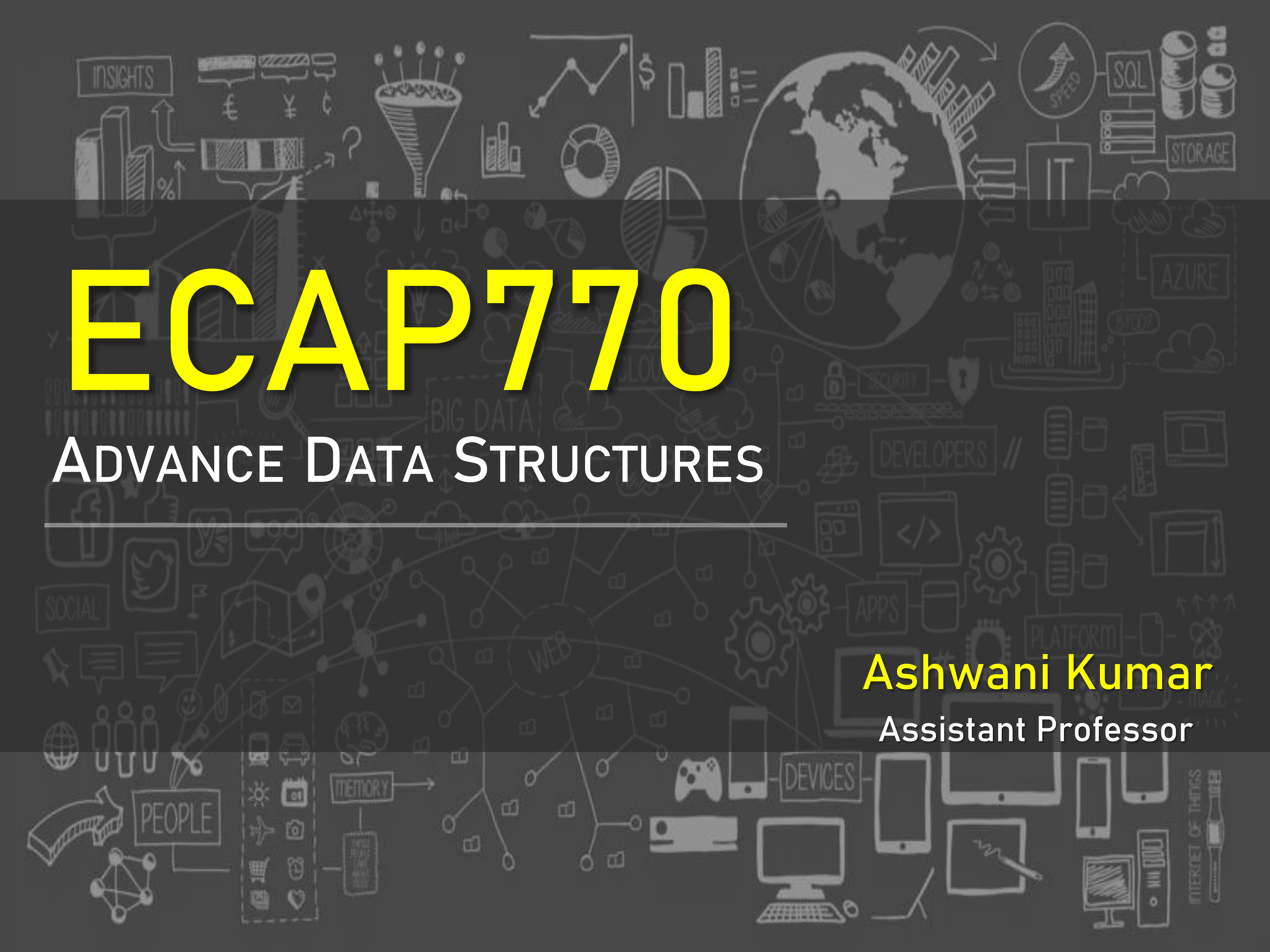


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

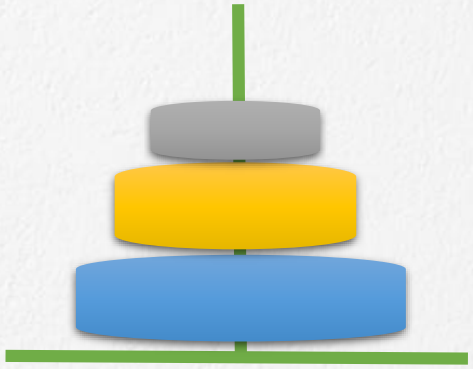
Learning Outcomes

After this lecture, you will be able to

- Understand tower of hanoi problem
- arithmetic expression conversion and evaluation

Tower of Hanoi

- The Tower of Hanoi, is a mathematical problem which consists of three rods and multiple disks.
- Initially, all the disks are placed on one rod, one over the other in ascending order of size similar to a cone-shaped tower.

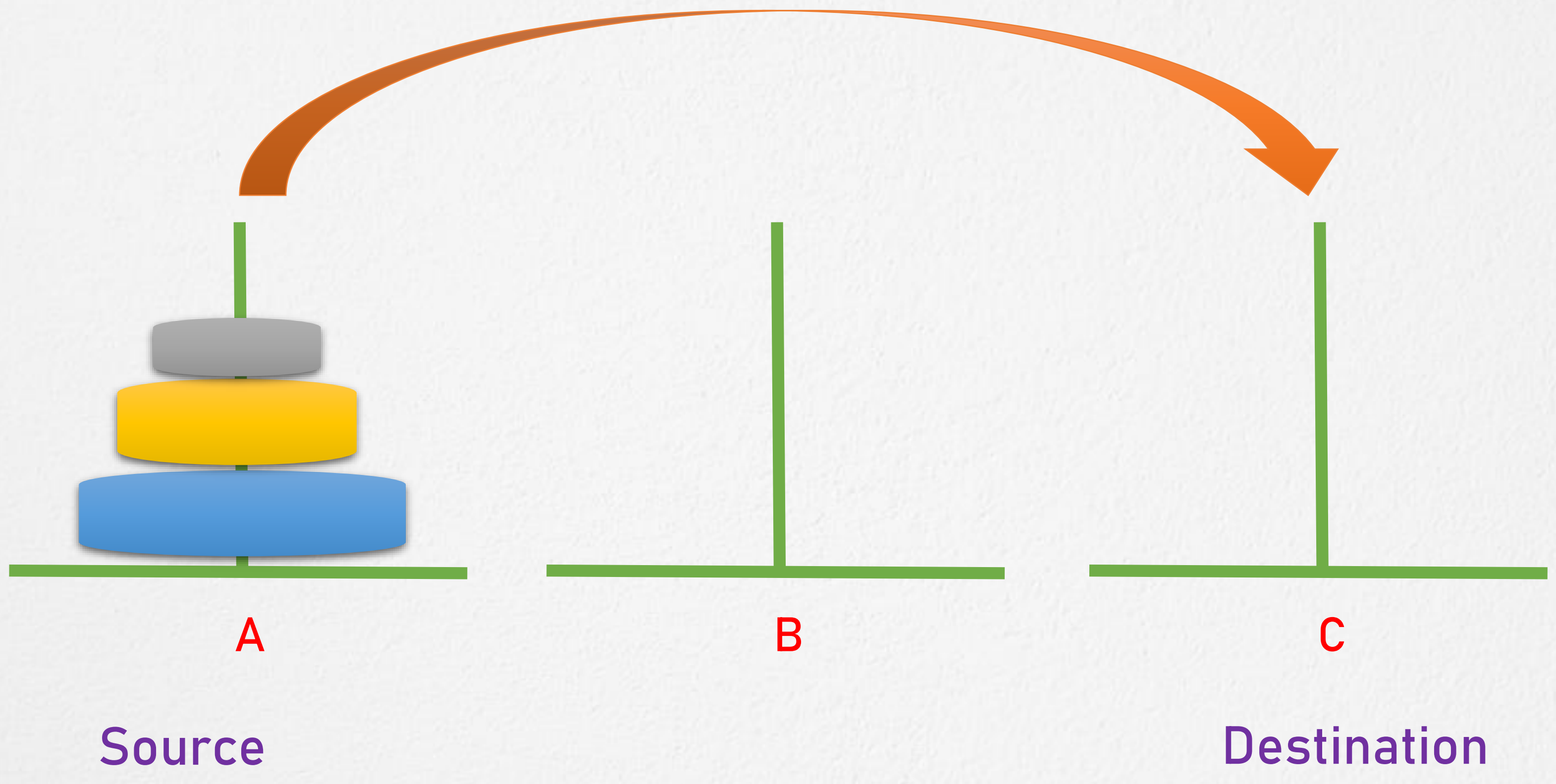


Tower of Hanoi

The objective of this problem is to move the stack of disks from the source to destination, following these rules:

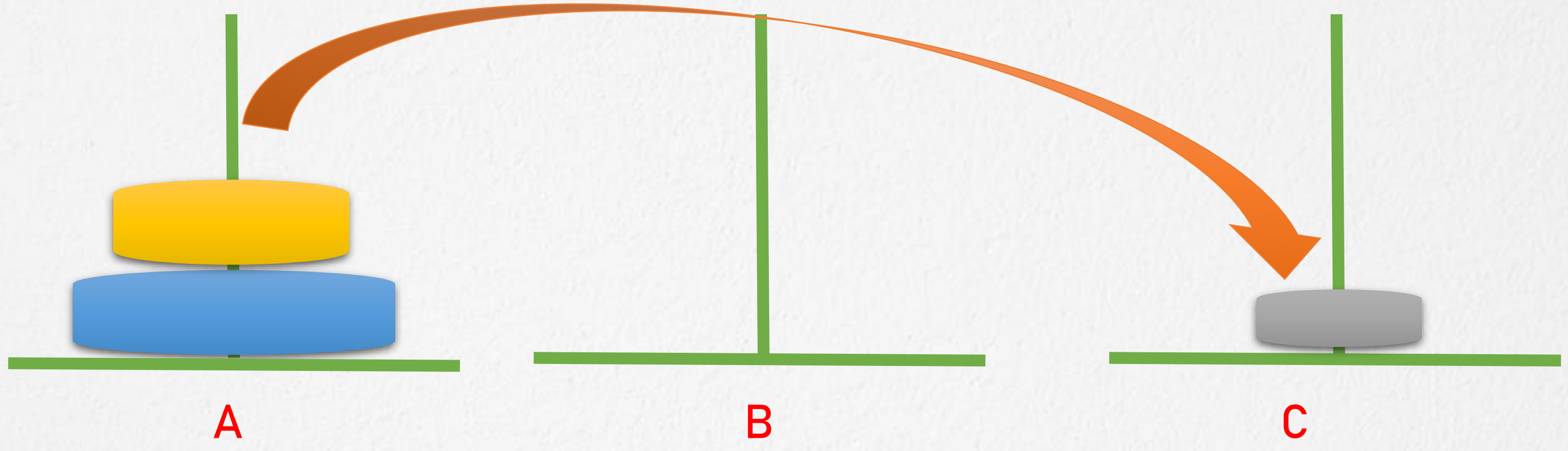
- Only one disk can be moved at a time.
- Only the top disk can be removed.
- No large disk can sit over a small disk.

Tower of Hanoi

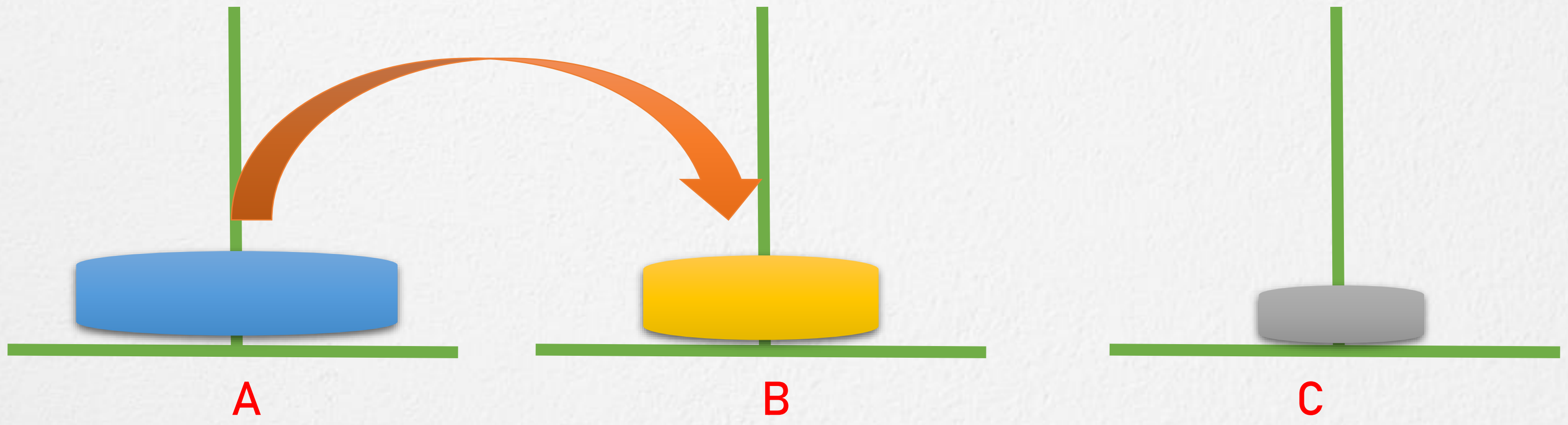


Tower of Hanoi

Step 1

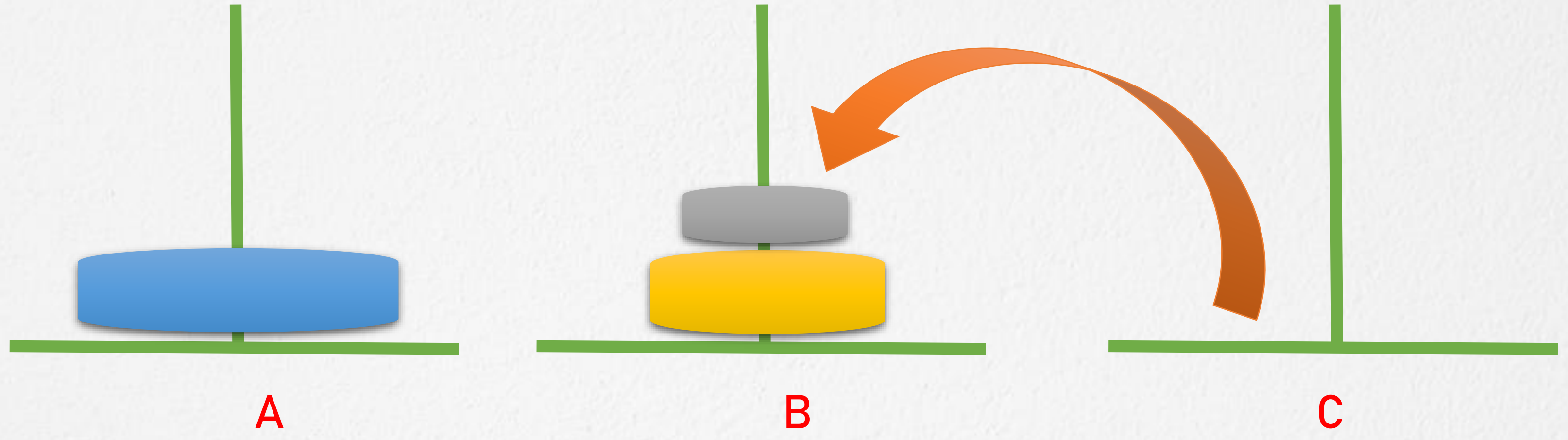


Step 2

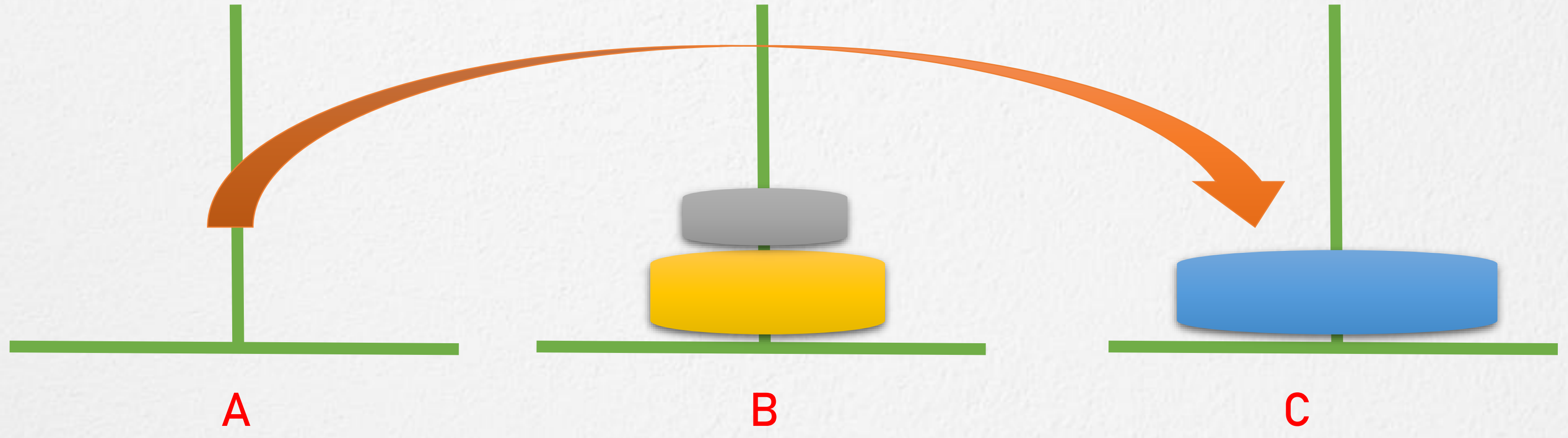


Tower of Hanoi

Step 3

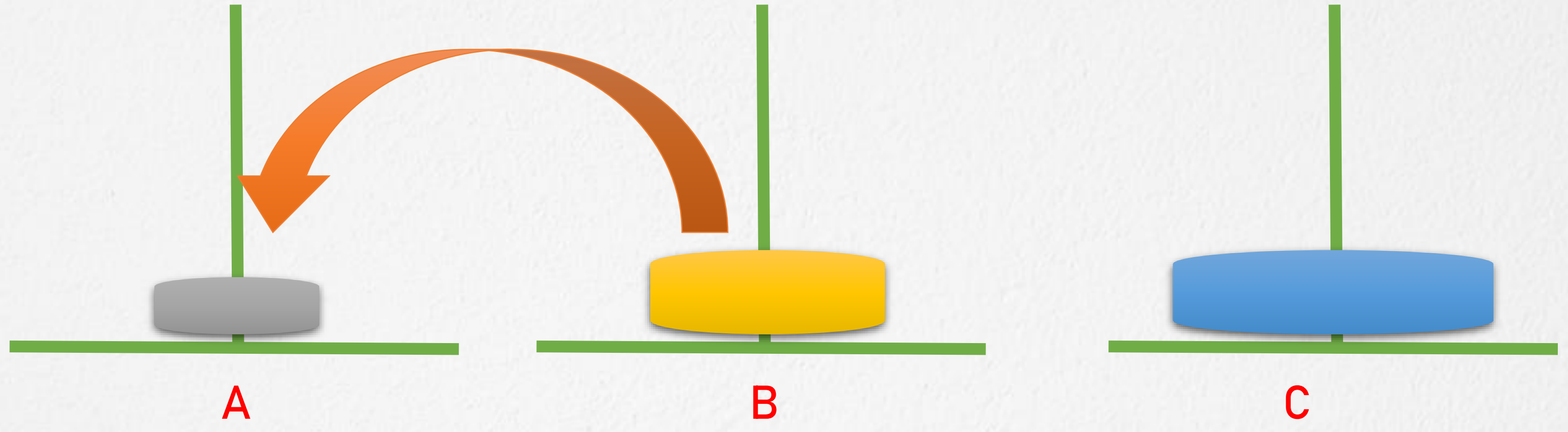


Step 4

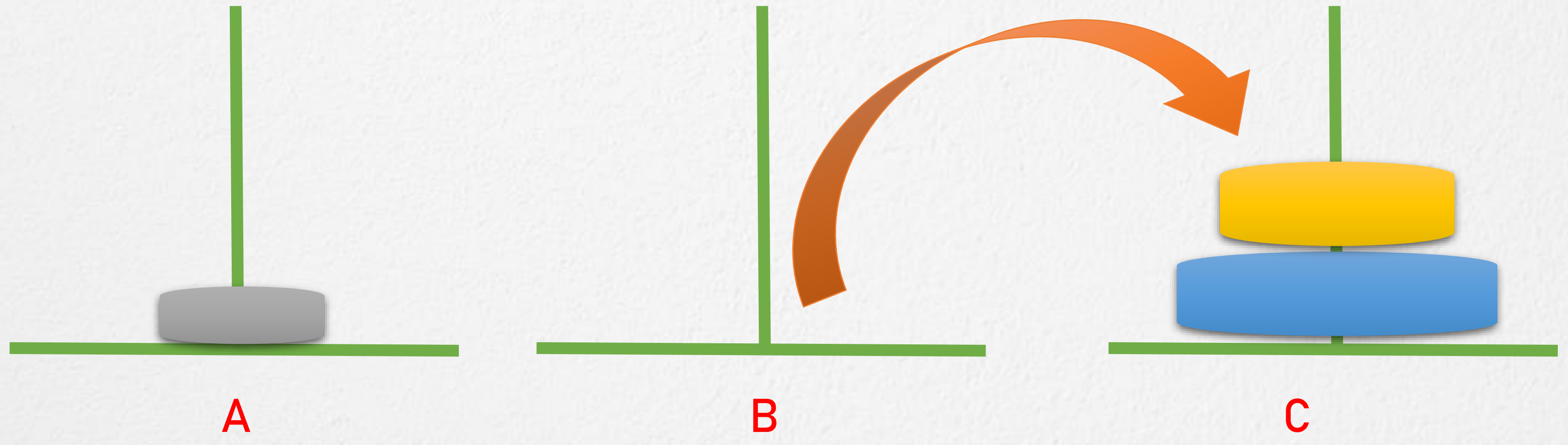


Tower of Hanoi

Step 5

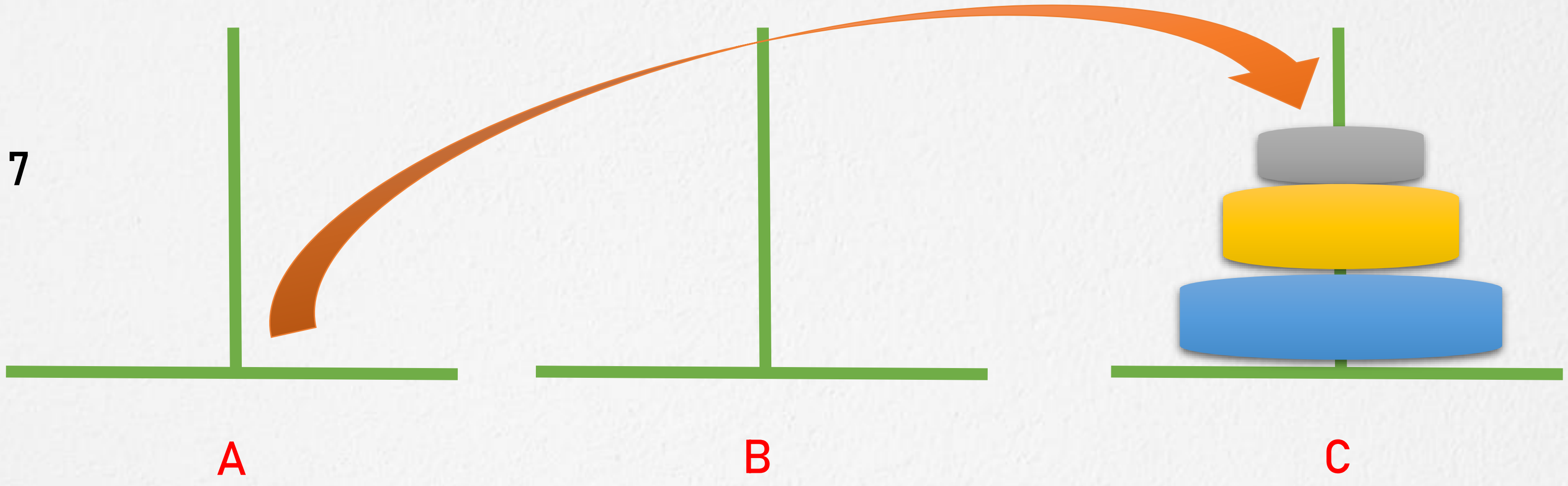


Step 6

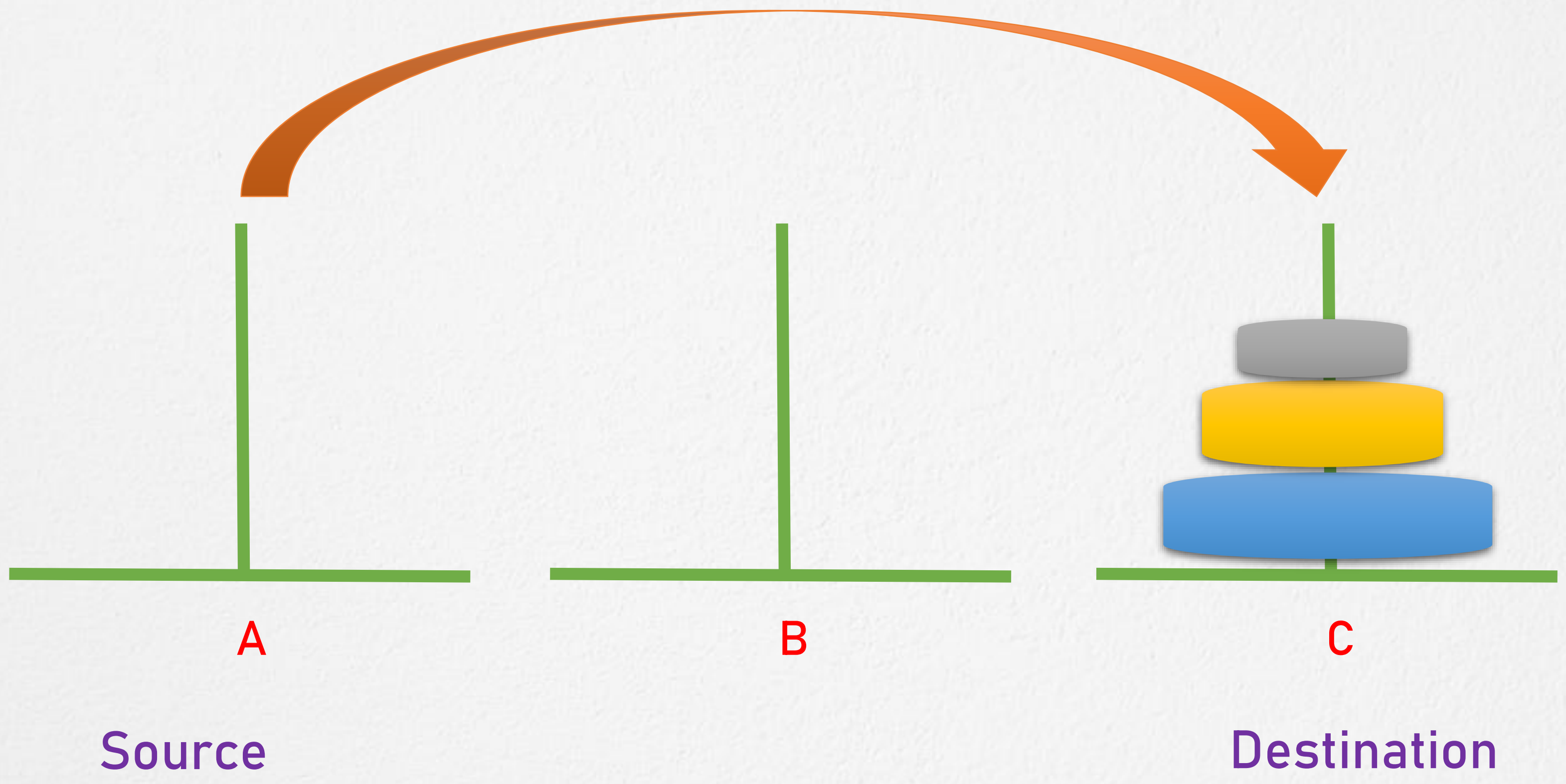


Tower of Hanoi

Step 7



Tower of Hanoi



Algorithm

```
START

Procedure Hanoi(disk, source, dest, aux)

  IF disk == 1, THEN

    move disk from source to dest

  ELSE

    Hanoi(disk - 1, source, aux, dest)

      move disk from source to dest

    Hanoi(disk - 1, aux, dest, source) END IF

 END Procedure

STOP
```

Expression conversion and evaluation

- Arithmetic expressions can be represented in 3 forms:
  - Infix notation
  - Postfix notation (Reverse Polish Notation)
  - Prefix notation (Polish Notation)

Infix Notation

- Infix Notation can be represented as:

operand1 operator operand1

Example: 10 + 20

a + b

Postfix Notation

- Postfix Notation can be represented as

operand1 operand2 operator

Example: 10 20 +

a b +

Prefix notation

- Prefix notation can be represented as

  operator operand1 operand2

  Example: + 10 20

  + a b

Expression conversion and evaluation

- Infix notation is used most frequently in our day to day tasks.
- Machines find infix notations tougher to process than prefix/postfix notations. Hence, compilers convert infix notations to prefix/postfix before the expression is evaluated.

Expression conversion and evaluation

- The precedence of operators needs to be taken care as per hierarchy

(^) > (*) > (/) > (+) > (-)

Brackets have the highest priority

Expression conversion and evaluation

- To evaluate an infix expression, We need to perform 2 steps:
- Convert infix to postfix
- Evaluate postfix

Example

expression: 10+20*5

Step 1: change this infix expression to postfix

10 20 5 + *

Step 2: Stack S = [ ], traverse the string

10 : Operand, push into the stack, S = [10], top = 10

20 : Operand, push into the stack, S = [10,20], top = 20

5 : Operand, push into the stack, S = [10,20,5], top = 5

Example

- * : Operator, pop top two elements, op1 = 5, op2 = 20.
- Stack after pop operations S = [10], top = 10. Now, we push the result of op1 * op2, i.e 5 * 20 = 100 into the stack. S = [10, 100], top = 100
- + : Operator, pop top two elements, op1 = 100, op2 = 10. Stack after pop operations S = []. Push the result of op1 + op2 into the stack, i.e 100 + 10 = 110, S = [110]
- The string has been completely traversed, the stack contains only 1 element which is the result of the expression = 110

Algorithm

Traverse the expression:

- If the character is an operand, push it into the stack.
- If the character is an operator, pop the 2 top most elements from the stack and perform the operation. Push the result back to the stack.

Once the expression is fully traversed, the element in the stack is the result.

That’s all for now…