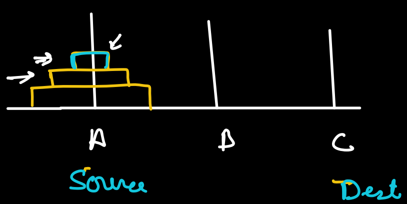
Q **Towers of Hanoi**

There are 3 towers A B & C

N disks are placed in tower A in sorted order

→ Move all disks from A (Source) to C (dest) using tower B (temp) making sure that a small disk is never below a larger disk.

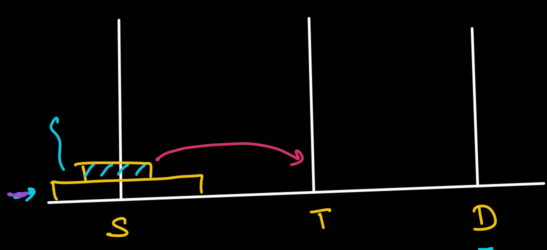


Implement the function

⇒ TOH (N, source, dest, temp)

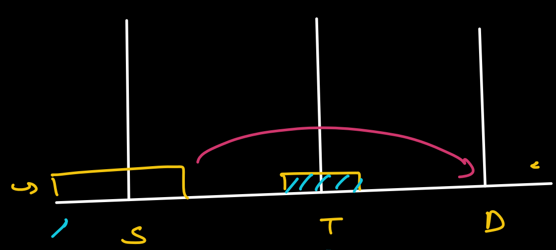
→ Print all steps of moving N disks from src to dest in correct order.

TOH(3, A, C, B)

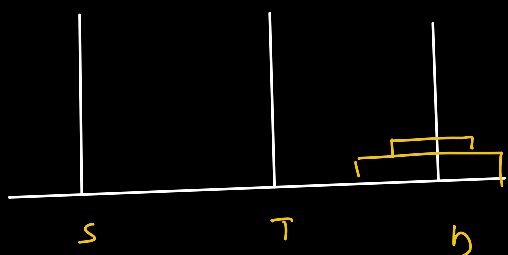


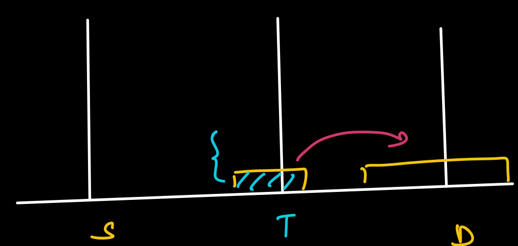
(N-1, S, T, D)

→

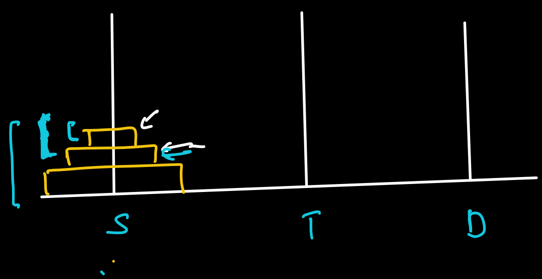


S → D

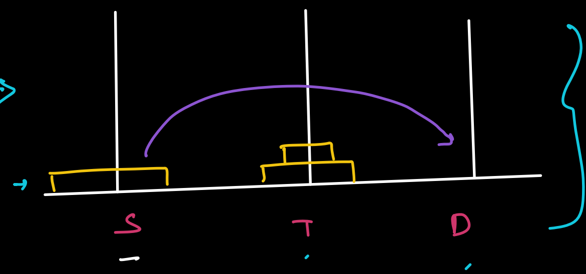


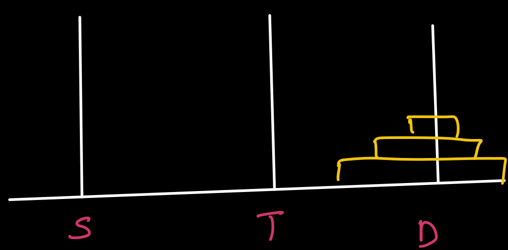


(N-1, T, D, S)

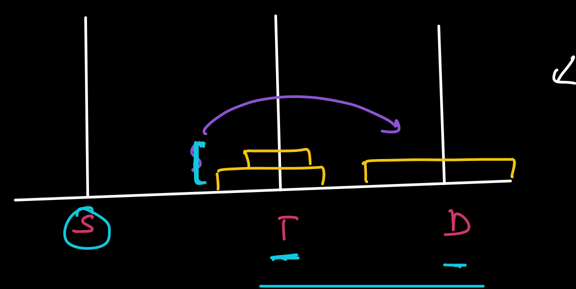


3 steps →





← 3 steps



7 steps

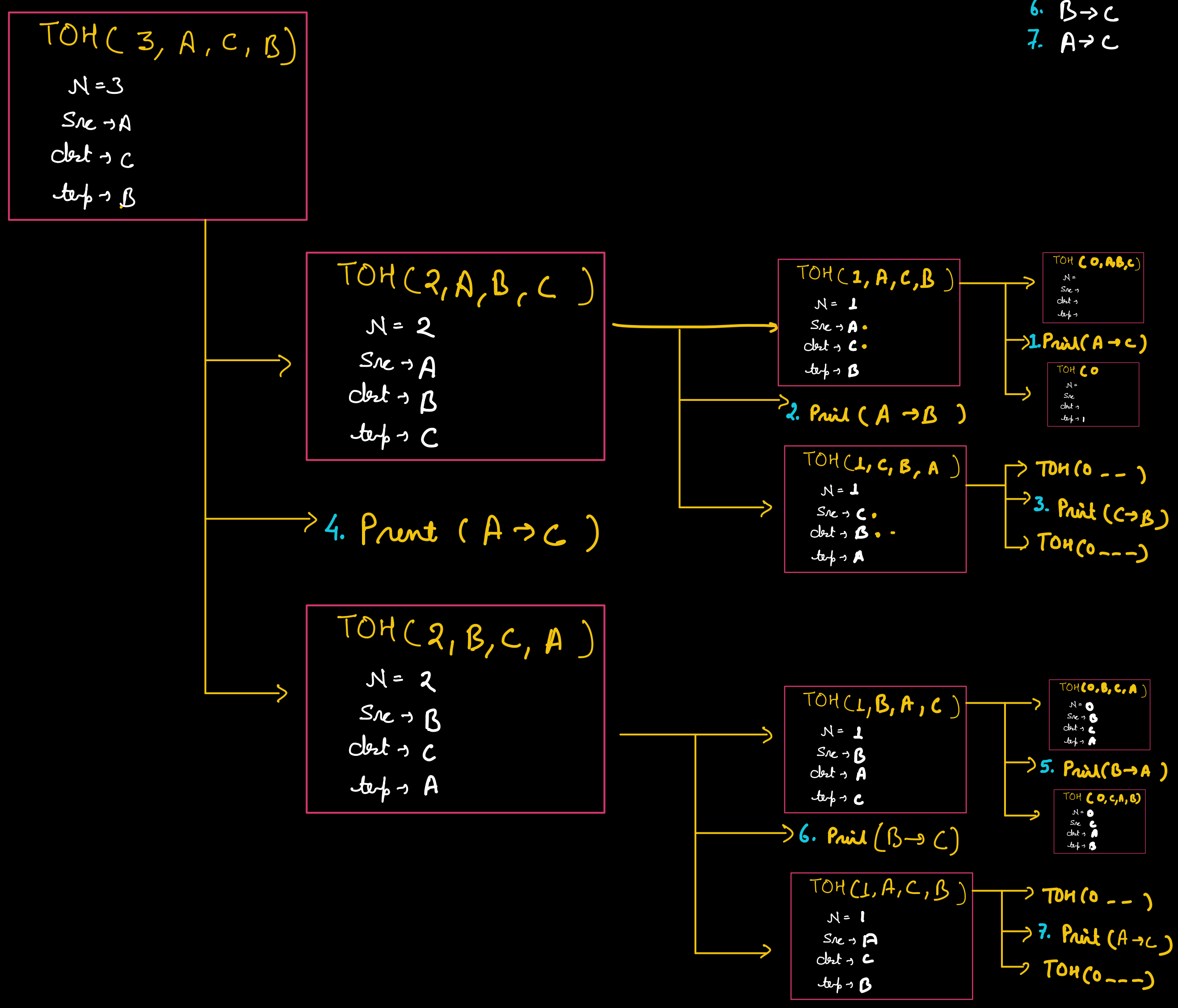
S → D
S → T
D → T
S → D
T → S
T → D
S → D

```
void TOH( N, source, destination, temp) {
    if ( N==0)
        return;

 // Assumption:
    TOH(x, A, C, B) → Print the correct steps of moving x disks
                       from A to C using B.

    ⇒ TOH ( N-1, source, temp, destination);
    ⇒ Print ( source → destination);
    ⇒ TOH ( N-1, temp, destination, source);
}
```

1. A → C
2. A → B
3. C → B
4. A → C
5. B → A
6. B → C
7. A → C

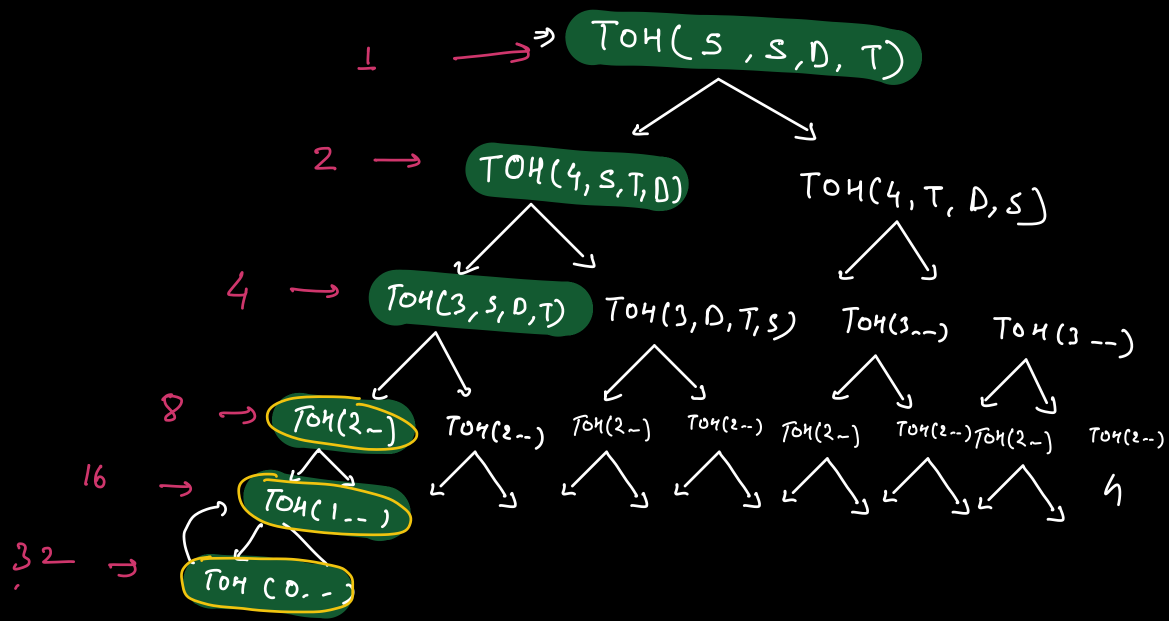


TC

Time Complexity of any recursive function:

No. of fn calls (no. of nodes in rec tree) × TC of every fn call

$2^N \times O(1)$



TC : $O(2^N)$ | SC : $O(N)$

$N = 2 \longrightarrow 3 \rightarrow 2^2 - 1$

$N = 3 \longrightarrow 7 \rightarrow 2^3 - 1$

$N = 4 \longrightarrow 15 \rightarrow 2^4 - 1$

$N = 5 \longrightarrow 31 \rightarrow 2^5 - 1$

⋮ ⋮

Q Given N & K. Return the value at K^{th} index in N^{th} row (Assume that input is always valid)

$K \le 10^{18}$ $\boxed{N <= 10^5}$

$$\begin{bmatrix} 0 \rightarrow 0\ 1 \\ 1 \rightarrow 1\ 0 \end{bmatrix}$$

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 :	0															
2 :	0	1														
3 :	0	1	1	0												
4 :	0	1	1	0	1	0	0	1								
5 :	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0

N : 5
K : 7 $\longrightarrow$ 1

N : 4
K : 2 $\longrightarrow$ 1

$\boxed{N < 20}$

$2^N \Rightarrow 2^{20} = 2^{10} \times 2^{10}$

$\approx 10^6$ ✓

N = 4
K = 5

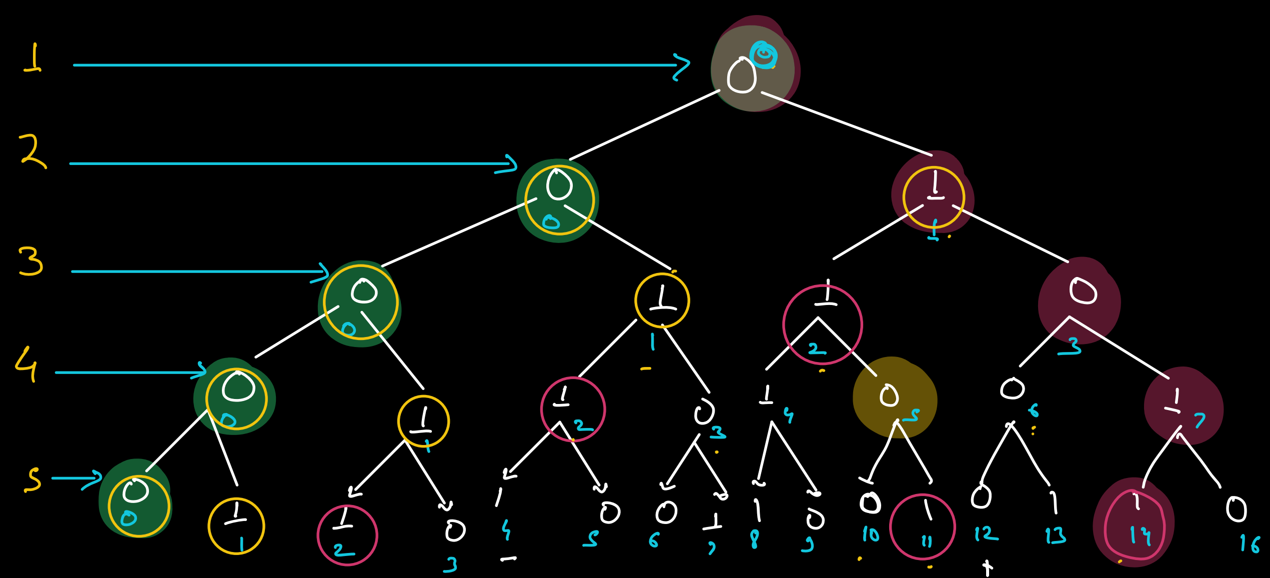
$2^0 + 2^1 + 2^2 - - - - 2^N$

1	0								$\rightarrow$ 1
2	0	1							$\rightarrow$ 2
3	0	1	1	0					$\rightarrow$ 4
4	0	1	1	0	1	0	0	1	$\rightarrow$ 8
	0	1	2	3	4	5			

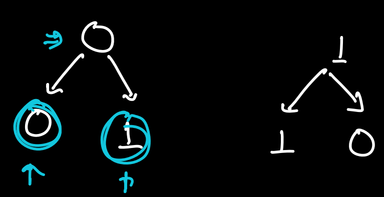
$O(2^N)$ ✗

$\rightarrow$ 16

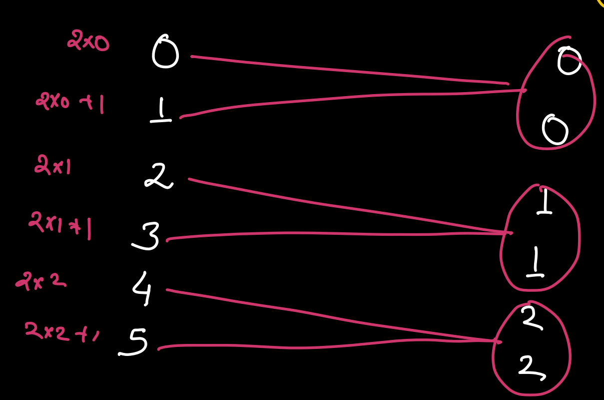
N, $K \rightarrow max \approx 2^N$



- To get K^{th} element in N^{th} row we need information about who generated it (Parent)

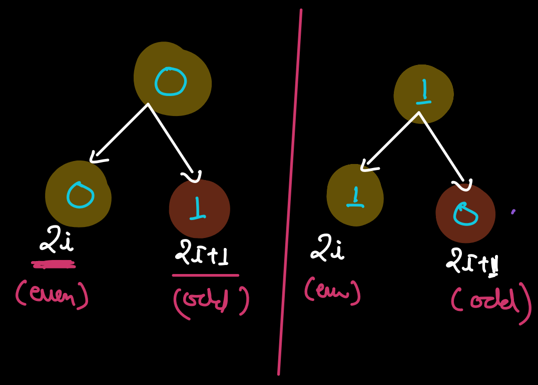


Index of Child   Index of Parent



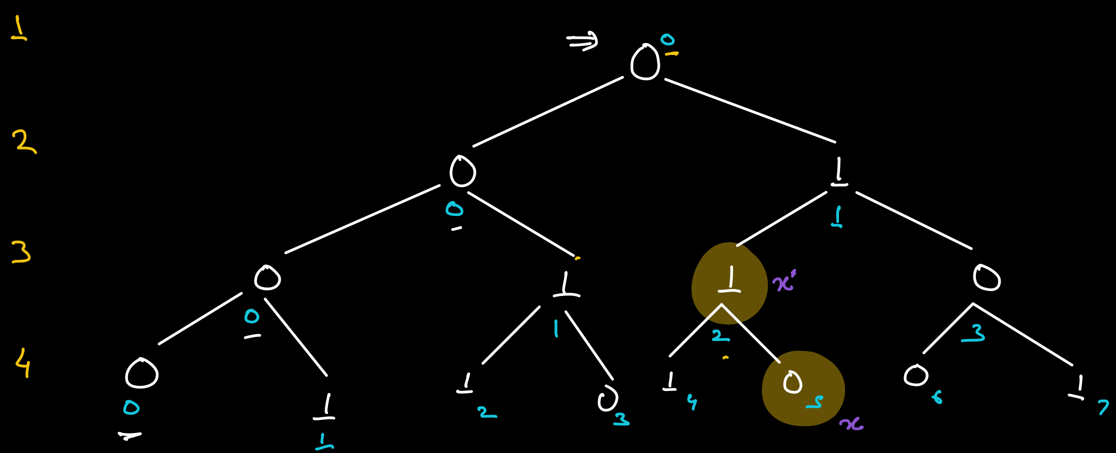
i ⟵ Parent

$2i$ $\quad$ $2i+1$ ⟵ Children



$K \longrightarrow K/2$

↑ Child (in N^{th} row) $\qquad$ ↑ Parent (in $(N-1)$ row)



$N = 4$

$K = 5$

5 (Row 4) ⟶ 2 (Row 3)

```
int find (N, K) {
* if (K == 0) ret 0;

// Assumption: find(N, K) returns the correct value at index K of
   row N.

    int parentVal = find (N-1, K/2);

    // if child at even index

    if (K%2 == 0)
          ret parentVal;

    else
          ParentVal ^1;          // or  1 - parentVal

}
```

$N = 10^5$

$K = 0$ $\longrightarrow$

$N = 10^5$, $K = 100$ $\longrightarrow$ $N = 10^5 - 1$, $K = 50$ $\longrightarrow$ $N = 10^5 - 2$, $K = 25$ $\longrightarrow$ $N = 10^5 - 3$, $K = 12$

$\downarrow$

$N = 10^5 - 7$, $K = 0$ $\longleftarrow$ $N = 10^5 - 6$, $K = 1$ $\longleftarrow$ $N = 10^5 - 5$, $K = 3$ $\longleftarrow$ $N = 10^5 - 4$, $K = 6$

$N = 10^5 - 7 \longrightarrow 0$

$\downarrow$

$N = 10^5 - 8$, $K = 0$ $\longrightarrow$ $N = 10^5 - 9$, $K = 0$ $\dots$

$$K \longrightarrow K/2 \longrightarrow K/4 \longrightarrow K/8 \; - - - - \longrightarrow 0$$

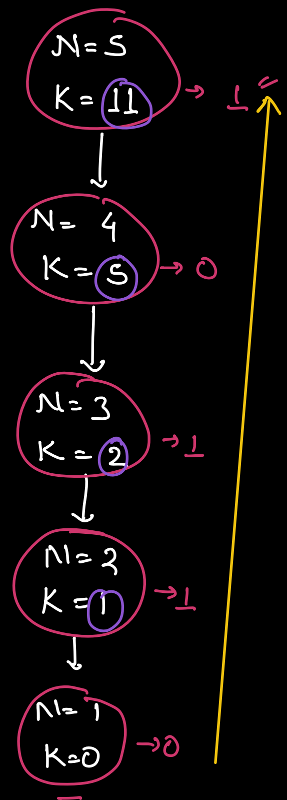
$\approx \log_2 K$ iterations

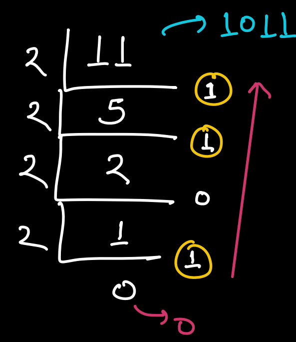
$K \downarrow 2^N$

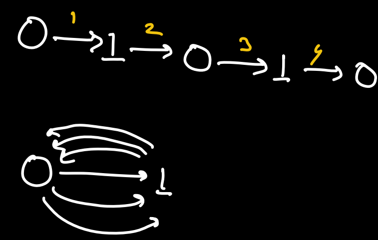
TC $\longrightarrow \log(K) \longrightarrow O(\log(2^N))$

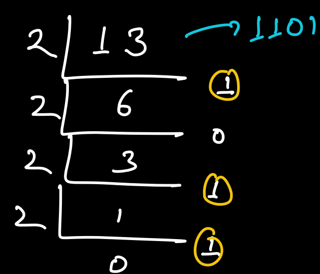
$\Rightarrow O(N)$

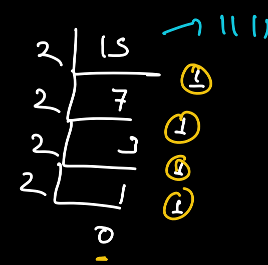
SC $\longrightarrow O(N)$











If no. of set bits is even → 0

If no. of set bits is even → 1

TC : $O(\log K) \longrightarrow O(\log 2^N) \rightarrow O(N)$

SC : $O(1)$

* Gray Code