Chapter: 6.2
Recursive Functions

Department of Electronics and Computer Engineering

Er. Pradip Khanal

Er. Bikash Acharya

Er. Bigyan Karki

ESTD. 2000

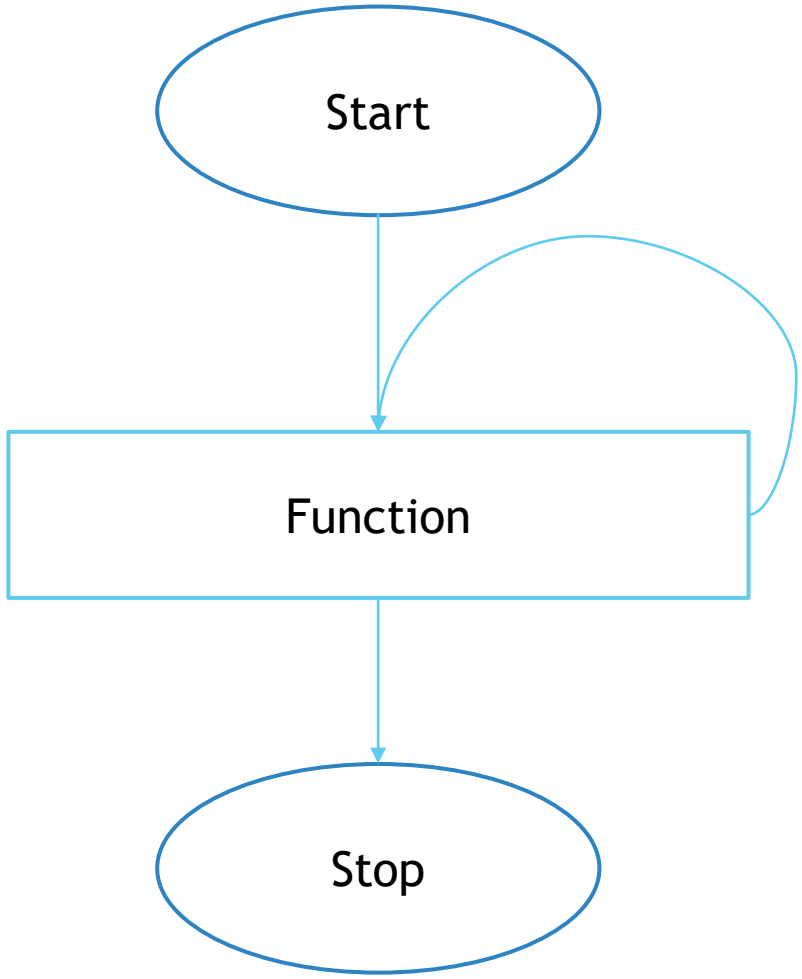
ADVANCED COLLEGE OF ENGINEERING & MANAGEMENT

Affiliated to Tribhuvan University (T.U.)

Introduction

- Recursion is the process of expressing a function in terms of itself.
- Function call itself unless some specific condition is satisfied i.e, base condition.
- The problem is solved by repeatedly breaking into smaller problems, which is similar in nature to the original problem.
- Each time a function calls itself and it must be closer to the solution.

Recursion



Principles of Recursion

- For implementing and designing the good recursive program we must follow certain conditions:
  - The problem must be written in recursive form and must be closer to solution
  - The problem statement must include a stopping condition called base condition.
- Base case is the terminating condition for the problem while designing any recursive algorithm.

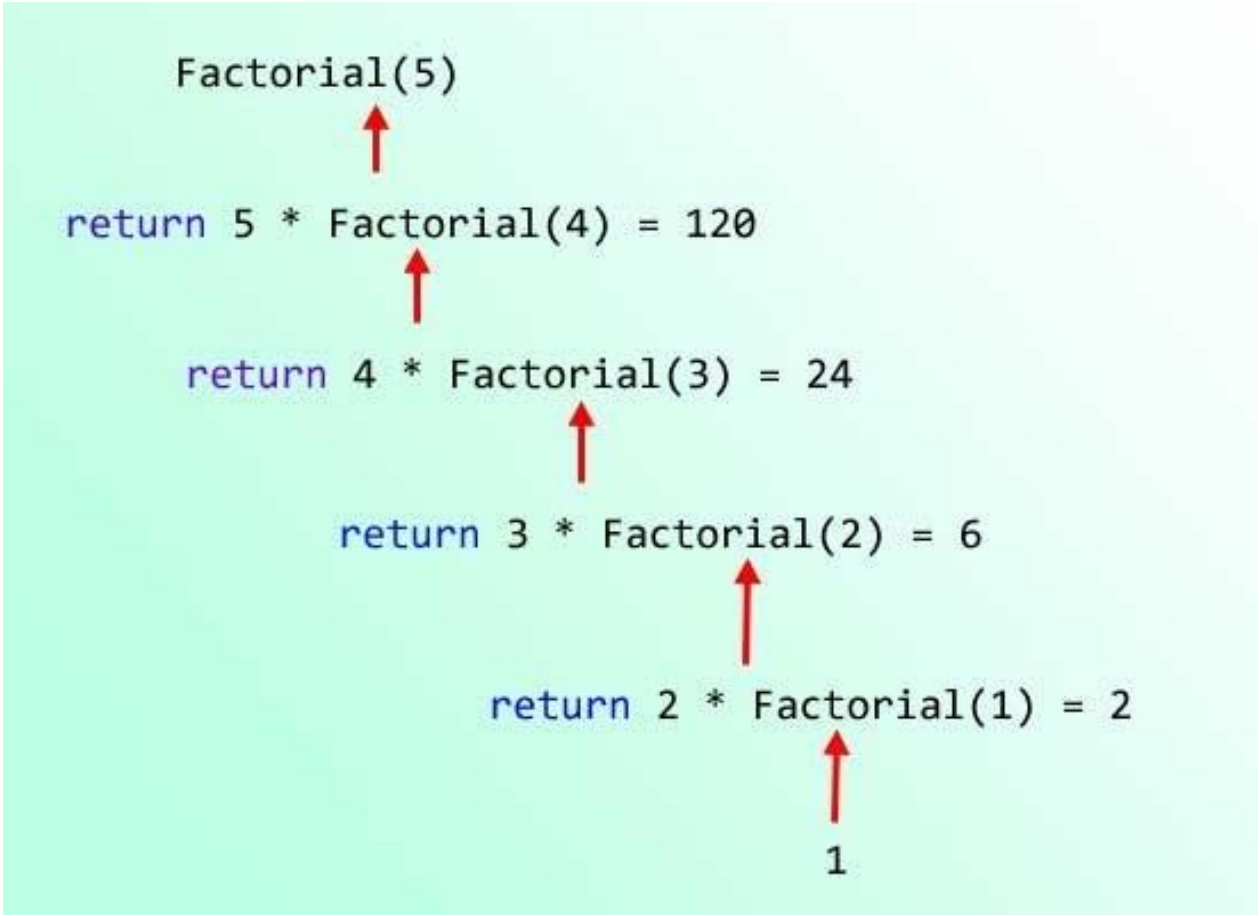
Example : Factorial [Main Function]

```
void main()
{
  int number;
  int f;
  printf("Enter a number: ");
  scanf("%d", &number);
  f = fact(number);
  printf("Factorial of %d is %d\n", number, fact);
  return 0;
}
```

Example: [Sub Function]

```
int fact (int n)
{
if(n==0 || n==1)
return 1; //base case
else
return (n * fact (n-1)) //recursive call
}
```

Tracing the Evaluation



Example 2: fibonacci Series [Main Function]

```
#include<stdio.h>
int fibo(int);
int main()
{
   int n, i = 0, c;
    scanf("%d",&n);
    printf("Fibonacci series\n");
   for ( c = 1 ; c <= n ; c++ )
   {
      printf("%d\n", fibo(i));
      i++;
   }
return 0;
}
```

Example: [Sub Function]

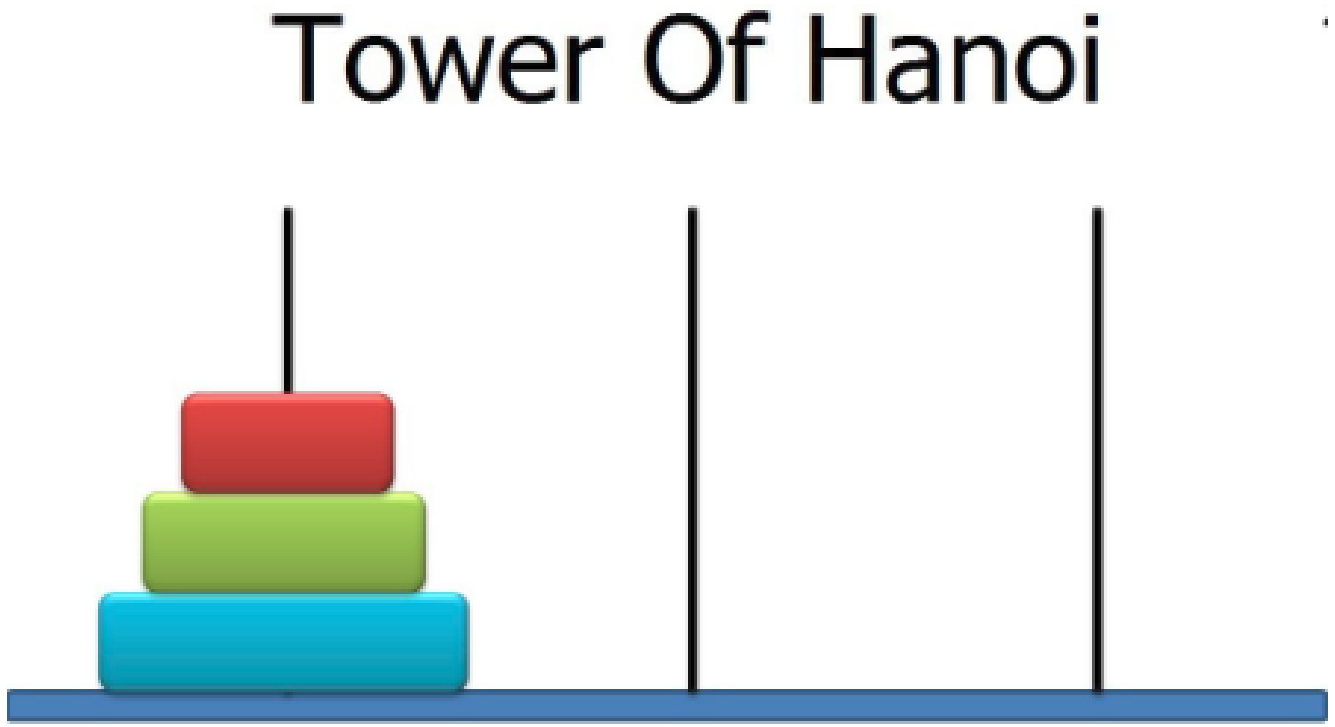
```
int fibo(int n)
{
   if ( n == 0 || n==1) return n;
else
      return (fibo(n-1) + fibo(n-2) );
}
```

Tracing the Evaluation

fibo(5) = fibo(4) + fibo (3)

= fibo(3) + fibo (2) + fibo(2) + fibo (1)
= fibo(2) + fibo (1) + fibo(1) + fibo (0) + fibo(1) + fibo (0) + fibo(1)
= fibo(1) + fibo (0) + fibo (1) + fibo(1) + fibo (0) + fibo(1) + fibo (0) + fibo(1)
= 1 + 0 + 1 + 1 + 0 + 1 + 0 + 1
= 5

Tower of Hanoi

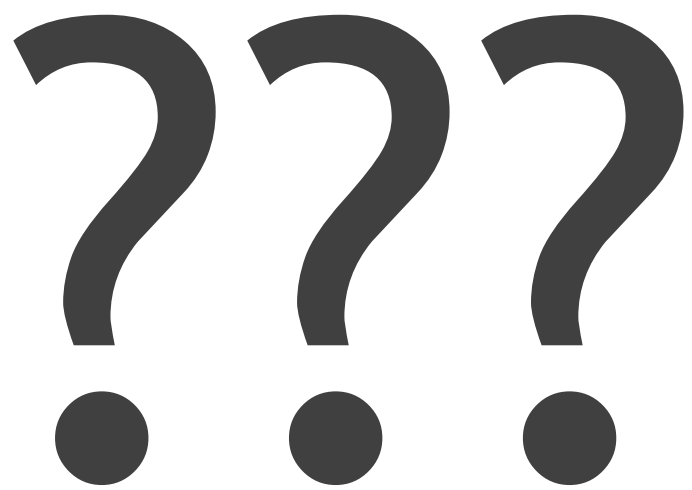


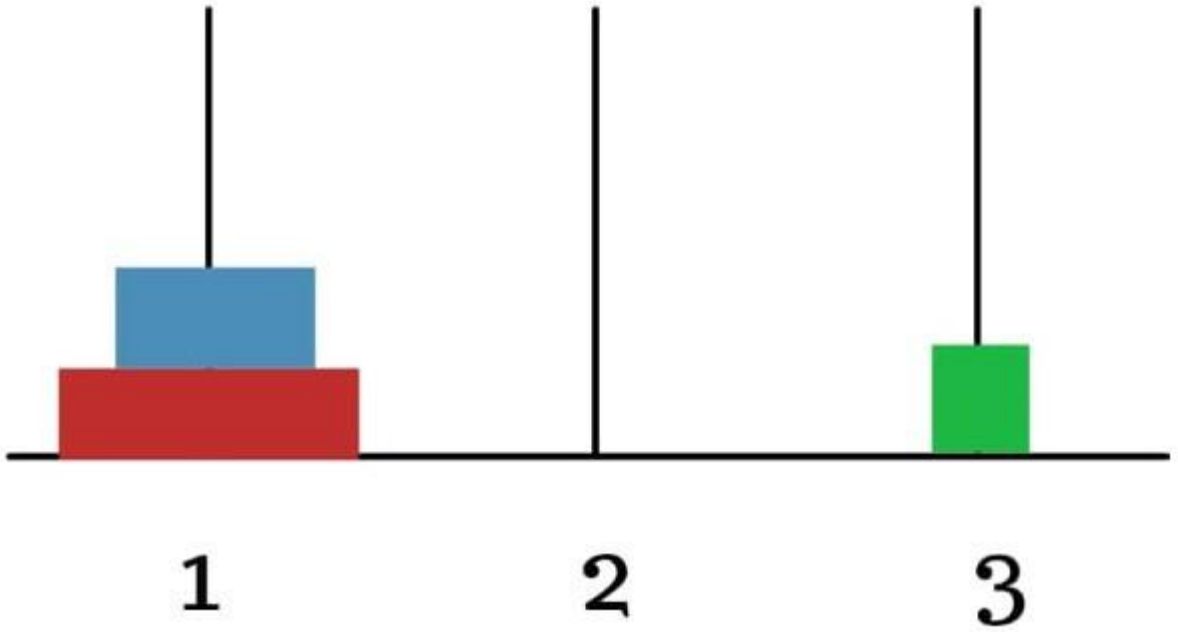
Rules

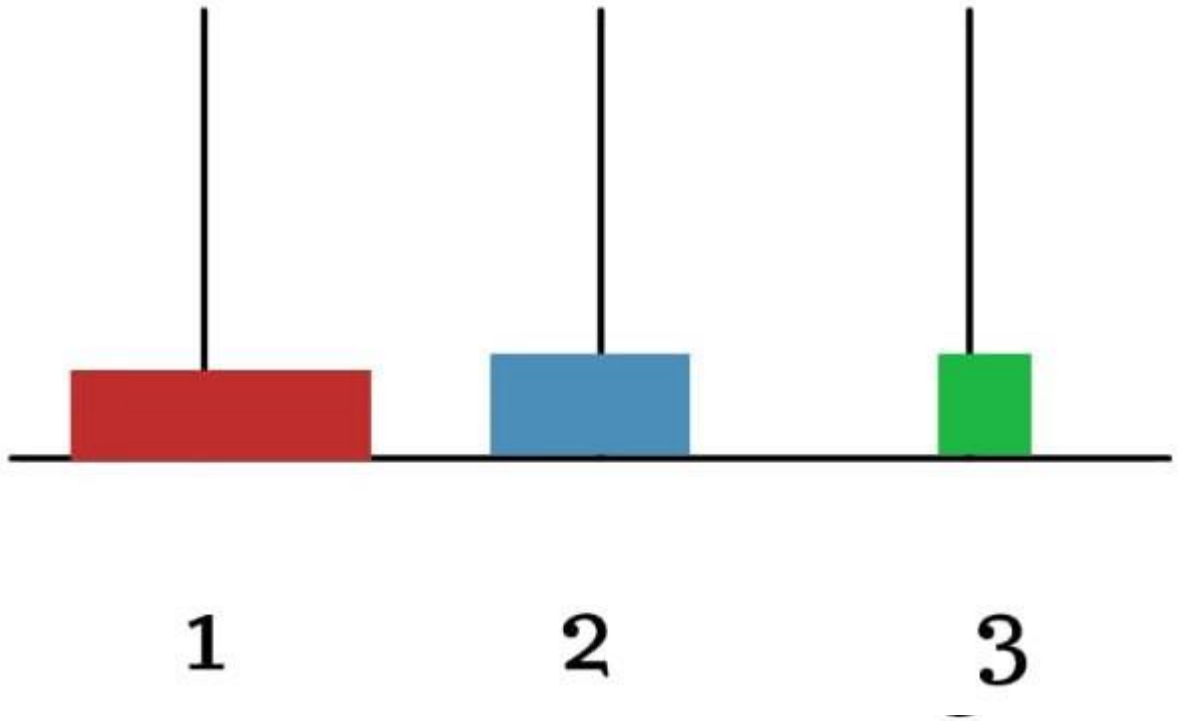
- The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

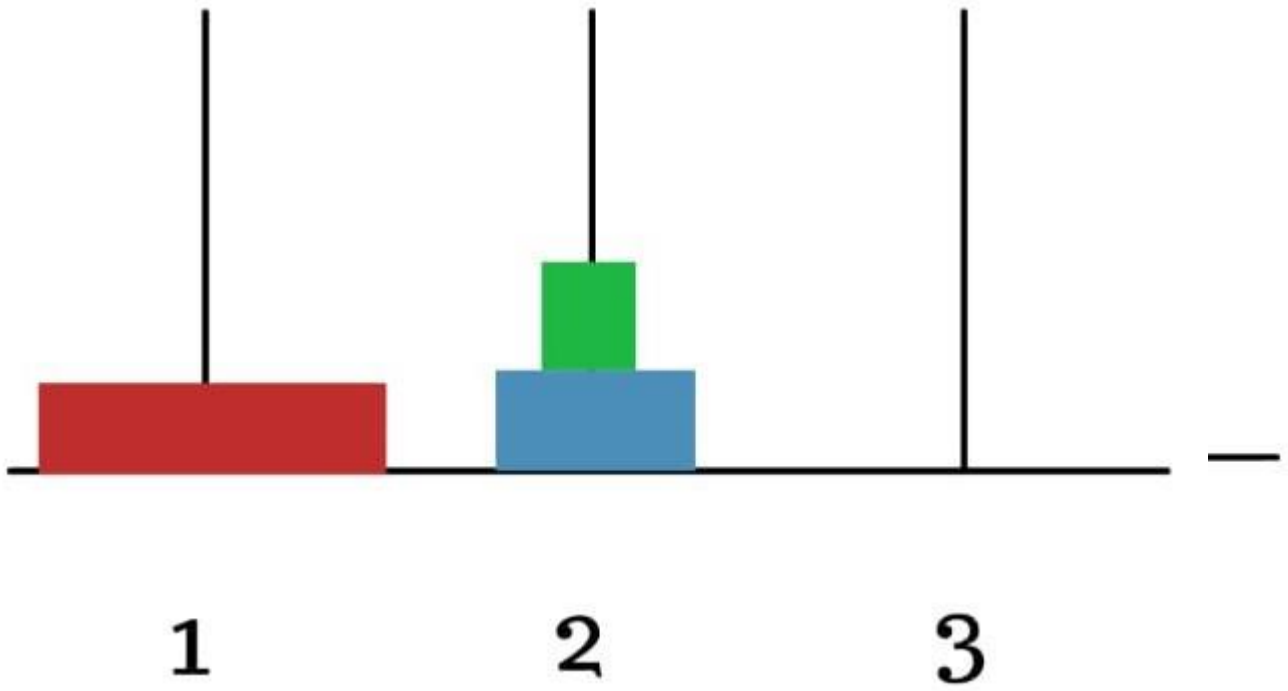
1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

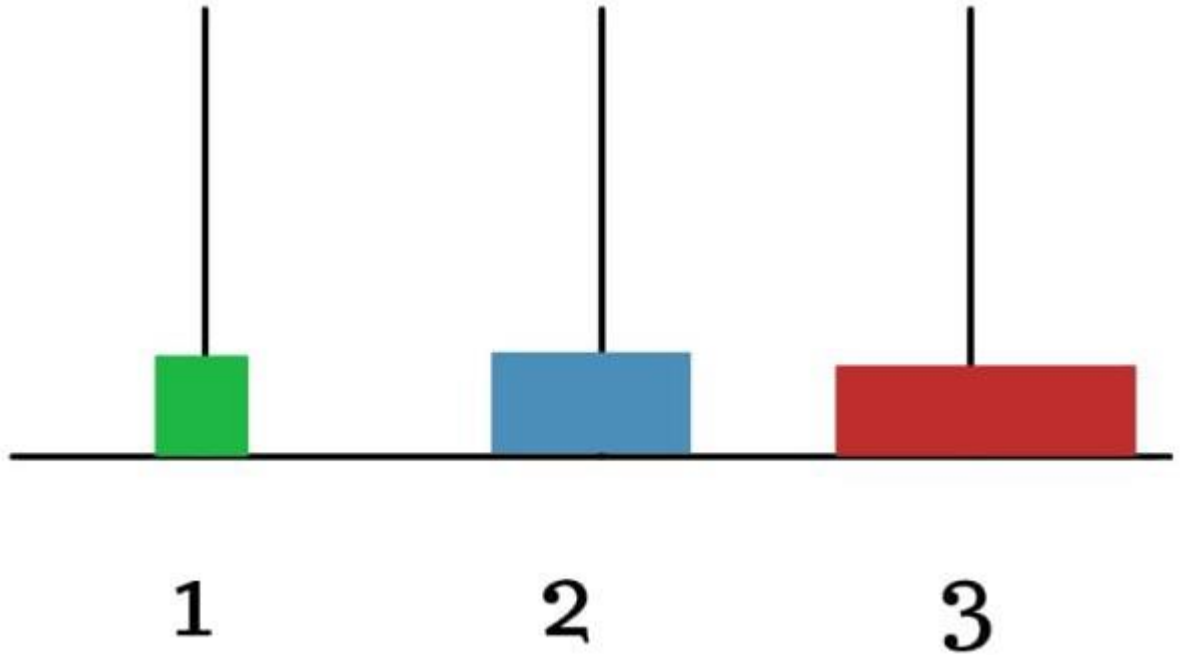
The minimal number of moves required to solve a Tower of Hanoi puzzle

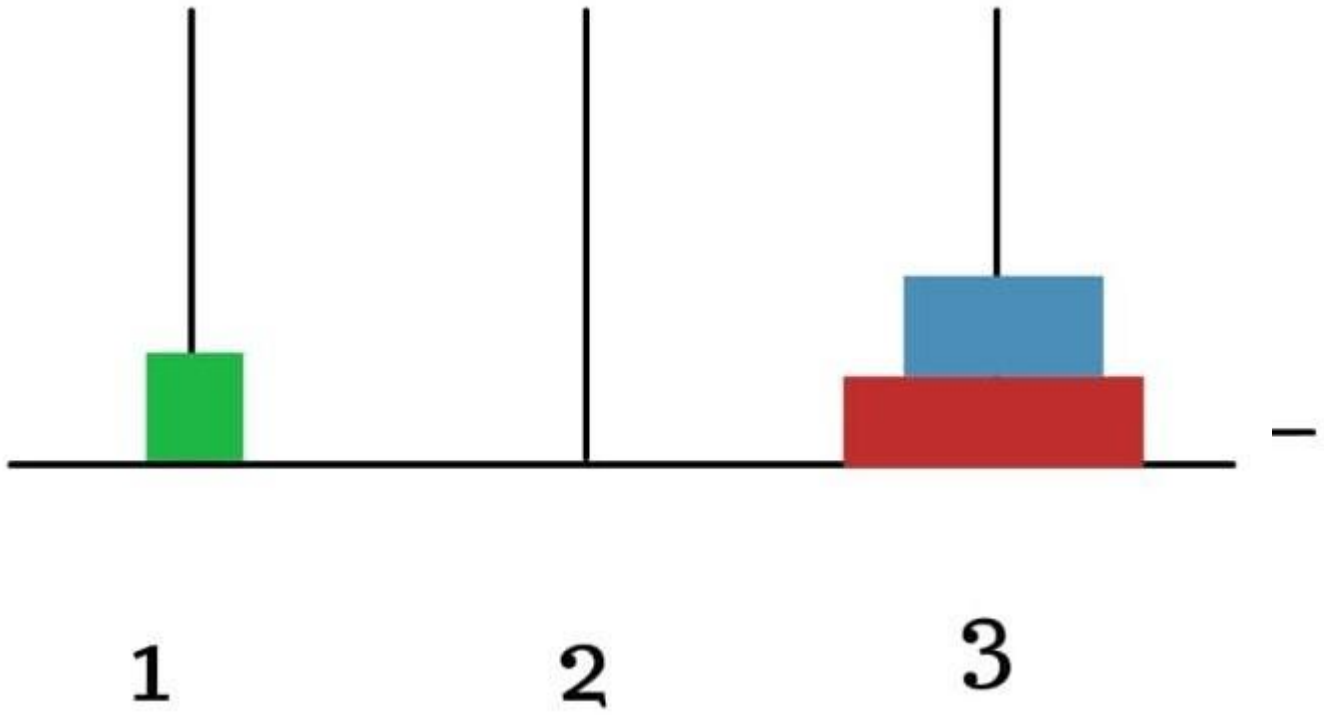


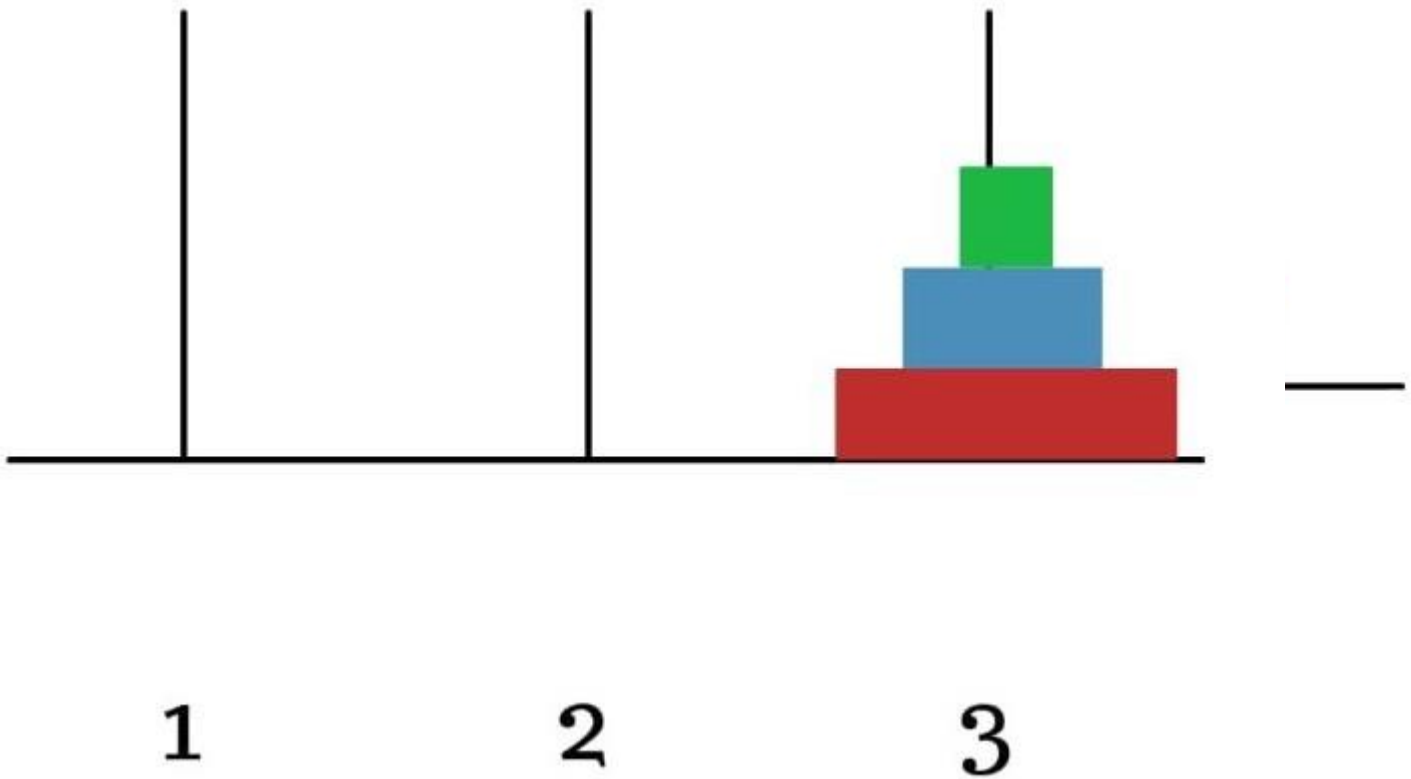

1
2
3


1
2
3


1
2
3


1
2
3


1
2
3


1
2
3

Advantage of Recursion

- Avoidance of unnecessary calling of function
- Reduce the size of code
- Easy to solve problem than iterative solution
- Length of the program can be reduced
- Extremely useful when applying same solution

Disadvantage of Recursion

- Use system stack
- Slow at execution time
- Occupies lot of memory space

ITERATION VS RECURSION

Criteria	Iteration	Recursion
Mode of implementation	Implemented using loops	Function calls itself
State	Defined by the control variable's value	Defined by the parameter values stored in stack
Progression	The value of control variable moves towards the value in condition	The function state converges towards the base case
Termination	Loop ends when control variable's value satisfies the condition	Recursion ends when base case becomes true
Code Size	Iterative code tends to be bigger in size	Recursion decrease the size of code
No Termination State	Infinite Loops uses CPU Cycles	Infinite Recursion may cause Stack Overflow error or it might crash the system
Execution	Execution is faster	Execution is slower

THE END