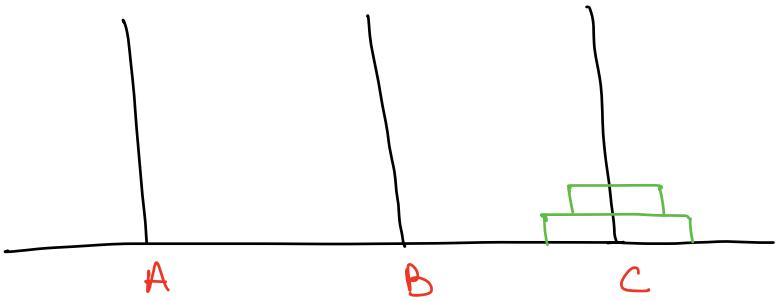
Tower of Hanoi

→ 3 towers A, B, C

→ There are N disks placed on A initially.

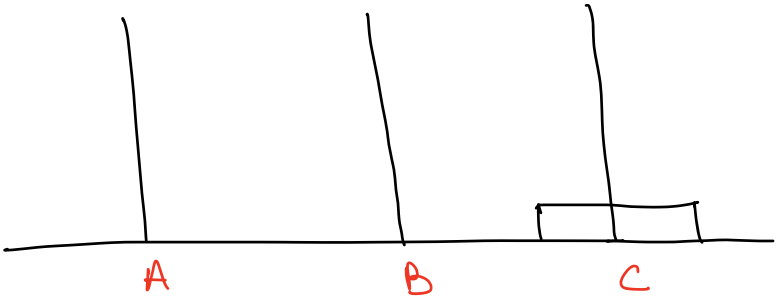
→ Move all the disks from A to C using B, making sure that at no point a smaller disk is below a larger disk.

Ex N = 2



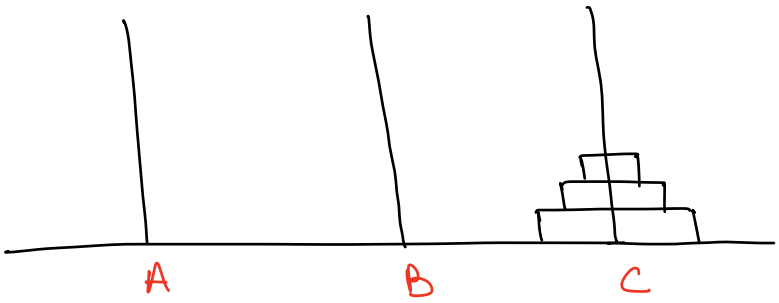
1) A → B
2) A → C
3) B → C

Ex N = 1



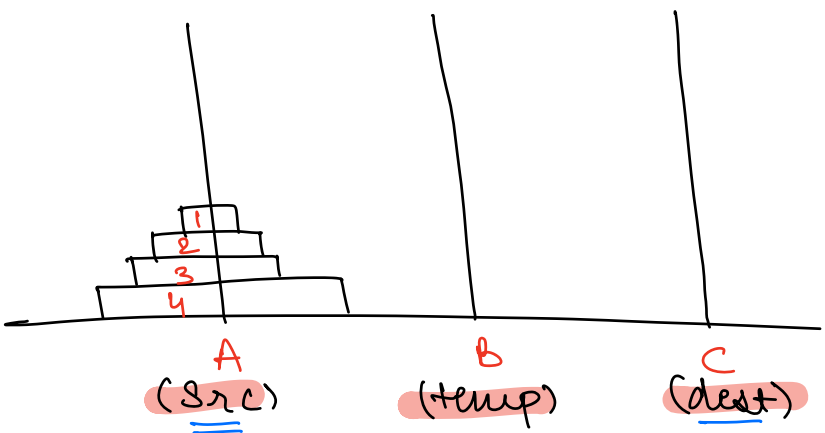
A → C

Ex N=3



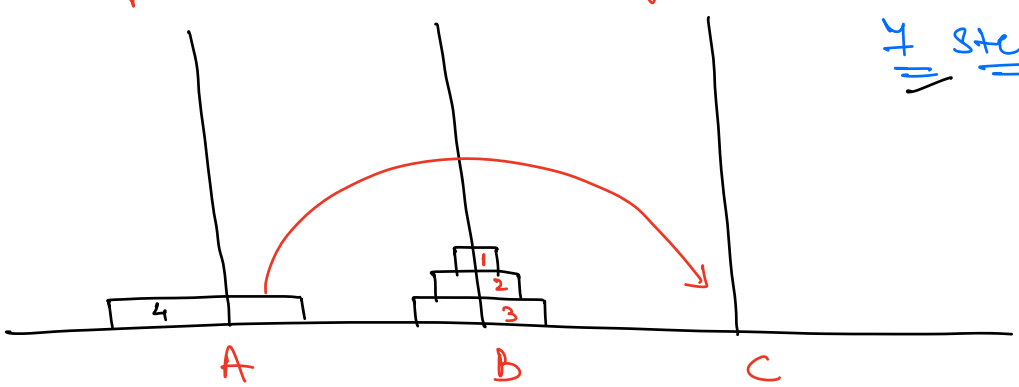
A → C
A → B
C → B
A → C
B → A
B → C
A → C

N=4 : Transfer 4 disks from tower (A) to (C) using tower B.

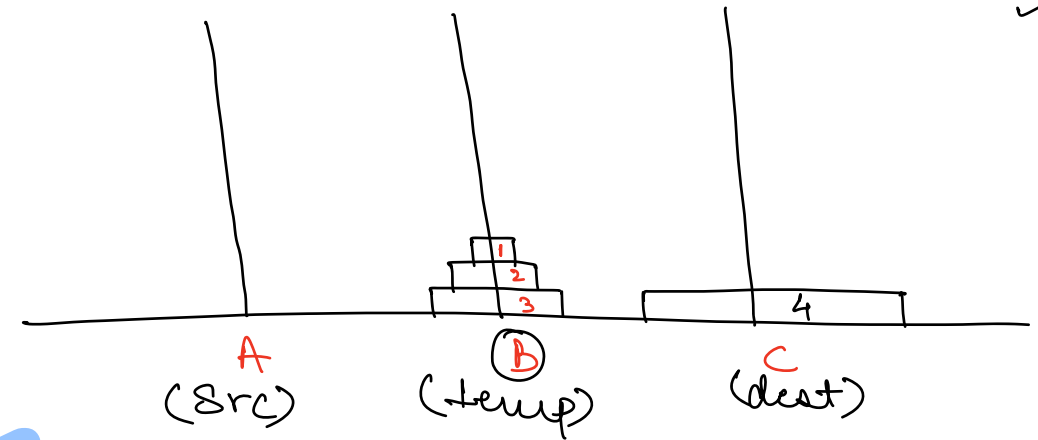


i) Transfer 1, 2, 3 disks from (A) to (B) using (C)

7 steps



ii) Transfer disk 4 from (A) to (C).



iii) Transfer 1, 2, 3 disks from (B) to (C) using (A)

↳ 7 steps

⇒ Total (15) steps

// Print steps to move N disks from src to dest using temp.

```
void TOH( N, src, dest, temp) {
    if (N == 0) {
        return;
    }
    TOH (N-1, src, temp, dest);
    print ("src → dest");
    TOH (N-1, temp, dest, src);
}
```

N=3

A → C
A → B
C → B
A → C
B → A
B → C
A → C

Src Dest Temp

TOH(3, A, C, B)

Src Dest Temp

TOH(2, A, B, C)

TOH(1, A, C, B)

TOH(0, ——) ✓

print(A → C)

TOH(0, ——)

print(A → B)

TOH(1, C, B, A)

TOH(0, ——)

print(C → B)

TOH(0, ——)

print(A → C)

TOH(2, B, C, A)

TOH(1, B, A, C)

TOH(0 ——)

print(B → A)

TOH(0, ——)

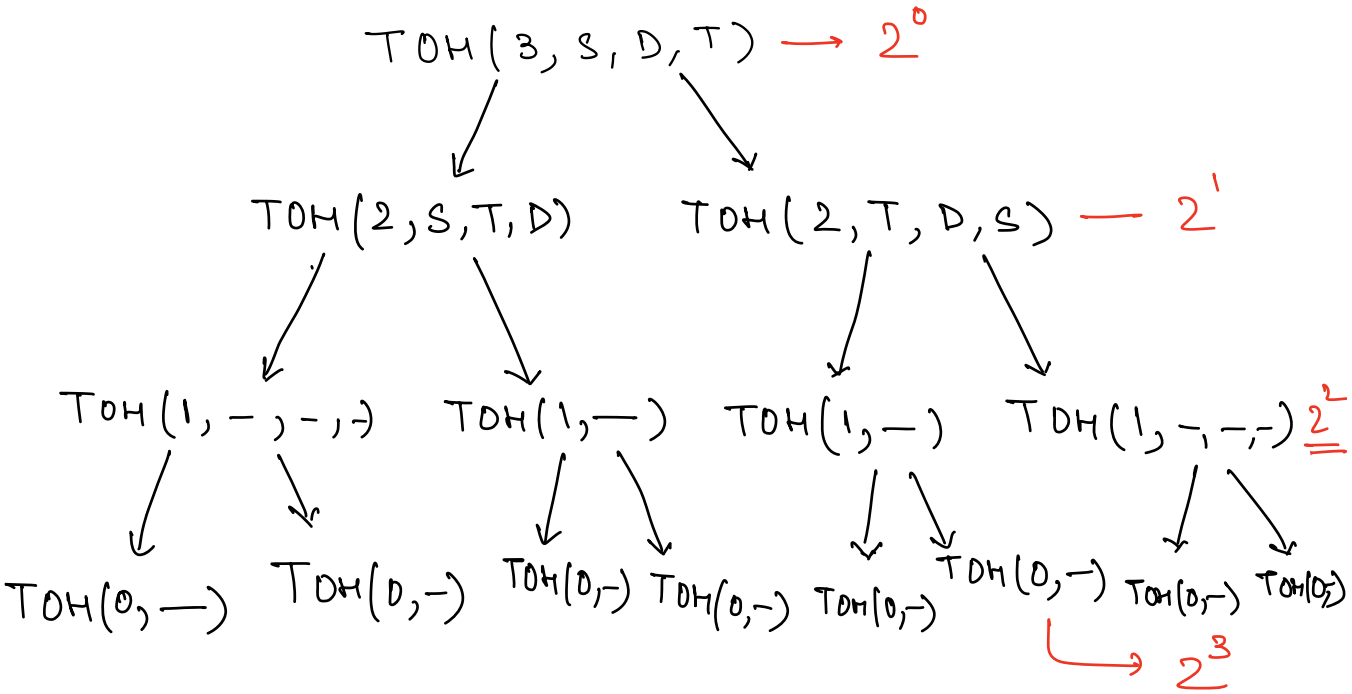
print(B → C)

TOH(1, A, C, B)

TOH(0 ——)

print(A → C)

↳ TOH(0, —)



TC: (# of function calls) * (work done in one fun^)

No. of fun^ calls = no. of nodes in Recursion Tree

$= 2^0 + 2^1 + 2^2 + \cdots 2^{N-1}$

$a = 2^0$

$r = 2$

# of terms $= x = N$.

$$\text{Sum} = \frac{2^0(1-2^N)}{1-2} = 2^N - 1.$$

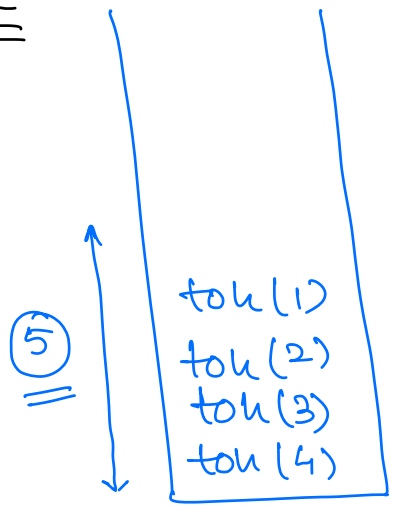
TC: $O(2^N)$

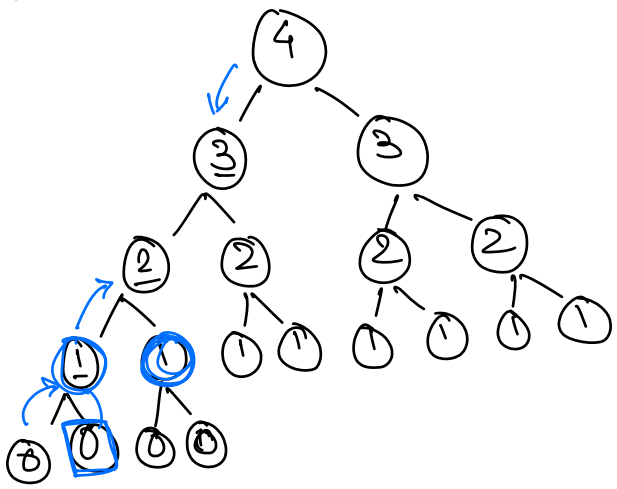
Recurrence Relation

$T(N) = 2T(N-1) + 1$

TC: $O(2^n)$

SC





SC: O(Height)

SC: O(N)

Sorting Basics

Arranging the data in a specific order based on same parameter.

{1, 2, 3, 4, 5} ⇒ Ascending order based on the value.

{5, 4, 3, 2, 1} ⇒ Descending order based on the value.

{7, 2, 4, 9, 6} ⇒ Asc. order based on the no. of factors.

↓ ↓ ↓ ↓ ↓

2 2 3 3 4

Why Sorting ?

↳ It makes searching faster.

Ex

	12th %
Sawan	77
Siddhesh	70
Deepti	76
Shiv	98
Ayush	70
Bhavesh	57
Shambhu	78
Sandhya	85
Deepak	85

Sort →

Bhavesh	57
Siddhesh	70
Ayush	70
Deepti	76
Sawan	77
Shambhu	78
Sandhya	85
Deepak	85
Shiv	98

Stable Sort

⇒ If two data points are equal then their relative order in sorted data should be maintained.

Inplace Algorithm

↳ SC: O(1)

No extra space.