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

Recursion → Solving a problem using smaller instances of the same problem

↓

Subproblem

Q: Calculate the sum of the first N natural no's

$Sum(N) = 1 + 2 + 3 + 4 \ldots + N-2 + N-1 + N$

$= Sum(N-1)$

$Sum(N) = Sum(N-1) + N;$

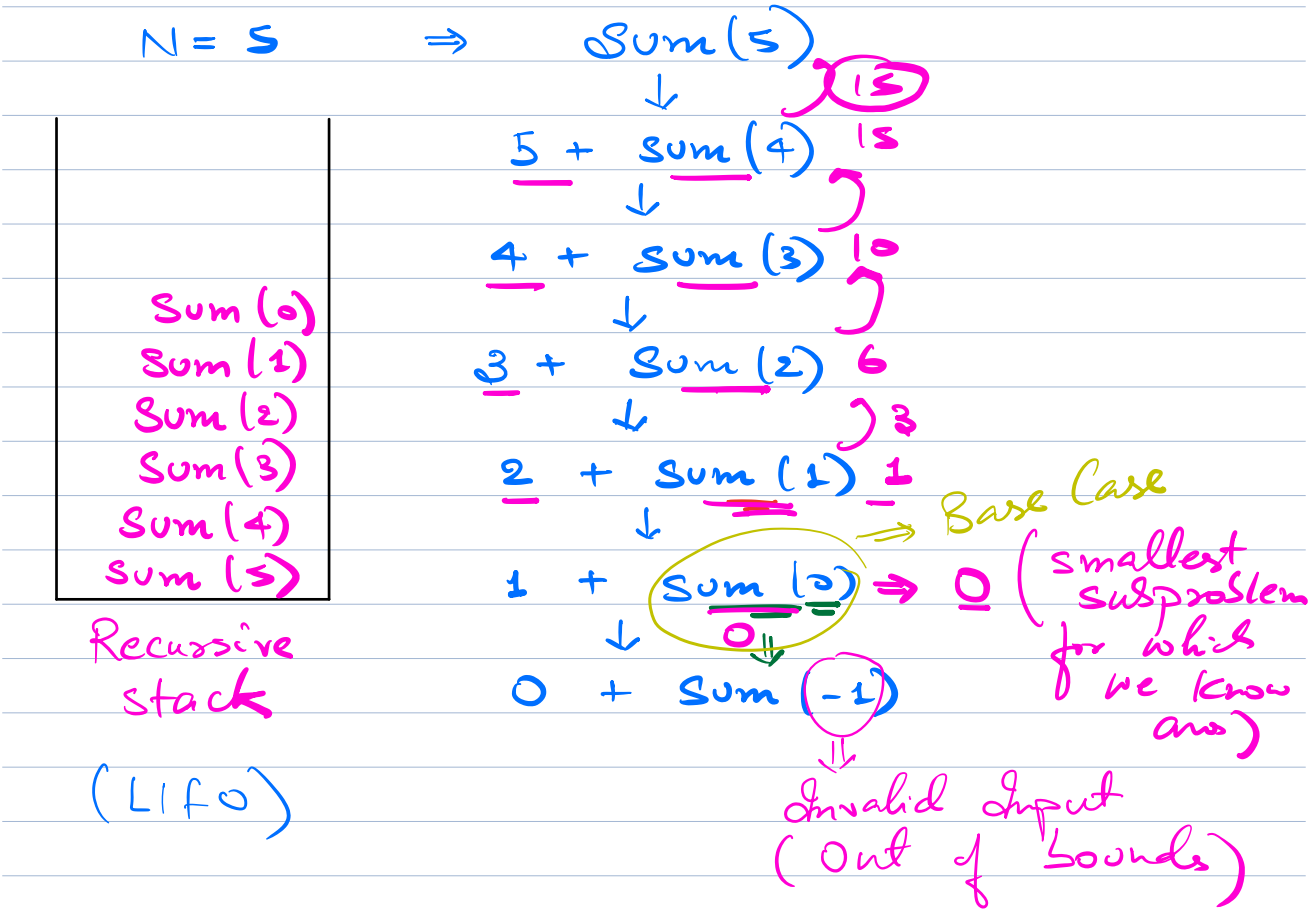
Steps for Recursion

1) Assumption : Decide what the function do & believes it does it

correctly

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

3) Base Case



Code

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

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

```
    return N + sum(N-1);
}
```

S.C. = O(N)

Q2 Fibonacci Series.

$fib(N) = fib(N-1) + fib(N-2)$;

does not follow main logic

$fib(1) = fib(0) + fib(-1)$

$fib(2) = fib(1) + fib(0)$

1 1 0

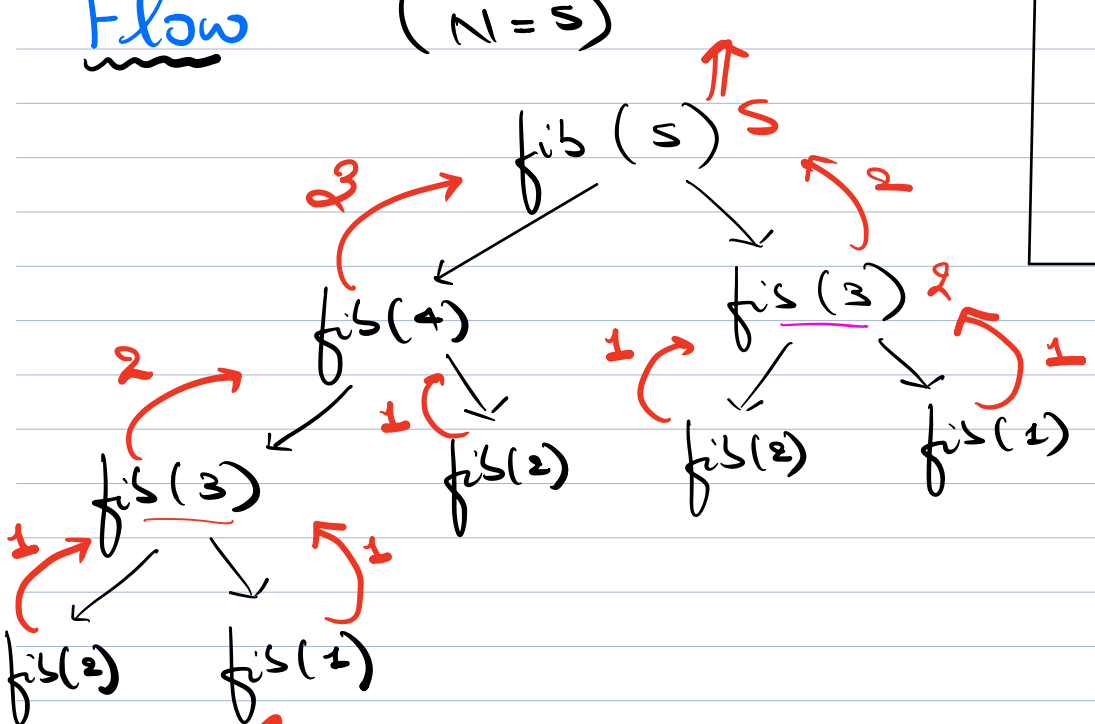
Base Case :

```
if (N==1 || N==2) {
    return 1;
}
```

Code

```
                    T(N)
int fib (N) {
    if (N == 1 || N == 2) {
        return 1;
    }
                          T(N-1)        T(N-2)
    return fib (N-1) + fib (N-2)
}
```

Flow



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

1) Sum

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

`return` → O(1); `N + sum(N-1);` → O(1); ⇒ recursive call to subproblem

Assume: Let $T(N)$ be the time taken to compute sum(N)

① $T(N) = T(N-1) + 1$ ⇒ Recursive relation

$T(0) = 1$ ⇒ Base Case

substitute $N = N-1$ in eqn ①

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

substitute to eqn ①

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

$N = N-2$ in eq (1)

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

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

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

$$\vdots$$

K times.

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

$\Downarrow$ Base Case

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

$$T(N) = T(N-N) + N$$
$$= T(0) + N$$ (1)

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

$\Downarrow$

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

2) Fast Power

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

$a^{12} = a \times a^{11}$

$= a^6 \times a^6$

$a^{13} = a \times a^6 \times a^6$

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

$N = N/2$ in eq (1)

$T(N/2) = T(N/4) + 1$

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

$N = N/4$ in eq (1)

$T(N/4) = T(N/8) + 1$

$T(N) = T(N/8) + 3$

$T(N) = T(N/2^3) + 3$

$N = N/2^3$

123

$$T(N/2^3) = T\left(\frac{N/2^3}{2}\right) + 1$$

$$= T\left(N/2^4\right) + 1$$

$$T(N) = T\left(\frac{N}{2^4}\right) + 1 + 3$$

$$T\left(\frac{N}{2^4}\right) + 4$$

$\vdots$

K times

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

a²

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

Take $\log_2$ both sides

$$\log(N) = K \log_2(2^{\cancel{K}})$$

$$\log(N) = K \times \log_{\cancel{2}}{}^{1}$$

$$K = \log(N)$$

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

$$\frac{N}{2^{\log_2 N}} = 1$$

$$T(N) = T(1) + \log_2 N$$

$$= \log_2 N + 1$$

$$\Downarrow$$

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

H.W: $T(N) = T(N-1) + T(N-2) + 1$

Towers of Hanoi

⇒ 3 towers A, B & C

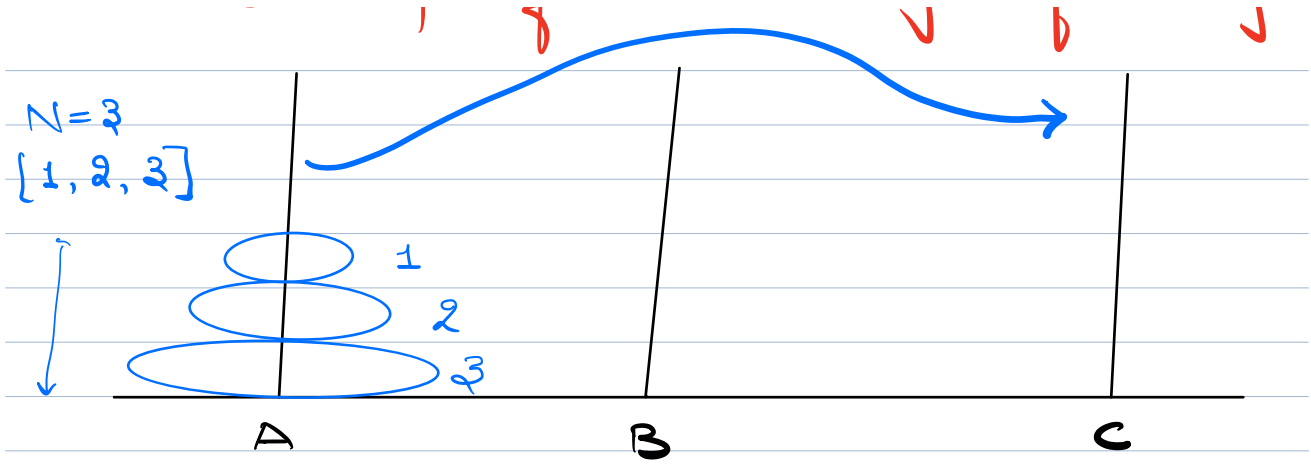
⇒ There are N disks [1 to N] on tower A

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

Const

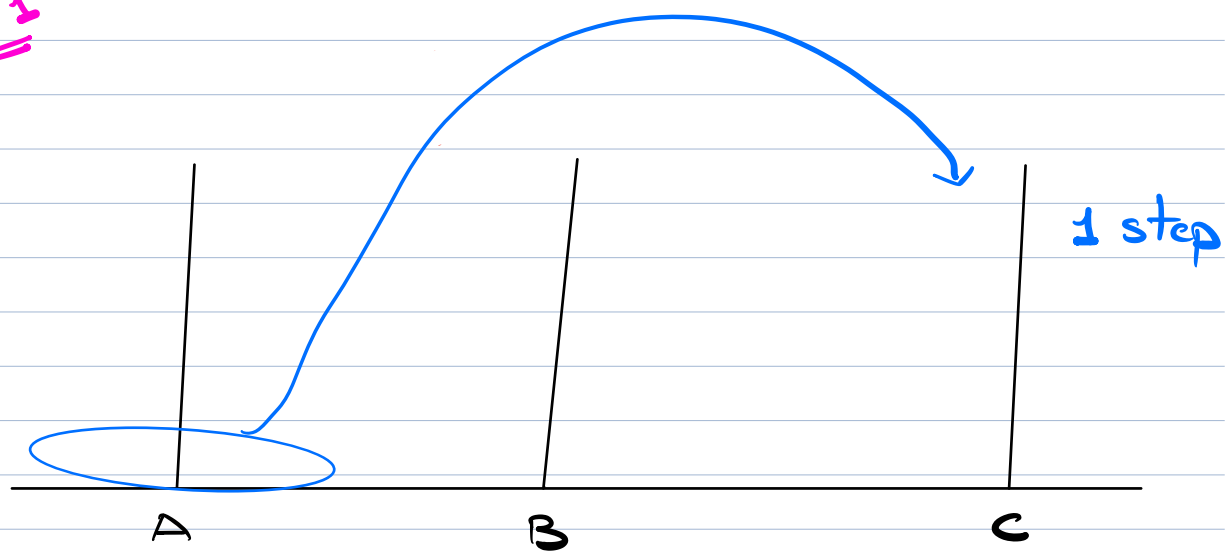
1) Move only one disk at a time

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

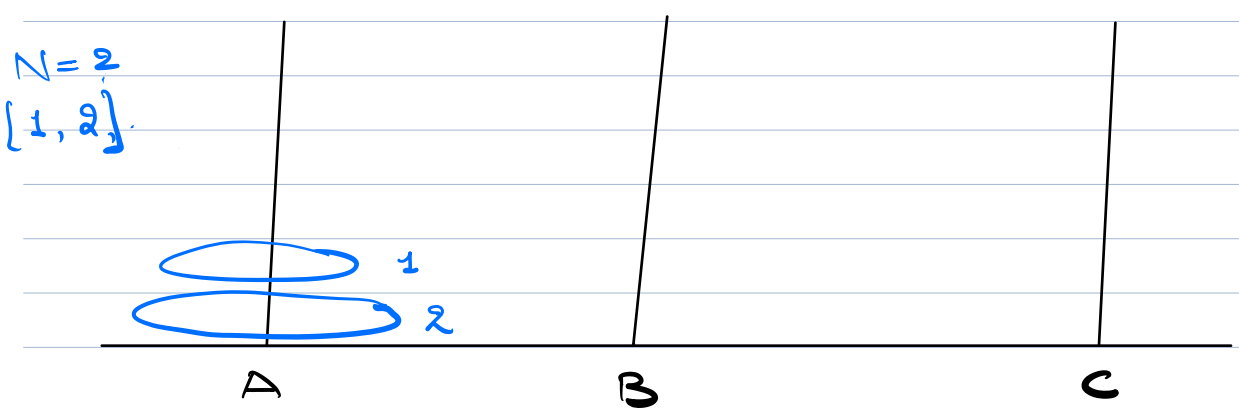


Calc min no. of steps to move N disks from A to C.

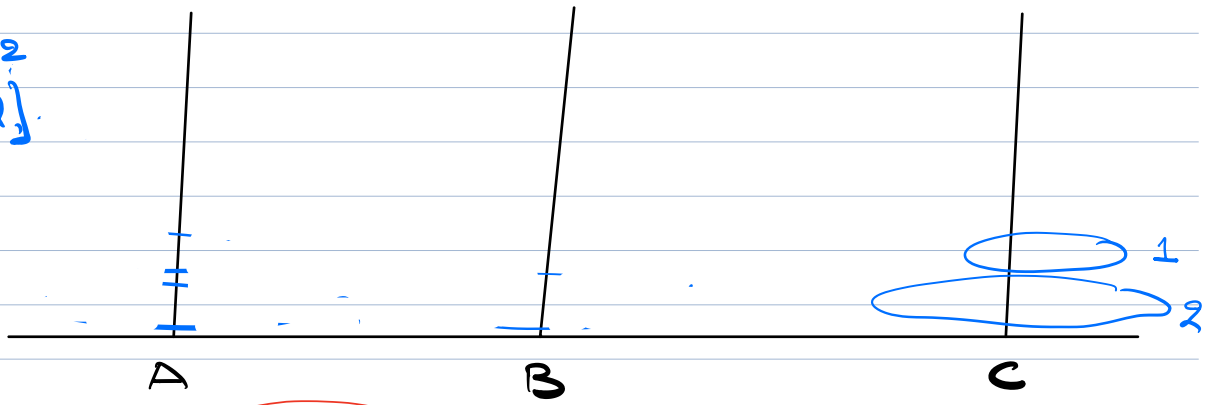
N = 1



N = 2



N = 2
[1, 2]



1	A → B
2	A → C
3	B → C

(3 steps)

Print

N = 3

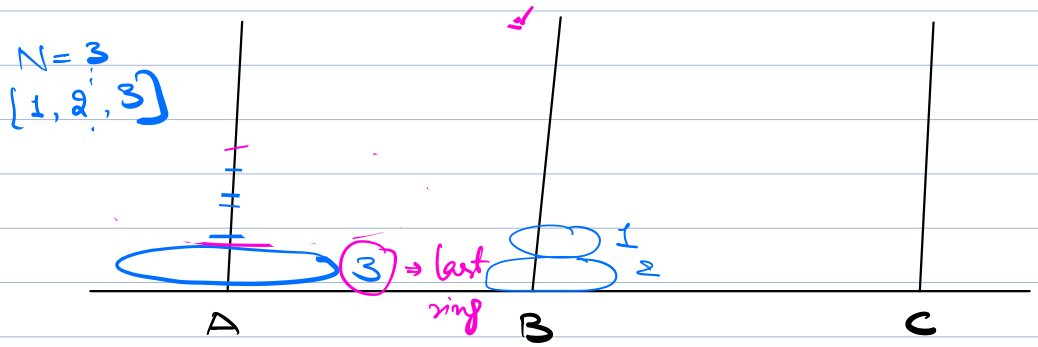
N = 3
[1, 2, 3]

Step 1 ⇒ Move the ring 1 & 2 to tower B

(source = A
dest = B
int = C)

3

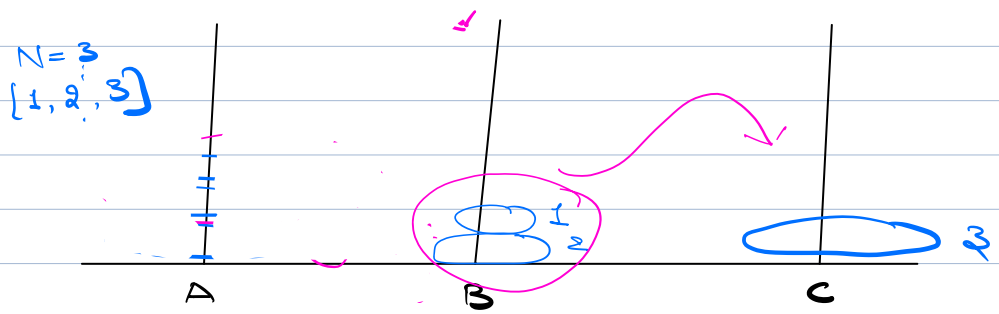
N=3



Step 2 ⇒ Move ring 3 from A to C

(1 step)

N=3



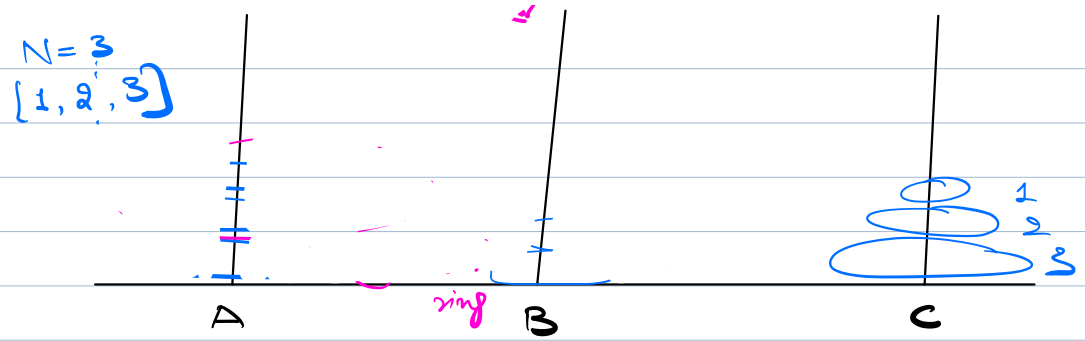
Step 3 ⇒ Move rings 1 & 2 from B to C using A

source = B
dest = C
int = A

⇒ 3 steps

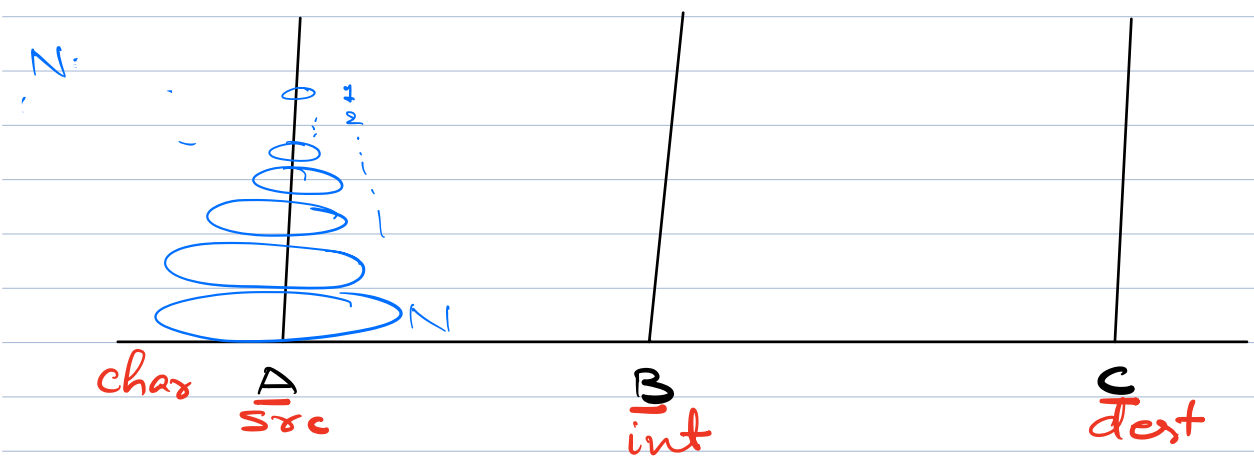
ind - C

N=3



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

using src.

Code

```
int TOH(int N, char src, char dest, char 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);
}
```

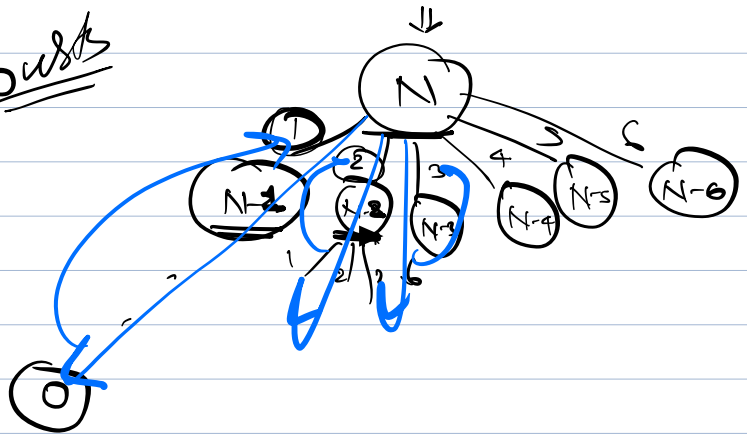
$T(N) = T(N-1) + T(1) + T(N-1) + 1$

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

H.W.

H.W: Print the movement.

Doubts

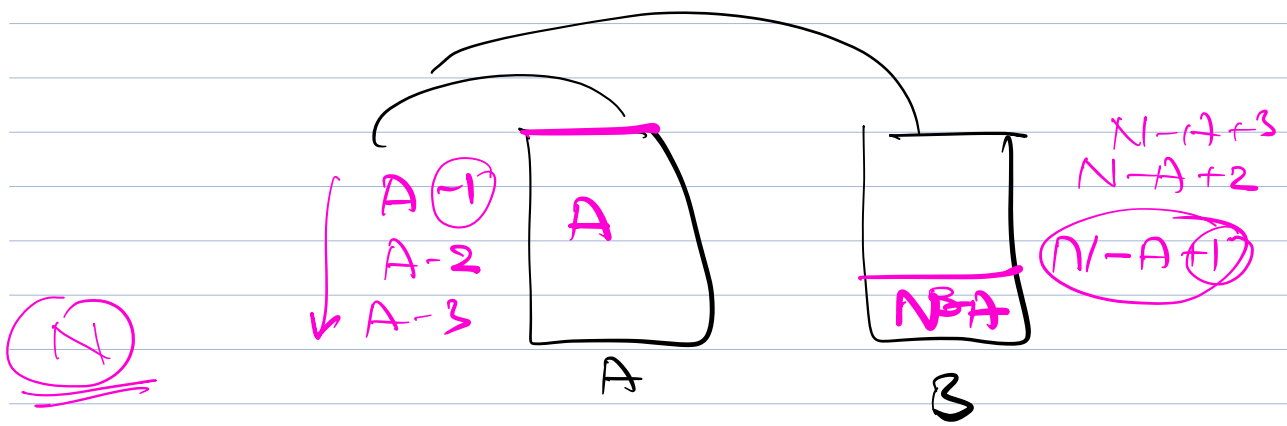


1, 2, 3, 4, 5, 6

⇒ Backtracking

H.W ⇒ GCD

stack ⇒ Abstract datatype



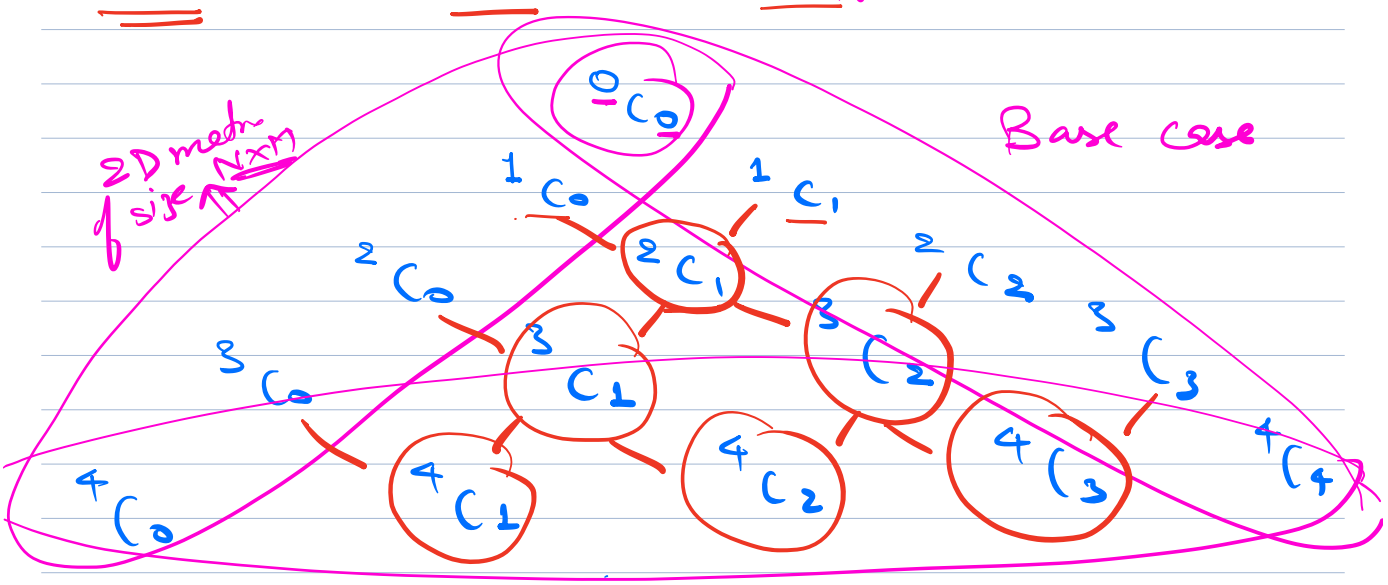
N > A

N X B

Pascal's

Triangle

$^{N}C_{R} = {}^{N-1}C_{R-1} + {}^{N-1}C_{R}$



N ≥ R

$^{N}C_{R}$

R == 0 → 1

N == R → 1

Max value of R = N

R <= N

i \ R	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

N

R > N

j = 0 1 2 3 4

```
for (i = 0; i <= N; i++) {
    for (j = 0; j <= i; j++) {
        if (j == 0 || (j == i)) {
            M[i][j] = 1;
        }
        M[i][j] = M[i-1][j-1] + M[i-1][j];
    }
}
```

T.C. = $O(N^2)$