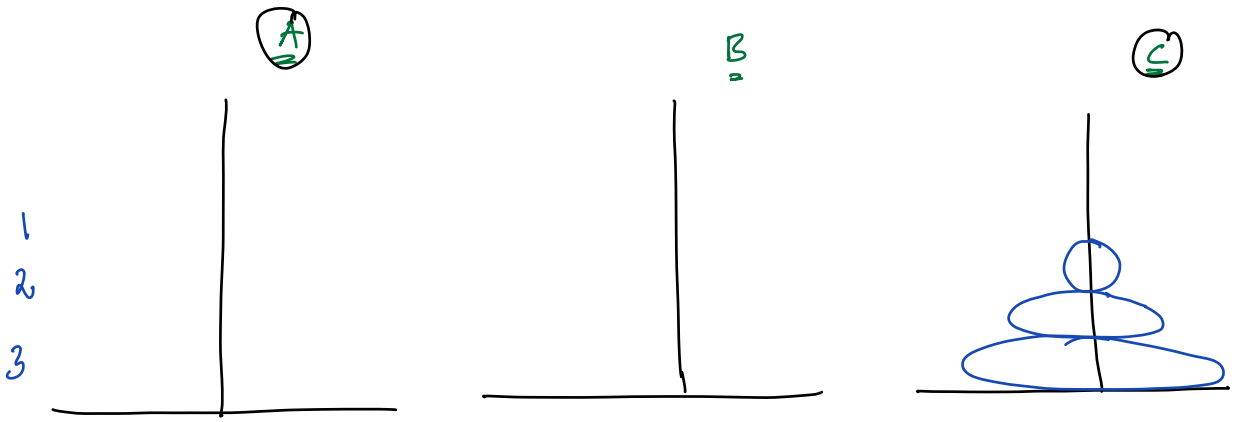
Recursion:

Ass : Decide what your function does

Main : Solving assumption with Subproblems

Base : When to stop

Towers of Hanoi → No of towers = 3



Rules:

1) We can move 1 disc at one

2) Bigger disc cannot be placed on top small disc

// Q) Given N disc move all discs from A → C & print movement of discs

// Movements

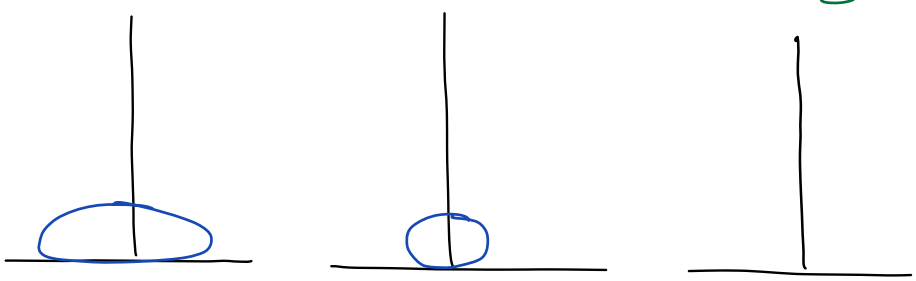
1: A → C
2: A → B
3: C → B
3: A → C
1: B → A
2: B → C
1: A → C

N=1

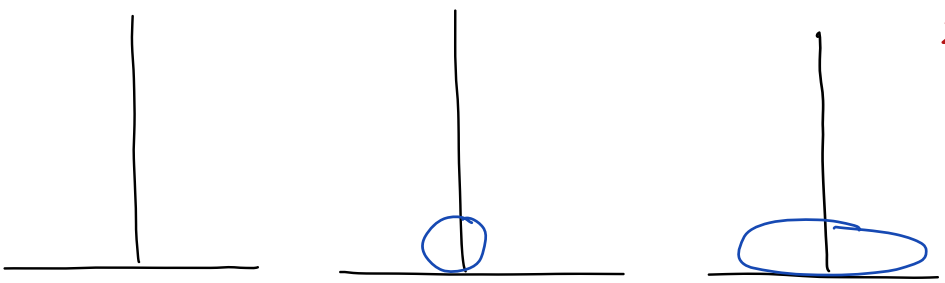
: $l : A \Rightarrow C$

N=2

A B C



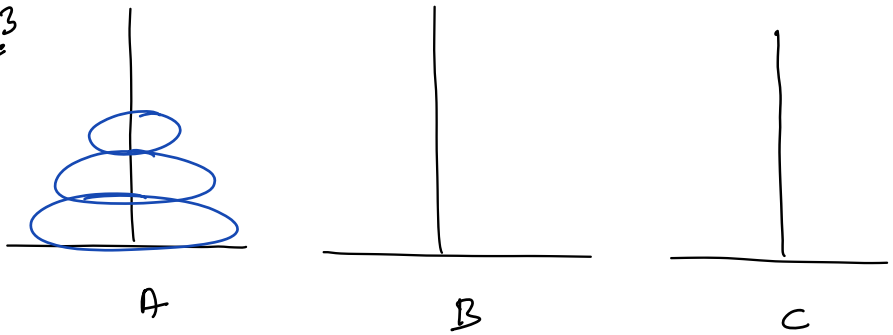
1 : A → B

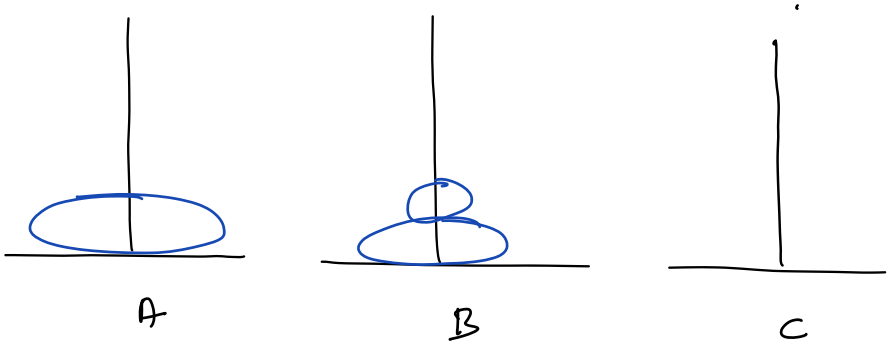


2 : A → C

1 : B → C

N=3



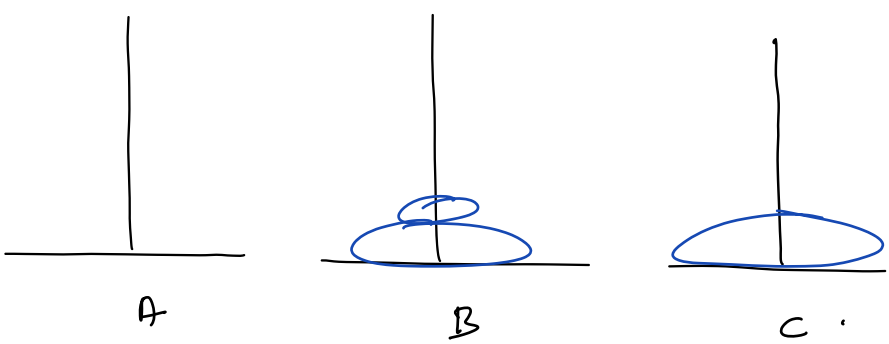


1 : A → C

2 : A → B

1 : C → B

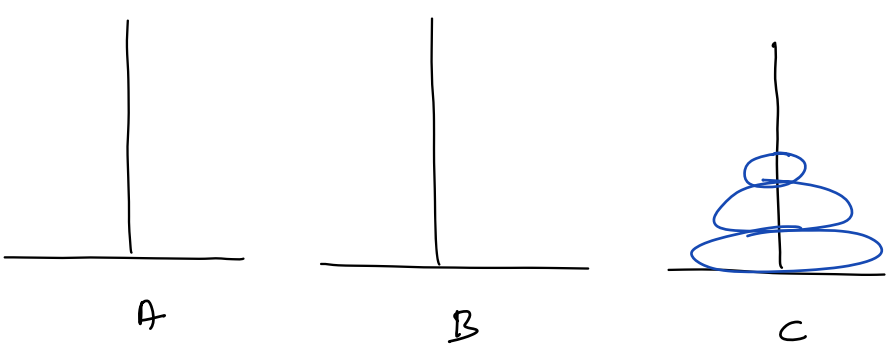
3 : A → C



1 : B → A

2 : B → C

1 : A → C



move 4 disc from A → C

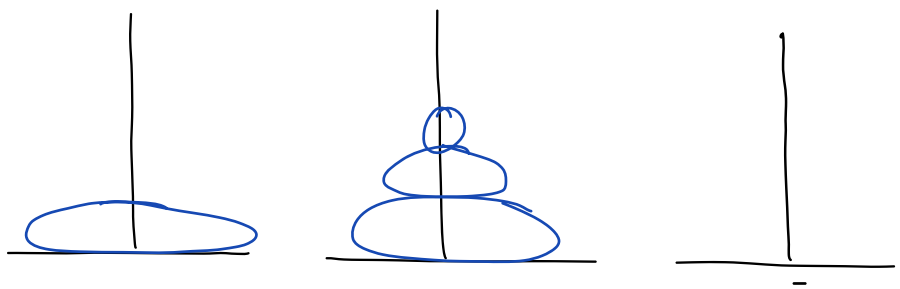
N=4

1
2
3
4

A B C

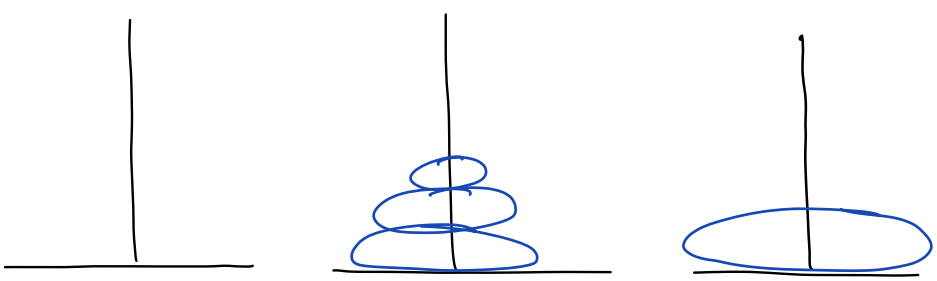
move 3 disc A → B

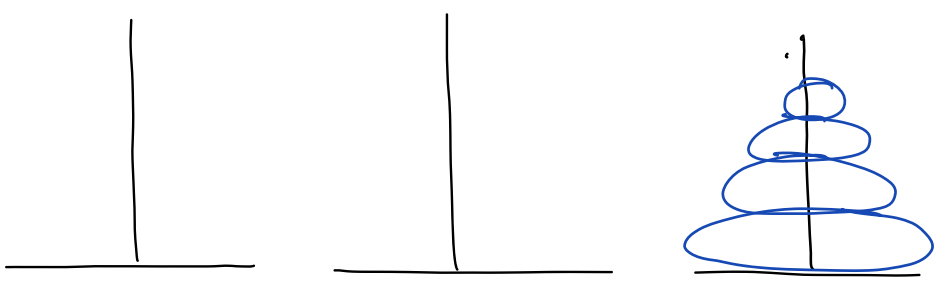
1 : A → B
2 : A → C
1 : B → C
3 : A → B
1 : C → A
2 : C → B
1 : A → B



A B C

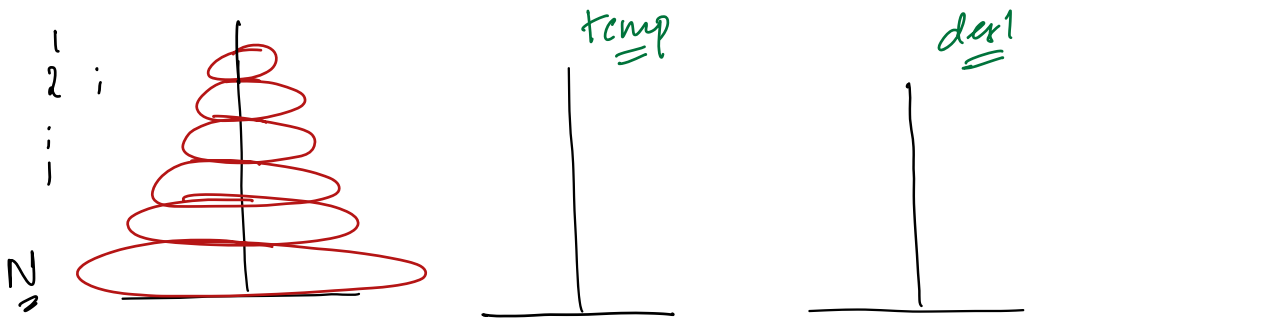
4 : A → C



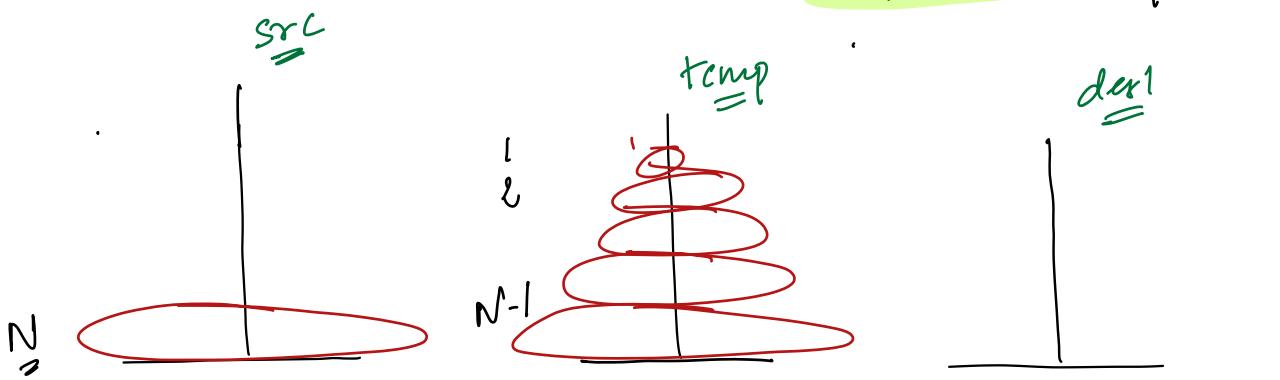


1 : B → C
2 : B → A
1 : C → A
3 : B → C
1 : A → B
2 : A → C
1 : B → C

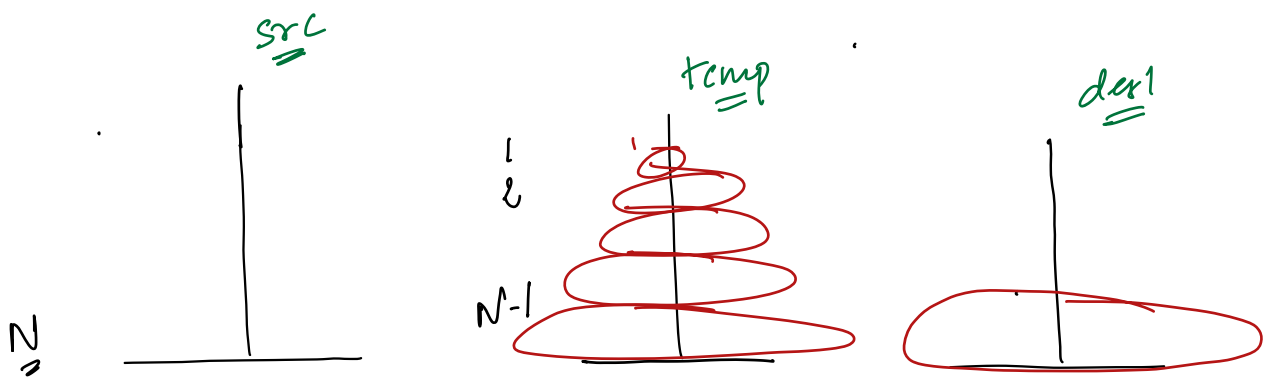
move 3 disc from B - C



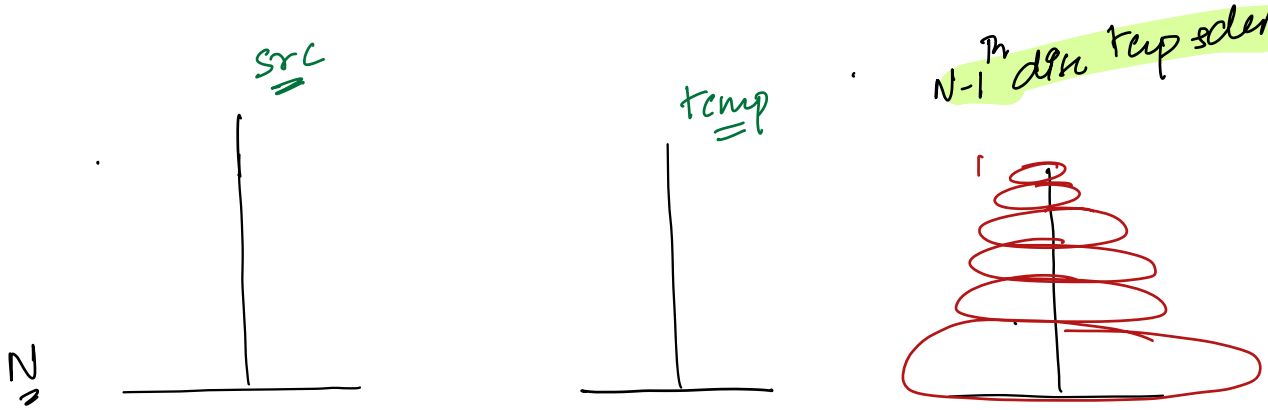
1 N-1 src → temp.



N th src → dest



N-1 th src temp → dest



// Pseudo Code

N, move N disc from S → D using T

1) N-1 disc from S → T using D ✓
2) N^{th} disc from S → D {print(N^{th} : S → D)}
3) N-1 disc from T → D using S ✓

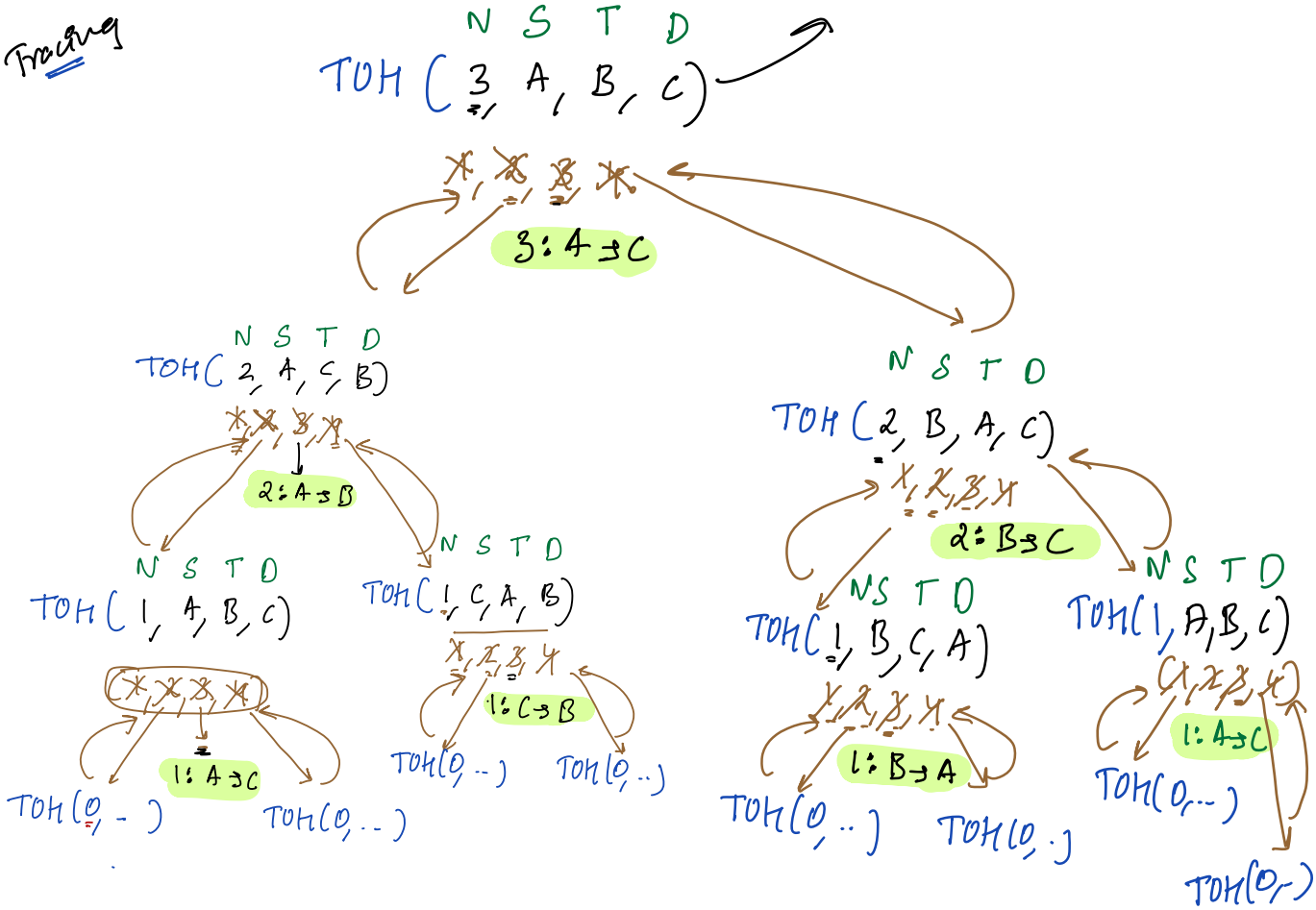
Code:

```
void TOH (N, char S, char T, char D)
{
1  if (N == 0) { return; }
2  TOH (N-1, S, D, T)        → Tower D acts as Temp
3  print(N^th : S → D)
4  TOH (N-1, T, S, D)        → Tower S acts as Temp
}
```

$T(N) = 2T(N-1) + 1$, $T(0) = 1$ } $k = N$ ⇒ Idea's break

TC: $O(2^N)$ SC: $O(N)$ ⇒ In doubt Session

Tracing



1 : A → C
2 : A → B
1 : C → B
} 2 disc from A → B

3 : A → C → 3rd disc from A → C

1 : B → A
2 : B → C
1 : A → C
} 2 disc from B → C

TC: $T(N) = 2^N - 1$ steps

// 64 discs

→ Every day 1 pujari will move 1 disc

→ All discs are moved it would be last

→ How many disc movements you need : $2^{64} - 1$ days

→ $2^{60} \approx 10^{18}$ days

→ $2^{64} \approx \left\{ \frac{16 * 10^{18} \text{ days}}{365} \right\}$ years

$\approx 32 * 10^{15}$

k^{th} Symbol, row generated by replacing all elements of of previous row $0 \to 01$ & $1 \to 10$

At N^{th} row find k^{th} symbol

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 → 2

0 1 2 3 4 5

2^0 2^1 2^2 2^3 2^4

Input N=5, k=1

5, 4, 0

En:

N	k	ans
4	6	0
5	13	1
5	8	1

Sol: Generate N^{th} row & get k^{th} Element

In N^{th} row → 2^{N-1} TC:

Total: $2^0 + 2^1 + 2^2 + \dots 2^{N-1} = O(2^N)$

Hint?

// 5^{th} 7 index generate | 4 | 7/2 | 3 index

// 5^{th} 6 index generate | 4 | 6/2 | index

// 5^{th} 11 index generate | 4 | 11/2 | index

row	index
4^{th}	7/2 = 3 index
4^{th}	6/2 index
4^{th}	11/2 index

Constraints:

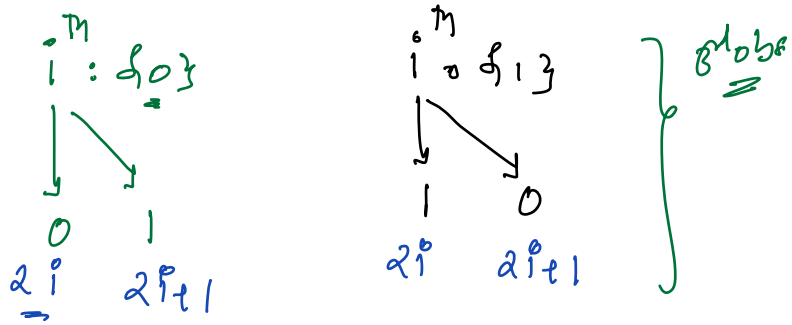
$1 \le N \le 10^5$

$1 \le K \le 10^{18}$

Obs1: N^{th} k^{th} index generated = $(N-1)^{th}$ $k/2$ index } = 1

Obs2: i^{th} index children = $(2i \;\&\; 2i+1)$ } =

Obs3:



} global

Obs4: { If you are even child data same as parent
If you are odd child data inverse of parent }

Recursive Relat — k==0, break

$T(k) = T(k/2) + 1$

{ At max we will only log k iterations }

TC = (log k)

```
int find(N, k){
    if(k==0){return 0}
    int val = find(N-1, k/2)
    if(k%2==0){
        // we are at even child
        return val;
    }
    else{
        // we are at odd child
        return !val
    }
}
```

//

// Input $N = 10^5$, $k = 1$

$(10^5 - 1, 0)$

$N = 10^5$, $k = 100$

$10^5 - 1$, 50

$10^5 - 2$, 25

$10^5 - 3$, 12

$10^5 - 4$, 6

$10^5 - 5$, 3

$10^5 - 6$, 1

$10^5 - 7$, 0

// → Simplified Master's Theorem

$T(N) = a\,T(N/b) + O(N^c)$ ↦ power of N

$t = \log_b^a$

if $t > c$: $T(N) = O(N^t)$

if $t = c$: $T(N) = O(N^t \log N)$

if $t < c$: $T(N) = O(N^c)$

$T(N) = 2T(N/2) + N^1$

$a = 2,\ b = 2,\ c = 1$

$t = \log_2^2 = 1$

TC = $O(N \log N)$

$T(N) = \underset{a}{2}\,T(N/\underset{b}{2}) + (N^0)$ → $c = 0$

$t = \log_b^a = \log_2^2 = 1,\ c = 0$

$T(N) = O(N^1) = O(N)$

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

$a = 1,\ b = 2,\ c = 0$

$t = \log_2^1 = 0,\ c = 0$

TC = $O(N^0 \log N)$ = $O(\log N)$

// Doubts ↦ Recursion

// $T(N) = 2T(N-1) + 1$ → $T(0) = 1$

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

$= 2[2T(N-2) + 1] + 1$ → $2^2 T(N-2) + 2^2 - 1$

$= 4T(N-2) + 3$

$T(N-2) = 2T(N-3) + 1$

$= 4[2T(N-3) + 1] + 3$ → $2^3 T(N-3) + 2^3 - 1$

$= 8T(N-3) + 7$

$T(N-3) = 2T(N-4) + 1$

$= 8[2T(N-4) + 1] + 7$

$= 16T(N-4) + 15$ → $2^4 T(N-4) + 2^4 - 1$

→ $T(0)$, $k = N$

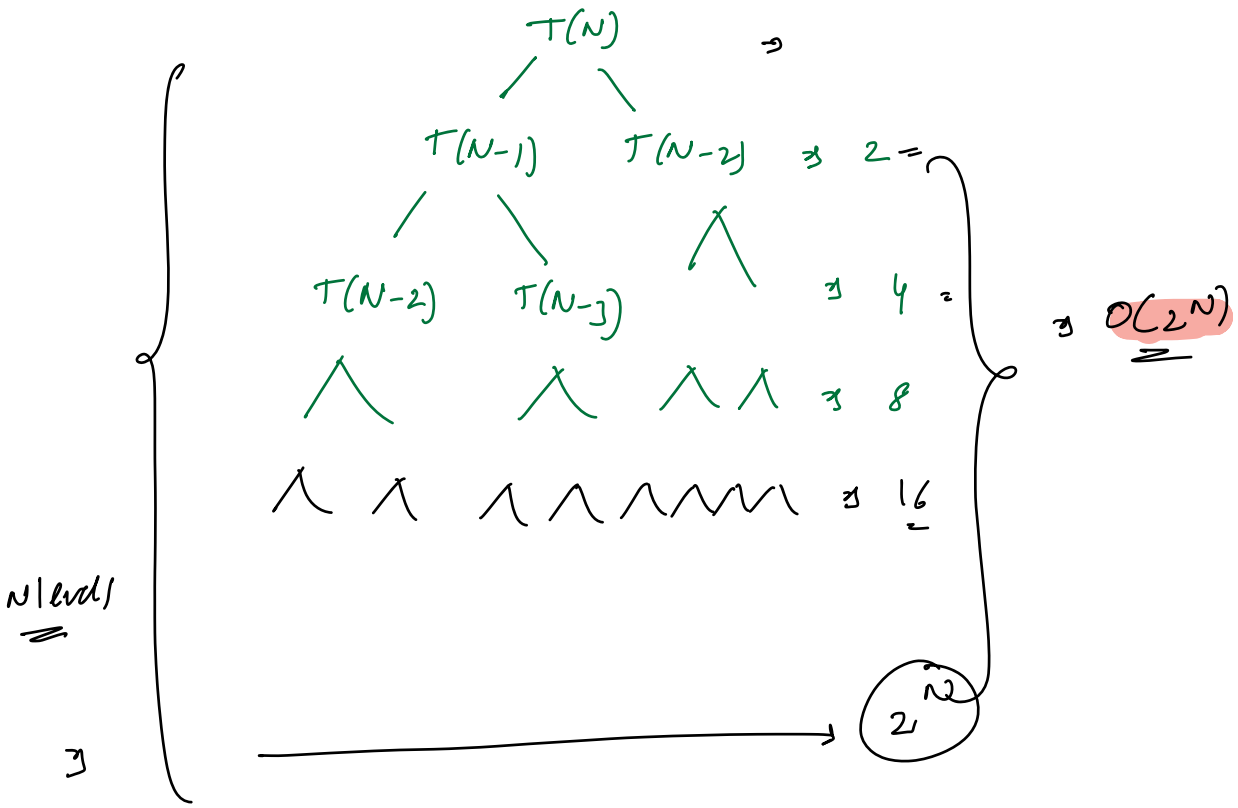
1) $2^k * T(N-k) + 2^k - 1$

2) $2^N * T(0) + 2^N - 1$

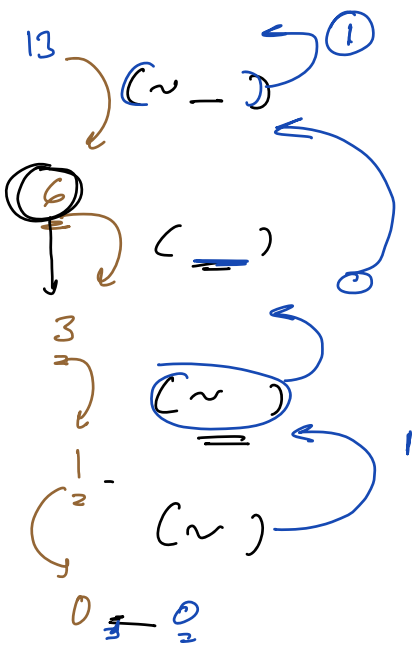
3) $2 * 2^N - 1 \Rightarrow O(2^N)$

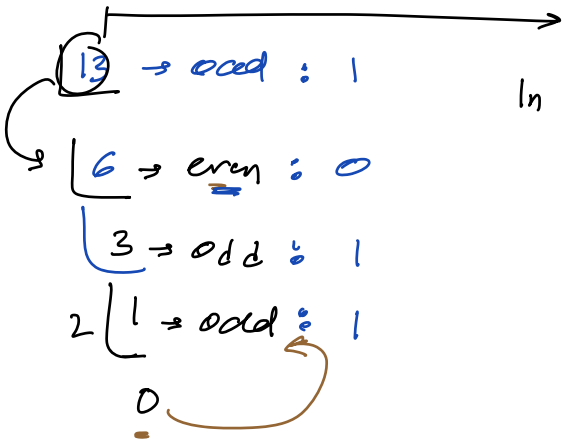
$TC = O(2^N)$

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



row
index
5th





In Binary representation

1 → flip ans

0 → same ans.

→ get no: of set bits in k = cnt

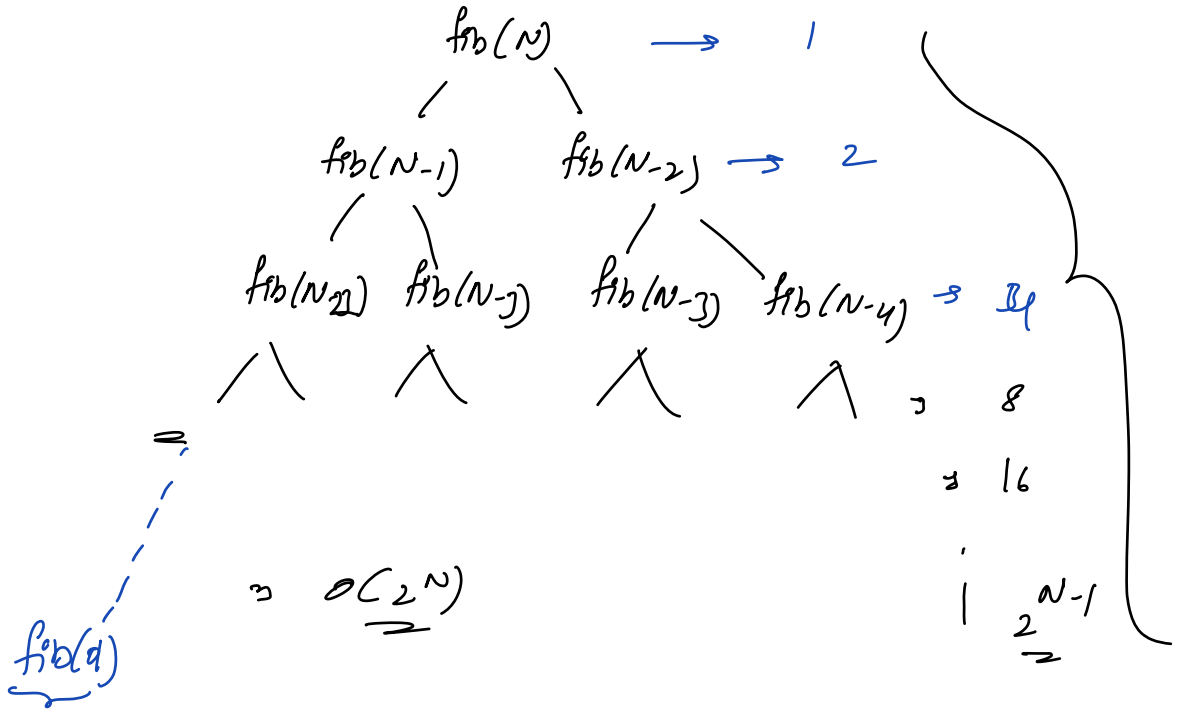
SC:

↓

O(1)

```
if( cnt%2==0) {return 0}
if (cnt%2==1) {return 1}
```

TC: O(1)



TA

coming monday