Example 2 :-

$$T(n) = 2T(n/2) + n\log n$$

Here

$$f(n) = n \log n$$

$$a = 2, \quad b = 2$$

use formula 4

$$f(n) = \theta(n^{\log_2 2} \log^1 n) \qquad \text{ie., } k = 1$$

Then $T(n) = \theta(n^{\log_b a} \log^{k+1} n)$

$$= \theta(n^{\log_2 2} \log^2 n)$$

$$= \theta(n^1 \log^2 n)$$

$$\therefore T(n) = \theta(n \log^2 n)$$

Example 3 :-

$$T(n) = 8T(n/2) + n^2$$

Here $f(n) = n^2$

$a = 8$ and $b = 2$, $d = 2$

$$\therefore \log_2 8 = 3$$

use formula 3 $\left[\because \begin{array}{l} a > b^d \\ 8 > 2^2 \end{array}\right]$

$$f(n) = O(n^{\log_2 8})$$

$$T(n) = \theta(n^3)$$

Example 4 :-

$$T(n) = 9T(n/3) + n^3$$

Here $a = 9$, $b = 3$ and $d = 3$

$a < b^d$ $\quad (\because 9 < 3^3)$

so use formula 1

$$f(n) = \theta(n^d)\cancel{\log n}$$

$$\therefore f(n) = \theta(n^3)\cancel{\log n}$$

MATHEMATICAL ANALYSIS OF RECURSIVE ALGORITHM:

→ GENERAL PLAN FOR ANALYZING EFFICIENCY OF RECURSIVE ALGORITHMS:

1. Decide the input size based on parameter n.
2. Identify Algorithm's basic operations.
3. Check how many times the basic operation is executed. Then find whether the execution of basic operation depends upon the input size n. Determine worst, best and average cases for input of size n.
4. Set up the recurrence relation with some initial condition and expressing the basic operation.
5. Solve the recurrence relation using forward and Backward Substitution method. And the prove the Correctness by using Mathematical Induction

EXAMPLES FOR RECURSIVE ALGORITHMS:-

Example 1:-

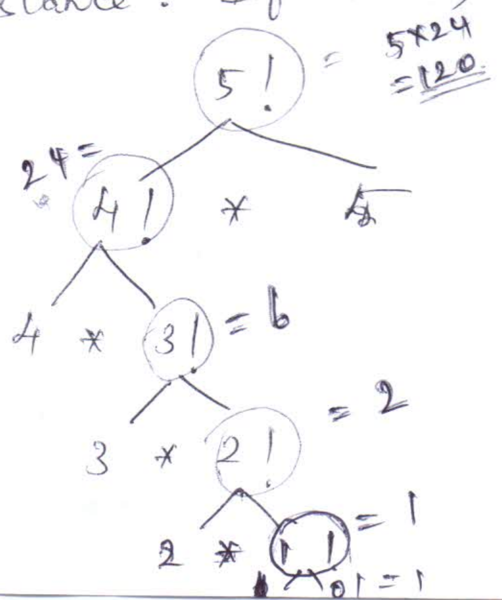
Computing Factorial of some number n.

Solution:- The Factorial of some number can be obtained by performing repeated Multiplication using recursive call.

For instance: If n=5, then

[Use Formula:- $n! = n * (n-1)$

Steps:



[We know that $0! = 1$]

Algorithm :- Fact(n)

// problem Description :- This Algorithm computes n! using recursive function.

// Input :- A non-negative integer n.

// output :- returns the factorial value

```
if (n = = 0)
    return 1
else
    return Fact (n-1) * n
```

$n! = (n-1)! * n$

MATHEMATICAL ANALYSIS :-

Step 1 :- The factorial algorithm works for input size

Step 2 :- The basic operation is multiplication.

Step 3 :- The recursive function call can be formula as,

$$F(n) = F(n-1) * n \quad \text{where } n > 0$$

Then the Recurrence relation is

$$M(n) = M(n-1) + 1 \quad \text{(1)}$$

these multiplications are required to compute factorial (n-1)

To multiply fact (n-1) by n

step 4 :-

Now we solve the recurrence using.

Forward Substitution :- use eqn (1)

put n=1, 2, 3...

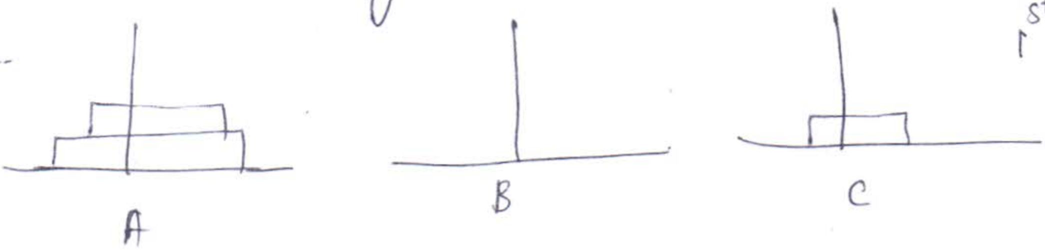
$$M(1) = M(0) + 1 = 1$$

$$M(2) = M(1) + 1 = 1+1 = 2$$

$$M(3) = M(2) + 1 = 2+1 = 3$$

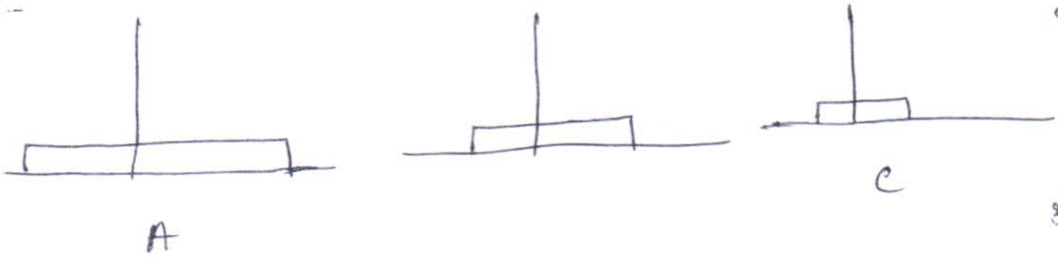
The problem of Tower of Hanoi states that move the three disks from peg A to peg C using peg B as an auxillary.

step 2 :-



1st move

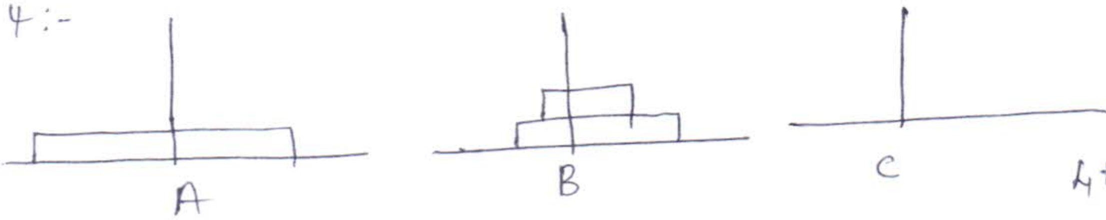
step 3 :-



2nd mov of disk

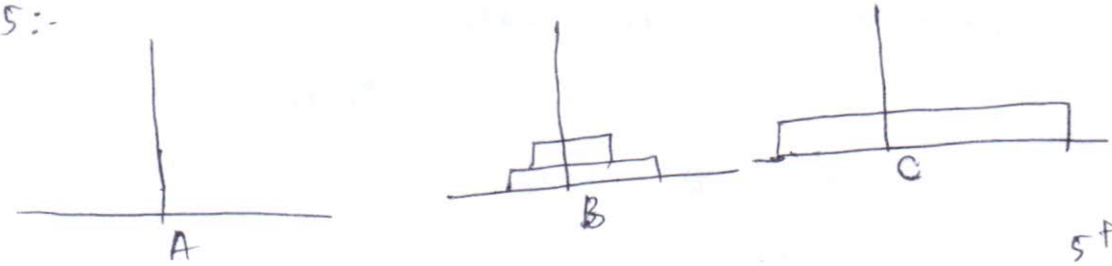
3rd moves

step 4 :-



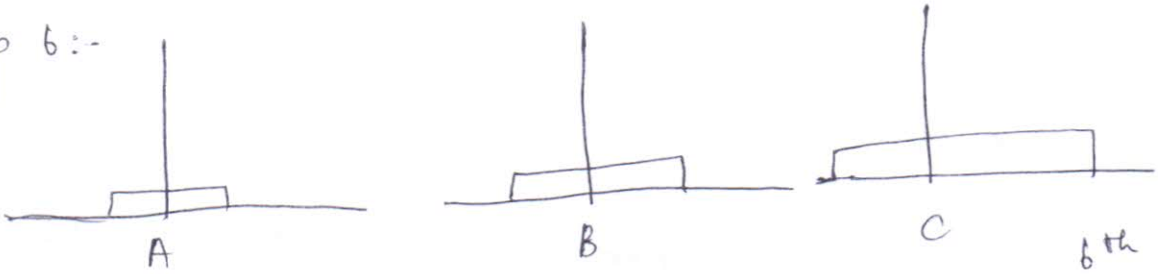
4th move

step 5 :-



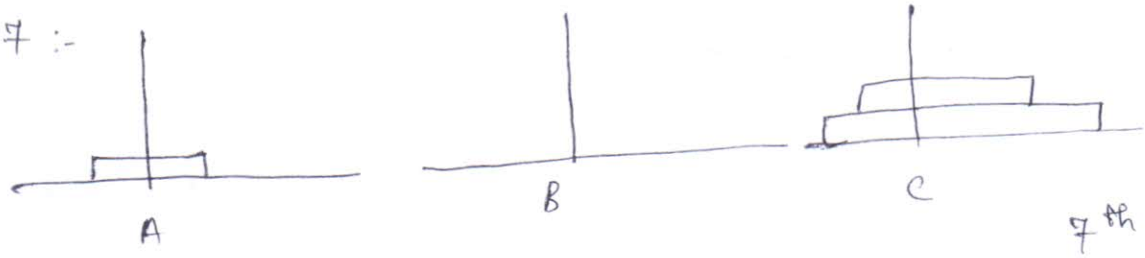
5th move

step 6 :-



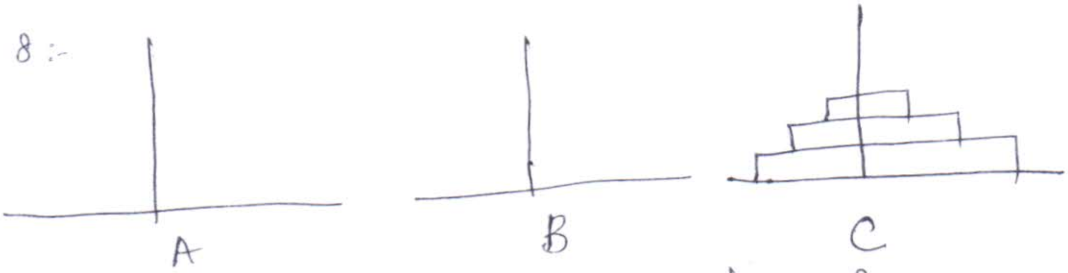
6th move

step 7 :-



7th move

step 8 :-



~~of movements of rod~~

```
Algorithm TOH (A, B, C, n)
{
 if (n==1) then    // if only one disk has to be
                      moved.
  {
   write (" The peg moved from A to C");
   return
  }
 else
  {
   // move top n-1 disks from A to B using C
   TOH (n-1, A, B, C);
   // move remaining disks from B to C using A
   TOH (n-1, B, C, A);
  }
}
```

MATHEMATICAL ANALYSIS :-

step 1:- The input size is n (total no/: of disks)

step 2:- The Basic operation in this problem is move disks from one peg to another.

If $n=1$, then we simply move the disks from peg A to peg C.

step 3:- The moves of disks are denoted by $M(n)$. $M(n)$ depends on number of disks n.

The recurrence relation can be setup as,

$M(1)=1$ ∵ Only 1 move is needed

If $n>1$, then we need two recursive calls plus one move. Hence

$$M(n) = M(n-1) + 1 + M(n-1)$$

To move (n-1) disks from peg A to B

To move largest disk

To move (n-1) disks from

$\therefore \; M(n) = 2M(n-1) + 1$

Step 4 :- Solving recurrence $M(n) = 2M(n-1)+1$ using forward and Backward Substitution method.

Forward Substitution :-

for $n > 1$,

$M(2) = 2M(1) + 1 = 2 + 1$

$M(2) = 3$

$M(3) = 2M(2) + 1$

$= 2(3) + 1$

$M(3) = 7$

$M(4) = 2M(3) + 1 = 2(7) + 1$

$M(4) = 15$

Backward Substitution :-

$M(n) = 2M(n-1) + 1$

$= 2[2M(n-2) + 1] + 1 = 4M(n-2) + 3$

$= 2^2 M(n-2) + 2^2 - 1$

$= 4[2M(n-3) + 1 + 3] = 8M(n-3) + 7$

(or) $2^3 M(n-3) + 2^3 - 1$

From this we can establish a general formula

$M(n) = 2^i M(n-i) + 2^{i-1} + 2^{i-2} + \ldots + 2 + 1$

this can be written as,

$M(n) = 2^i M(n-i) + 2^i - 1 \longrightarrow (1)$

Prove correctness using Mathematical Induction :-

put $i = n-1$ in eqn (1)

$M(n) = 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1$

$= 2^{n-1} M(1) + 2^{n-1} - 1$ $\qquad \because M(1) = 1$

$= 2^{n-1} + 2^{n-1} - 1$ $\qquad$ n = one disk need one move

$M(n) = 2^n - 1$ Now if $n = 1$ then

$M(1) = 2^1 - 1 = 1$ is proved.

Example 3:-

3. finding Number of bits in Integer.

Algorithm: Binary_Rec (n)

// Problem Description : The Algorithm is for counting the binary digits from a decimal Integ

// Input :- The decimal integer n.

// output :- Returns total number of digits from the ing

if (n==1) then
    return 1
else
    return (Binary_Rec($\lfloor n/2 \rfloor$) + 1

Mathematical Analysis :-

step 1:- The input Size n.

step 2:- The basic operation which is performed is division by 2.

step 3:- Now we will set up recurrence relation for this Algorithm.

→ Let D(n) be a count of performing division to calculate Binary_Rec (n)

→ when n = 1, then there is no division operation performed

$$\therefore D(1) = 0$$

If n>1 then Binary_Rec(n) makes a recursive ca with $\lfloor n/2 \rfloor$

$$D(n) = D(\lfloor n/2 \rfloor) + 1 \xrightarrow{\text{for } n>1} (1)$$

step 4 :-

Solving recurrence for n to be power of 2, we can compute the efficiency of an Algorithm.

Forward substitution :-

$$D(2) = D(1) + 1 \qquad \because D(1) = 0$$

$$D(2) = 1$$

Similarly $D(4) = D(2) + 1$

$= 1 + 1$

$D(4) = 2$

Similarly, $D(8) = D(4) + 1$

$= 2 + 1$

$D(8) = 3$

Backward substitution :-

$$D(n) = \underbrace{D(\lfloor n/2 \rfloor)}_{} + 1 \qquad D(\lfloor n/2^2 \rfloor) + 2$$

$$= \boxed{D(\lfloor n/4 \rfloor) + 1} + 1 = D(\lfloor n/4 \rfloor) + 2$$

$$= D(\lfloor n/8 \rfloor + 1] + 2 = D(\lfloor n/8 \rfloor) + 3.$$

Put $n = 2^k$, in eqn (1).

$$D(2^k) = D(2^{k-1}) + 1 \longrightarrow (2)$$

Prove correctness using Mathematical Induction :-

If $k = 0$ then,

$D(2^k) = D(2^0) = D(1)$

But as $D(1) = 0$ we get,

$D(2^k) = k$ is proved. $\longrightarrow (3)$

We have assumed $n = 2^k$. By taking log on both the sides $k = \log_2 n$.

using (3) $D(n) = \log_2 n \in \Theta(\log_2 n)$