Recursion 3

AGENDA:

- Find kth character
- Time and Space Complexity

Revise Classes & Objects before next class.

Q1 Find kth character

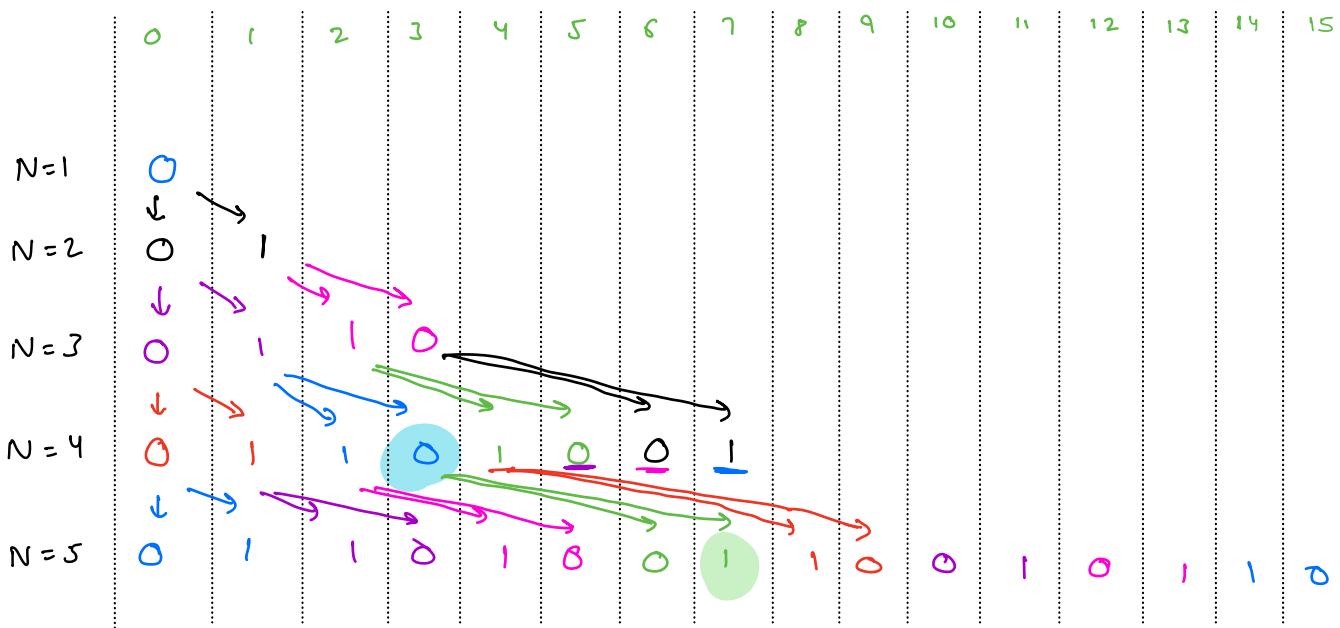
Each row is generated by replacing all elements of the previous row such that,

0 -> 01

1 -> 10

We always start with a 0 for N=1.

Given N, k. Find the kth element in Nth row.



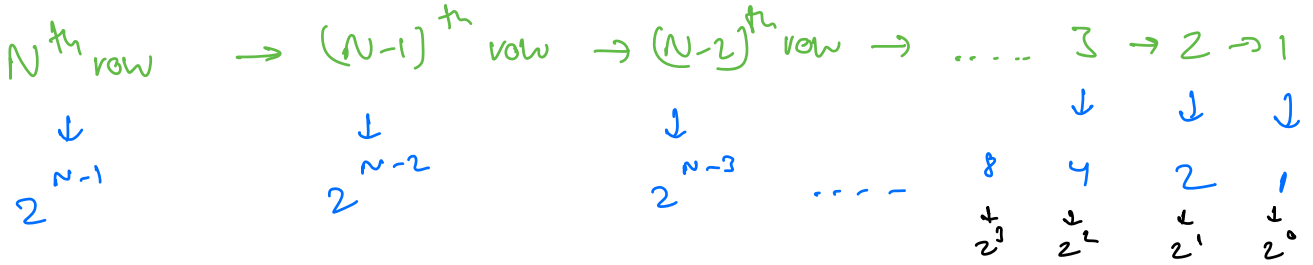
N	k		
5	10	⇒	0
4	3	⇒	0

$1 <= N <= 10^5$

$1 <= k <= 10^9$

Brute Force

Generate N^{th} row & get k^{th} char.



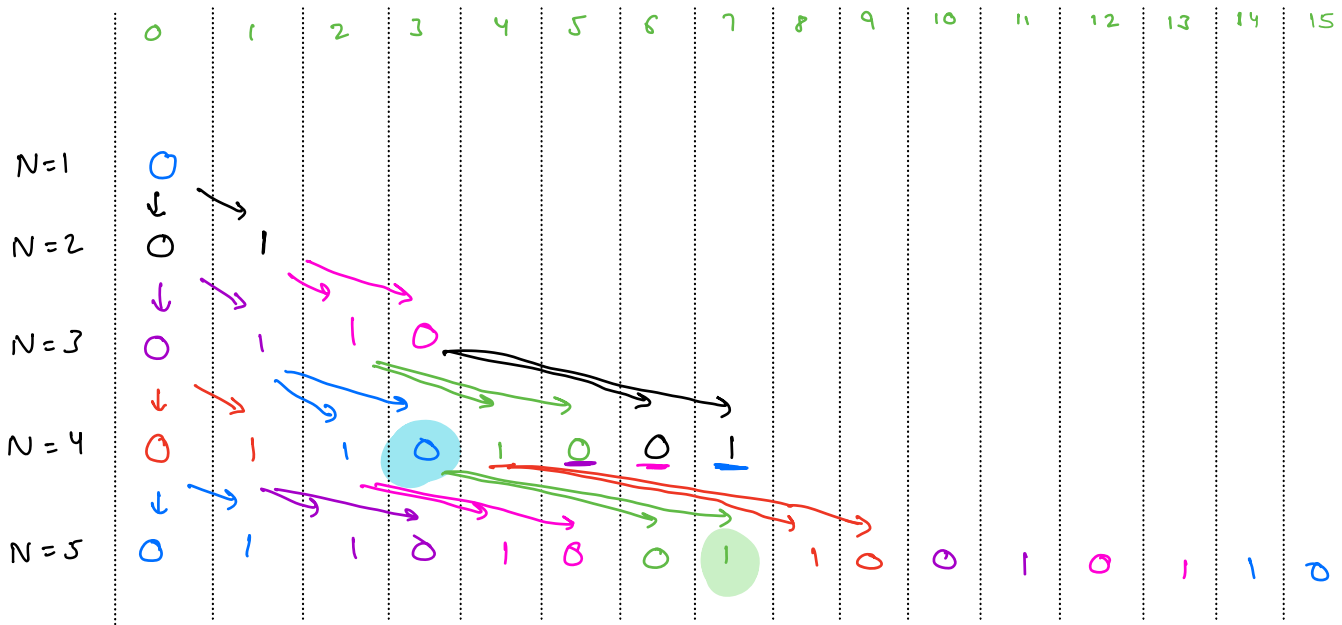
Total TC: $2^{N-1} + 2^{N-2} + 2^{N-3} + \ldots\ldots + 2^3 + 2^2 + 2^1 + 2^0$

$= 2^N - 1$

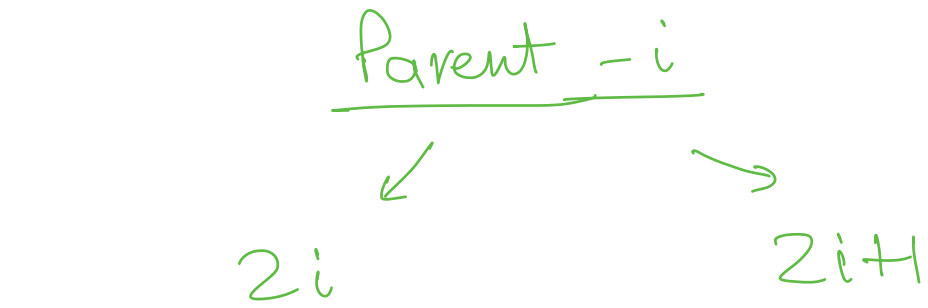
$= O(2^N)$

Using GP formula

Observations



index	children
2	4, 5
1	2, 3
4	8, 9
5	10, 11
i	2i, 2i+1



Child - j → Parent

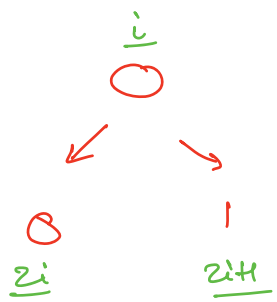
$\frac{j}{2}$ $j//2$

---

i^{th} : Data = 0

2i : Data = 0 (even)

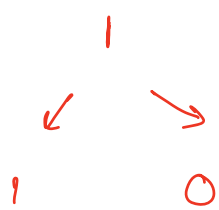
2i+1 : Data = 1 (odd)



i^{th} : Data = 1

2i : Data = 1 (even)

2i+1 : Data = 0 (odd)



→ for even indexes, they have the same data as their parent

→ for odd indexes, they have the inverse data as their parent

---

Pseudocode

row col

```
int find (N, K) {
    if (K == 0)
        return 0

    parentData = find(N-1, K/2)
    if (K is even)
        return parentData
    else
        return 1 - parentData
}
```

0 → 1
1 → 0

Time Complexity in Recursion

1. Assume time taken for input N to be some function T(N).
2. Write down recursive relationship based on your code's main logic.
3. Pick N value as such you know the result of T(N) directly, i.e. the base case.
4. Find generic expression for your recursive relationship.
5. Solve the recursive relationship by substituting value from step 3.

Example - Sum till N

```
int sum(N) {
    if (N == 1)
        return 1

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

$1+2+3+4+5 \ldots N$

$T(N)$ represents time taken for input of size $= N$

$T(1) = 1$

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

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

Quiz 1

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

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

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

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

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

$T(N) = T(N-5) + 5$

$T(N) = T(N-k) + k$ ← Generic Expression

Put $T(N-k) = T(1)$

$T(1)=1$

$\Rightarrow N-k = 1$

$\Rightarrow k = N-1$

Put value of k back in eq.

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

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

$= T(1) + N-1$

$T(1)=1$

$= 1 + N-1$

$= N$

TC: $O(N)$

Example - Factorial

```
int fact(N) {
    if (N == 0)
        return 1

    return N * fact(N - 1)
}
```

$T(0) = 1$

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

↳ $T(N) = T(N-k) + k$ ← Generic Expression

Put $T(N-k) = T(0)$

$\Rightarrow N - k = 0$

$\Rightarrow k = N$

Put it back in generic expression

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

$= 1 + N$

$T(0) = 1$

TC: $O(N)$

Break till 10:15

Example - Power function

```
int pow(a, n) {
    if (n == 0)
        return 1

    return a * pow(a, n - 1)
}
```

$T(0) = 1$

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

Same as factorial

TC : $O(N)$

Example - Power 2

$T(0) = 1$

$T(1) = 1$

```
int pow(a, n) {
    if (n == 0)
        return 1

    if (n % 2 == 0)
        return pow(a, n / 2) * pow(a, n / 2)
    else
        return a * pow(a, n / 2) * pow(a, n / 2)
}
```

$pow(a,1) = 1 \times \underbrace{pow(0)}_{O(1)} \times pow(0)$ (O(1))

12%

$T(N) = 2\,T\left(\frac{N}{2}\right) + 1$ ← (2-1)

Quiz 2

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

$T(N) = 4\,T\left(\frac{N}{4}\right) + 3$ ← (4-1)

$T\left(\frac{N}{4}\right) = 2\,T\left(\frac{N}{8}\right) + 1$

$T(N) = 8\,T\left(\frac{N}{8}\right) + 7$ ← (8-1)

$T(N) = 16\,T\left(\frac{N}{16}\right) + 15$ ← (16-1)

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

Generic Expression

Put $T\left(\frac{N}{2^k}\right) = T(0)$

$\Rightarrow \frac{N}{2^k} = 0$

$\rightarrow N = 0$

This is useless

We need another base case

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

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

$\Rightarrow N = 2^k$

$\Rightarrow \log_2 N = k$

Put k value back in generic expr.

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

$$= N\,T\left(\frac{N}{N}\right) + (N-1)$$

$N = 2^k$

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

$$= N + N - 1$$

$$= 2N - 1$$

TC: $O(N)$

Example - Power 3

```
int pow(a, n) {
    if (n == 0)
        return 1

    int p = pow(a, n / 2)

    if (n % 2 == 0)
        return  p * p
    else
        return a * p * p
}
```

$T(0) = 1$

$T(1) = 1$

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

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

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

$$T\left(\frac{N}{4}\right) = T\left(\frac{N}{8}\right) + 1$$

$$T(N) = T\left(\frac{N}{8}\right) + 3$$

$$T\left(\frac{N}{8}\right) = T\left(\frac{N}{16}\right) + 1$$

$$T(N) = T\left(\frac{N}{16}\right) + 4$$

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

Put $T\left(\frac{N}{2^k}\right) = T(0)$

$$\Rightarrow \frac{N}{2^k} = 0$$

$$\Rightarrow N = 0$$

This is meaningless

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

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

$$\Rightarrow N = 2^k$$

$$\Rightarrow k = \log_2 N$$

Put k in generic expr

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

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

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

$T(1) = 1$

$$= 1 + \log_2 N$$

TC: $O(\log_2 N)$

Example - Power 4

```
int pow(a, n, m) {
    if (n == 0)
        return 1

    int p = pow(a, n / 2, m)

    if (n % 2 == 0)
        return  (p * p) % m
    else
        return (a * (p * p) % m) % m
}
```

$T(0) = 1$

$T(1) = 1$

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

Same as Power 3

TC: $O(\log_2 N)$

Example - Fibonacci

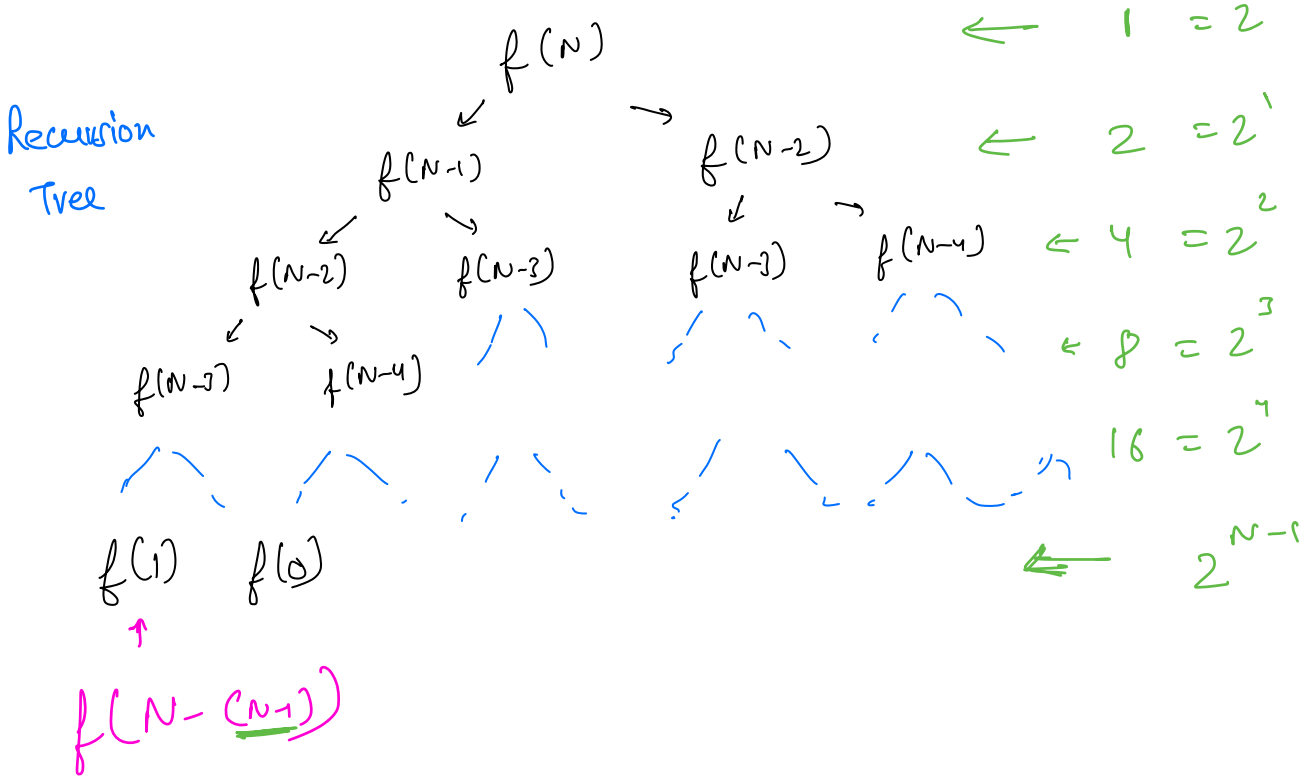
```
int fib(N) {
    if (N == 0 or N == 1)
        return 1

    return fib(N - 1) + fib(N - 2)
}
```

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

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

Solving through substitution is very tricky.

Recursion Tree



f(N - (N-1))

Total function calls

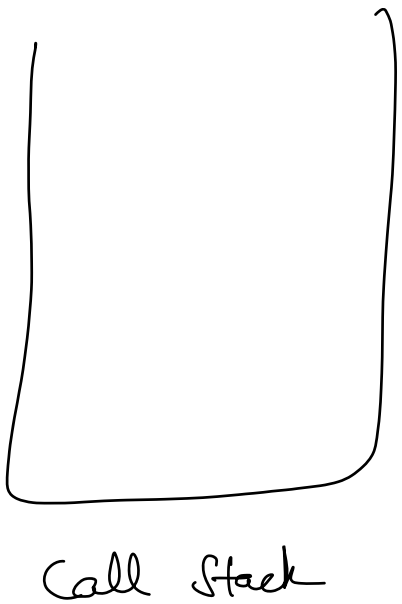
$= 2^0 + 2^1 + 2^2 + 2^3 \ldots\ldots 2^{N-1}$

$= O(2^N)$

Sum using GP

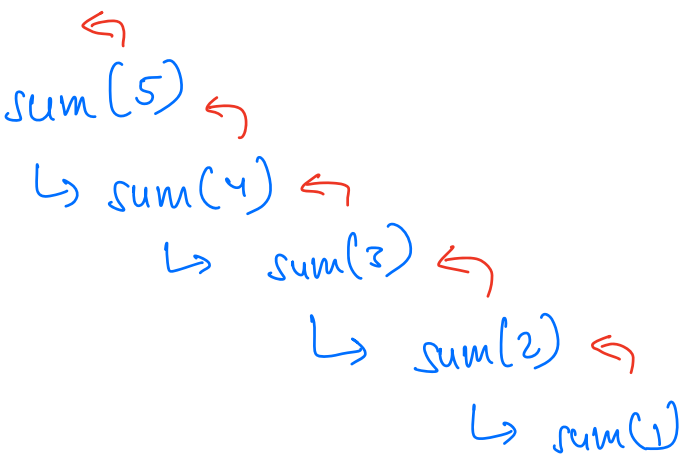
Space Complexity in Recursion

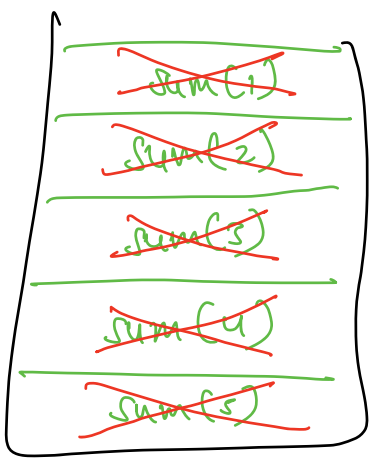
Max no. of functions present in call stack.



Example - Sum till N

```
int sum(N) {
    if (N == 1)
        return 1

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





Call Stack

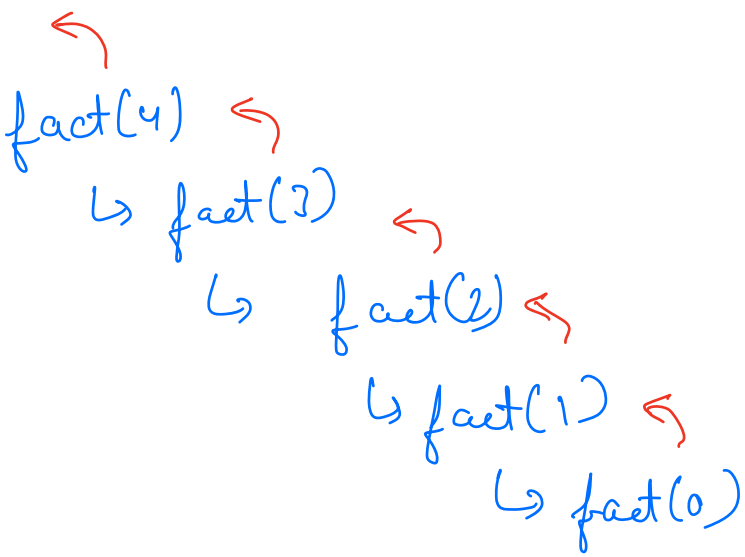
N = 5 → 5 function frames

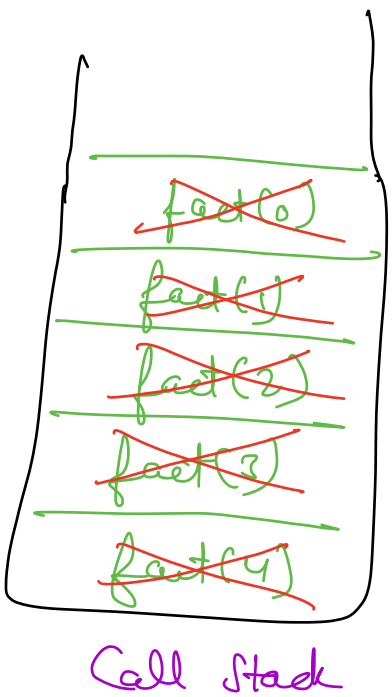
for input N → N

SC: O(N)

Example - Factorial

```
int fact(N) {
    if (N == 0)
        return 1

    return N * fact(N - 1)
}
```





N=4 → 5 frames

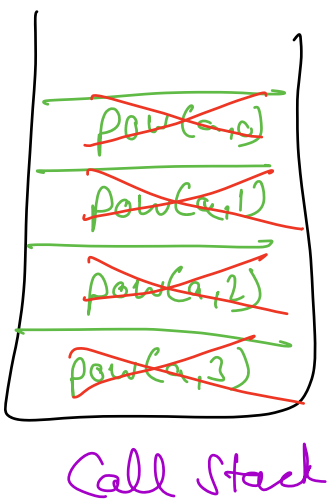
N → N+1 frames

SC : O(N)

Example - Power function

```
int pow(a, n) {
    if (n == 0)
        return 1

    return a * pow(a, n - 1)
}
```

pow(a,3)
↳ pow(a,2)
↳ pow(a,1)
↳ pow(a,0)



Call Stack

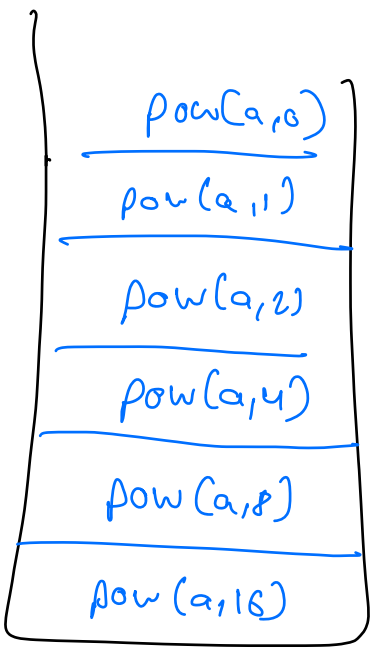
N=3 → 4 frames

N → N+1 frames

SC: O(N)

Example - Power 3

```
int pow(a, n) {
    if (n == 0)
        return 1

    int p = pow(a, n / 2)

    if (n % 2 == 0)
        return  p * p
    else
        return a * p * p
}
```



pow(a,16)
↳ pow(a,8)
↳ pow(a,4)
↳ pow(a,2)
↳ pow(a,1)
↳ pow(a,0)

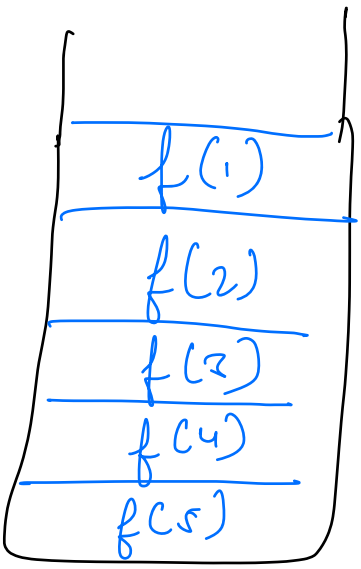
Input	Frames
16	6
32	7
64	8
128	9

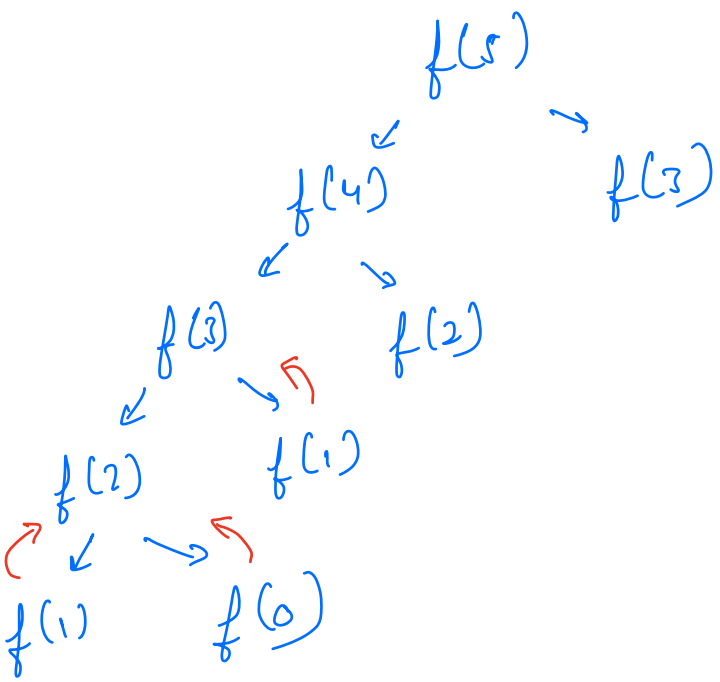
TC: $O(\log_2 N)$

$N \to \frac{N}{2} \to \frac{N}{4} \to \frac{N}{8} \cdots \to 1 \to 0$

Example - Fibonacci

```
int fib(N) {
    if (N == 0 or N == 1)
        return 1

    return fib(N - 1) + fib(N - 2)
}
```





Max frames = 5

For N → Max frames = N

Space = O(N)

Doubts

Thank You

Mongo → Database ⟷ Backend

MySQL ↗
PostgreSQL ↗

Backend → ↳ Springboot

Fullstack → ↳ Express (NodeJS)

↳ Django (Python)

↳ Go

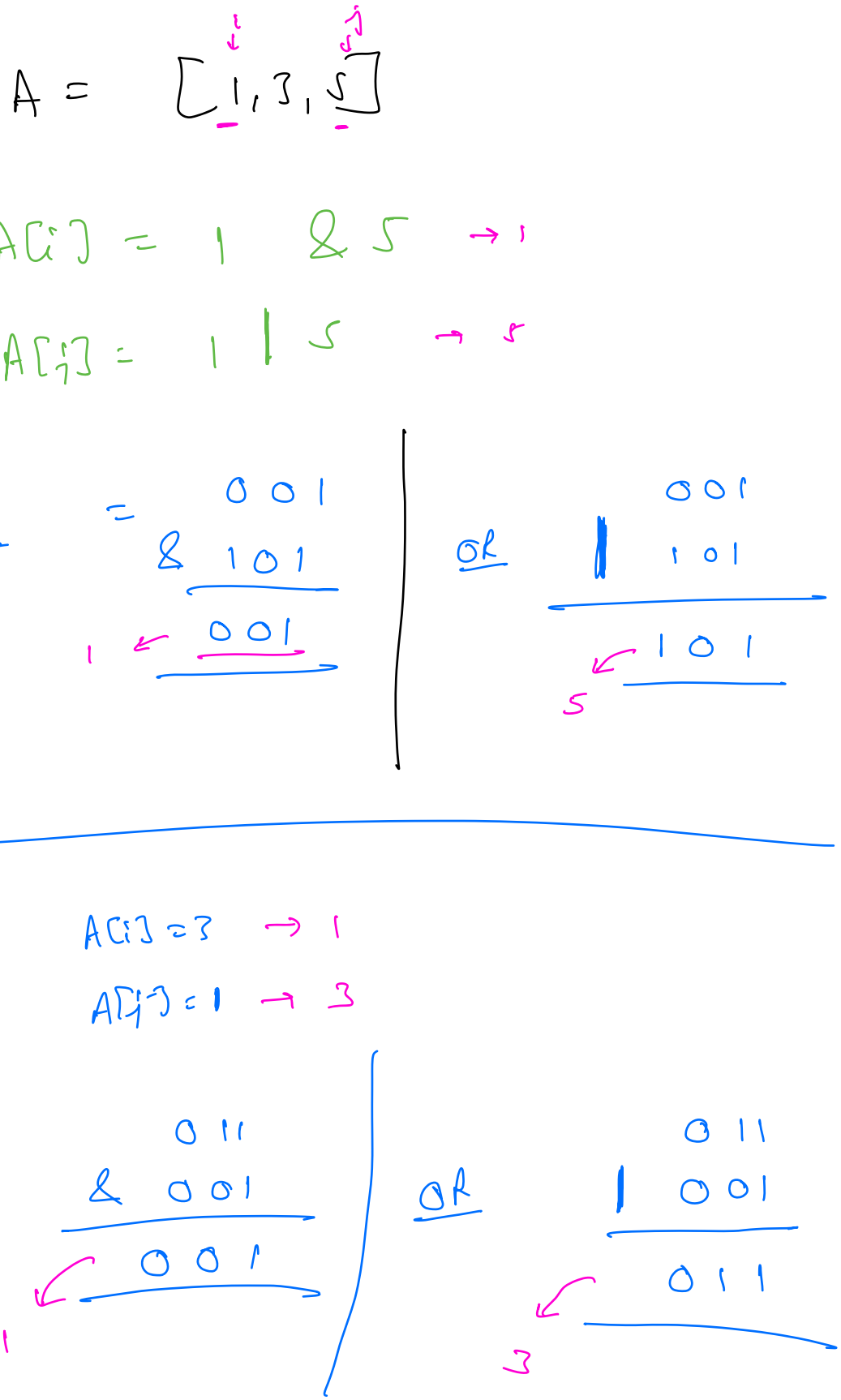
Bit Compression

Indexes — i, j

A[i] = A[i] & A[j]

A[j] = A[i] | A[j]

Do this for all pairs & return the XOR of entire array.


A = [1, 3, 5]
i
j
A[i] = 1 & 5 → 1
A[j] = 1 | 5 → 5
AND
1 = 001
& 5 = & 101
001
1
OR
001
| 101
101
5
A[i] = 3 → 1
A[j] = 1 → 3
AND
011
& 001
001
1
OR
011
| 001
011
3

$1 \wedge 3 = 3 \wedge 1$

$A = [1, 2, 4]$

Pairs = (1,2), (1,4), (2,4)

$x = 101$

$y = 010$

$x = 101$

$\&y = \&010$

000 ←

$x = 101$

$|y = |010$

$\rightarrow 111$

New $000 \wedge 111 \rightarrow 111$

Odd $101 \wedge 010 \rightarrow 111$

N rows

Row 1 → 2^0

Row 2 → 2^1

Row 3 → 2^2

Row 4 → 2^3

Row N → 2^{N-1}

Total = $2^0 + 2^1 + 2^2 + 2^3 + 2^{N-1}$

SC = $O(2^N)$

Revise Classes & Objects

before next lecture.

No assignments / HW


Good
Night


Thank
You


Friday