Recursion

→ k^{Th} Char

→ Towers of Hanoi

→ Basic back tracking problems

↳ Advanced before DP

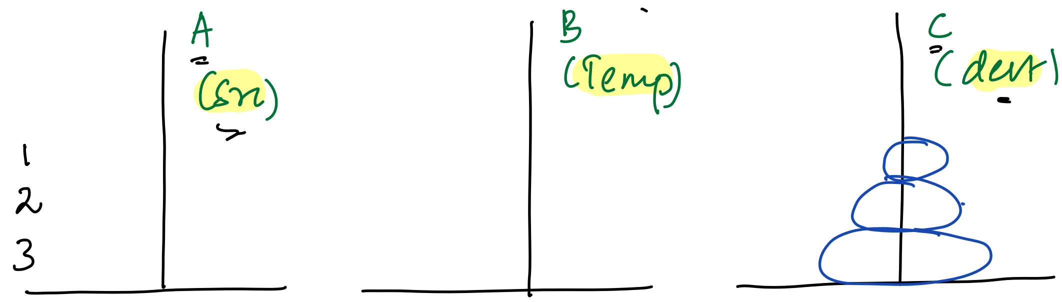
Recursion :

Ass : Decide what your function does

Main : Solving Assumption with Subproblems

Base : When to stop

Towers of Hanoi → No: of Towers → 3



Rules:

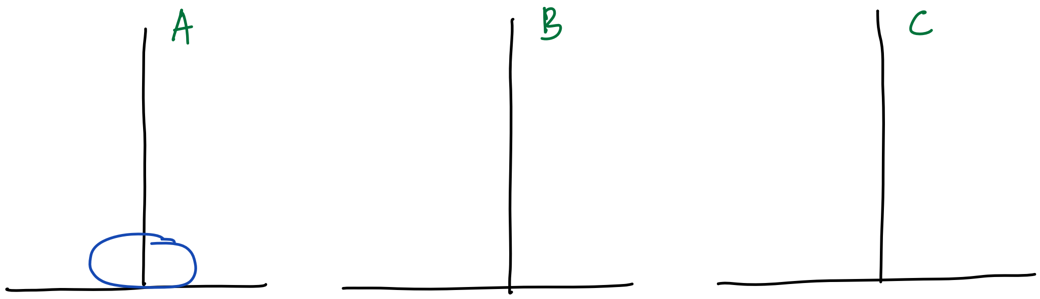
1) 1 disc at a time

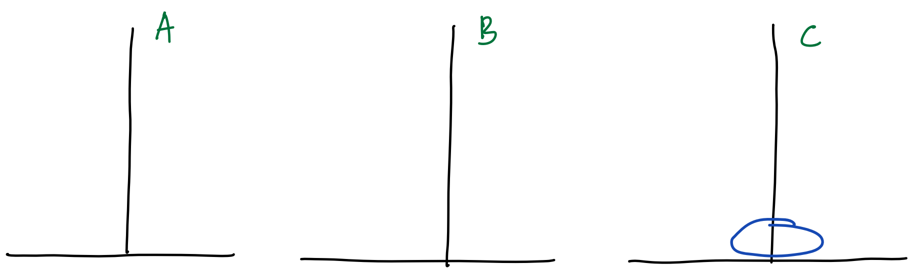
2) larger disc cannot be placed on a smaller disc

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

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

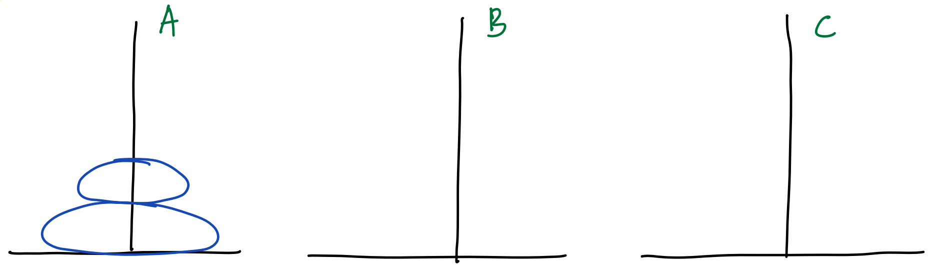
N=1:





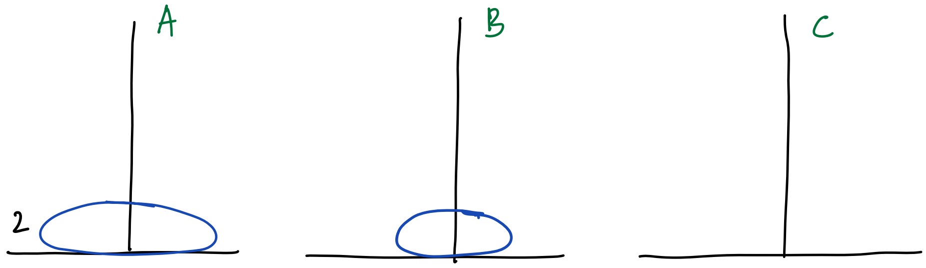
1: A → C

$N=2$



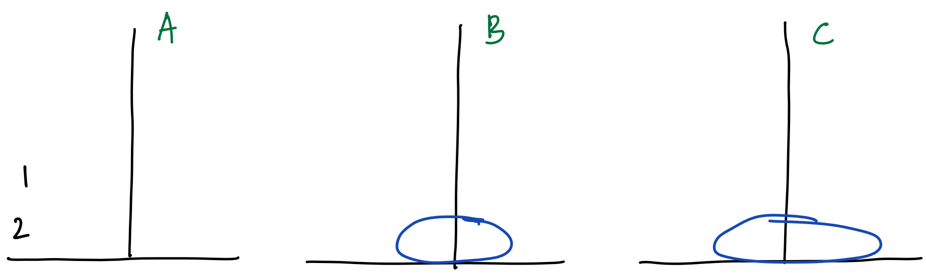
1 disc from $A \to B$ ✓

$1 : A \to B$



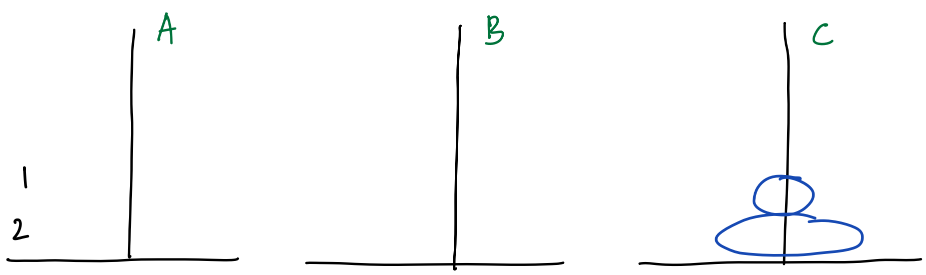
2nd disc from $A \to C$

$2 : A \to C$

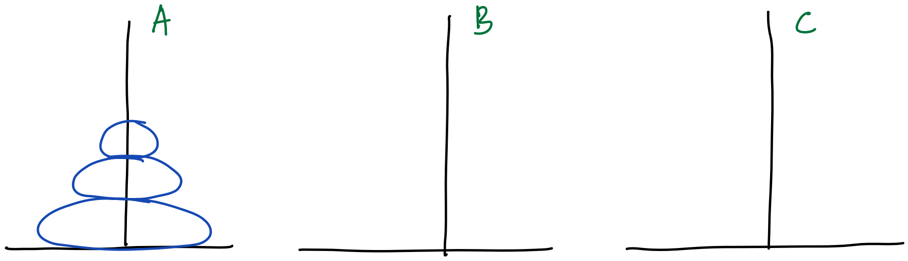


1 disc from $B \to C$

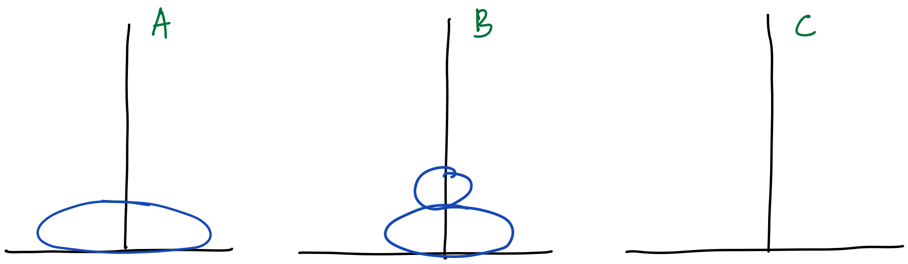
$1 : B \to C$



N=3



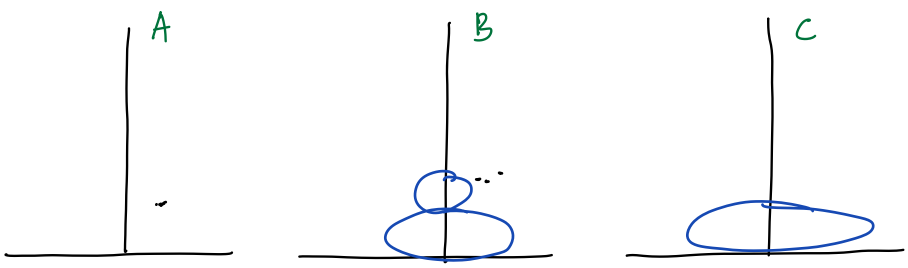
// 2 discs from A → B



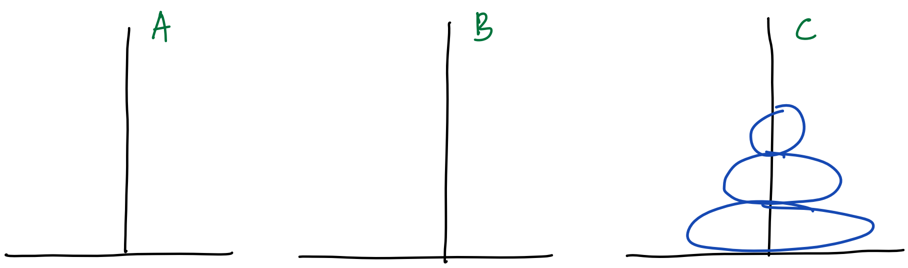
1 : A → C
2 : A → B
1 : C → B

3rd disc from A → C

3 : A → C

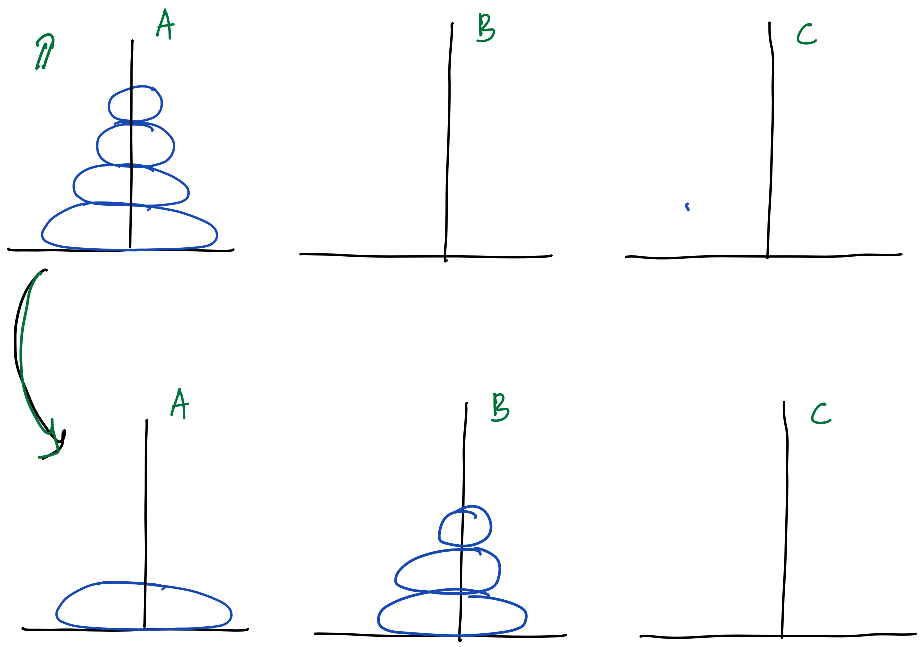


2 disc from B → C



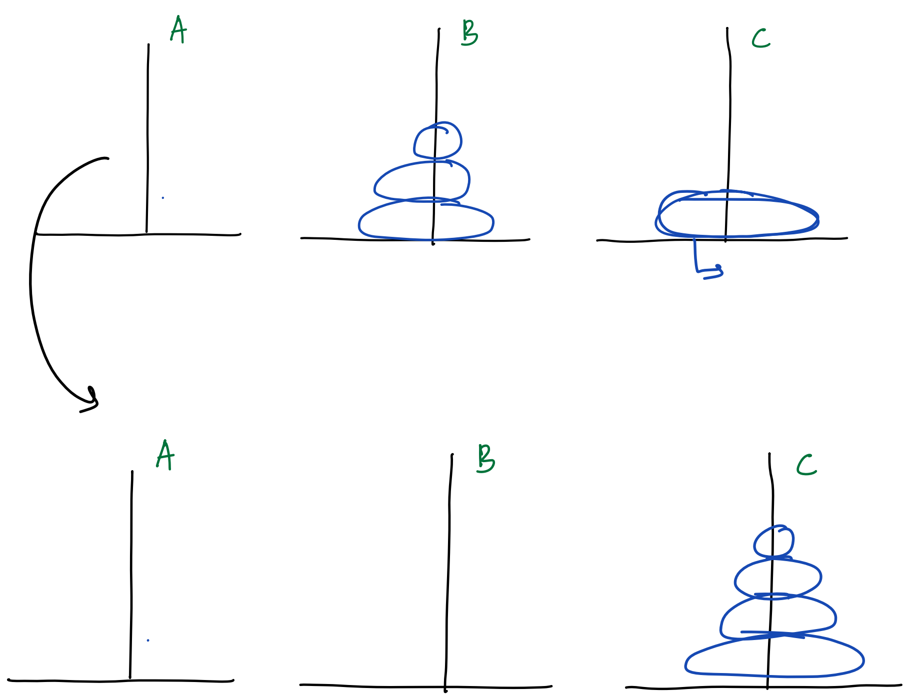
1 : B — A
2 : B — C
1 : A → C

$N=4 \Rightarrow$ 4 discs from $A \to C$



// 3 disc from $A \to B$

1 : $A \to B$
2 : $A \to C$
1 : $B \to C$
3 : $A \to B$
1 : $C \to A$
2 : $C \to B$
1 : $A \to B$



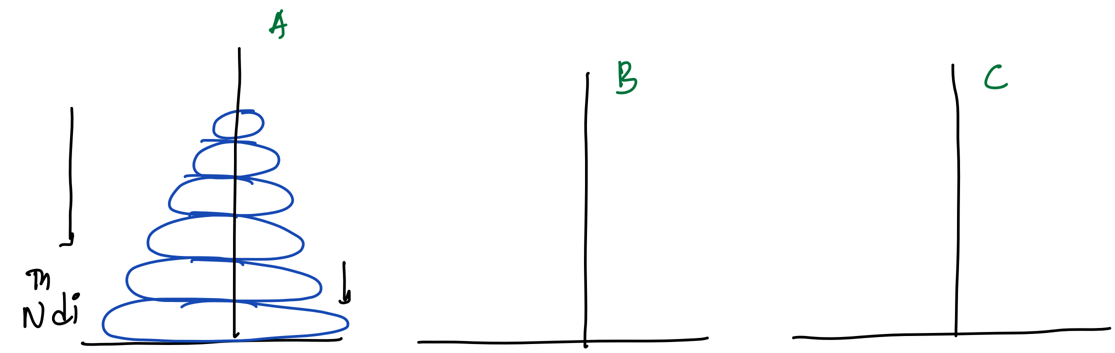
4 : $A \to C$

4^{th} disc $A \to C$

// 3 disc from $B \to C$

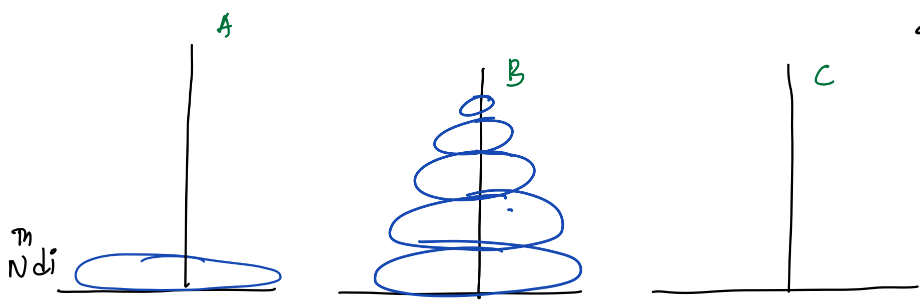
1 : $B - C$
2 : $B - A$
1 : $C - A$
3 : $B \to C$
1 : $A \to B$
2 : $A \to C$
1 : $B \to C$

In general N discs :

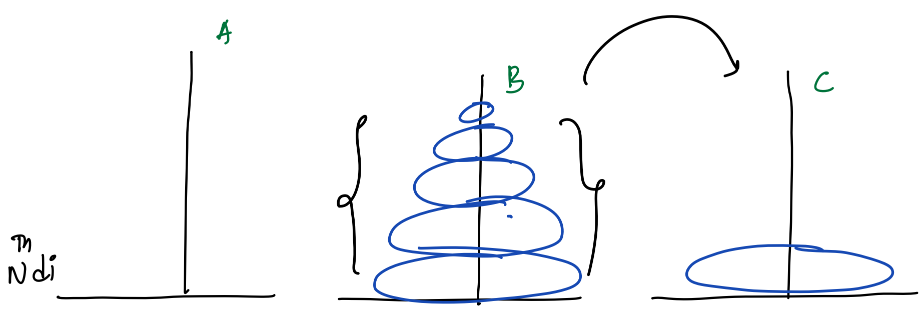


Step1:

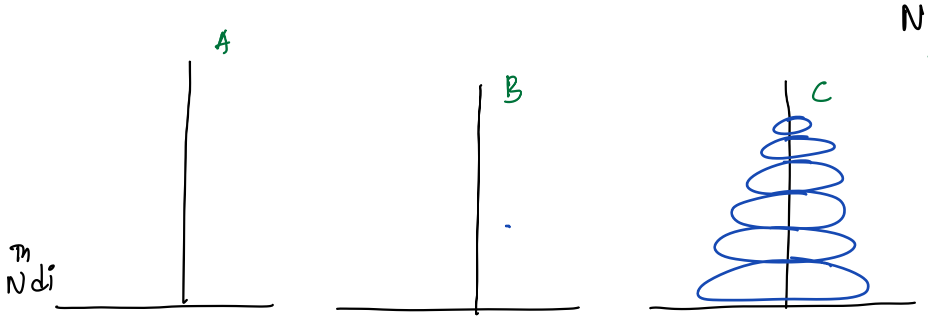
N-1 discs from A → B



N^{th} disc : A → C



N-1 discs from B → C



// Pseudo Code

Move N discs from Src → dest, temp

N → no. of discs

1) N-1 discs from Src → Temp

2) N^{th} disc from Src → dest } print (N^{th} disc : Src → dest)

A particular disc

3) N-1 disc from temp → dest

Code:

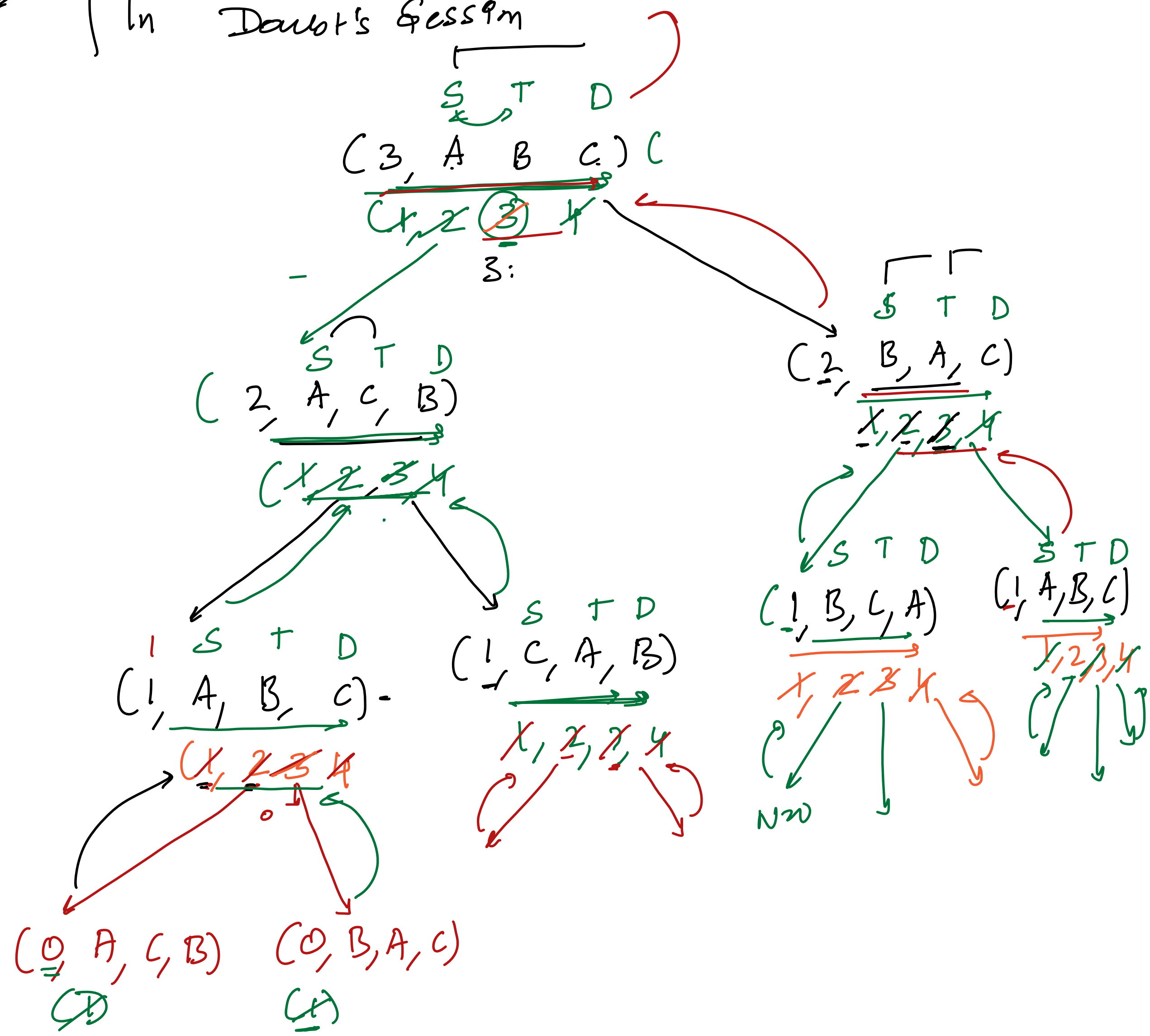
```
void TOH ( N, char S, char T, char D) {
  1  if ( N == 0) { return; }        Since no disc return
  2  TOH ( N-1, S, D, T) }           N-1 disc from S → T
  3  print ( Nth : S → D) }          Nth disc from S → D
  4  TOH ( N-1, T, S, D) }           N-1 disc from T → D
}
```

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

$T(N) = 2T(N-1) + O(1)$ } TODO

Tracing
N=3

on code
In Doubt's Session



1: A → C
2: A → B
1: C → B

} Moving 2 discs from A → B

3: A → C } Moving 3rd disc from A → C

1: B → A
2: B → C
1: A → C

} Moving 2 discs from B → C

K^{th} Symbol, row generated using previous row by $0 \to 01$ & $1 \to 10$

	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^{th} : 2^{N-1} Elements

N	K	Output
5	8	1
3	2	1
5	3	0
4	8	{Invalid}
1	0	0

BF:

Generate N^{th} row & get K^{th} index

TC: $O(2^N)$

SC: $O(2^N)$

$1^{st} + 2^{nd} + 3^{rd}$ - - —— N^{th} row

$2^0 \quad 2^1 \quad 2^2 + - -$ —— 2^{N-1} } $\to O(2^N)$

// Constraint

$1 <= N <= 10^5$ } Iterations $\Rightarrow 2^N \to 2^{10^5} >>>>> 10^8$

Observation :

For a given row 1^{st} half = Previous row

(16)

$N=5$: First half $N=4$ full row

$[0-7]$ = $[0-7]$

$[0-2^3)$

$N=6$: First half = $N=5$ full row $N=5$ is 16

$[0-15]$ $[0-15]$ $[0 \quad 15]$

$[0-2^4)$

$N=i$: First half $i-1^{th}$ full

$[0, 2^{i-2})$ = $[0, 2^{i-2})$

For a given row 2^{nd} half = Toggle Previous row

$N=5$ 2^{nd} half = Toggle previous row

$[8-15]$ = $\sim[0-7]$

$N=6$ 2^{nd} half

$[16-31]$ = $\sim[0-15]$

$N=i$ 2^{nd} half

$[2^{i-2}, 2^{i-1})$

$8^{th} = \sim[0]$

$9^{th} = \sim[1]$

$10 = \sim[2]$

$11 = \sim[3]$

$12 = \sim[4]$

$13 = \sim[5]$

$14 = \sim[6]$

$15 = \sim[7]$

Ex:

N^{th}	k^{th}	
5	3	$\Rightarrow$ 4^{th} row 3^{rd} index
4	1	$\Rightarrow$ 3^{rd} row 1^{st} index
5	10	$\Rightarrow$ Toggle { 4^{th} row 2^{nd} index }
5	13	$\Rightarrow$ Toggle { 4^{th} row 5^{th} index }

How many Elements = 2^{N-1}

1^{st} Half = $[0, 2^{N-2})$

2^{nd} Half = $[2^{N-2}, 2^{N-1})$

```
// int find (N, k) {
    if(k==0) return 0
    if(N==1) return 0

    if( k < 2^(N-2) ) { // k is 1st half
        return find(N-1, k) }
    }
    else { // k is 2nd half
        return 1 - find(N-1, k - 2^(N-2)) }
    }
}
```

TC: $T(N) = T(N-1) + 1$

$T(N) = O(N)$

//

$T(N) = 2T(N-1) + 1, \quad T(0) = 1$ $\quad K = N$

SC: $O(N)$

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

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

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

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

$= 8T(N-3) + 7 \Rightarrow 2^3 T(N-3) + 2^3 - 1$

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

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

$T(0) = 1$

$T(N) = 2^k T(N-k) + 2^k - 1$ }

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

$\Rightarrow 2 * 2^N - 1$

$\Rightarrow 2^{N+1} - 1$ }

TC: $O(2^N)$

$T(N) = 2T(N/2) + N \quad \} \; T(1) = 1$

$T(N/2) = 2T(N/4) + N/2$

$= 2 * [2T(N/4) + N/2] + N$

$= 4T(N/4) + 2N \Rightarrow 2^2 T(N/2^2) + 2N$

$T(N/4) = 2T(N/8) + N/4$

$= 4[2T(N/8) + N/4] + 2N$

$= 8T(N/8) + 3N \Rightarrow 2^3 T(N/2^3) + 3N$

$= 16T(N/16) + 4N \Rightarrow 2^4 T(N/2^4) + 4N$

// Generalized exp: $2^k T(\underbrace{N/2^k}_{\text{ukn}}) + kN$

$N/2^k = 1 \} \; 2^k = N,$ A

$TC \Rightarrow 2^{\log_2 N} * T\left(N/2^{\log_2 N}\right) + (\log_2 N)(N)$

$= N * T(1) + N\log N$

$TC \Rightarrow N\log N$

→ Simplified Masters Theorem

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

$t = \log_b^a$

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

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

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

1) $T(N) = T(N/2) + O(N^0)$

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

$t = \log_2^1 = 0$

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

$= O(\log N)$

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

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

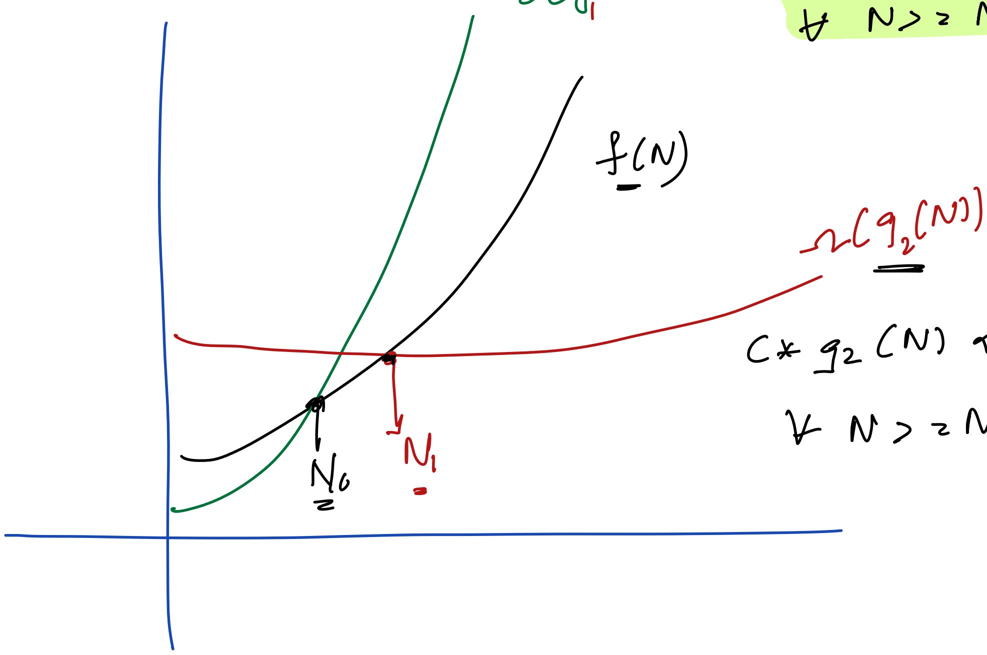
$t = \log_b^a = \log_2^1 = 0$

$c > t$: $TC: O(N^c) = O(N^1)$

3) $T(N) = 2T(N-1) + O(1)$ } This cannot be solved using master's theorem.

Doubts

$O(g_1(N))$ } $\Rightarrow$ $C * g(N) >= f(N)$ $\forall N >= N_0$



$C * g_2(N) <= f(N)$ $\forall N >= N_1$

$f(N) = \Theta(g(N))$, N_0, c_1, c_2

$c_1 * g(N) <= f(N) <= c_2\, g(N)$ $\forall N >= N_0$

$g(N)$ is acting as lower bound $\longleftrightarrow$ $g(N)$ is acting upper bound

$g(N)$ is both upper / lower $\Rightarrow$ Tight bound

$\rightarrow$ `i = 0; i < N; i++) {`

`}` `print(arr[i])`

$N \to O(N)$, $\Omega(N)$ } $\Theta(N)$

$gcd(a, b) = gcd(b, a \% b)$

$T(N, M) = T(M, N/2) = T(N/2, M/2)$

$\log_2 N + \log_2 M$