Follow

# Data Structure & algorithm

## Recursion-3


**Tower of Honoi**

#Vale_freeConten
t

SWIPE

# Tower of Hanoi

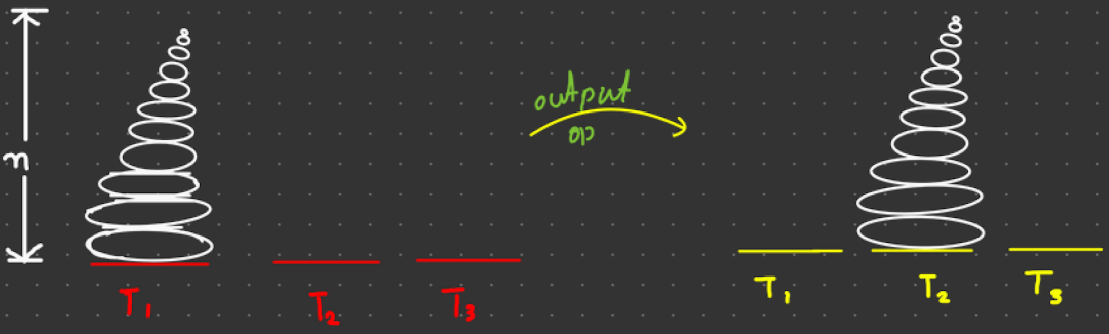
**Statement :-** We have 3 Tower ($T_1, T_2, T_3$). Initialy on tower $T_1$ we have n number of coins. In final all the coins should be move from $T_1$ to $T_2$. You can use additional Tower $T_3$ in from of helper

**Condition :-**

1. In $T_1$ coins are arranged in Decreasing order/ size.
2. At one time only one coin can be moved.

**AIM :-** How many moves will be require to achive that.

$T_1 \longrightarrow T_2$

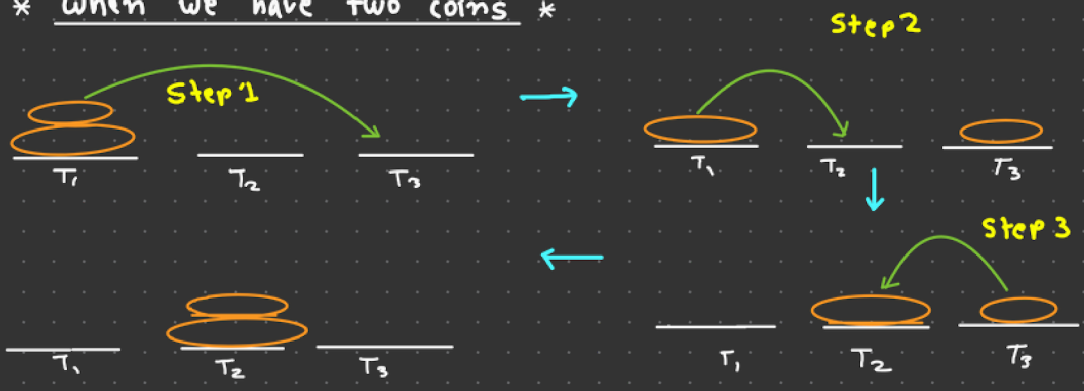



**Ans**

* if we have one coin *

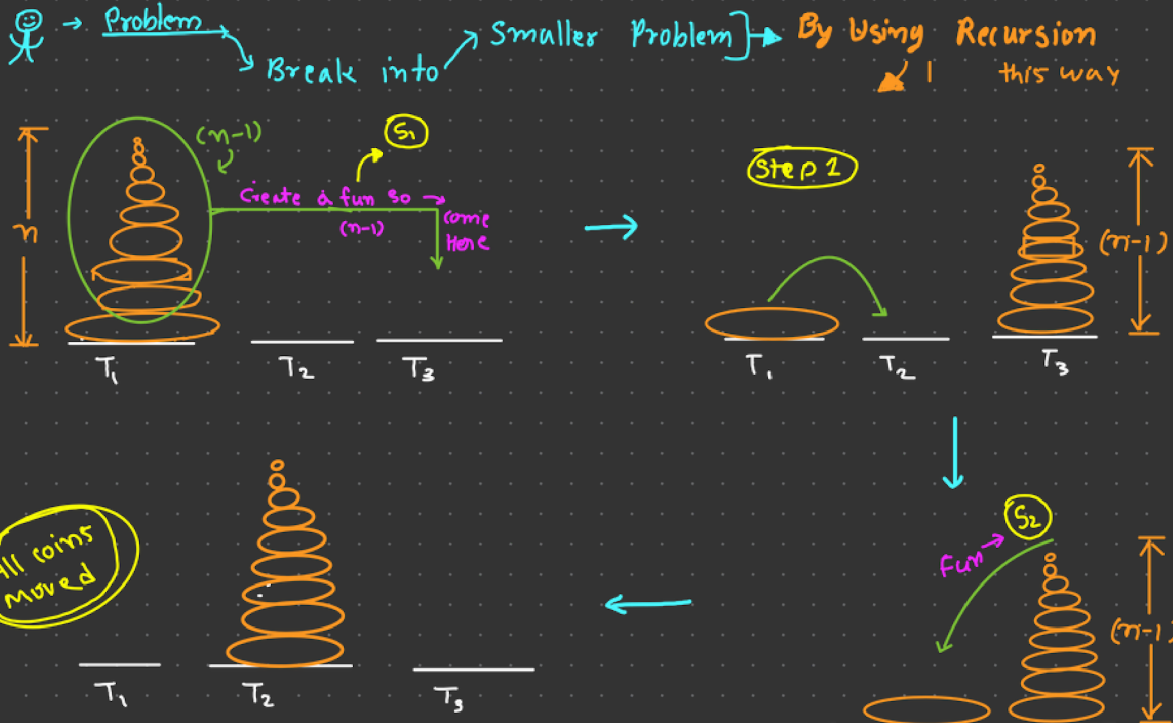

No. of step = 1

## * When we have two coins *




I needed = 3 steps

As the number of coins increase doing it manually became harder.




Total Steps → $S_1 + 1 + S_2$

Let move is the function, How the move function works?

Here we can say, $T_1$ → Starting point (S)

$T_2$ → Destination point (d)

$T_3$ → Helper (a)

Fun → move (n, S, d, a)

n → No. of coins, d → destination, S → Starting, a → Helper

↳ n coins, S → d using a } → Get no. of move that required

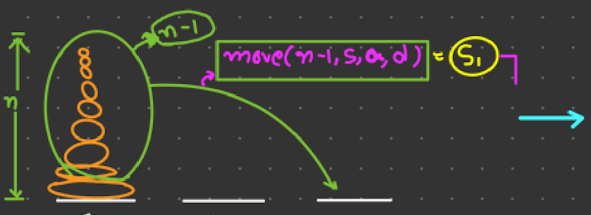
ex → mean(5, s, d, a)

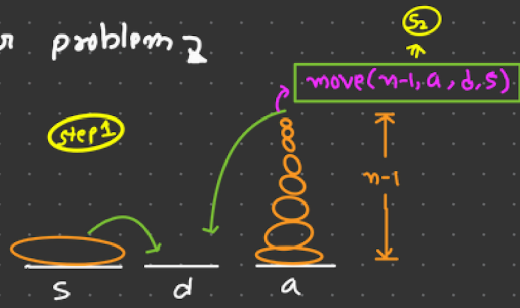
↳ 5 coins, S → d using a } → Get no. of moves that required

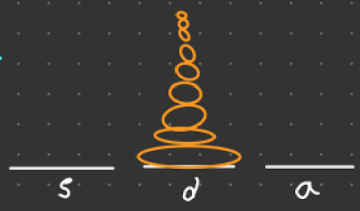
ex → move(n-1, S, a, d)

↳ (n-1) coins, S → a } → Get no. of moves that required.

⇒ using this mean function in our problem?




mean(n, s, d, a) ⇒

mean(n-1, S, a, d)
+
1
+
mean(n-1, a, d, S)

This will get the no. of step that we need to solve our Problem

# Write the code using recursion

```python
def moves(n, s, d, a):

    #Base condition
    if n == 1:
        return 1

    #logic
    # moves n-1 s->a, then moves 1 biggest coin s->d, then moves n-1 coins a->d
    return moves(n-1,s,a,d) + 1 + moves(n-1,a,d,s)

#Driver code
print(moves(1,'s','d','a'))
print(moves(2,'s','d','a'))
print(moves(3,'s','d','a'))
print(moves(4,'s','d','a'))
```

```
1
3
7
15
```