INTRODUCTION TO DATA STRUCTURES & ALGORITHMS

OMAR AL-MUKHTAR UNIVERSITY

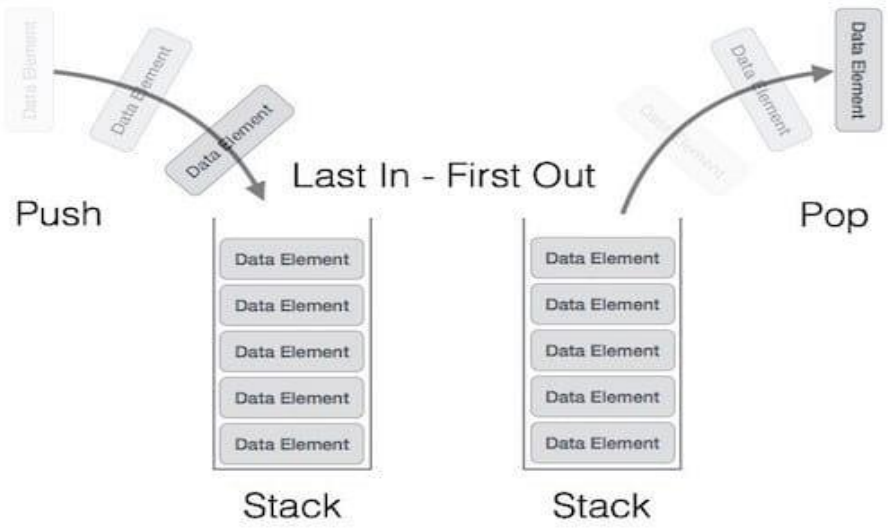
FACULTY OF SCIENCE

COMPUTER SCIENCE DEPARTMENT

LECTURE 2

THE STACK

- A stack is one of the most important and useful non-primitive linear data structure in computer science.
- It is an ordered collection of items into which new data items may be added or deleted at only one end, called the top of the stack.

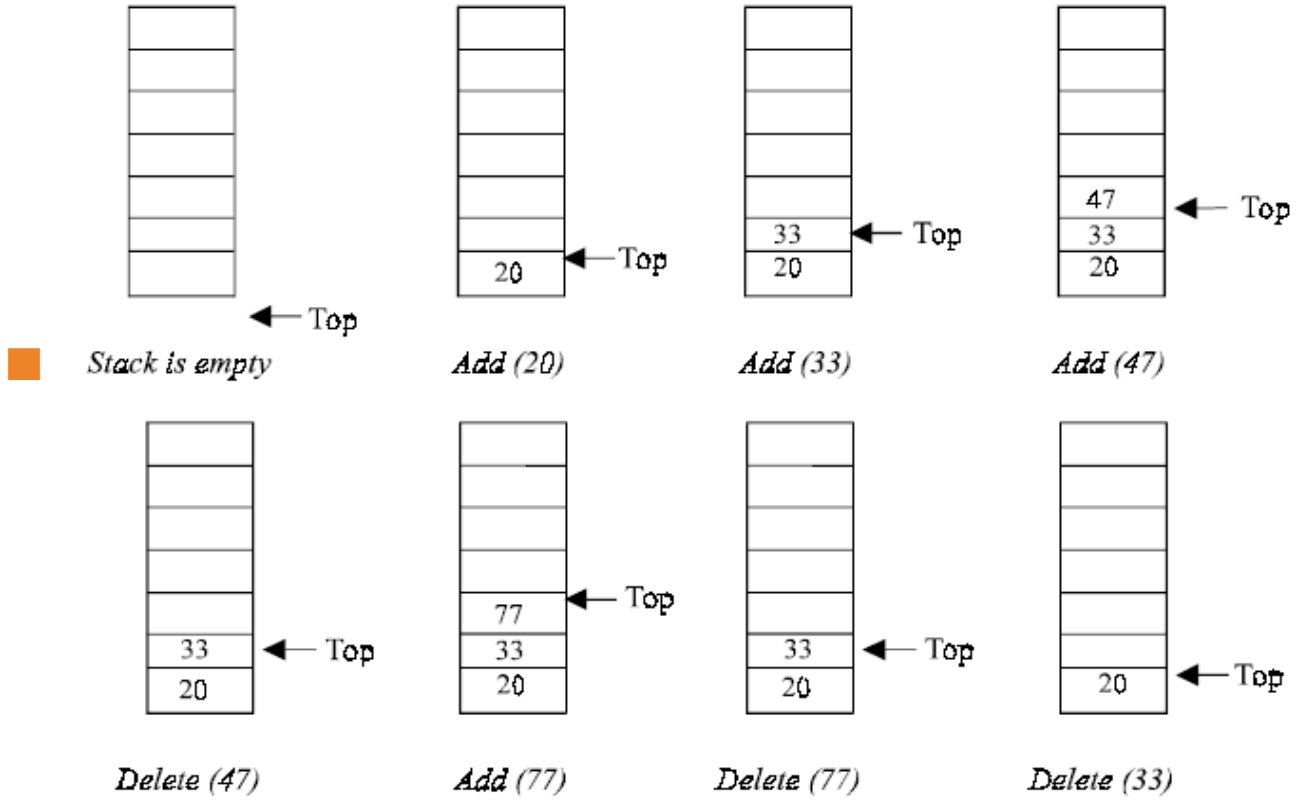


STACK ADT

- A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only.

- At any given time, we can only access the top element of a stack.
- That is why the stack is also called Last-in-First-out (LIFO).

STACK BASIC OPERATIONS



Stack operation.

BASIC OPERATIONS PERFORMED ON STACK

- A stack is used for the following two primary operations:

- push() − Pushing (adding) a new element to the top of the stack. After every push operation the top is *incremented* by one.

- pop() − Removing (deleting) an element from the top of the stack. After every pop operation the top stack is *decremented* by one.

- *A stack is said to be *empty* or *underflow*, if the stack contains no elements. And it is *overflow* when the stack becomes *full*, i.e., no other elements can be pushed onto the stack.

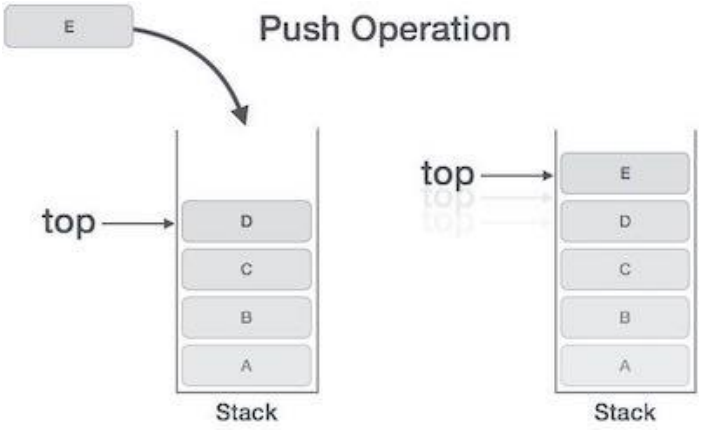
STACK IMPLEMENTATION

- A Stack can be implemented in two ways:
  - 1. Static implementation (using arrays).
  - 2. Dynamic implementation (using pointers).

- Static implementation using arrays is a very simple technique but is not a flexible way , the size of the stack has to be defined during the program design, because after that, the size cannot be varied.

STACK USING ARRAYS

- - Push Operation Steps:
  - Step 1 . Checks if the stack is full.
  - Step 2 . If the stack is full, error and exit.
  - Step 3 . If the stack is not full, increments top to point next empty space.
  - Step 4 . Adds data element to the stack location, where top index is.
  - Step 5 . Exit.



STACK USING ARRAYS

- Algorithm for push:

➢ Assume STACK [SIZE] is a one dimensional array which will hold the data items.

➢ TOP is the pointer that points to the top most element of the stack. DATA is the data item to be pushed.

1) If TOP = SIZE – 1, then:

- (a) Display “The stack is in overflow condition”
- (b) Exit

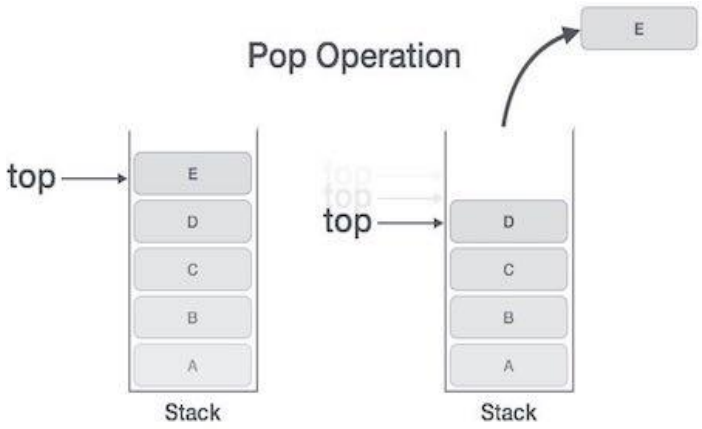
2) TOP = TOP + 1

3) STACK [TOP] = ITEM

4) Exit

STACK USING ARRAYS

- - Pop Operation Steps:
  - Step 1 . Checks if the stack is empty.
  - Step 2 . If the stack is empty, error and exit.
  - Step 3 . If the stack is not empty, accesses the data element at which top is.
  - Step 4 . Decreases the value of top by 1.
  - Step 5 . Exit.



STACK USING ARRAYS

- Algorithm for pop:

➢ Assume STACK [SIZE] is a one dimensional array which will hold the data items.

➢ TOP is the pointer that points to the top most element of the stack. DATA is the popped data item from the top of the stack.

1) If TOP < 0, then
   - (a) Display "The Stack is empty"
   - (b) Exit
2) Else remove the Top most element
3) DATA = STACK[TOP]
4) TOP = TOP – 1
5) Exit

APPLICATIONS OF STACKS

- Applications related to computers:

1. “Back” button of Web Browser - Page-visited history in a Web browser
2. “Undo” functionality of a text editor.
3. Reversing the order of elements in an array
4. Saving local variables when one function calls another, and this one calls another, and so on.

APPLICATIONS OF STACKS

- 1) RECURSION:
- Recursion occurs when a function is called by itself repeatedly; the function is called recursive function.

■ The general algorithm model for any recursive function contains the following steps:

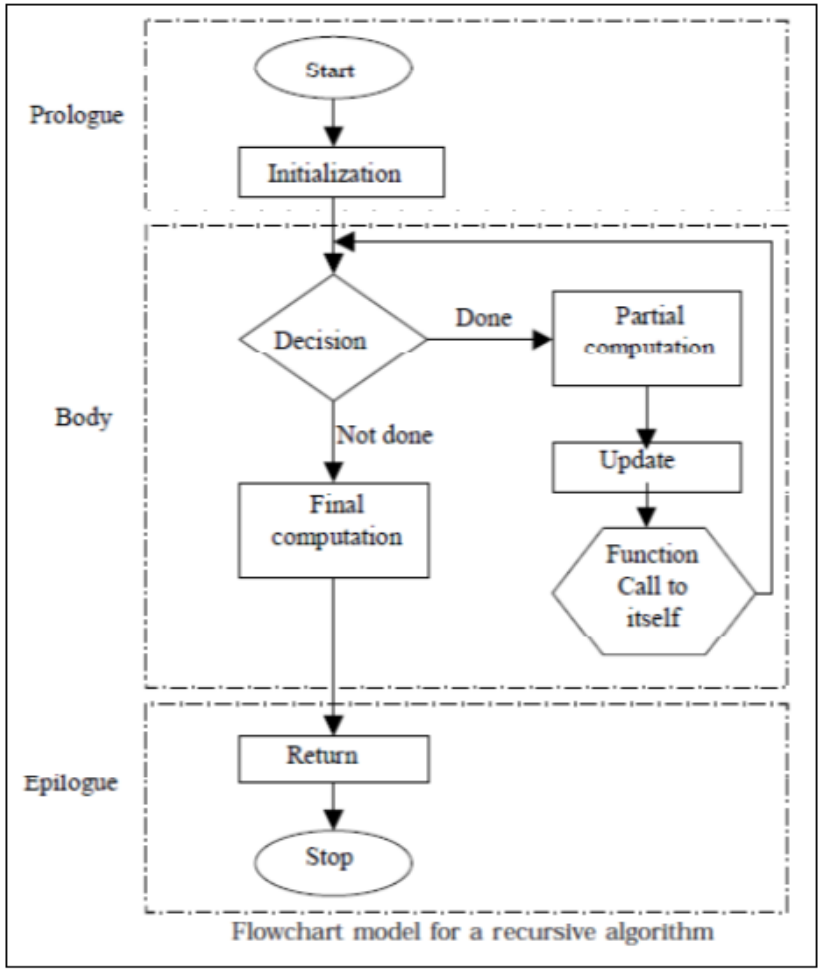
1. Prologue: Save the parameters, local variables, and return address.

2. Body: If the base criterion has been reached, then perform the final computation and go to **step 3** ; otherwise, perform the partial computation and go to **step 1** (initiate a recursive call).

3. Epilogue: Restore the most recently saved parameters, local variables, and return address.

RECURSION ALGORITHM

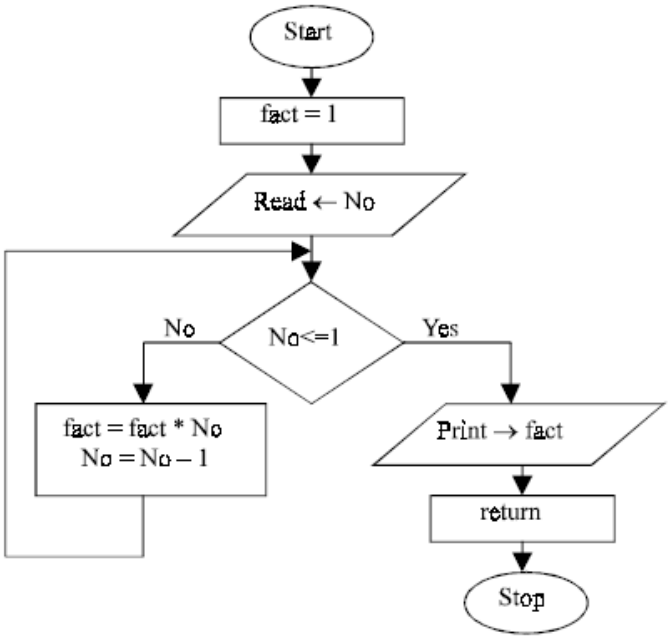
- The Last-in-First-Out characteristics of a recursive function points that the stack is the most obvious data structure to implement the recursive function.



Flowchart model for a recursive algorithm

RECURSION: FACTORIAL NUMBER

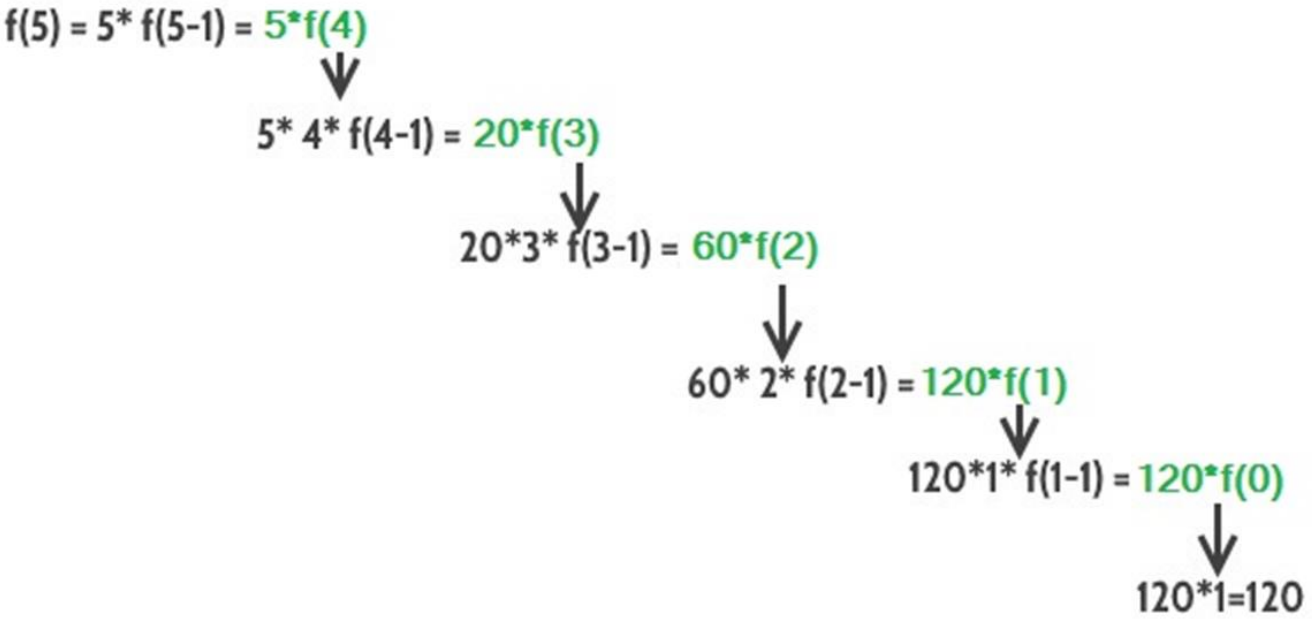
- The recursive mechanism can be best described by an example.
- Consider the following flowchart is to calculate factorial of a number recursively.



RECURSION: FACTORIAL NUMBER

Factorial function: f(n) = n*f(n-1)

Lets say we want to find out the factorial of 5 which means n =5



RECURSION X ITERATION

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
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

DISADVANTAGES OF RECURSION

1. It consumes more storage space because the recursive calls.
2. The computer may run out of memory if the recursive calls are not checked.
3. It is not more efficient in terms of speed and execution time.
4. Recursion is not advocated when the problem can be solved through iteration.

APPLICATIONS OF STACKS

- 2) TOWER OF HANOI:
- The Tower of Hanoi problem can use recursive technique to produce a logical and elegant solution.

- **Problem Definition** :

➢ We have to transfer all the disks from source peg X to the destination peg Z by

➢ using an intermediate peg Y taking into account the following conditions:

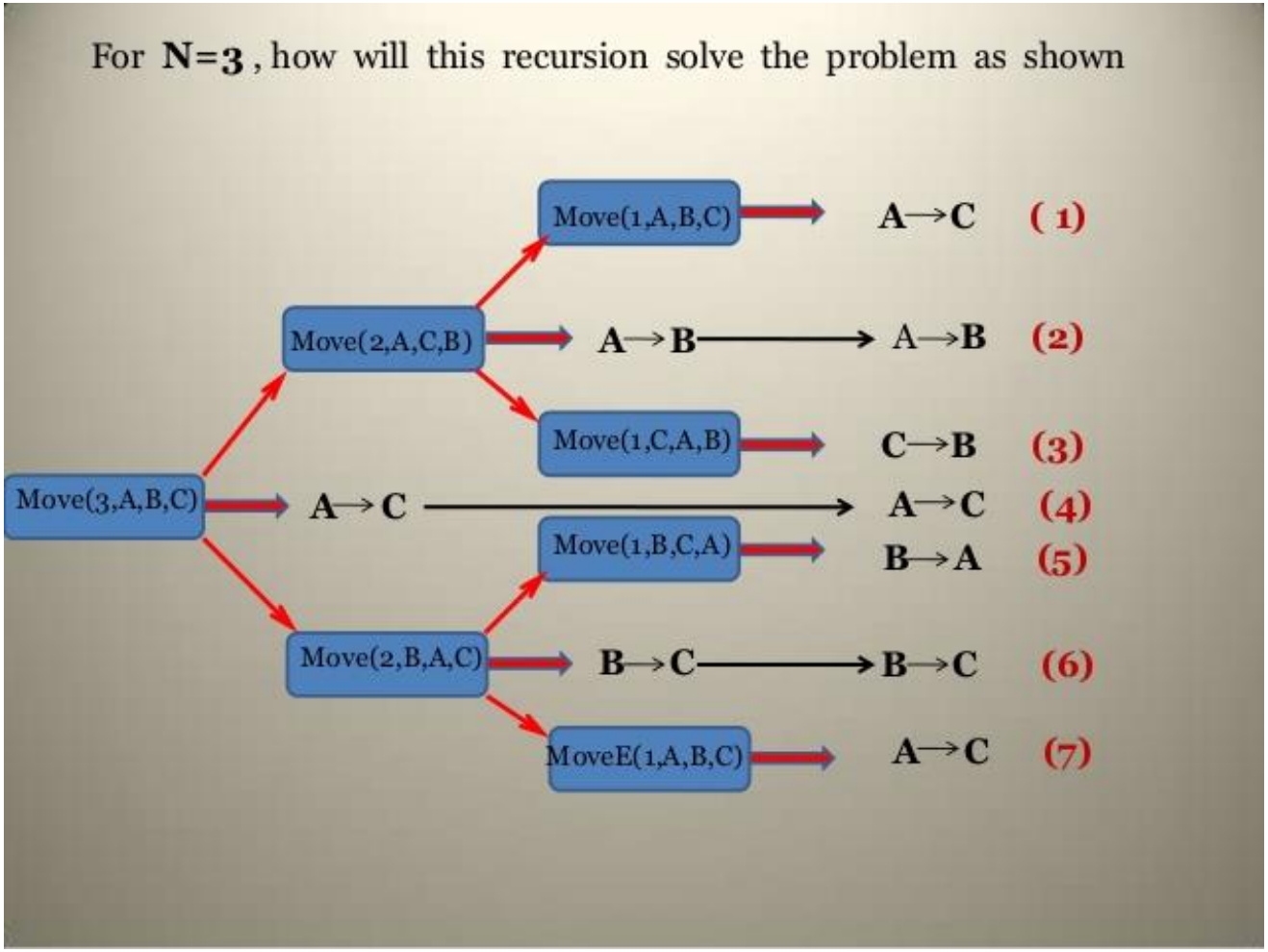
1. Transferring the disks from the source peg to the destination peg such that at any point of transformation no large size disk is placed on the smaller one.
2. Only one disk may be moved at a time.
3. Each disk must be stacked on any one of the pegs.

TOWER OF HANOI

- The Question Now is !?
- **How many moves does it take to solve the Tower of Hanoi puzzle??**
- *The Answer is,,*
  - "The number of moves needed to solve the Tower of Hanoi problem depends on the number of >> Disks<<"

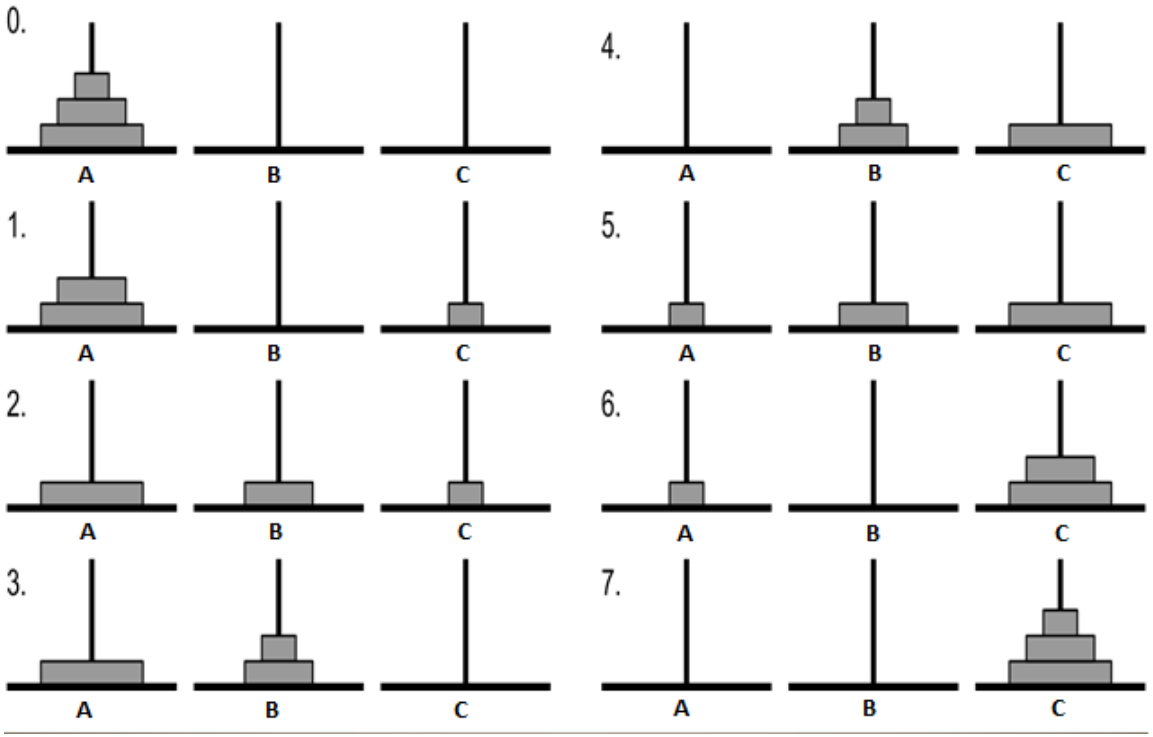
- * For example suppose we have 3 disks >> N=3 and the 3 pegs are A,B,C. How many moves it take??

TOWER OF HANOI



TOWER OF HANOI

- Now Tower of Hanoi problem can be solved as shown below:



TOWER OF HANOI

- The problem of "Tower of Hanoi" can be solved recursively as follows :
- To move all disks from peg X to peg Z, using Y as auxiliary peg:

1. If n = 1, move the single disk from X to Z and stop.
2. Move the top(n – 1) disks from the peg X to the peg Y, using Z as auxiliary.
3. Move nth disk to peg Z.
4. Now move n – 1 disk from Y to Z, using A as auxiliary.