

- 1) Basics
- 2) Problems
- 3) Recursive relation
- 4) Time & Space complexity
- 5) Tower of Hanoi
- 6) Gray Code

Recursion \Rightarrow Solving a problem using
 Smaller instances of the
 same problem
 ↓
 Subproblem

I Calculate the sum of the first N natural no's

$$\underline{\text{Sum}(N)} = \underline{1 + 2 + 3 + 4 + \dots + N-2 + N-1 + N}$$

↓
Sum ($N-1$)

$$\text{Sum}(N) = \text{Sum}(N-1) + N;$$

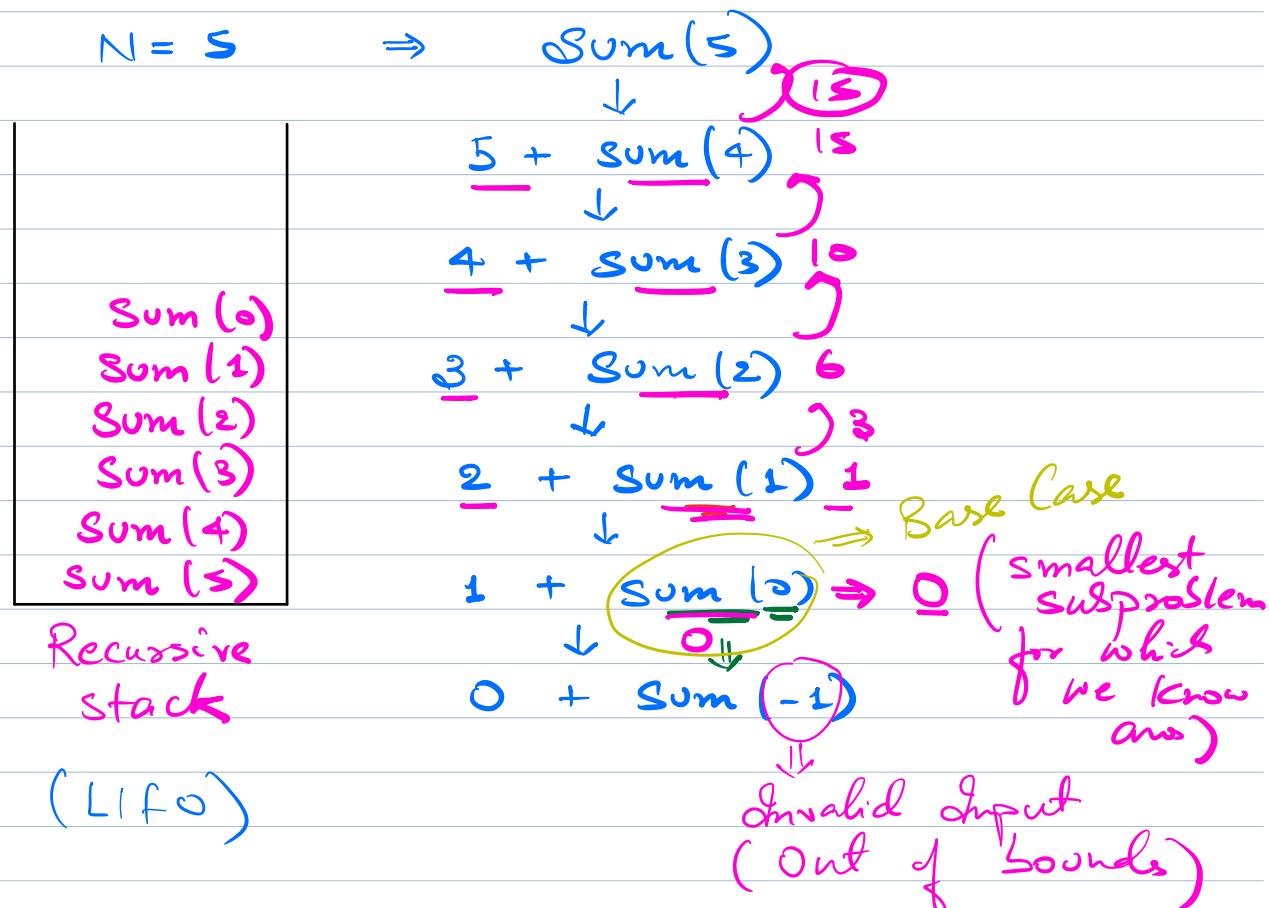
Steps for Recursion

1) Assumption : Decide what the function does & solves it

correctly

2) Main Logic : How to use subproblem to calculate answer

3) Base Case



Code

```
int sum(N) {
```

Base Case if ($N == 0$) { return 0; }

return $N + \text{sum}(N-1);$

5

S.C. = $O(N)$

Θ^2 Fibonacci Series.

$$\text{fib}(N) = \text{fib}(N-1) + \text{fib}(N-2);$$

N	0	1	2	3	4	5	...
fib	0	1	1	2	3	5	...
		1	1				
			1				
				2			
					3		
						5	

does not
follow main
logic

$$\text{fib}(1) = \text{fib}(0) + \text{fib}(-1)$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$$

Base Case :

if ($N == 1 \text{ || } N == 2$) {
 return 1;}

5

Code

$T(N)$

```
int fib (N) {
    if ( $N == 1 \text{ || } N == 2$ ) {
        return 1;
    }
}
```

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

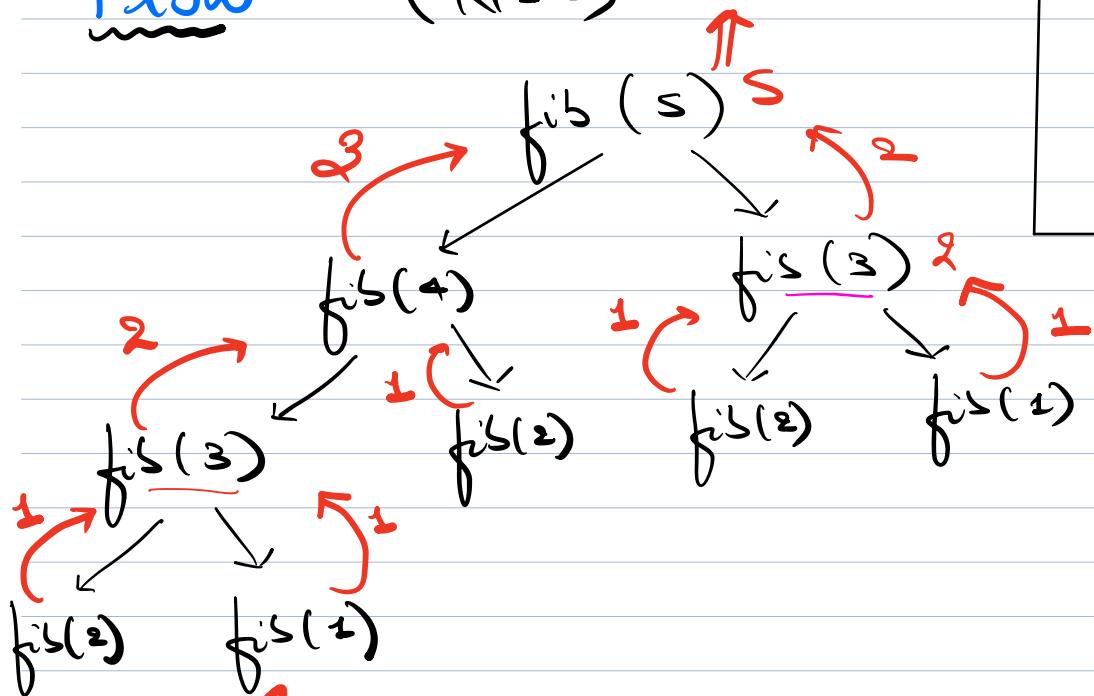
$$\text{return } fib(\underline{N-2}) + fib(\underline{N-2})$$

↳

Flow

$(N = 5)$

size = 4



S.C. ↗ recursion = Height ↗
recursive tree.

1) Sum

int sum (N) {
Base Case if (N==0) { return 0; } // O(1)}

↳ return $O(1)$ $N + \boxed{\text{sum}(N-1)}$ \Rightarrow recursive call to subproblem

Assume : Let $T(N)$ be the time taken to compute $\text{sum}(N)$

$$\textcircled{1} \quad T(N) = T(\underline{N-1}) + 1 \Rightarrow \text{Recursive relation}$$

$$T(0) = 1 \Rightarrow \text{Base Case}$$

substitute $N = N-1$ in eqn ①

$$\underbrace{T(N-1)}_{\Downarrow} = T((N-1)-1) + 1 = T(N-2) + 1$$

substitute
to eqn ①

$$T(N) = \left(T(N-2) + 1 \right) + 1$$

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

$N = N-2$ in eq ①

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

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

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

⋮

K times.

$$T(N) = T(N-K) + K \Rightarrow \text{Generalized}$$

Base Case

$$N-K = 0 \Rightarrow \text{Base Case}$$

$$T(N) = T(N-K) + K \\ = T(0) + K$$

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

$$T.C. = O(N)$$

2) Fast Power

$$a^n = a \times a^{n-1} \Rightarrow T(n) = T(n-1) + 1$$

$$\underline{a^2} = a \times a^1$$

$$= \underline{\underline{a^6}} \times a^6$$

$$\underline{a^8} = a \times \underline{\underline{a^6}} \times a^6$$

$$\underline{T(n)} = T(\underline{n/2}) + 1 \quad \text{--- ①}$$

$n = n/2$ in eq ①

$$\underline{T(n/2)} = T(\underline{n/4}) + 1$$

$$T(n) = \underline{T(n/4)} + 2$$

$n = n/4$ in eq ①

$$\underline{T(n/4)} = T(\underline{n/8}) + 1$$

$$T(n) = T(\underline{n/8}) + 3$$

$$T(n) = \underline{T(n/8)} + 3$$

$n = n/8$

12.

$$T(N/2^i) = T\left(\frac{N/2^i}{2}\right) + 1$$
$$= T\left(\frac{N}{2^{i+1}}\right) + 1$$

$$T(N) = T\left(\frac{N}{2^k}\right) + 1 + 2^k$$

$$T\left(\frac{N}{2^k}\right) + 2^k$$

\dots k times

$$T(N) = \underbrace{T\left(\frac{N}{2^k}\right)}_{\Downarrow} + k$$

a^k

$$\boxed{\frac{N}{2^k}} = 1 \Rightarrow N = 2^k$$

Take \log_2 both sides

$$\log_2(N) = k \log_2(2)$$

$$\log_2(N) = k \times \log_2 2^1$$

$$\boxed{k = \log_2(N)}$$

$$T(N) = T\left(\frac{N}{2^{\log_2 N}}\right) + \log_2 N$$

↓

$$\begin{aligned} T(N) &= T\left(\frac{N}{2}\right)^1 + \log_2 N \\ &= \log_2 N + * \end{aligned}$$

↓

$$T.C. = O(\log_2 N)$$

H.W. $T(N) = T(N-1) + T(N-2) + 1$ $O(2^n)$



Towers of Hanoi

→ 3 towers A, B ≡ C

[1 to N]

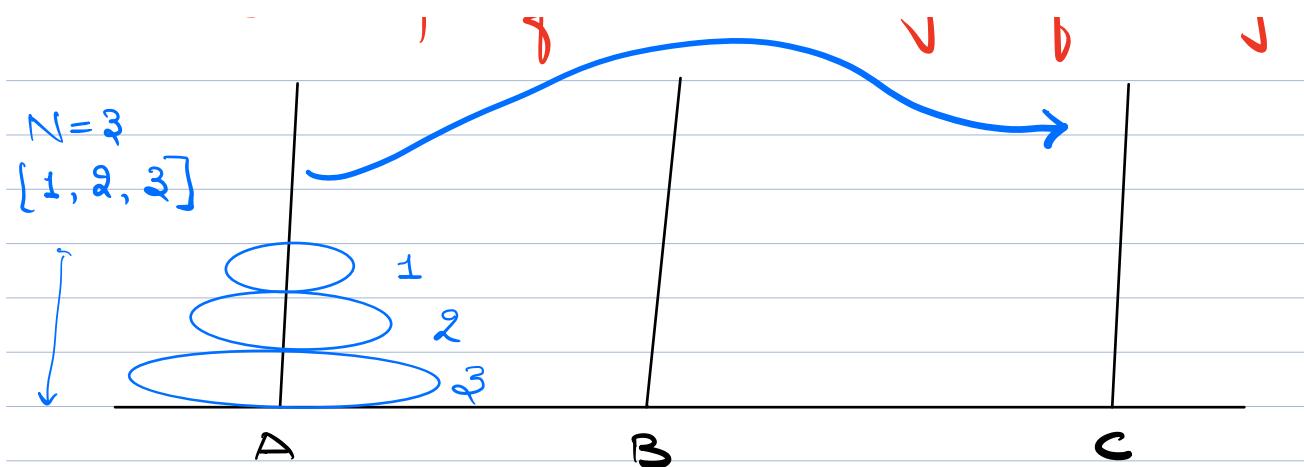
→ There are N disks on tower A

→ Move all disks from A to C (source destination
use B as int. tower)

Const

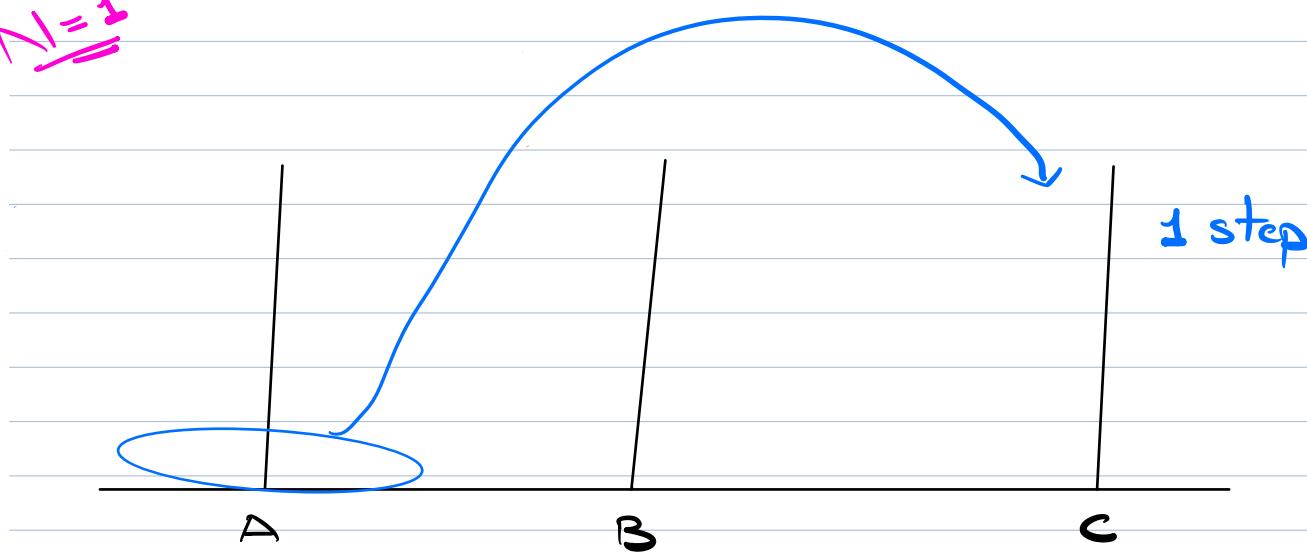
1) Move only one disk at a time

2) A disk numbered i can only sit
 on top of a disk j if $j < i$



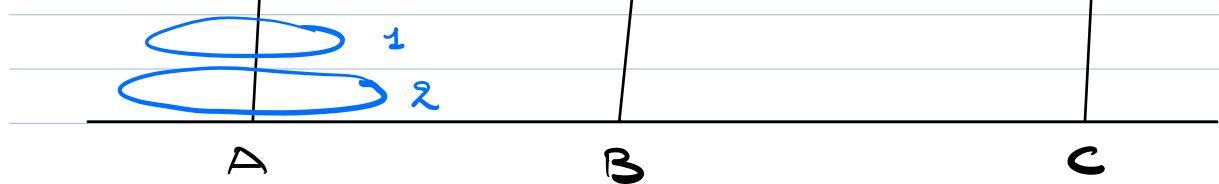
Calc min no. of steps to move N disks from A to C.

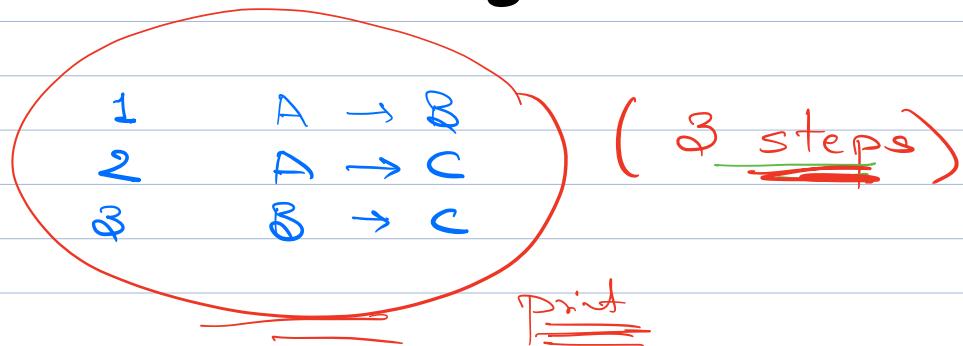
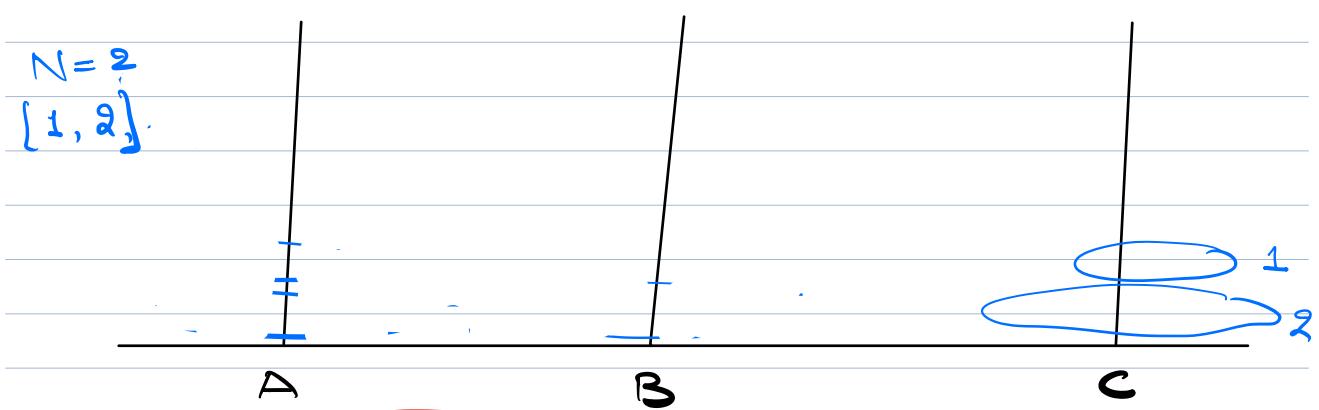
$\cancel{N=1}$



$\cancel{N=2}$

$\cancel{N=2}$
[1, 2]

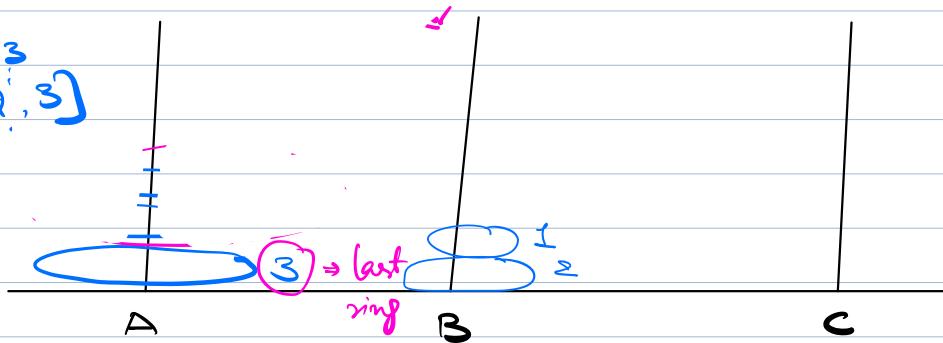




Step 1 \Rightarrow Move the ring 1 & 2 to tower B
 (source = A)
 dest = B
 int = C

$N=3$

$N=3$
[1, 2, 3]

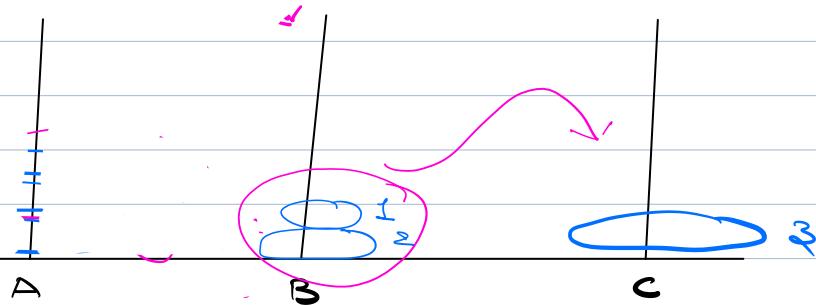


Step 2 \Rightarrow Move ring 3 from A to C

(1 step)

$N=3$

$N=3$
[1, 2, 3]



Step 3 \Rightarrow

Move rings 1 & 2 from B to C using A

source = B
dest = C
int = A

int - C

\Rightarrow 3 steps

$N=3$



$$\text{Ans} = 3 + 1 + 3 = 7$$

Generalise



Goal : Move N rings from src to dest using int.

- 1) Move $N-1$ rings from src to int
using dest tower
- 2) Move n^{th} ring from src to dest.
- 3) Move $N-1$ rings from int to dest

1 using src. 0 0

Code

```
int TOH (int char char char  
         ^ N , ^ src , ^ dest , ^ int) {
```

// Assumption : TOH function returns the
correct value of min no.
of steps to move N rings
from src to dest using
int.

if (N == 1) { return 1; } Base Case

```
int x = TOH(N-1, src, int, dest);
```

```
int y = TOH(1, src, dest, int);
```

```
int z = TOH(N-1, int, dest, src);
```

```
return (x + y + z);
```

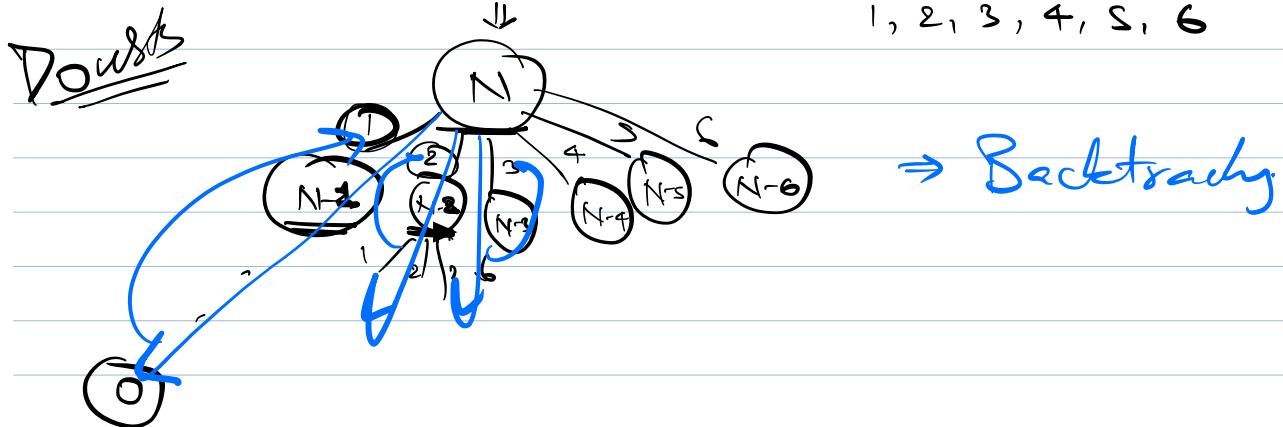
b

$$T(N) = T(N-1) + \overset{1}{T(1)} + T(N-1) + \overset{1}{1}$$

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

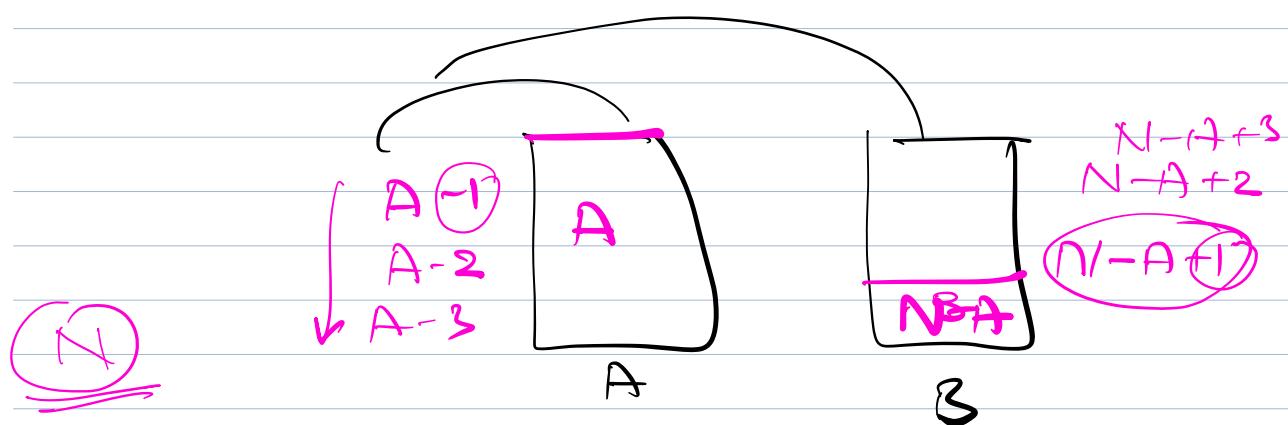
H.W:

H.W. Point the movement.



H.W. → GDP

stack \Rightarrow Abstract datatype

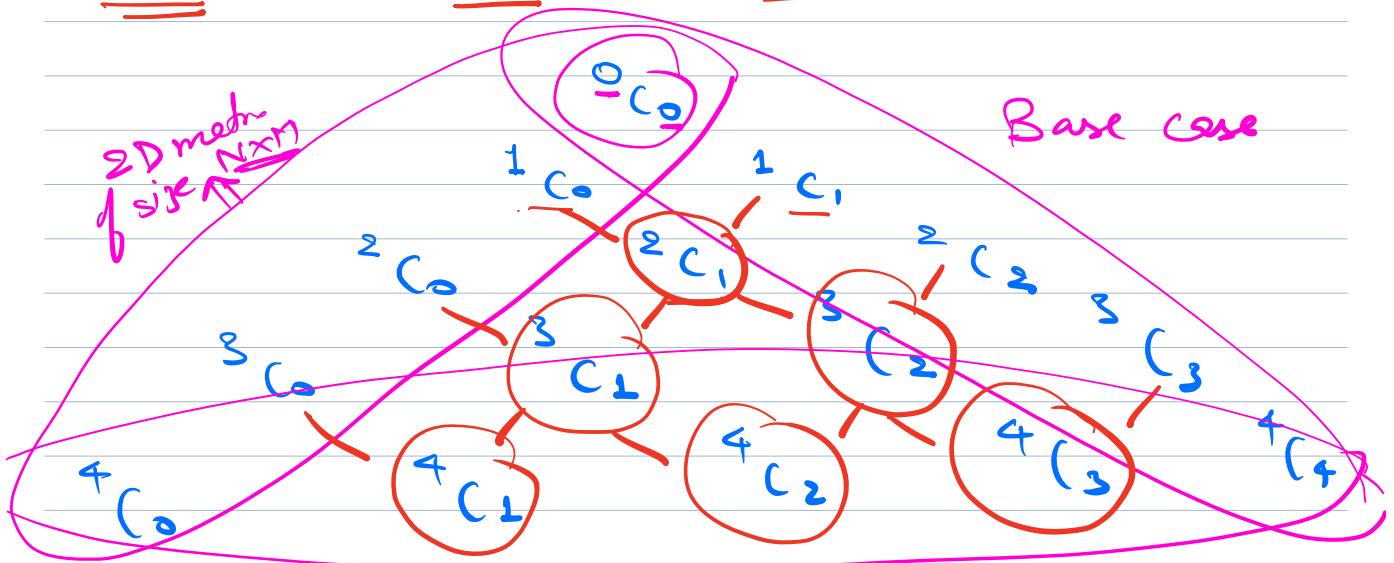


N > A
N > B

Pascal's

triangle

$$\underline{\underline{N}} \underline{\underline{C}}_R = \underline{\underline{N-1}} \underline{\underline{C}}_{R-1} + \underline{\underline{N-1}} \underline{\underline{C}}_R$$



$$\underline{\underline{N}} \underline{\underline{C}}_R$$

$$N \neq R$$

$$R = \infty \quad N = R \rightarrow ①$$

Max value of $R = N$

		0	1	2	3	4
		0	1	1	1	1
		1	1	2	1	1
		2	1	3	3	1
		3	1	4	6	4
		4	1	4	6	4

$$R > N$$

$$j = \emptyset \neq \Sigma \neq \Gamma$$

for (i = 0; $i \leq N$; $i++$) {

 for (j = 0; $j \leq i$; $j++$) {

 if ($j == 0 \text{ || } (j == i)$) {

$$M[i][j] = 1;$$

$$M[i][j] = M[i-1][j-1] + M[i-1][j];$$

}

 }

$$\text{T.C.} = \underline{\underline{O(N^2)}}$$