RECURSION

Recursion Example

- A **recursive algorithm** is one that invokes (makes reference to) itself repeatedly until a certain condition (also known as termination condition) matches

```
void recursion() {
   recursion(); /* function calls itself */
}

int main() {
   recursion();
}
```

Recursion VS Iteration

- An Iterative algorithm **will be faster** than the Recursive algorithm because of overheads like calling functions and registering stacks repeatedly.
- Recursive algorithm uses a **branching structure**, while iterative algorithm uses a **looping construct**.
- **Recursive algorithms** are not efficient as they take more space and time.
- Recursive algorithms are mostly used **to solve complicated problems** when their application is easy and effective.
- Some problems are **inherently recursive**, like tree traversal, quick sort, merge sort, etc.

Factorial – A case study

- The factorial of a positive number is the product of the integral values from 1 to the number:

$$n! = n*(n-1)*(n-2)……….3*\ 2*\ 1$$

$$4! = 4.3.2.1 = 24$$

- **Iterative Factorial Algorithm definition:**

$$\text{Factorial}(n) = \begin{bmatrix} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1 & \text{if } n > 0 \end{bmatrix}$$

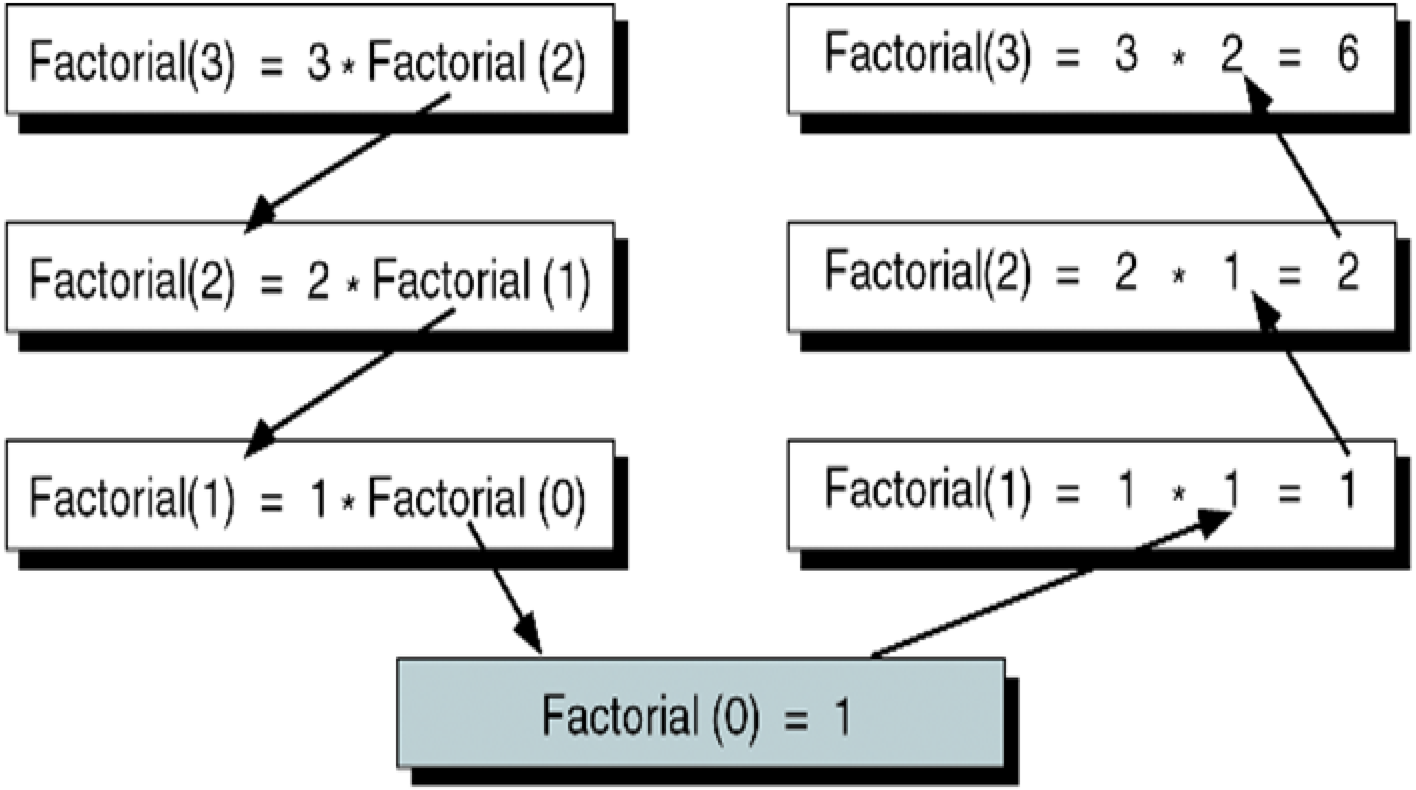
- **Recursive Factorial Algorithm definition:**

$$\text{Factorial}(n) = \begin{bmatrix} 1 & \text{if } n = 0 \\ n \times (\text{Factorial}(n-1)) & \text{if } n > 0 \end{bmatrix}$$

Factorial (3): Decomposition and solution

•The recursive solution for a problem involves a two-way journey:

- ➢First we decompose the problem from the top to the bottom
- ➢Then we solve the problem from the bottom to the top



ALGORITHM 2-1 Iterative Factorial Algorithm

```
Algorithm iterativeFactorial (n)
Calculates the factorial of a number using a loop.
   Pre  n  is the number to be raised factorially
   Post n! is returned
1 set i to 1
2 set factN to 1
3 loop (i <= n)
   1  set factN to factN * i
   2  increment i
4 end loop
5 return factN
end iterativeFactorial
```

ALGORITHM 2-2 Recursive Factorial

```
Algorithm recursiveFactorial (n)
Calculates factorial of a number using recursion.
   Pre    n  is the number being raised factorially
   Post   n! is returned
1 if (n equals 0)
   1  return 1
2 else
   1  return (n * recursiveFactorial (n - 1))
3 end if
end recursiveFactorial
```

Example-Problem GCD

- Determine the greatest common divisor (GCD) for two numbers.
- Euclidean algorithm: GCD (a,b) can be recursively found from the formula

$$GCD(a,b) = \begin{cases} a & \text{if } b=0 \\ b & \text{if } a=0 \\ GCD(b, a \bmod b) & \text{otherwise} \end{cases}$$

ALGORITHM 2-4 Euclidean Algorithm for Greatest Common Divisor

```
Algorithm gcd (a, b)
Calculates greatest common divisor using the Euclidean algo-
rithm.
   Pre  a and b are positive integers greater than 0
   Post greatest common divisor returned
1  if (b equals 0)
   1  return a
2  end if
3  if (a equals 0)
   2  return b
4  end if
5  return gcd (b, a mod b)
end gcd
```

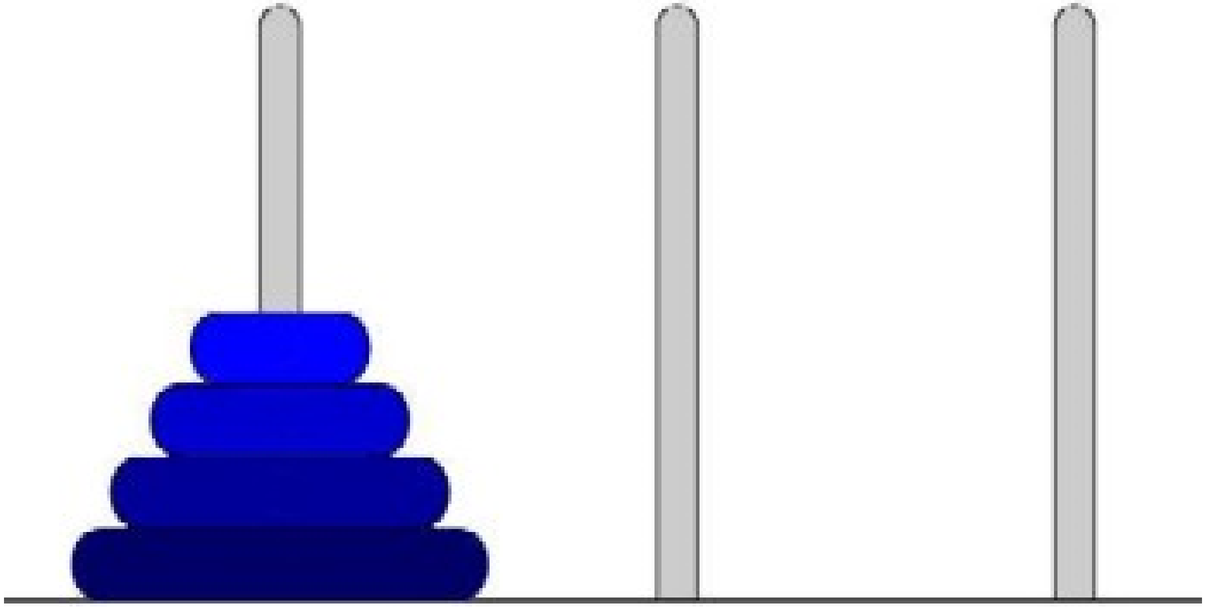
Example-Problem Fibonacci series

- Generation of the Fibonacci numbers series.
- Each next number is equal to the sum of the previous two numbers.
- A classical Fibonacci series is 0, 1, 1, 2, 3, 5, 8, 13, …
- The series of *n* numbers can be generated using a recursive formula

$$Fibonacci(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ Fibonacci(n-1) + Fibonacci(n-2) & \text{otherwise} \end{cases}$$

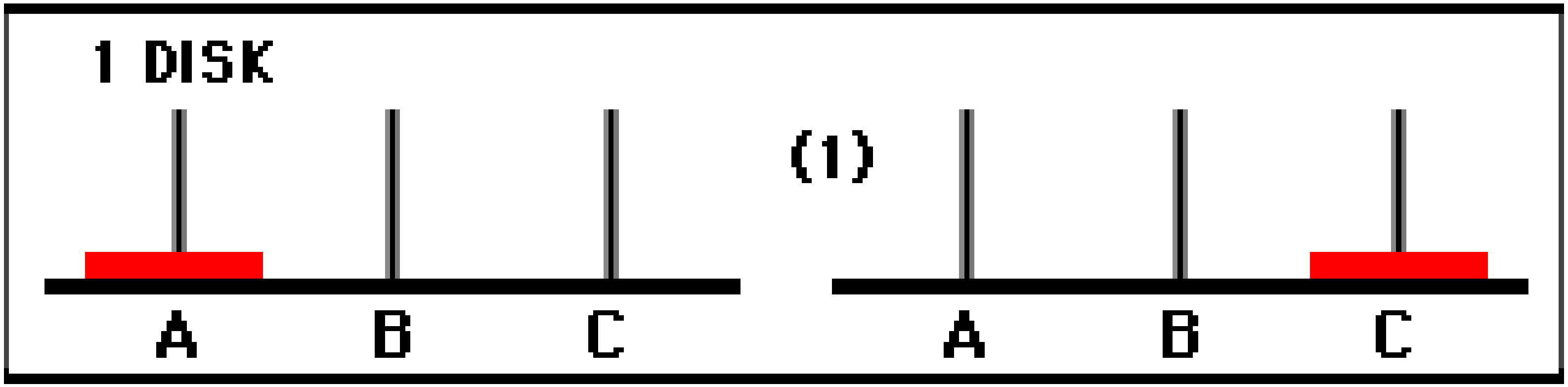
TOWERS OF HANOI

- Disks of different sizes (call the number of disks "n") are placed on the left-hand post,
- Arranged by size with the smallest on top.
- Your job is to transfer all the disks to the right-hand post in the fewest possible moves without ever placing a larger disk on a smaller one.



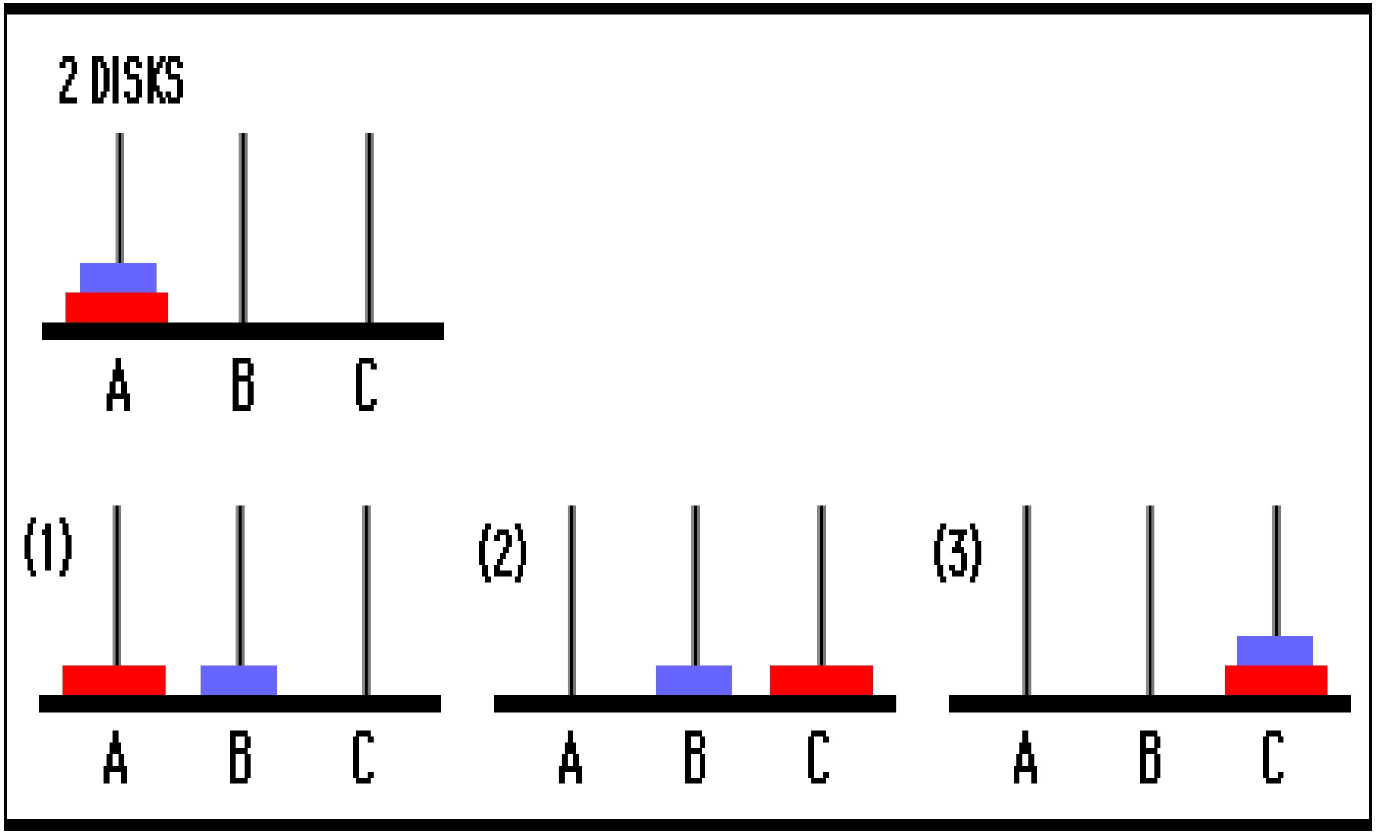
TOWERS OF HANOI

- **How many moves will it take to transfer n disks from the left post to the right post?**
- Let's look for a pattern in the number of steps it takes to move just one, two, or three disks. We'll number the disks starting with disk 1 on the bottom.**1 disk: 1 move**
- Move 1: move disk 1 to post C



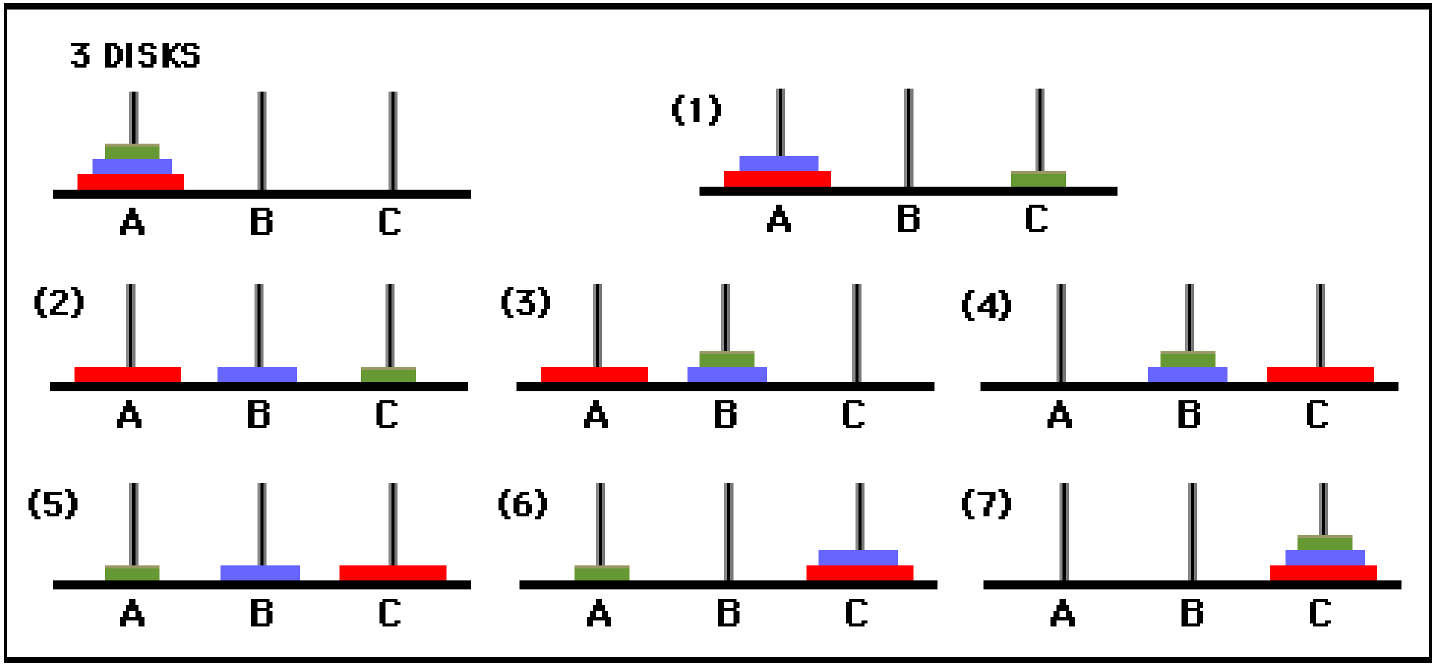
2 disks: 3 moves

Move 1: move disk 2 to post B
Move 2: move disk 1 to post C
Move 3: move disk 2 to post C



3 disks: 7 moves

Move 1: move disk 3 to post C
Move 2: move disk 2 to post B
Move 3: move disk 3 to post B
Move 4: move disk 1 to post C
Move 5: move disk 3 to post A
Move 6: move disk 2 to post C
Move 7: move disk 3 to post C



Example: Recursive solution to the Towers of Hanoi problem for N=3.

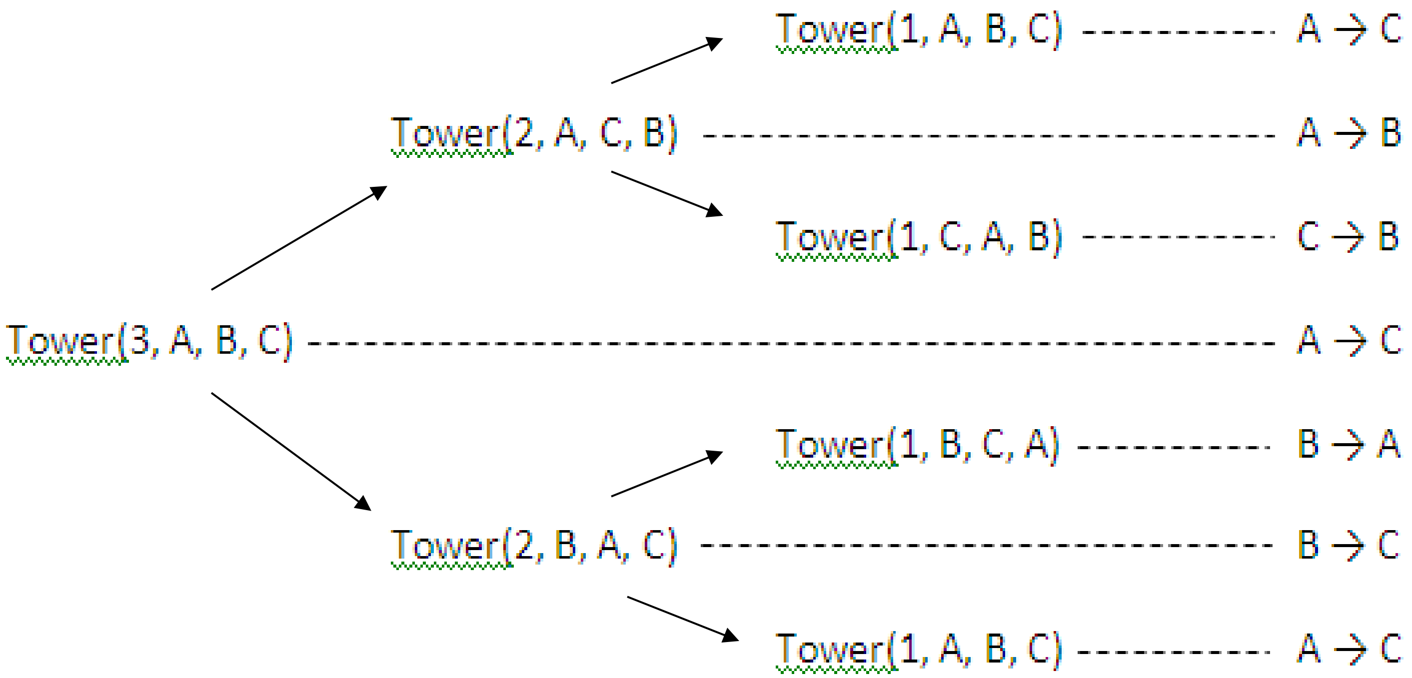


Figure: Recursive solution to the Towers of Hanoi problem for N=3.

<u>Assignment:</u> Write a recursive solution to the Towers of Hanoi for N=5.

TOWER(N, BEG, AUX, END)

This procedure gives a recursive solution to the Towers of Hanoi problem for N disks.

1. If N=1, then:
   (a) Write : BEG   END
   (b) Return.
   [End of If structure]
2. [Move N-1 disks from peg BEG to peg AUX]
   call TOWER(N-1, BEG, END,AUX)
3. Write : BEG   END
4. [Move N-1 disks from peg AUX to peg END]
   call TOWER(N-1, AUX, BEG, END)
5. Return.

Review Questions

- Write a recursive function to compute the sum of digits of a number.
- Write a recursive function to generate all subsets of a given set.

  Input: {1, 2, 3}

  Output: { {}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3} }
- Write a recursive function to print all permutations of a given string.

  Input: "ABC" Output: "ABC", "ACB", "BAC", "BCA", "CAB", "CBA”
- Given the Fibonacci numbers F_{11}= 89 and F_{12} = 144.
  - Implement a recursive function to compute F_{16} .
  - Implement an iterative function to compute F_{16} .
  - Compare recursion and iteration for computing Fibonacci numbers in terms of efficiency (time and space complexity).

Review Questions

Recursion

6.13 Let J and K be integers and suppose Q(J, K) is recursively defined by

$$Q(J, K) = \begin{cases} 5 & \text{if } J < K \\ Q(J - K, K + 2) + J & \text{if } J \geq K \end{cases}$$

Find Q(2, 7), Q(5, 3) and Q(15, 2)

6.14 Let A and B be nonnegative integers. Suppose a function GCD is recursively defined as follows:

$$GCD(A, B) = \begin{cases} GCD(B, A) & \text{if } A < B \\ A & \text{if } B = 0 \\ GCD(B, MOD(A, B)) & \text{otherwise} \end{cases}$$

(Here MOD(A, B), read "A modulo B," denotes the remainder when A is divided by B.) (a) Find GCD(6, 15), GCD(20, 28) and GCD(540, 168). (b) What does this function do?

6.15 Let N be an integer and suppose H(N) is recursively defined by

$$H(N) = \begin{cases} 3 * N & \text{if } N < 5 \\ 2 * H(N - 5) + 7 & \text{otherwise} \end{cases}$$

(a) Find the base criteria of H and **(b)** find H(2), H(8) and H(24).