Recursion 2

Content

- Quizzes on Recursion
- Power Function
- Print Array
- Max/Min of an array
- Indices of an array
- Check Palindrome
- Tower of Hanoi {Idea if time permits}

Amreshwar
Anil
Aravind Krishnan S
Arunava Basak
Bhavesh Pandey
Chandra Shekhar Bhatt
Deepshikha Arora
Dhasthagiri Reddy
Dipika Malik
Girish Pawar
Harikrishnan A
jeevananthan
M S Haseeb Khan
Mahesh Baswaraj
Manohar A N
MOHAMMAD ALI
Pallavi V Rao
Piyush
Pranjul Kesharwani
Priyank Varshney
SHIVAM SHIV
Suresh
Uddeepta Saikia
Ved Verma

Q Given two +ve integer a and n, find a^n

$2^3 = 8$

$3^3 = 27$

$$2^5 = \underbrace{2 * 2 * 2 * 2}_{2^4} * 2$$

$2^4 * 2 \longrightarrow$ subproblem

Assume $\longrightarrow$ pow(a, n) calculates a^N

$\downarrow$

pow(a, n-1) * a

Base case = $a^0 = 1$

n==0 $\longrightarrow$ return 1

```
int pow ( int a , int n ) {
    if ( n==0 ) return 1
    return pow( a, n-1) * a
}
```

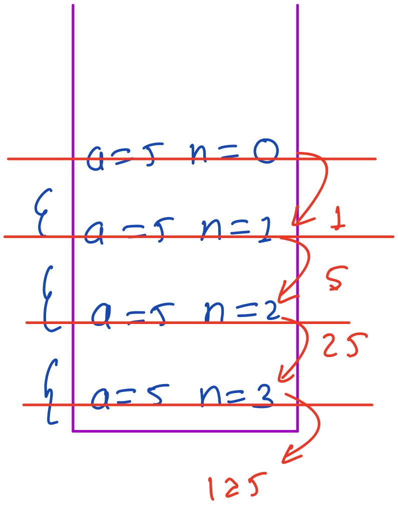
SC: O(N)

max stack space we

a=5 n=3

$$\text{TC} : \underbrace{\#\text{ fn calls}}_{n+1} * \underbrace{\text{time per fn call}}_{O(1)}$$

TC : O(N)



Idea 2>

$2^8 = 2^4 * 2^4$

$2^{16} = 2^8 * 2^8$

$5^{10} = 5^5 * 5^5$

$2^5 = 2^2 * 2^2 * 2$

$5^9 = 5^4 * 5^4 * 5$

```
int pow2 ( int a , int n ) {
        if ( n==0 ) return 1

        if ( n%2 ==0) {
            return    pow2(a, n/2) * pow2(a, n/2)
        }
        ret pow2(a, n/2) * pow2(a, n/2) * a
}
```

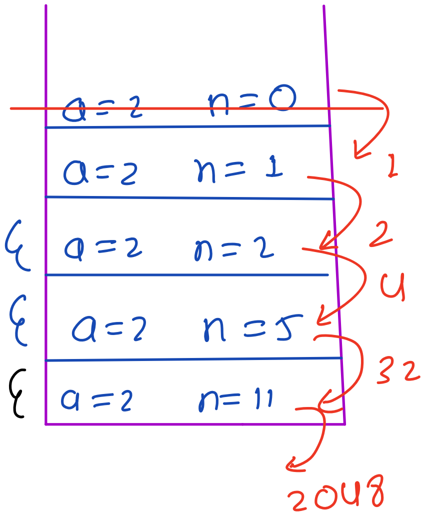
2^{11}

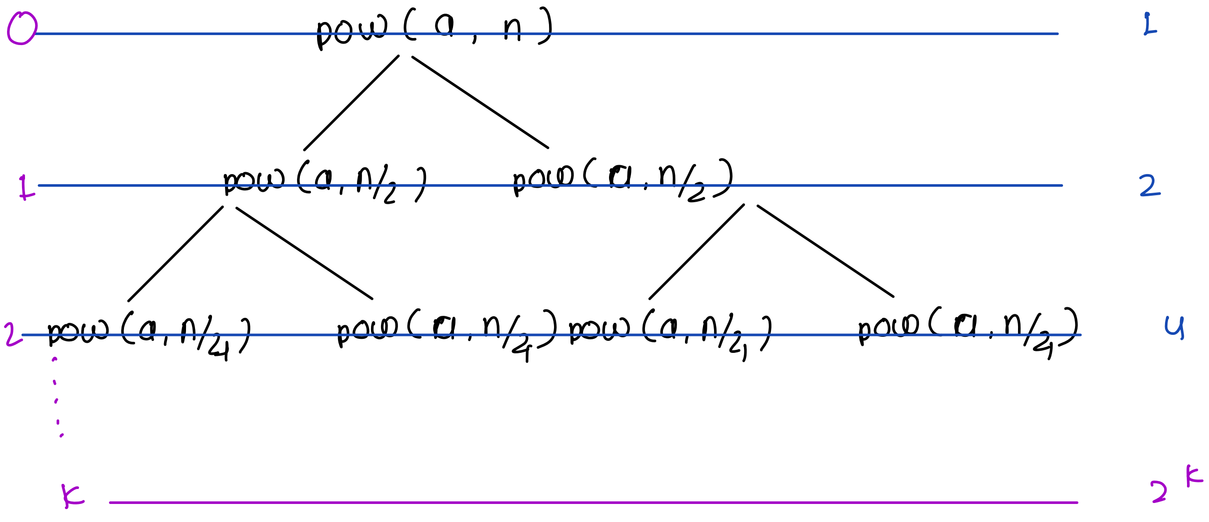
a = 2   n = 11

TC : # fn calls

SC : O(log N)

TC : O(N)





Total fn calls = sum of fn calls at all levels

$$\underbrace{1 + 2 + 4 + \dots 2^k}_{k+1} \longrightarrow \text{sum of GP}$$

$$\frac{1 * (2^{k+1} - 1)}{2 - 1}$$

$$2 * 2^k \longrightarrow 2 * 2^{\log_2 N}$$

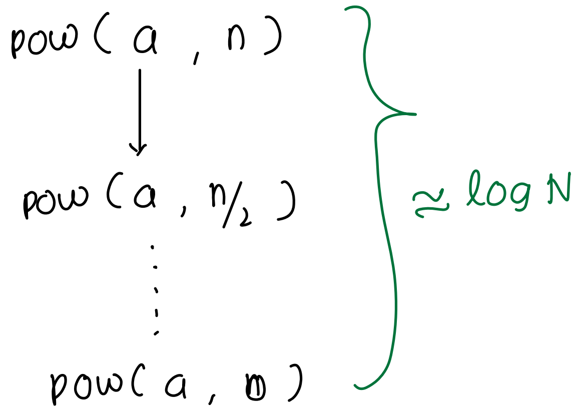
$$2 * N^{\log_2 2}$$

$$\underline{\underline{2 * N}}$$

```
int pow2 ( int a , int n ) {
        if ( n==0 ) return 1
           p = pow2(a, n/2 )
         if ( n%2 ==0) {
            return    p* p
         }
        ret    p* p * a
}
```

TC : O(log N)

SC: O(log N)



Print array using recursion

Given an array of integers, write a recursive function to print all the elements of the array.

A = [1, 2, 3, 4, 5]

Output — 1

2
3
4
5

print (A, 1)

sub prob

print(A, 0)

A = [10 2 1 9 6]

Output — 10
2
1
9
6

Assume ⟶ print (A, idx) prints elements from idx to N-1

```
void  printA(int[] A , int idx ){
    if ( idx == A.length )  return
    print( A[idx] )
    printA( A, idx+1)
}
```

TC: O(N)

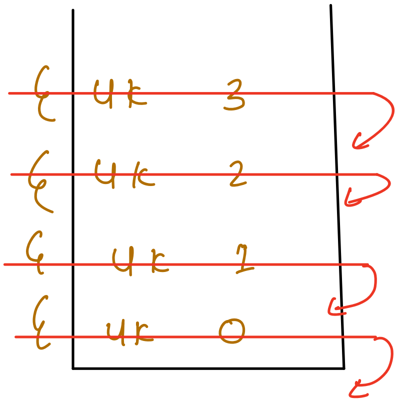
SC: O(N)

A

mem addr 4k

0	1	2
2	1	9

Output 2
1
9



Return max element using recursion

Given A[N], write a recursive function to find max element in the array.

A = 5 8 2 10 3 Output = 10

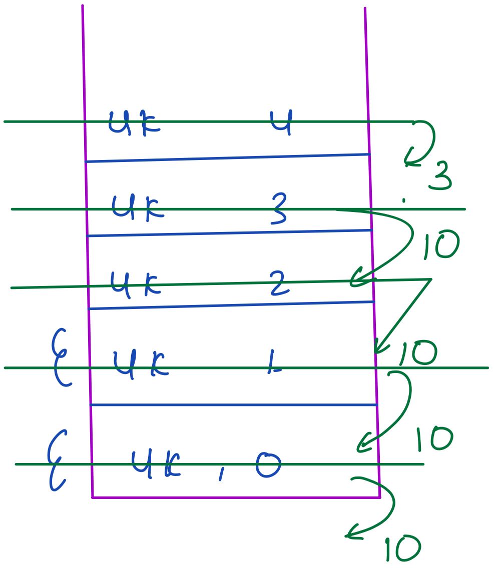
given problem sub problem

Assume ⟶ Amax (A, idx) returns max from $\underbrace{\text{idx} \ldots\ldots N-1}$

```
int  Amax ( A , idx ) {
        if ( idx == N-1)  return A[idx]

        r = Amax ( A, idx + 1 )
        return  max ( A[idx] , r )
}
```

TC: O(N)
SC: O(N)

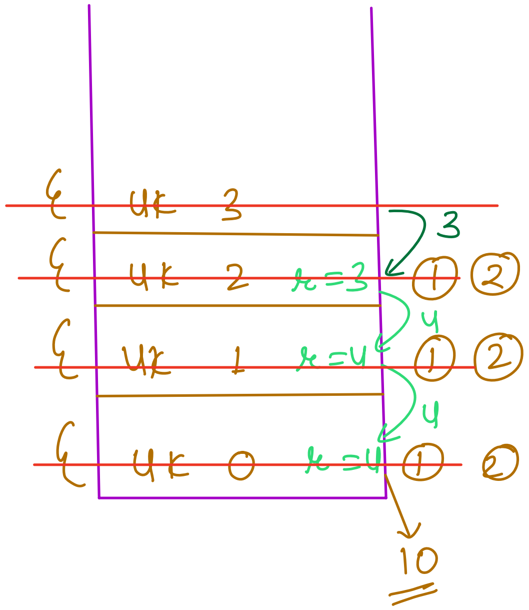
0 1 2 3 4
5 8 2 10 3
4K



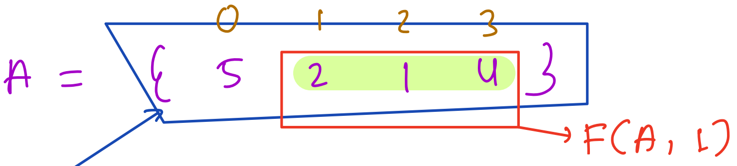
```
int   Amax ( A , idx ) {
         if ( idx == N-1 )   return  A[idx]

         r  =  Amax ( A, idx+1 )  ①
         return   max ( A[idx] , r ) ②
}
```

$$\begin{array}{ccccc} & 0 & 1 & 2 & 3 \\ A = & 10 & 1 & 4 & 3 \end{array}$$

↙
uk



To find   sum



Assume   F(A, 0) find sum from idx 0 to N-L

```
int   F ( int [] A ,   idx ) {
          if ( idx == N-1)    return   A [idx]
          return   A[idx]  +  F( A ,  idx + 1)
}
```

All indices of Array

Given A[N] , target B . The task is to find all the indices at which B occurs in the array.

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ [\,4 & 5 & 3 & 1 & 5 & 4 & 5\,] \end{array} \qquad ans = [1, 4, 6]$$

$$B = 5$$

QUIZ

$$A = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ [\,1 & 2 & 3 & 1 & 1\,] \end{array} \qquad ans = [0, 3, 4]$$

$$B = 1$$

Break : 22:38

Approach 1 { maintain global ans }

```
main (int[] A , int B ) {
    ans = []    // ArrayList , List
    helper ( A, B , idx , ans )
    return ans
}
```

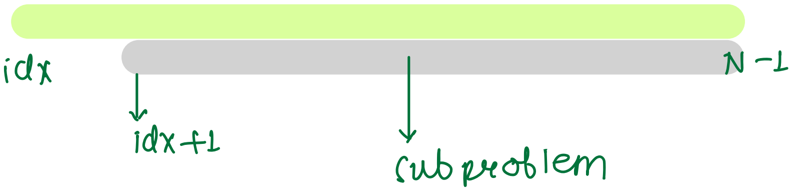
```
void helper ( int[] A , int B , int idx , int[] ans ){
    if ( idx == A.length )  return

    if ( A[idx] == B) { ans.add (idx) }

    helper ( A , B, idx +1, ans )
```

|3

Approach 2 > No dynamic arrays allowed.

0	1	2	3	4
1	2	3	1	1

Assume findAll (A, B, idx, count) return ans array with all populate indexes



8k

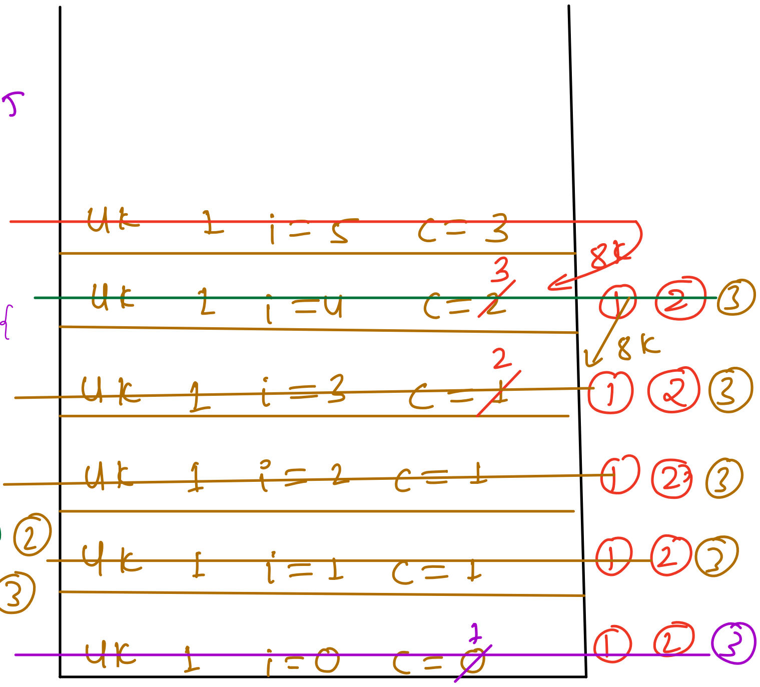
ans [] = [~~0~~, ~~0~~, ~~0~~] 0 3 4 0 1 2

B=1 N=5

4k

0	1	2	3	4
1	2	3	1	1

```
·[]   findAll (int[] A , int B , int idx , int count){
       // Base
        if (idx == A.length) {
              return new int[count] ;

       if( A[idx] == B) { count++}  ①

       int[] ans  =  findAll( A, B, idx+1 , count)  ②

       if( A[idx] == B) {ans[count-1] = idx}  ③

       ret ans
```



Pseudocode

```
int[]   findAll (int[] A , int B , int idx , int count){
            // Base
            if (idx == A.length) {
                    return new int[count];
            }

            if ( A[idx] == B) { count++ }

            int[] ans = findAll( A, B, idx+1 , count)

            if ( A[idx] == B) { ans[count-1] = idx }

            return ans
}
```

TC : O(N)

SC : O(N)

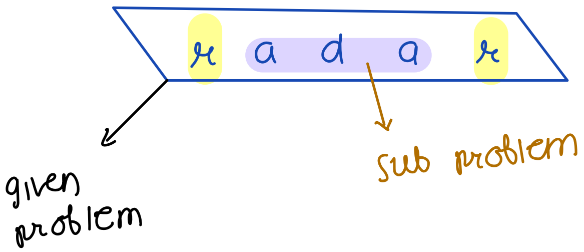
Check palindrome using recursion

Given a string s. Write a recursive function to check if it is a palindrome.

s = "radar"
output true

s = "area"
output = false

Assume → palin fn returns true if s is palindr..
else false



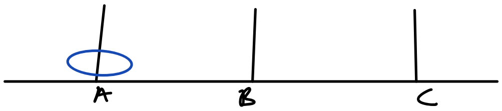
TC : O(N)
SC : O(N)

```
boolean isPalin ( String s , int l , int r ) {
        if ( l >= r ) return true

        if ( s[l] != s[r] ) { return false }
        return isPalin ( s , l+1, r-1 )
}
```

0 1 2 3 4 5
a b b d b a

l=2   r=3
l = 1   r = 4   false
l = 0   r = 5   false
false

Tower of Hanoi ***



There are n disks placed on a tower A
{different sizes}

Move all disks from tower A to C using B

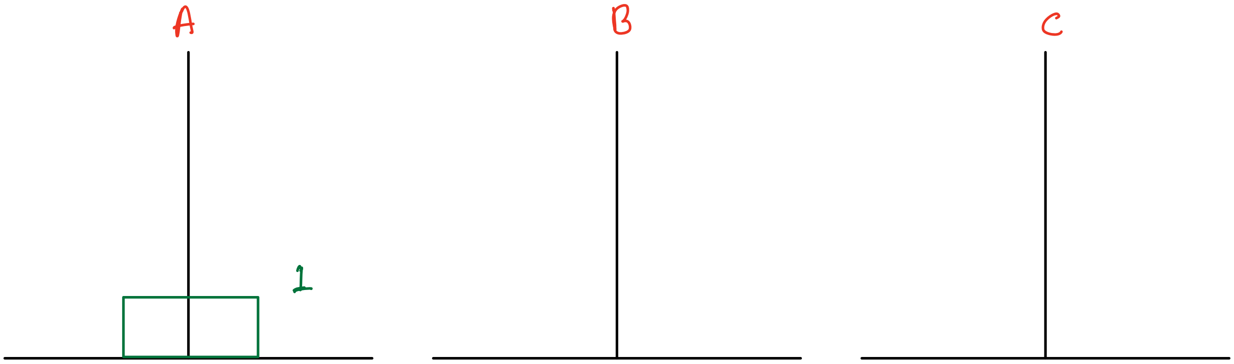
Constraints

— Only 1 disk can be moved at a time
— Larger disk cannot be placed on a small disk at any step.

Print the movement of disks from A to C in minimum steps.

Example

N = 1

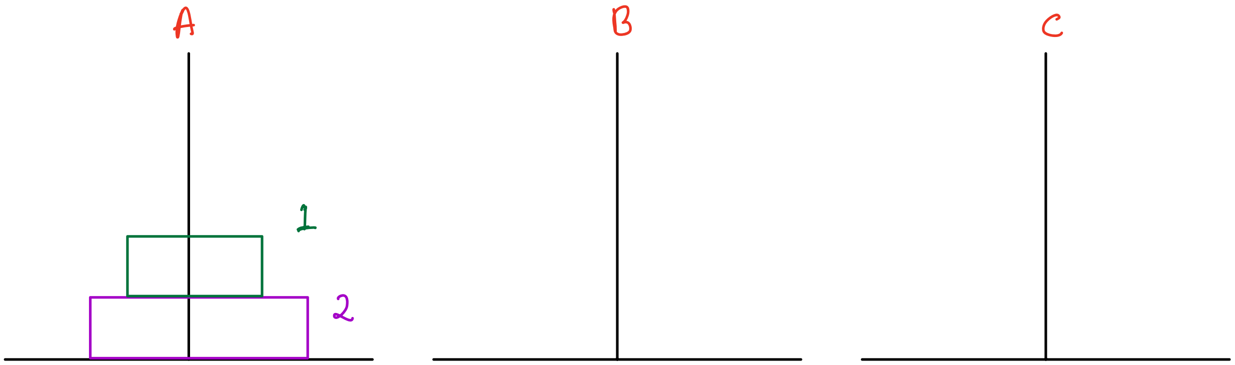


Move disk 1 from A to C

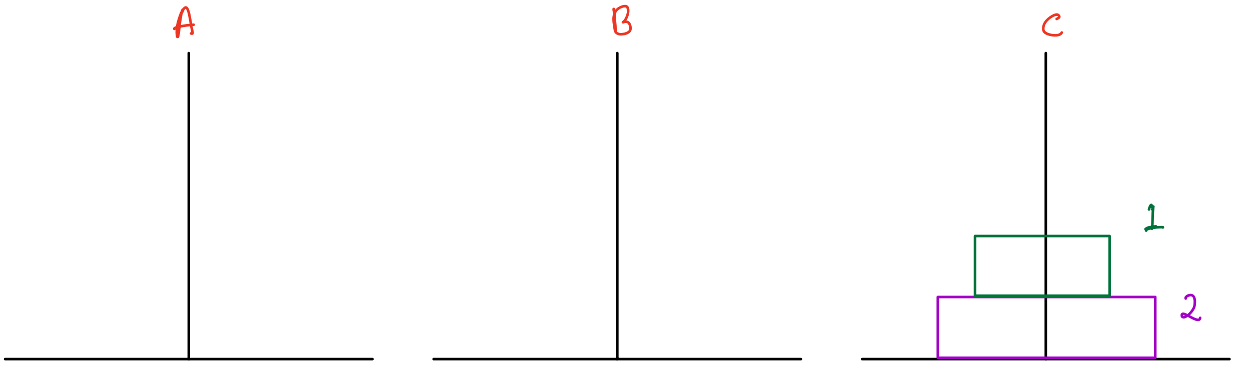
N = 2

TOH (2, A, B, C)

Initial State



Final state

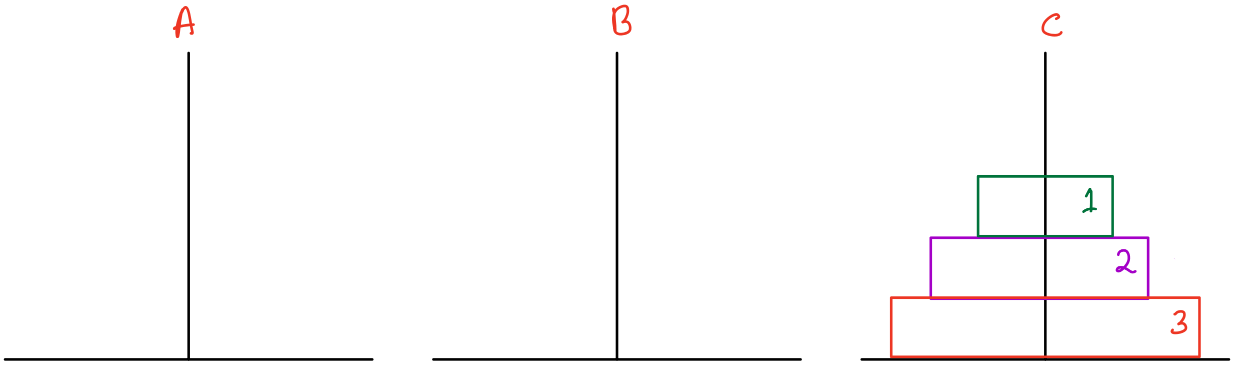


→ move disk 1 from A to B

→ move disk 2 from A to C

→ move disk 1 from B to C

N = 3



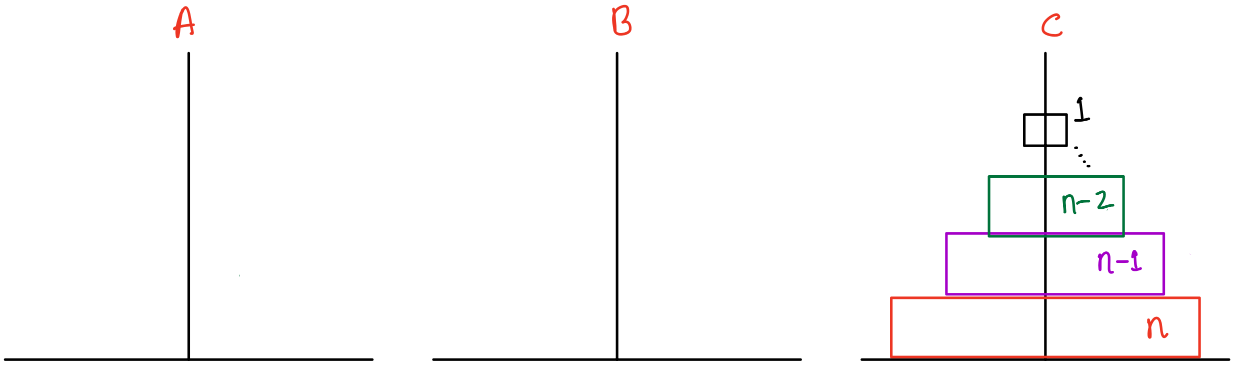
move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B

moved 2 disks from A to B using spare tower C

move disk 3 from A to C

move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C

moved 2 disks from B to C using spare tower A



move n-1 disks from A to B using spare C

move disk n from A to C

move n-1 disks from B to C using spare A

Pseudocode

move n-1 disks from A to B using spare C

move disk n from A to C

move n-1 disks from B to C using spare A

```
                       spare
                 fr      ↓       to
void   TOH ( N, A , B , C ) {
         if ( N==0 ) return
         TOH (N-1 , A , C , B )
         print ("move disk N from A to C")
         TOH(N-1, B , A , C )
}
```

Doubt session

```
f ( n) {
    if (n == 1)   return 1
    return    n * fact(n-1)
              0 * f(-1)
              -1 * f(-2)
}
```