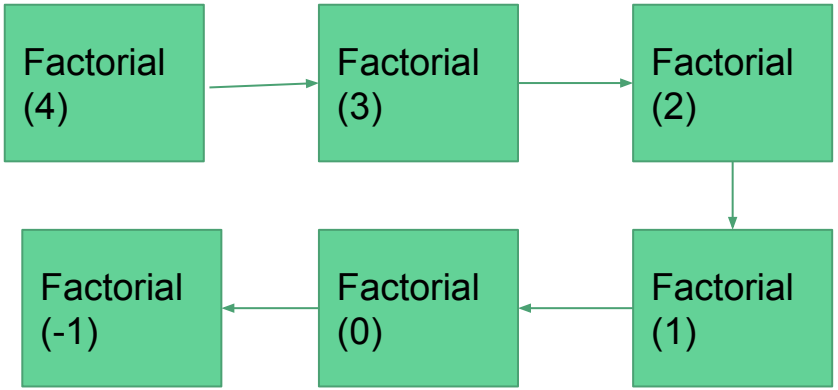
n! = n(n-1)(n-2).....1

(n-1)! = (n-1)(n-2).....1

n! = n* (n-1)!

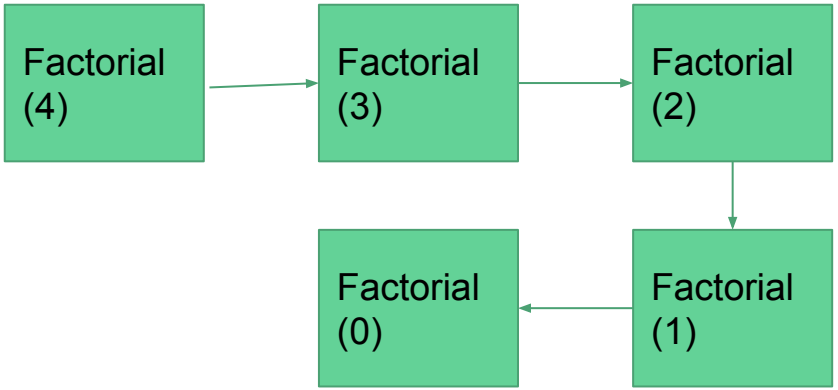


Factorial Function




```
int factorial(int n){

    if(n==0){

        return 1;

    }


    return n*factorial(n-1);

}
```

Factorial Function




```
int factorial(int n){

    if(n==0){

        return 1;

    }


    return n*factorial(n-1);

}
```

3 Steps to Solve Recursion Problems

1. Define Function
2. Recursive Call
3. Base Case

How to Solve a Recursion Problem?

1. Define the function

Function definition

- Parameters (input)
- Returns? (output)
- domain (assumptions on input)

How to Solve a Recursion Problem?

2. Recursive Call

Write f(n) = g(n)*f(n-1)

a. Write expression for f(n)
b. Write expression for f(n-1)
c. Write f(n) in terms of f(n-1)

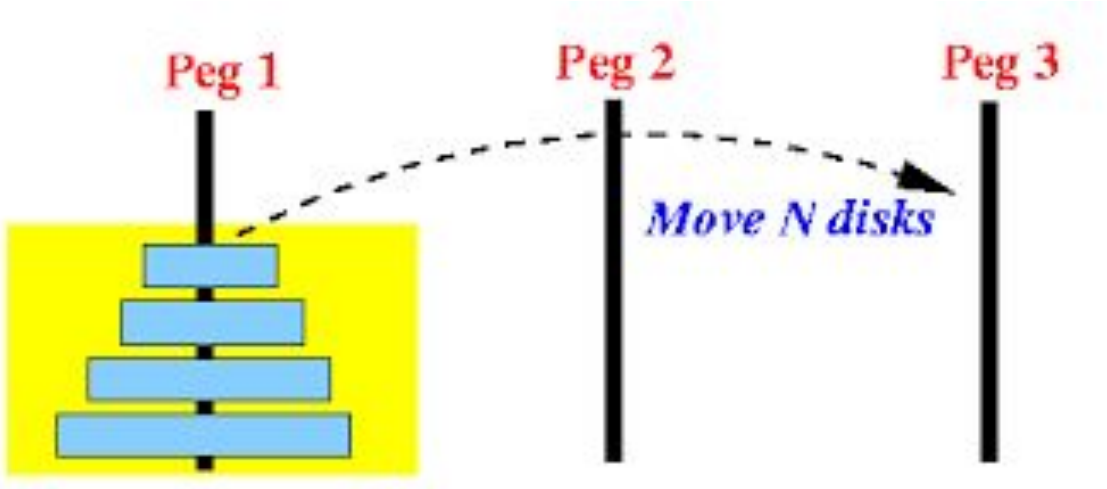
How to Solve a Recursion Problem?

3. Base Case

Generally, think of a border value of function's domain

- Why not take factorial(2) as base case?

Tower of Hanoi



1. A disk can only be placed on a larger disk
2. Only the topmost disk can be moved
3. In every step exactly one disk can be moved

Solution

1. Defining the function

```
int hanoi(
    int n,
    char source_peg,
    char dest_peg,
    char aux_peg,
)
```

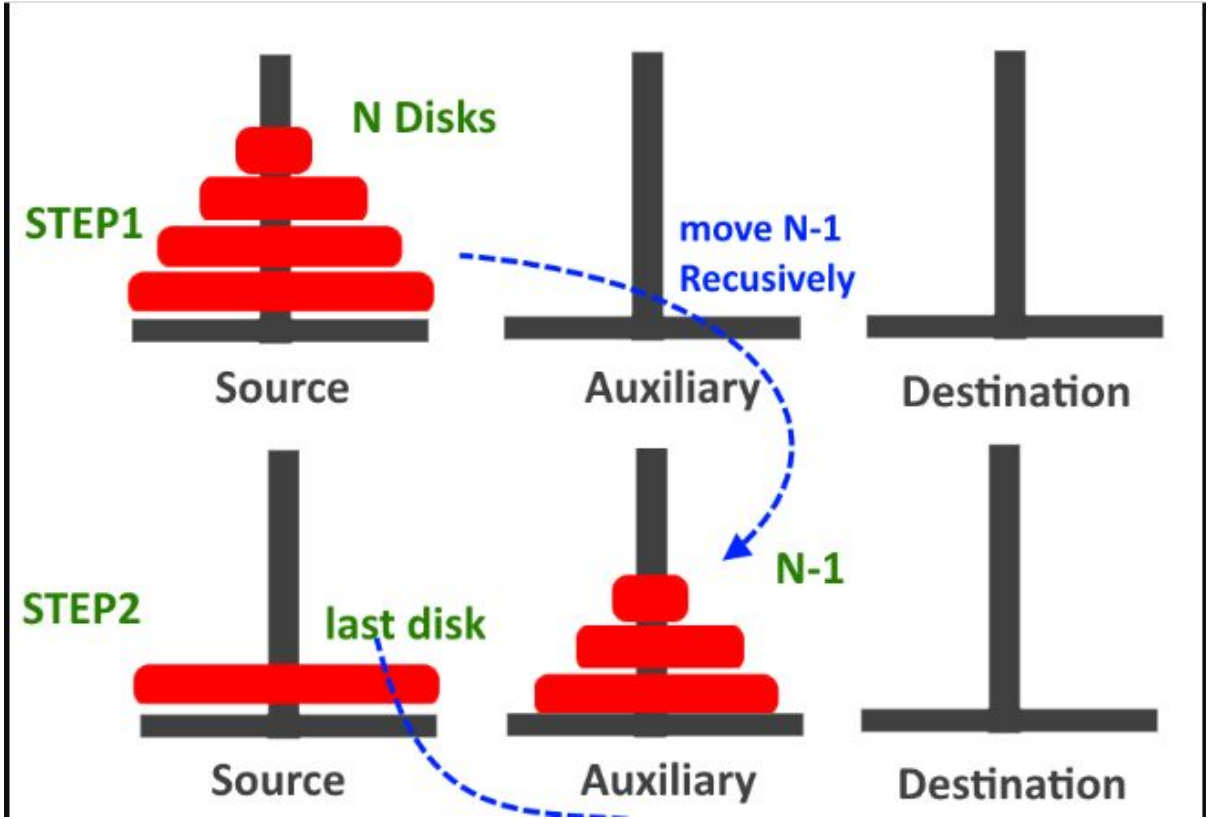
- Output = no. of steps (int)
- Input
  - int n
  - char source_peg
  - char dest_peg
  - Char aux_peg
- Domain
  - n>=1

2. Recursive Call

Part 1

Shift (n-1) disks to auxiliary peg

Steps += hanoi(n-1, source_peg, aux_peg, dest_peg)

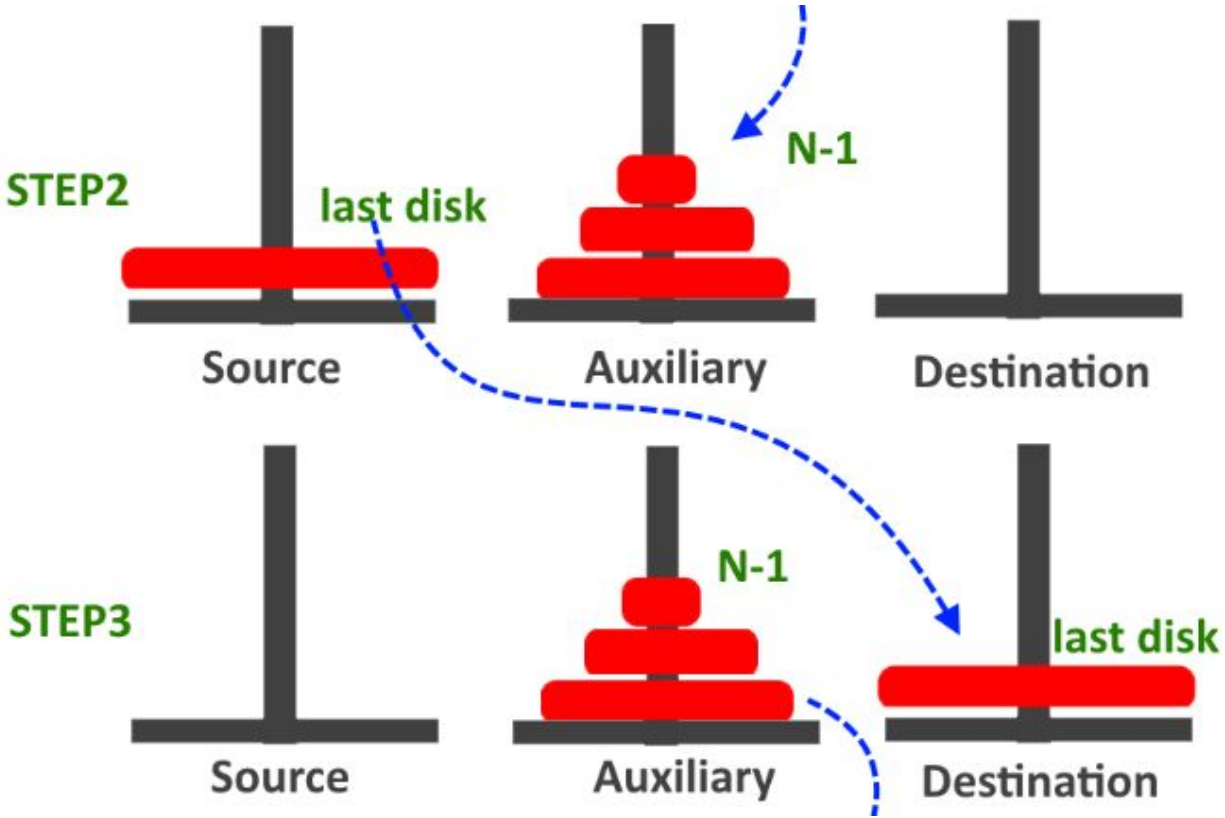


2. Recursive Call

Part 2

Shift largest disk to destination peg

Steps += 1

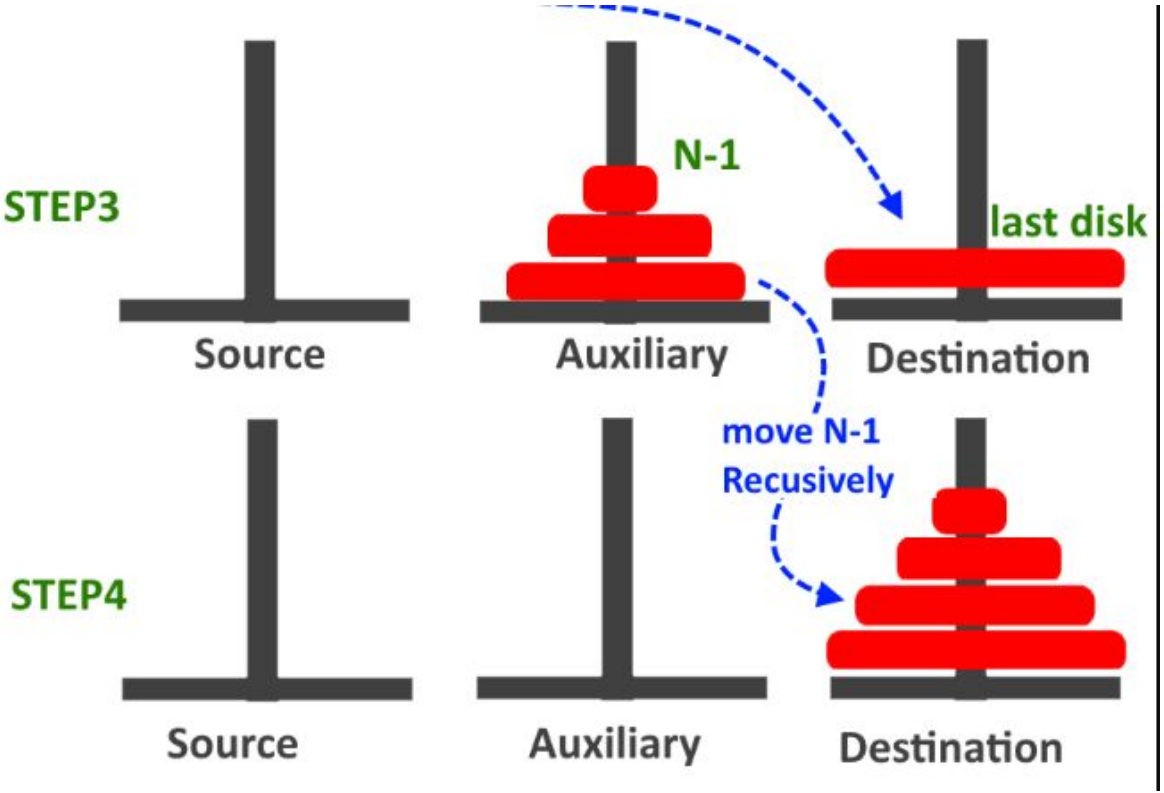


2. Recursive Call

Part 3

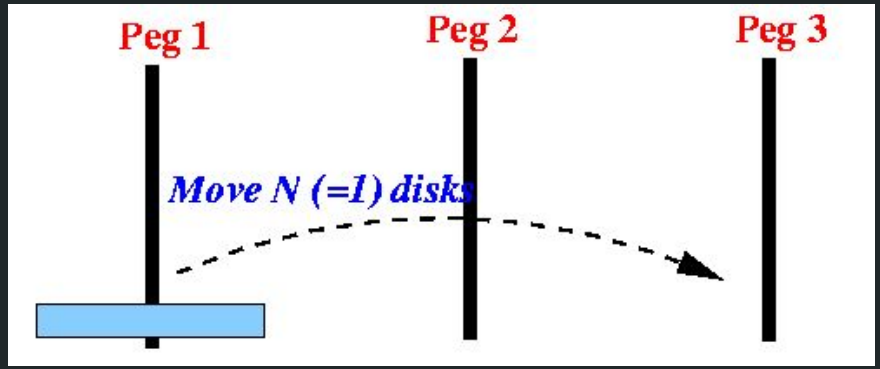
Shift (N-1) disks from auxiliary peg to destination peg

Steps += hanoi(n-1, aux_peg, dest_peg, source_peg)



3. Base Case

Thumbrule: Take border of the function domain




```
if (n==1){

    return 1;

}
```

Code

1.Function Definition

```
int hanoi(int n, char source_peg, char
dest_peg, char aux_peg){
```

2. Recursive Call

- **Part 1**
- **Part 2**
- **Part 3**

```
    if (n == 1){
        return 1;
    }
```

3. Base Case

```
   return hanoi(n-1, source_peg, aux_peg,
 dest_peg)
    + 1
    + hanoi(n-1, aux_peg, dest_peg,
 source_peg);

}
```

Pros of Recursion

1. Very intuitive
2. Compact Code Length

Cons of Recursion

1. Slower than an iterative solution
2. Stack overflow
3. Harder to debug

Thank you!

Questions?