Your Ultimate Guide To Landing Top AI roles

2.3.9

Tower of Hanoi

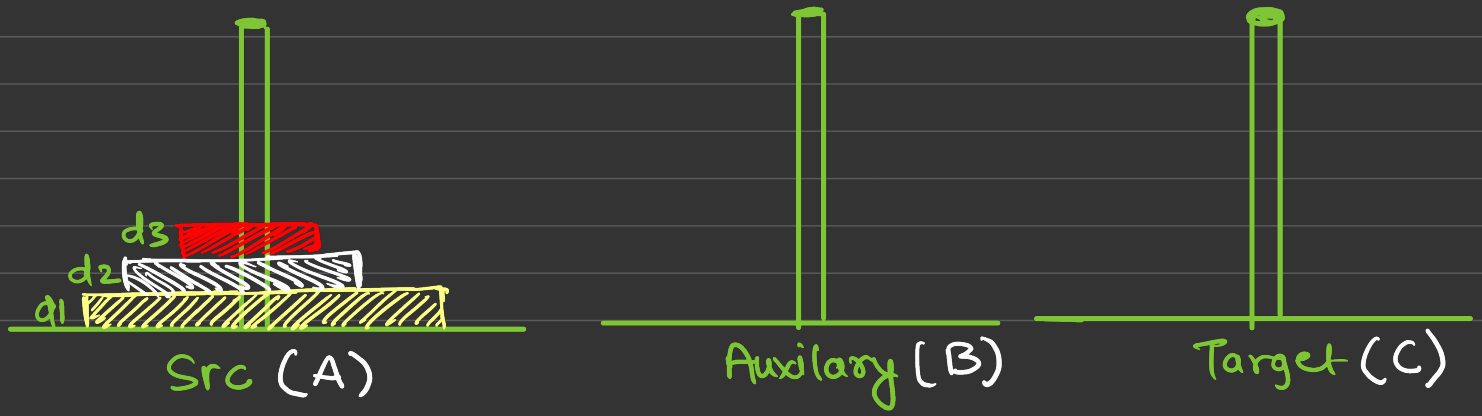
Problem: Tower of Hanoi Puzzle consists of 3 towers namely source, Auxilary and Destination. you have n disks arranged in the source tower in such a way that smaller disk is kept over larger disks.

Goal: you need to move n disks from src tower to dest tower using Aux tower by below rules

① only 1 disks can be moved at a time

② No disks can be placed on top of smaller disks.

For n=3




```
def TOH(n, src, Target, aux):
```

Base Case

Recursive Case

Base Case


```
def TOH(n, src, Target, Aux)
```

→ Move 1 from A to C

Recursive Case

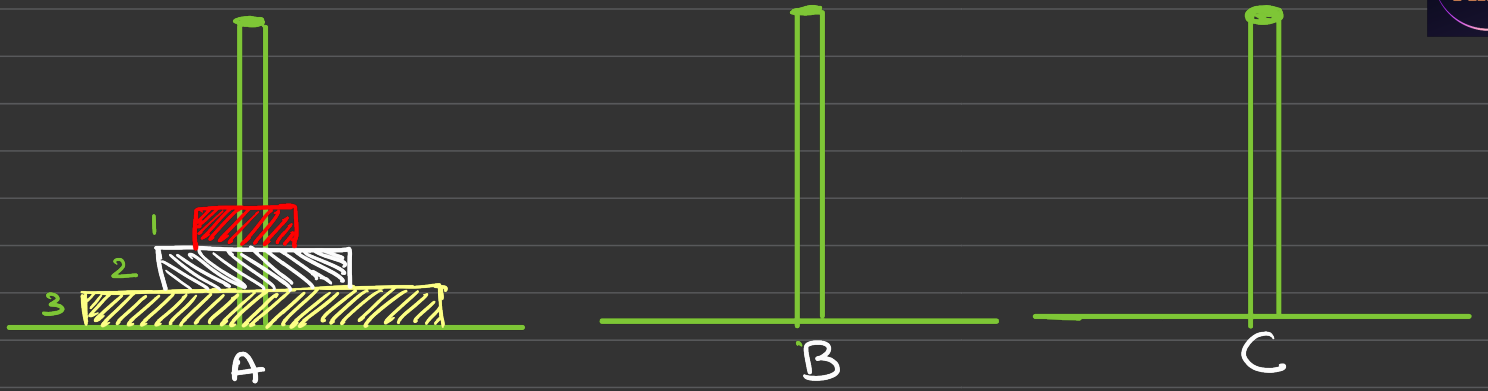

```
def TOH(n, src, Target, Aux)
```

→ Move 1 from A to B ⇒ TOH(1, A, B, C)

→ Move 2 from A to C

→ Move 1 from B to C ⇒ TOH(1, B, C, A)

Recursive Case




```
def TOH(3, src, Target, Aux)
    → Move 1 from A to C
    → Move 2 from A to B
    → Move 1 from C to B
    → move 3 from A to C
    → move 1 from B to A
    → move 2 from B to C
    → move 1 from A to C
```

TOH(2, A, B, C)

TOH(2, B, C, A)

```
def TOH(n, src, Target, aux):
    if n==1:
        print(f" Move 1 disk from {src} to {Target}");
    TOH(n-1, src, Aux, Target)
    print(f" Move {n} disk from {src} to {target}")
    TOH(n-1, Aux, Target, src)
```

Base Case

Recursive Case

Time Complexity

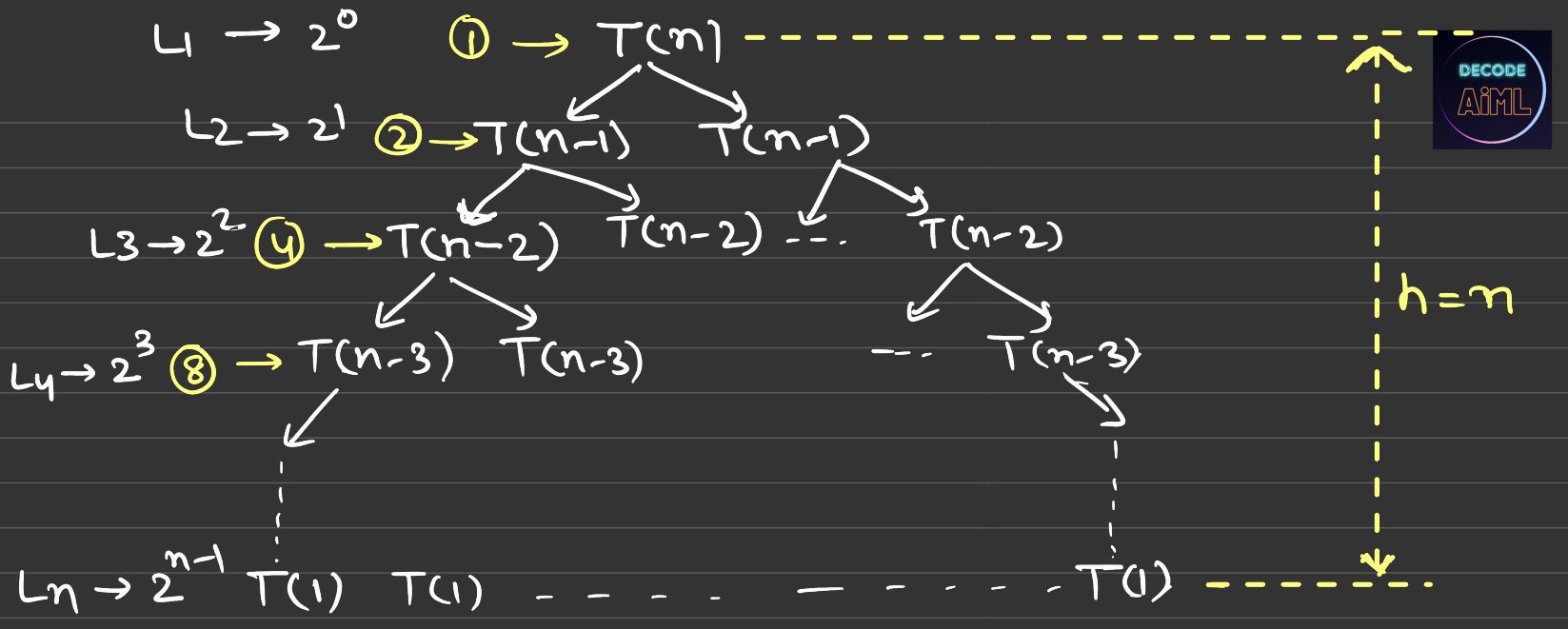
```
def TOH(n, src, Target, aux):  → T(n)
    if n==1:
        print(f" Move 1 disk from {src} to {Target}");   } O(1)
    TOH(n-1, src, Aux, Target)  → T(n-1)
    print(f" Move {n} disk from {src} to {target}")  → O(1)
    TOH(n-1, Aux, Target, src)  → T(n-1)
```

$$T(n) = O(1) + 2 \cdot T(n-1) + O(1)$$

$$T(n) = 2T(n-1) + O(1)$$
$$T(1) = O(1)$$

← Recurrence Relation

→ Let's solve this using Recursion Tree



Total # Recursion call = $2^0 + 2^1 + 2^2 + \dots + 2^{n-1}$

[GP] $S_n = \frac{a(r^n - 1)}{r - 1} = \frac{2 \cdot (2^{n-1} - 1)}{2 - 1}$

$= \boxed{2^n - 2}$

Total time Complexity = # Recursion Calls × Time at each rec. call

$$= (2^n - 2) \times O(1)$$

$$= (2^n - 2) \times c$$

$$= c \cdot 2^n - 2c$$

$$= O(2^n)$$

$$\boxed{T(n) = O(2^n)}$$

Space Complexity

→ Input space = O(1)

Auxilary space = Max depth of Recursion [From Recursion Tree]

= O(n)

Space Complexity = O(n)

Total space complexity = O(n)


Like
Subscribe