Power function → easy
fast power

Indices of an Array

Tower of Hanoi

Problem Statement

Given two integers **a** and **n**, find $\mathbf{a^n}$ using recursion.

Input

```
a = 2
n = 3
```

$2^3 = 8$

$a = 3$, $n = 4$ $\quad 3^4 = 81$

Output

```
8
```

```
int pow( int a, int n) {
```

a^n

3

1. P , SP
   ↳ ans

2. P = $\underbrace{SP}_{ans}$ + EW

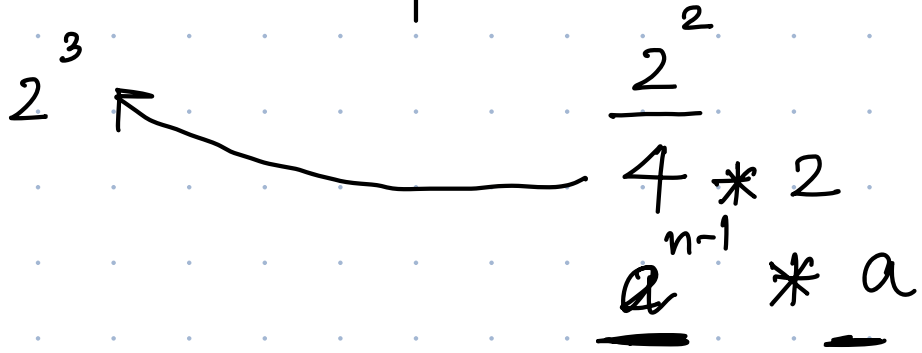
3. High level code

4. dry run ⟶ base case

$P = a^n$

$= pow(a, n)$

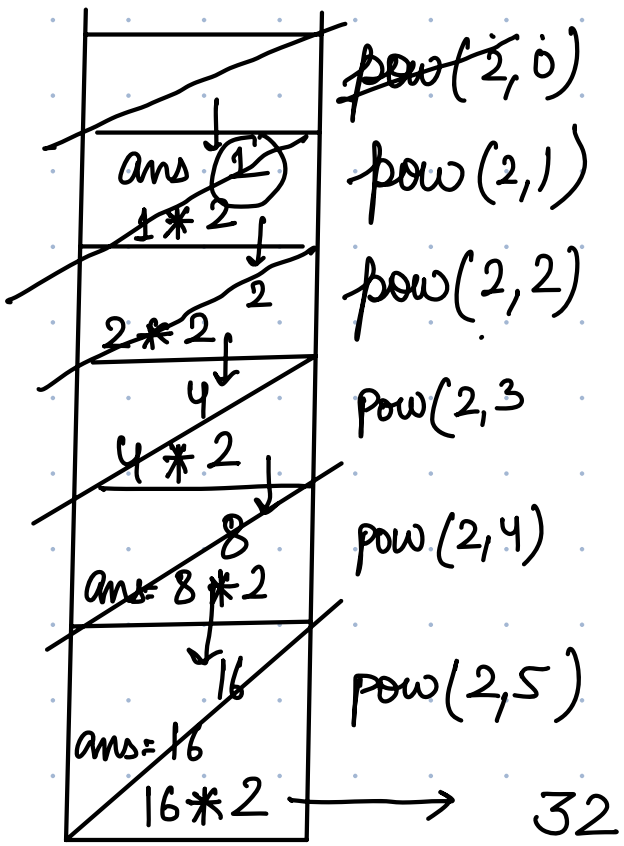
$SP = pow(a, n-1)$



```
int pow( int a, int n) {
    ans = pow( a, n-1);
    return ans * a
}
```

I

$N = 10^6 \rightarrow 10^6$ ✓




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

↓

Stack overflow

Is

10 mins

Optimized approach

a, n → a, n-1

a, n → a, $\frac{n}{2}$

P = pow(a, n)

SP = pow(a, $\frac{n}{2}$)

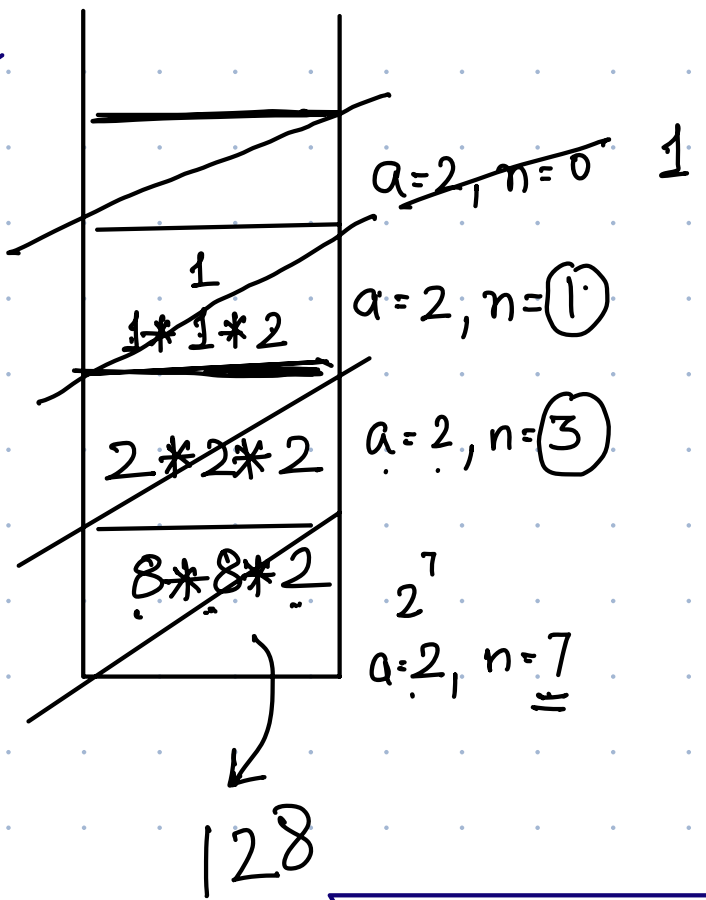
$2^4 = 16$

2^8	=	$2^4 * 2^4$
256	=	16 * 16
.	=	$a^{n/2} * a^{n/2}$

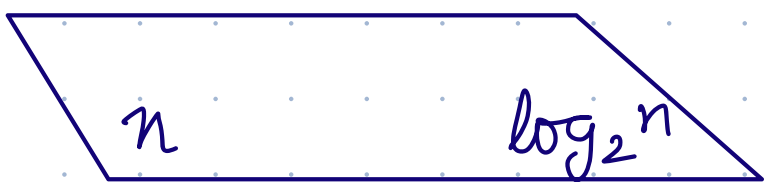
(7)

2^7	$2^{7/2}$ 2^3	8
128	8 * 8 * 2	
	$2^{n/2} * 2^{n/2} * 2$	
	$a^{n/2} * a^{n/2} * a$	

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



$\log_2 n$



n = 2 40 000 → $\log_2 240000$ ≈ 18

240 000

Problem Statement

Given an array of integers A with N elements and a target integer B, the task is to find all the indices at which B occurs in the array.

It is guaranteed that the target B, exist atleast once in the Array A.

Example

Input :

```
A = [4, 5, 3, 1, 5, 4, 5]
B = 5
```

Output :

```
[1, 4, 6]
```

Find all the Indices at which B occurs in the given array, A:

```
A = [1, 2, 3, 1, 1]
B = 1
```

[0, 3, 4]

Array index

P = int[] allI(int[] A, int B)

A = [i → N-1 : 1, 2, 3, 1, 1], B = 1, i = [0, 3, 4]

[1, i+1 → : 2, 3, 1, 1], B = 1

[3, 4]

```
int[] allI(int[] A, int B, int i ) {

    int[] ans = allI(A, B, i+1)
    if (A[i] == B) {
        int[] newA = new int[ans.length+1]
        newA[0] = i;
        for (i = 1; i < newA.length i++) {
            newA[i] = ans[i-1]
        }
        return newA;
    }
    else {
        return ans;
    }
}
```

[3,4]

P = A = [1 2 3 1 1], B = 1, i = 0 => [0, 3, 4]

SP = A = [1, 2, 3, 1, 1], B = 1, i = 1

ans = [3, 4]

P → A = [1, 2, 3, 1, 1], B = 1, i = 0

↓

[0, 3, 4]

SP : A = [1, 2, 3, 1, 1], B = 1, i = 1

↓

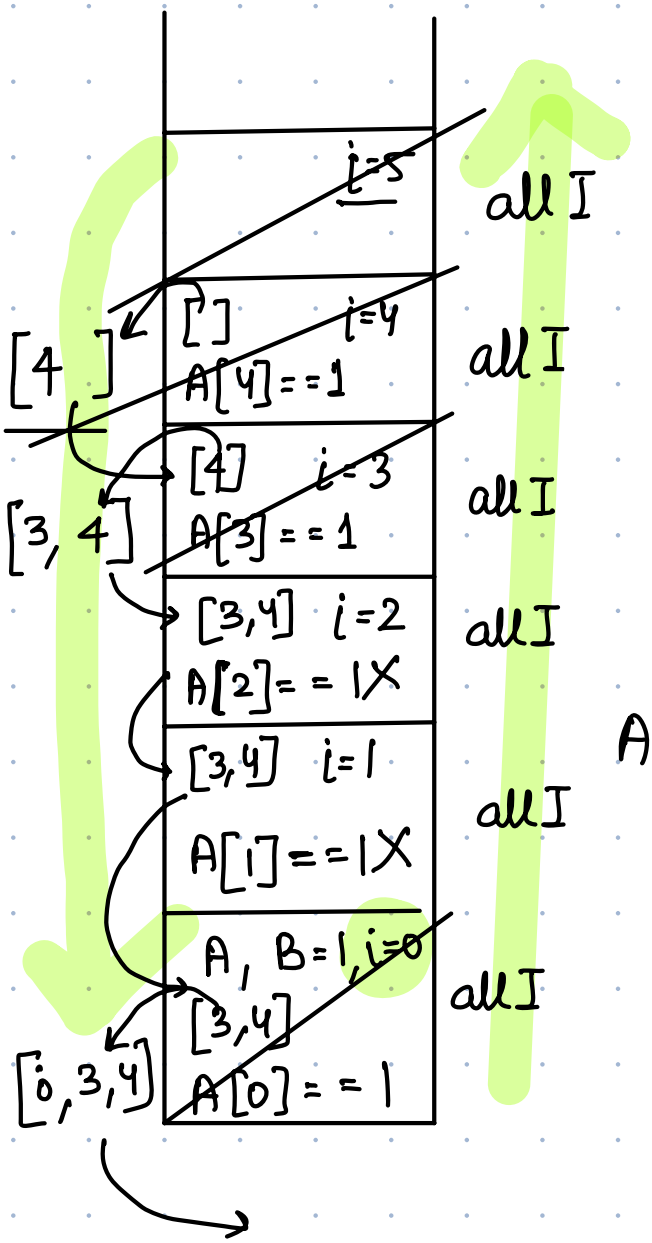
[3, 4]

P = SP + EW

ans

```
if (A[i] == B) {
    newA ⟶ A.+1
    newA[0] = 0/i
    for (i = 1; i < newA.length; i++) {
        newA[i] = ans[i - 1]
    }
    return newA.
}
```

```
else {
    return ans
}
```



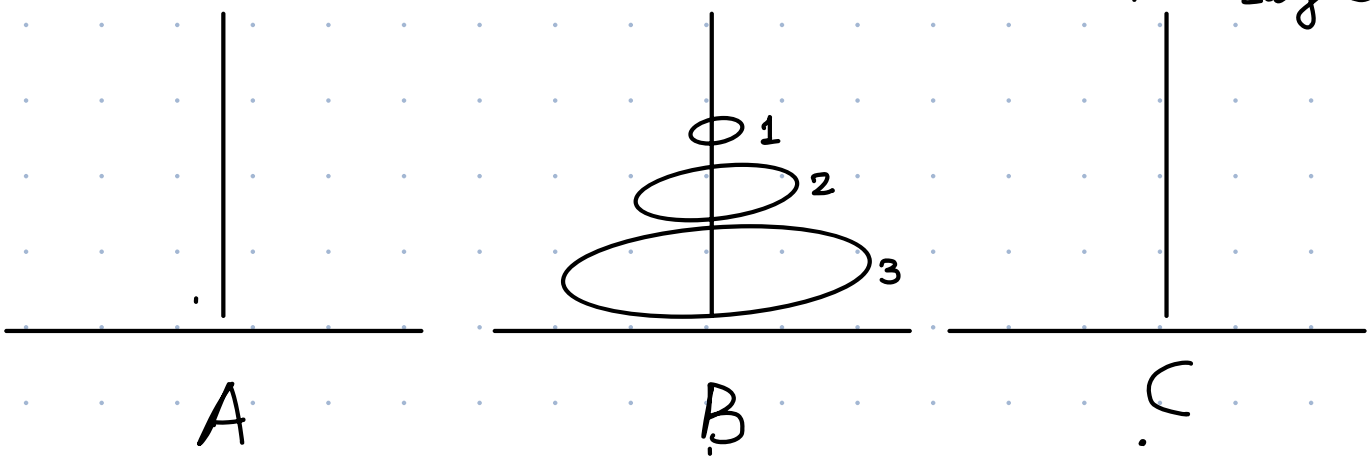

```
int[] allI(int[] A, int B, int i) {
    if(i == A.length) return new int[0];
    int[] [3,4] ans = allI(A, B, i+1)
    if (A[i] == B) {
        int[] newA = new int[ans.length+1]
        newA[0] = i;
        for (i=1; i < newA.length i++) {
            newA[i] = ans[i-1]
        }
        return newA;
    }
    else {
        return ans;
    }
}
```

$$A = [\overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{1}, \overset{4}{1}]$$

Q4 Tower of Hanoi → Print ins to move disks from A to B → following rules.

1. One disk at a time
2. Small D can be placed on Large D

N = 3



1	A	→	B
2	A	→	C
1	B	→	C
3	A	→	B

1	C	→	A
2	C	→	B
1	A	→	B

N = 3 → 7 ins

⇒ N = 4 → 15 ins

N = 5 → 31 ins

N

$2^N - 1$

P: void ToH (int n, A, B, C)
src dest ex

⇒ Print instructions of moving n disks from A to B, following rules

N = 3

1 A → B
2 A → C
1 B → C

N-1 DISKS A to C

Print
3 A → B
N

1 C → A
2 C → B
1 A → B

N-1 disks C to B

1 A → C
2 A → B
1 C → B
3 A → C
1 B → A
2 B → C
1 A → C

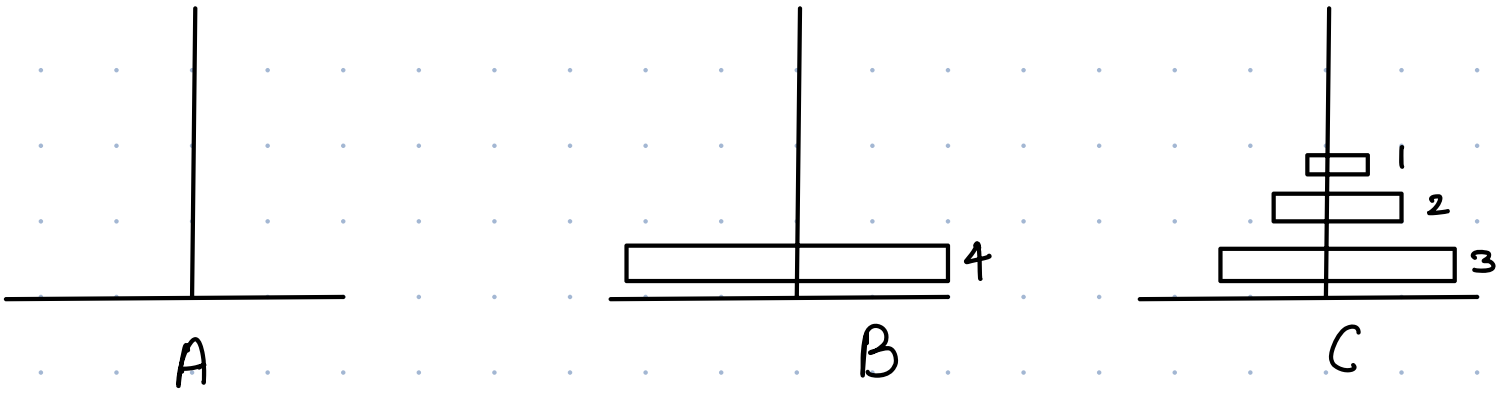
N-1 disks A to C

Print
4 → A to B
N

1 C → B
2 C → A
1 B → A
3 C → B
1 A → C
2 A → B
1 C → B

N-1 disks C to B

N → $2^N - 1$



1 A → C
2 A → B
1 C → B
3 A → C
1 B → A
2 B → C
1 A → C

4 → A to B

1 C → B
2 C → A
1 B → A
3 C → B
1 A → C
2 A → B
1 C → B

Print instructions of moving n disks from A to B, following rules

SP Print instructions of moving n-1 disks from A to ~~B~~ C, following rules

W → N → A to B

SP Print instructions of moving n-1 disks from C to B, following rules

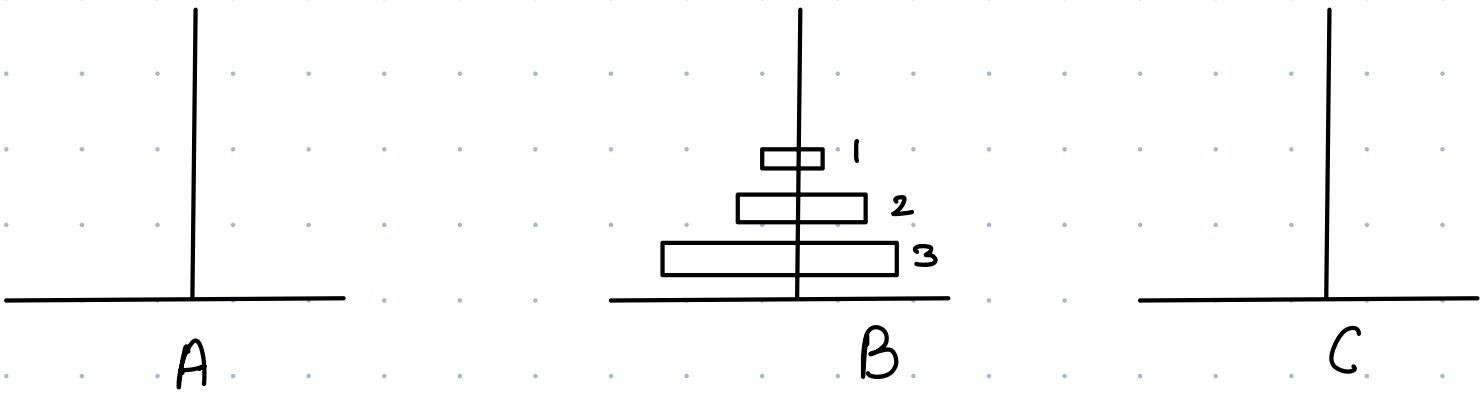
Print instructions of moving n disks from A to B, following rules

- SP: Print instructions of moving n-1 disks from A to C, following rules
  - W → N → A to B
- SP: Print instructions of moving n-1 disks from C to B, following rules

```
void TOH (int n, A, B, C) {
    TOH (n-1, A, C, B)
    print (n → A to B)
    TOH (n-1, C, B, A)
}
```

src dest ex

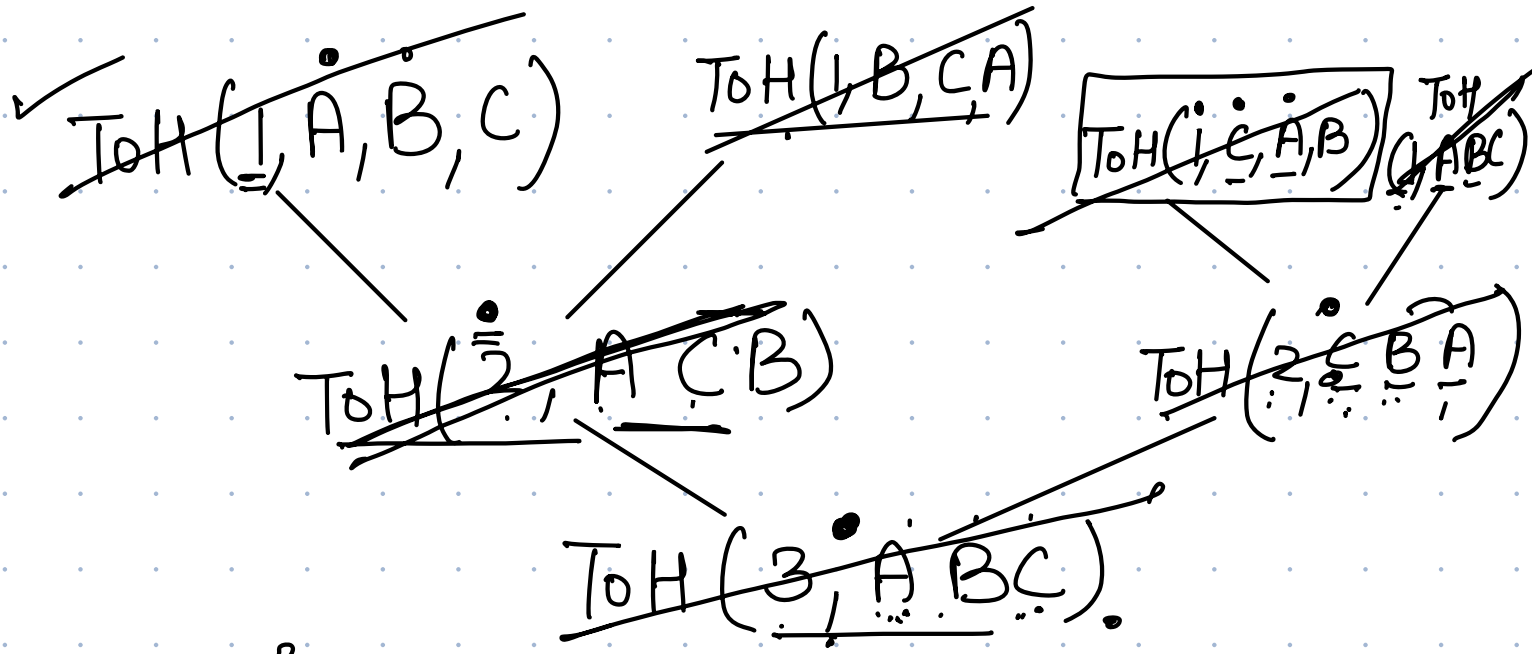
```
void TOH (int n, A src, B dest, C ex) {
    if (n==1) { print N / A → B / return }
    1 TOH (n-1, A, C, B)
    2 print (n → A to B)
    3 TOH (n-1, C, B, A)
}
```



1 A B
2 A C
1 B C

3 A → B

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



N=3 2^3-1 N=4 15 N=5 31

```
void TOH (int n, A(src), B(dest), C(ex)) {
    if(n==1) { print N 'A→B' return; }
    TOH (n-1, A, C, B)      O(1)
    print (n → A to B)      O(1)
    TOH (n-1, C, B, A)      O(1)
}                           O(1)
```

$O(1)$ [1, 2, 3] ?

TC = No of recursive calls * TC of 1 recursive call

$= (2^N - 1) * O(1)$

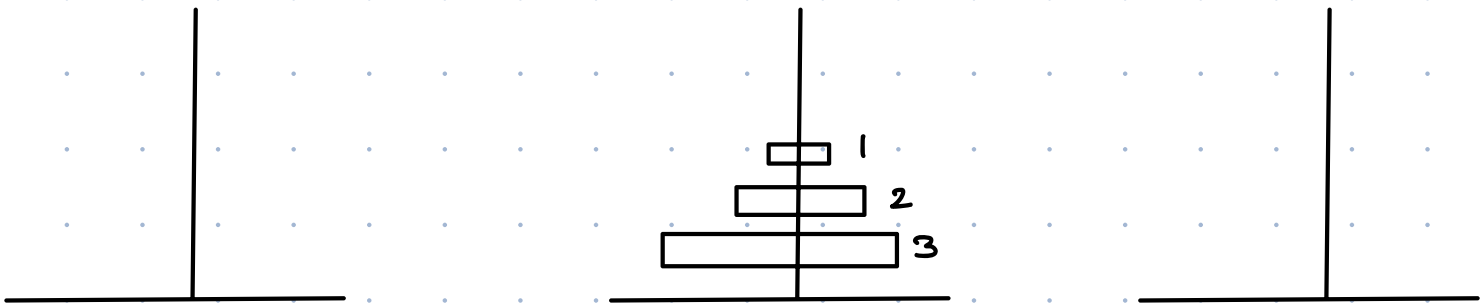
$= O(2^N)$

Space Complexity = max stack size * SC of 1 call

$= N * O(1)$

$= O(N)$

```
void TOH (int n, A src, B dest, C ex) {
    if(n==1) { print N /A→B/ return }
    1 TOH (n-1, A, C, B)
    2 print (n → A to B)
    3 TOH (n-1, C, B, A)
}
```



1 A B
2 A C
1 B C

3 A B

1 C A
2 C B
1 A B