Recursion 2

Agenda

1) Power function
2) Print array
3) Max/min of an array
4) Check palindrome
5) Tower of Hanoi

Q) Given 2 integer a & n, find a^n using recursion.

Eg. a=2 ; n=3

Output: $2^3 = 8$

$2 * 2 * 2$

$a^n = \underbrace{a * a * a * \ldots * a}_{n \text{ times}}$

$a^n = a * a^{n-1}$

$pow(a,n) = a * pow(a, n-1)$

$2^5 = 2 * 2^4$

$2^4 = 2 * 2^3$

$2^3 = 2 * 2^2$

$2^2 = 2 * 2^1$

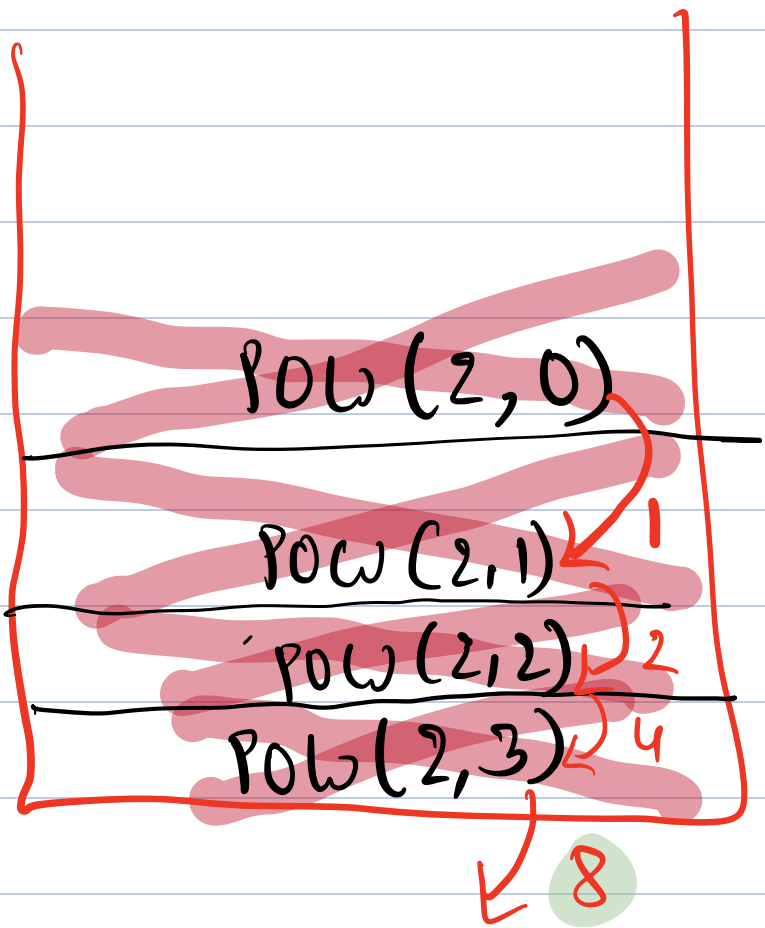
$2^1 = 2 * 2^0$

$2^0 \rightarrow 1$ $\quad\quad$ $a^0 = 1$

Code:

```
int pow(a, N)
{
    if (N==0) return 1;        ①
    return a * pow(a, N-1);    ②
}
```

pow(2, 3)



TC: O(N)

SC: O(N)

$2^{10} = 2^5 * 2^5$

$a^N = a^{N/2} * a^{N/2}$ — N is even

$2^{11} = 2^5 * 2^5 * 2$

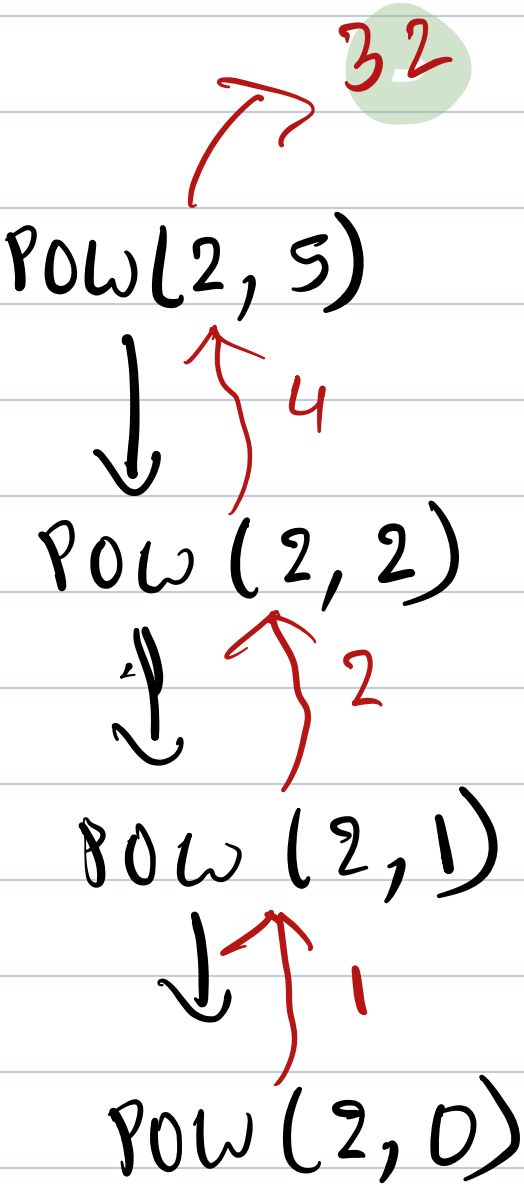
$a^N = a^{N/2} * a^{N/2} * a$ — N is odd

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

    if (n % 2 == 0)
    {
        return pow(a, n/2) *
               pow(a, n/2);
    }
    else
    {
        return pow(a,n/2) * pow(a,n/2) * a;
    }
}
```

Approach 3 : Fast power-

```
int pow(a, n)
{
    if (n==0) return 1;
    int p = pow(a, n/2)

    if (n%2==0)
    {
        return p*p;
    }else
    { return p*p*a;
    }
```

```
int pow(a, n)
{
    if (n == 0) return 1;
    int p = pow(a, n/2)
    if (n % 2 == 0)
    {
        return p * p;
    }else
    {   return p * p * a;
    }
```



TC: $O(\log n)$

SC: $O(\log n)$

9) Given an integer array, print all the array element using recursion.

Eg. A[] = {1, 2, 3, 4, 5}
(0 1 2 3 4)
Output: 1, 2, 3, 4, 5

```
void printArray(A[], index)
{
    if(index == A.length) return;
    print(A[index]);
    printArray(A, index+1);
}
```

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

Q) find max element of array using recursion.

A[] = {3, 5, 8, 1} O/P → 8

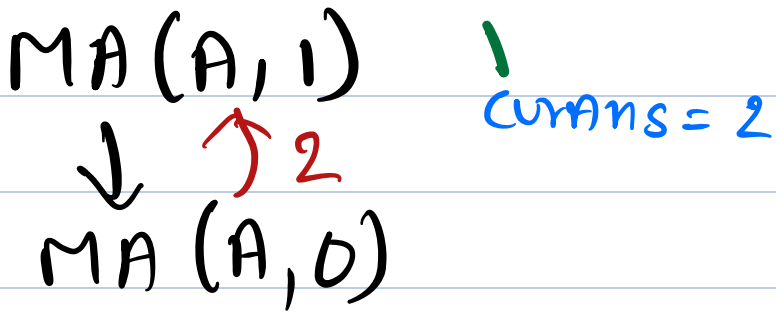
```
int MaxArray(A[], index)
{
    if (index == 0) return A[index];
    int currAns = MaxArray(A, index-1);
    if (currAns > A[index])
    {
        return currAns;
    }
    else
    {
        return A[index];
    }
}
```

① ② ③

A[] = {2, 5, 8, 3} (0 1 2 3)

MA(A, 3) → currAns = 8

MA(A, 2) → currAns = 5



TC : O(N)

SC : O(N)

9) Check if the given string is palindrome.

Eg. → racecar o/p → True
race o/p → False

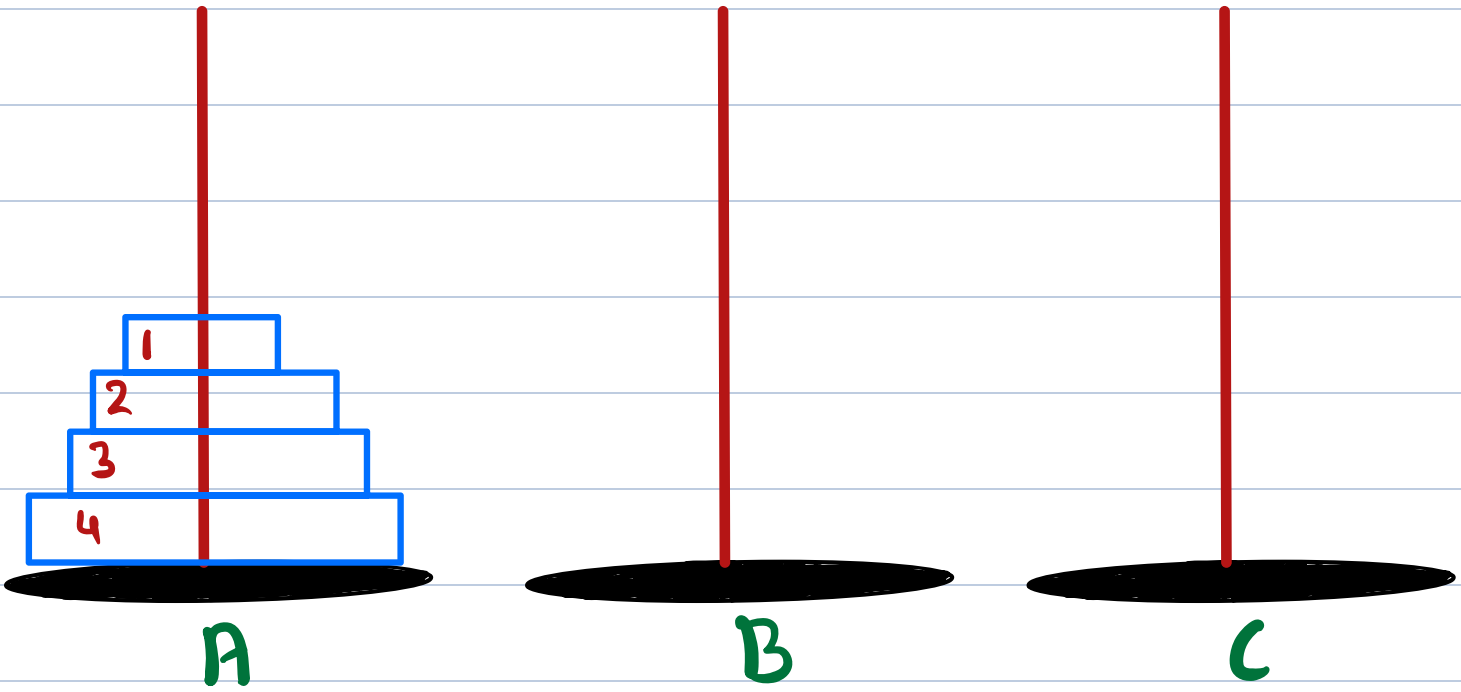
```
boolean isPalindrome(s[], l, r)
{
    if(l>=r) return true;

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

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

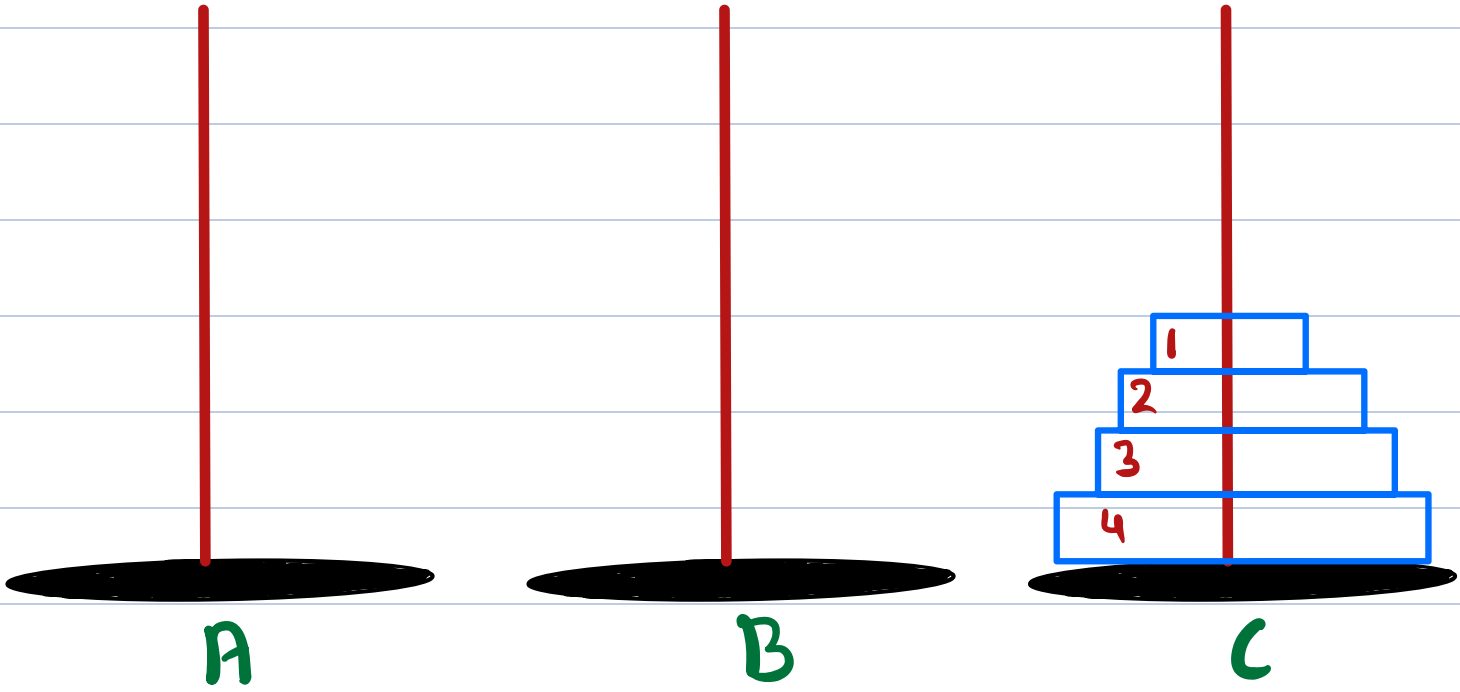
Break : 10.10 pm

9) Tower of Hanoi - 3 tower A, B & C.

There are N disks placed on tower A of different sizes.



Goal :- Move all the disks from A to C using tower B.



Rules:

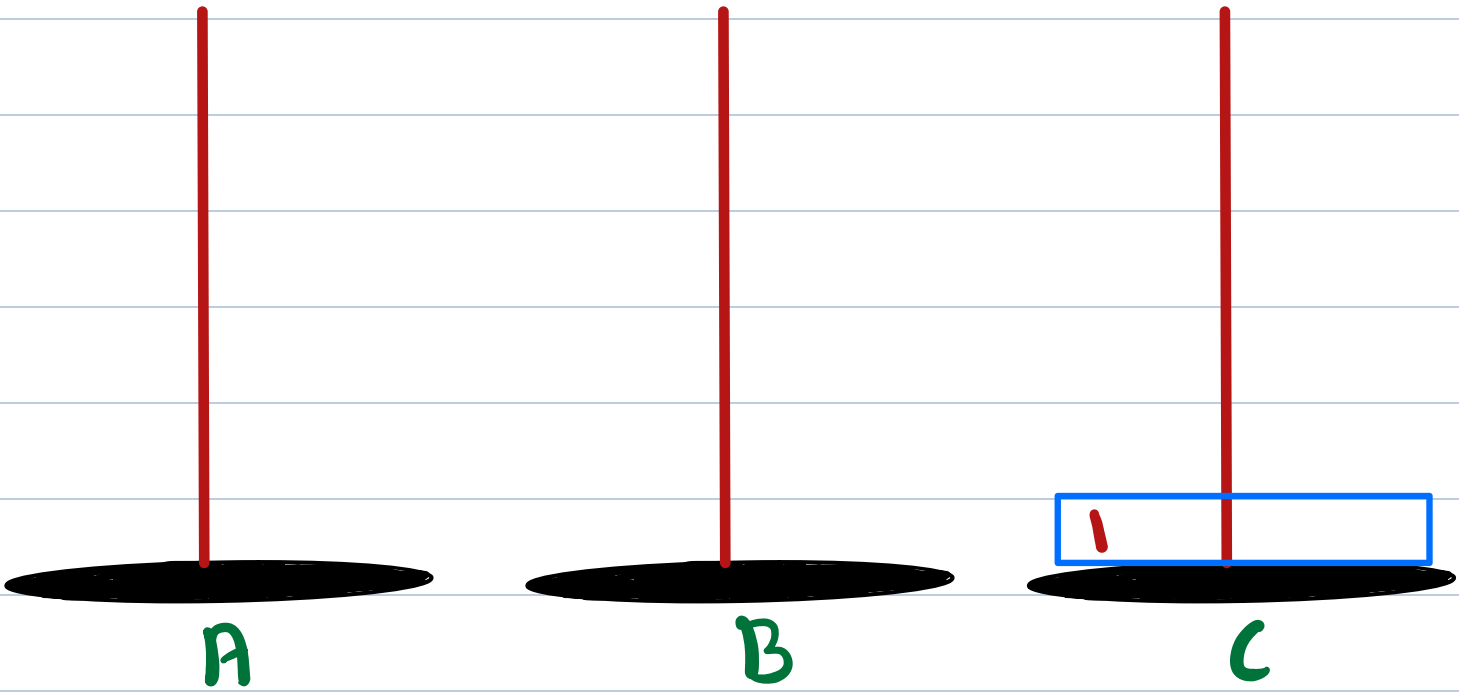
1) A larger disk cannot be placed on a smaller disk.

2) Only 1 disk can be moves at a time.

Print the movement of disks. (minimum steps).

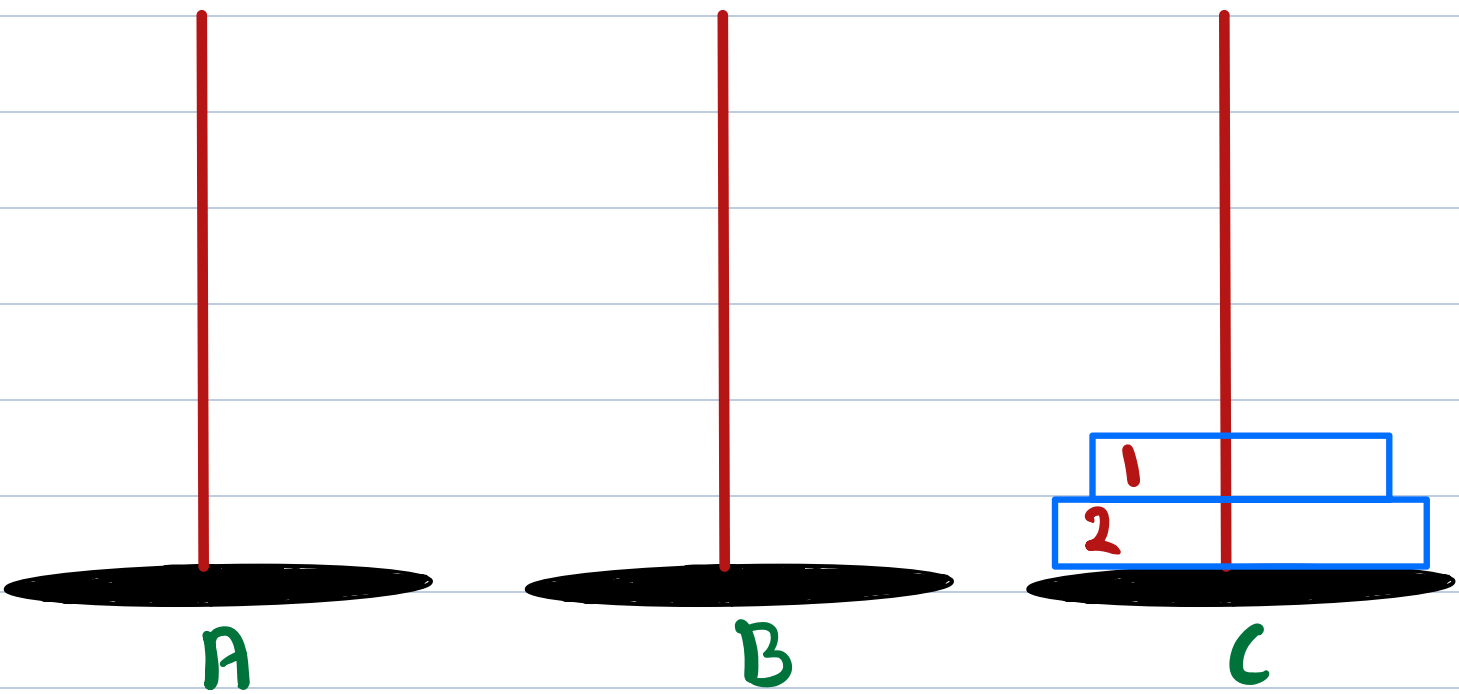
1 A to C

Eg. N = 1

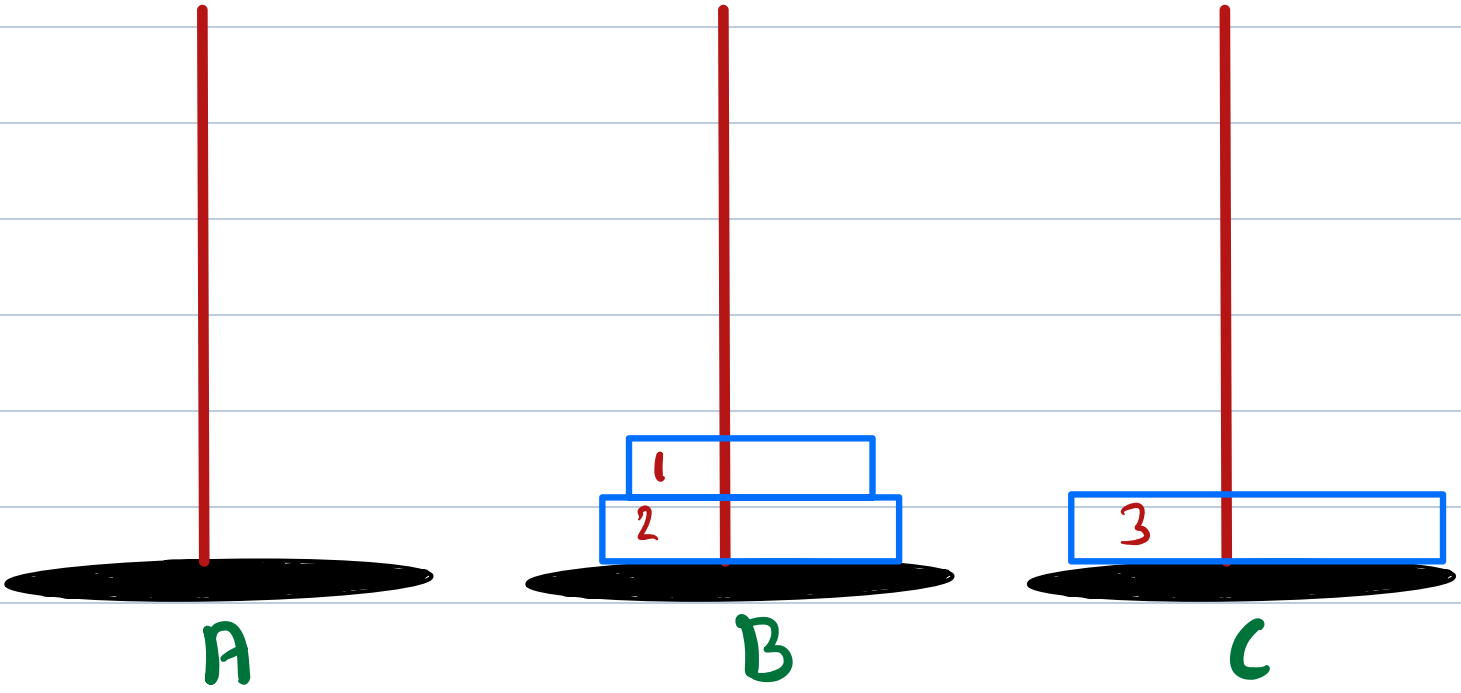


1 A to B
2 A to C
1 B to C

Eg. N = 2



N = 3



1 A to C
2 A to B
1 C to B
3 A to C → moving n^{th} disk from A to C.
1 B to A
2 B to C
1 A to C

move N-1 disk from A to B using help of tower C.

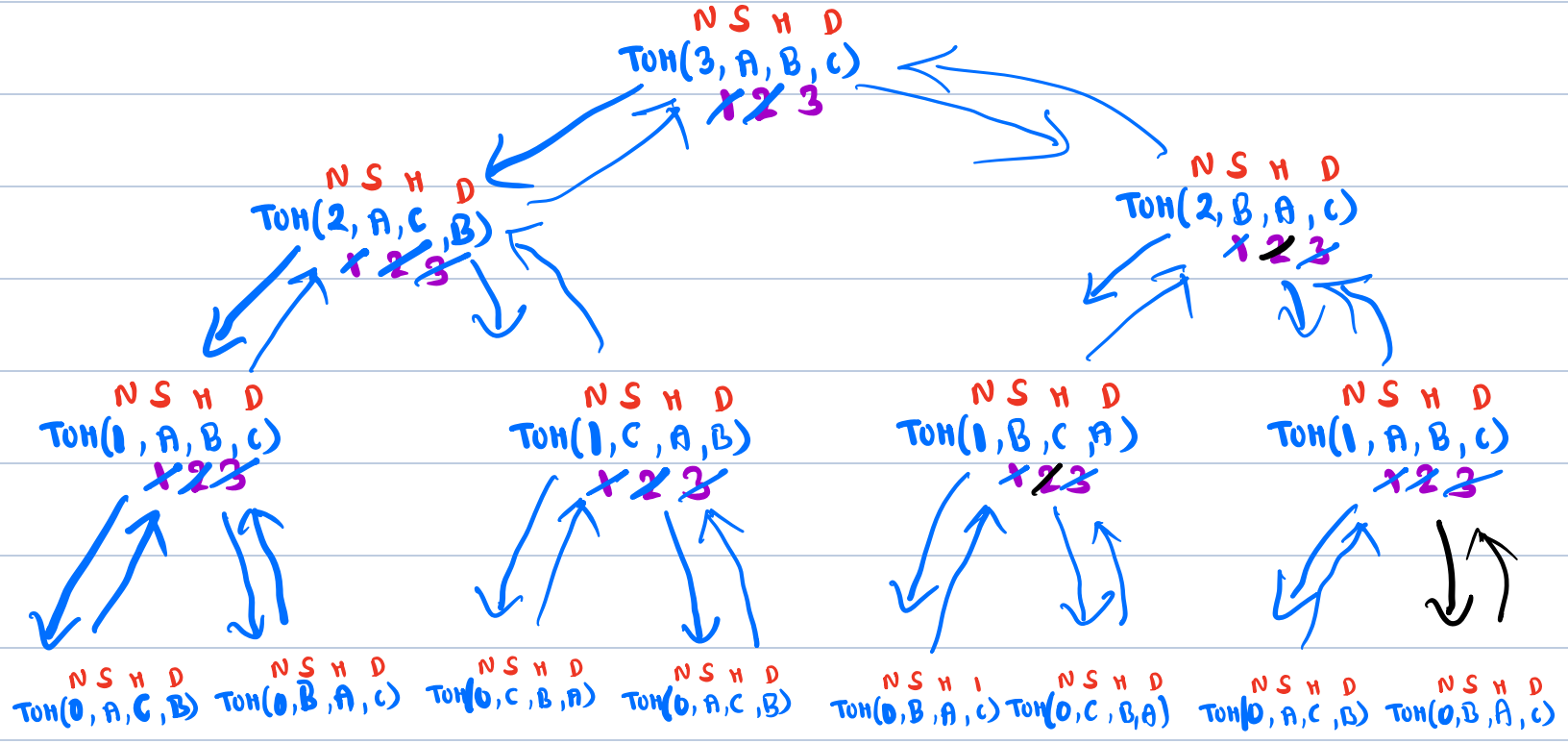
move N-1 disk from B to C by taking help of tower A.

#Code

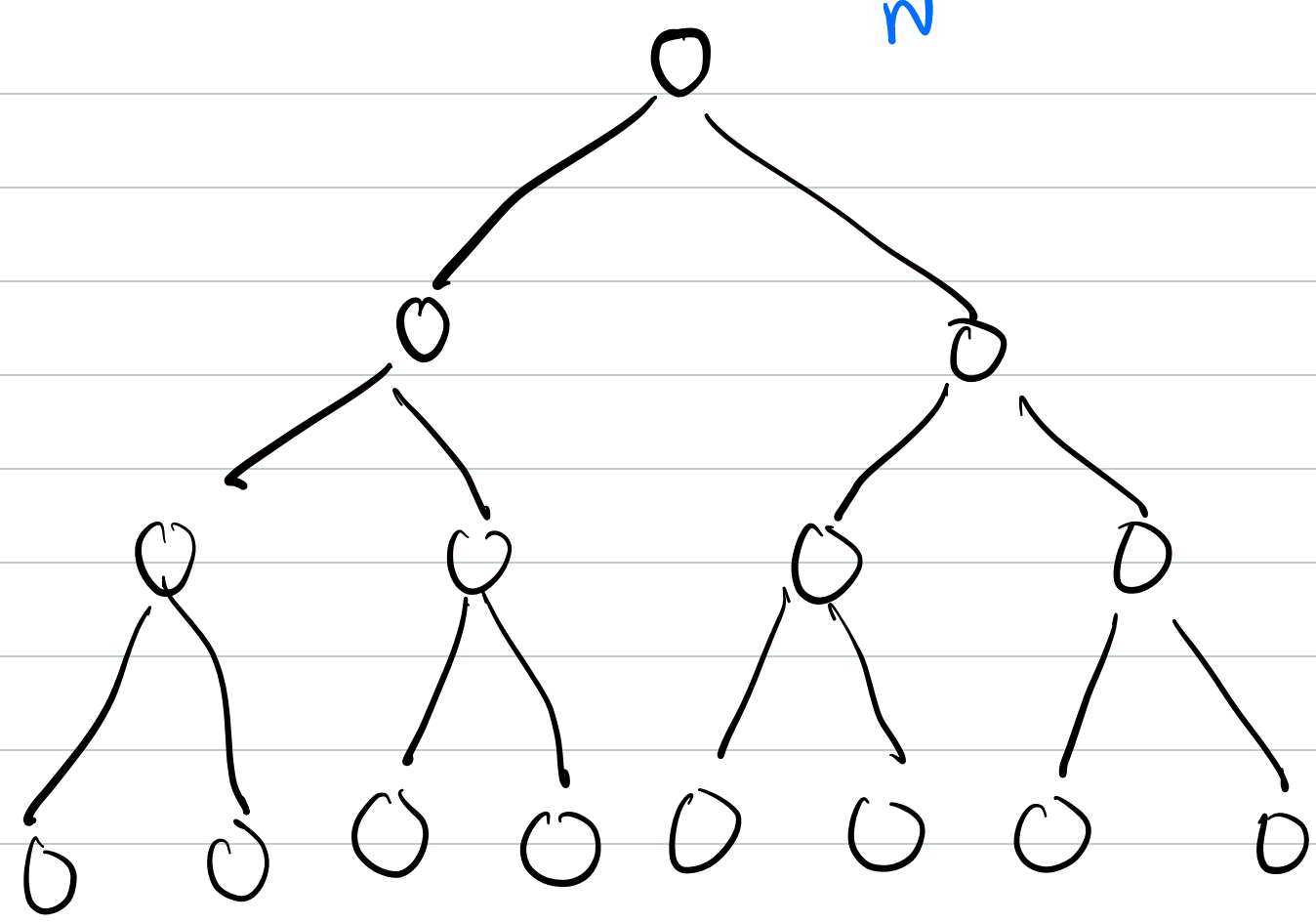
S H D

```
void TOH(N, A, B, C)
{
    if(N==0) return;
    TOH(N-1, A, C, B)  -- (1)
    Print("move disk {N} from {S} to {D}");
    TOH(N-1, B, A, C)  -- (2)
}
```

DRY RUN → N=3



Disk	1	A → C		1	A	to	C
Disk	2	A → B		2	A	to	B
Disk	1	C → B		1	C	to	B
Disk	3	A → C		3	A	to	C
Disk	1	B → A		1	B	to	A
Disk	2	B → C		2	B	to	C
Disk	1	A → C		1	A	to	C



TC: $O(2^N)$

SC: $O(N)$

9) Given an integer Array A, and integer B is also given. you have to find the indices of B in array A.

Eg. A[] = { 4, 5, 3, 5, 2, 1, 5 }
(0 1 2 3 4 5 6)
B = 5

Output: { 1, 3, 6 }

```
int[]  recur(A[], B, index, cnt)
{
        if (index == A.length)
        {   return new int[cnt];
        }
        if (A[index] == B)
        {
            int[] ans = recur(A, B,
                              index+1, cnt+1);
```

(0 0)

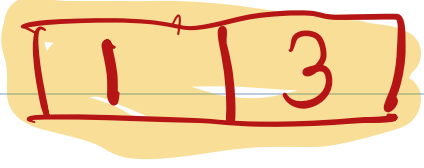
```
            ans[count] = index;
        }
        else
        {
            int[] ans = recur(A, B,
                              index+1, cnt);
        }
    }
    return ans;
}
```

B = 5

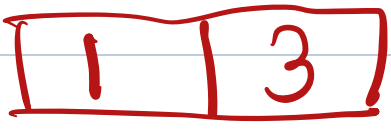
A[] = {4, 5, 3, 5}

index

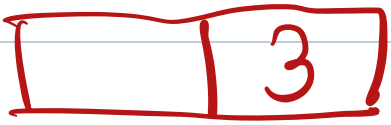
recur(A, 5, 0, 0)



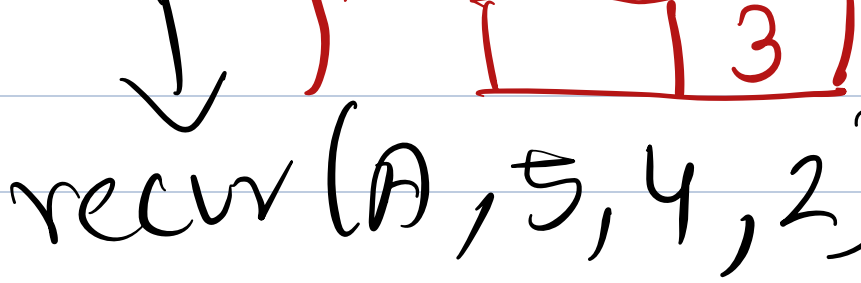
recur(A, 5, 1, 0)



recur(A, 5, 2, 1)



recur(A, 5, 3, 1)



recur(A, 5, 4, 2)