Algorithm Design

- Iterative Algo — Top-down design
- Recursive Algo — Bottom up design

For eg: Factorial(n)

1×2
$1 \times 2 \times 3$
⋮
$1 \times 2 \times 3 \times \cdots \times n$ — $n!$

n
$n \times (n-1)$
⋮
$n \times (n-1) \times \cdots \times 2 \times 1$ — $n!$

```
Fact = 1
for i = 2 to n
    Fact = Fact * i
```

↑ Iterative

```
if (n == 1) then return 1
else return n * factorial(n-1)
```

↑ Recursive

Problem

- Iterative Algo
- Recursive Algo

→ Which of the two is efficient

→ For every iterative Algo, is it true that there is an equivalent recursive algorithm.

Algorithm design

- Iterative Algorithm
  - Top-down design
- Recursive Algorithm
  - Bottom up design

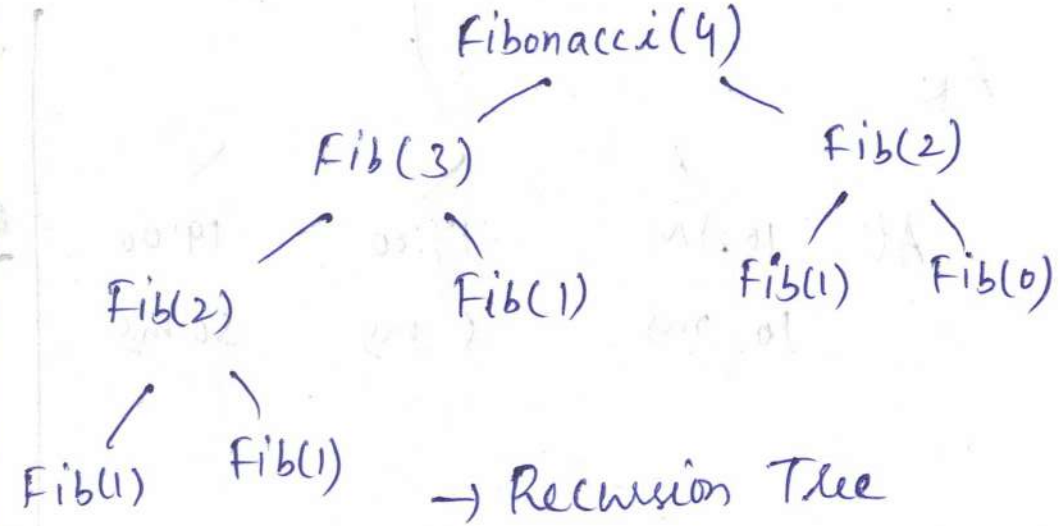
For eg:- Computing n^{th} Fibnocci number

1, 1, 2, 3, 5, 8, 13, 21, ...

```
a = 1
b = 1
for i=2 to n
    c = a+b
    a = b
    b = c
print c / return c
```

c=2	c=3	c=5
a=1	a=2	a=3
b=2	b=3	b=5

```
Fibonacci(int n)
If(n==0 || n==1) set 1
else set Fibonacci(n-1) + Fibonacci(n-2)
```



→ Recursion Tree

→ Is recursive code doing more work than iterative code.??

Analysis

Problem $\left\{\begin{array}{l} \text{Algo 1} \\ \text{Algo 2} \\ \text{Algo K} \end{array}\right.$

Time | Space

↓ ↳ # locations / # words used by the algorithm

System time??

Intel I5

$A_1 \sim Avg(I_1, I_2, -- I_K)$

$A_2 \sim Avg(I_1, I_2, -- I_K)$

$\vdots$

A_K

Ex:- A_3 is Minimum

AMD

A_7 is minimum

Intel A7

A_5 is minimum.

At	10 AM	17:00	19:00
	10 ms	5 ms	50 ms

'Measure' which is System Independent

Time Complexity Analysis, we shall focus on

'Step Count' → { The Count / frequency of fundamental / primitive operation involved in the algo

Analysis

Find Max

operations

```
Max = A[1]              → Assignment
for i = 2 to n          → Assignment, Comparison (Conditional check), Increment
  if (A[i] > max)       → Comparison
    Max = A[i]          → Assignment

Let Max
```

"Step Count Analysis"

1) Count of Assignment ops
   Comparison
   Increment

} which operation is frequent / dominant "primitive operation"

As part of time Complexity analysis (Step Count analysis)

our focus on "primitive ops"

"How many times we perform the primitive ops"

"Count of primitive ops"

Analysis

Find Max

operations

1 —	Max = A[1]	→ Assignment — 1
n-2+1+1 —	for i = 2 to n	→ Assignment, Comparison (Conditional check), Increment (n) (n) (n)
True = n, False = 1		
(n-1) —	if (A[i] > max)	→ Comparison — (n-1)
(n-1) —	Max = A[i]	→ Assignment — (n-1)
Worst Case → (n-1) —		
1 —	Set Max	

1 + n + (n-1) + (n-1) + 1 = 3n steps

Fund Indy

"Step Count Analysis"

↳ Count of Assignment ops
Comparison
Increment

} which operation is frequent / dominant "primitive operation"

As part of time Complexity analysis (Step Count analysis)

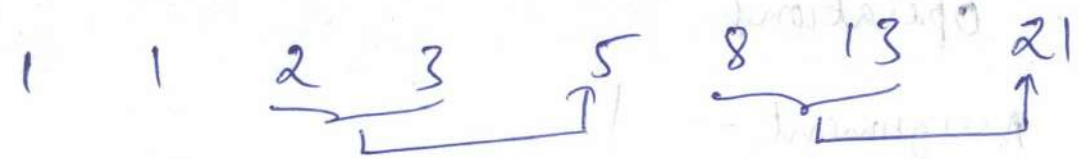
our focus on "primitive ops"

"How many times we perform the primitive ops"

"Count of primitive ops"

fibonacci sequence

1 1 2 3 5 8 13 21



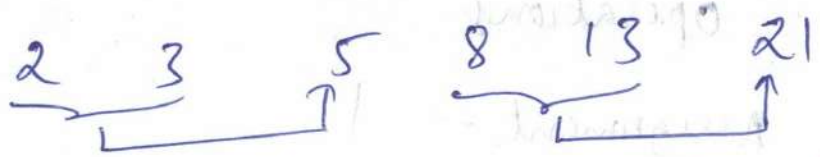
```
a = 1                          1
b = 1                          1
for i = 2 to n                 n-2+1+1 = n
    c = a + b                  n-1
    print c                    n-1
    a = b                      n-1
    b = c                      n-1
```

5n-2 steps

instructions

fibonacci sequence

1 1 2 3 5 8 13 21



```
a = 1                          1
b = 1                          1
for i = 2 to n                 n-2+1+1 = n
    c = a + b                  n-1
    print c                    n-1
    a = b                      n-1
    b = c                      n-1
```

5n-2 steps

instructions

```
1
1
3n (n: Inc, n: Assign, n: Comp)
2(n-1) (Addition, Assign)
n-1
n-1
n-1
```

8n-3

Recursive Factorial

```
Fact(int n)
if(n==1) then ret 1
else ret n * Fact(n-1)
```

n=1	n ≥ 2
1 (if)	1 (If)
1 (ret 1)	1 (Multiplication)
2	1 (ret)
	T(n-1)

↓ Cost of recursive subproblem of size (n-1)

T(n): Cost / Step Count of recursive subproblem of size 'n'

Recurrence relation

$$T(n) = 2 \text{ if } n=1$$
$$= 3 + T(n-1) \text{ if } n \geq 2$$

Suppose n=3 →

$$T(n) = 3 + T(2)$$
$$= 3 + 3 + T(1)$$
$$= 3 + 3 + 2$$

Recursive Fibonacci

```
Fib(int n)
if(n==0 || n==1) ret 1
else ret Fib(n-1)+Fib(n-2)
```

n=0 or 1	n≥2
1 (If)	1 (If)
1 (ret)	1 (Addition)
	1 (ret)
	T(n-1) for recursive subproblem of size(n-1)
	T(n-2) " " " " (n-2)
	3+T(n-1)+T(n-2)

Recurrence relation is

$$T(n) = \begin{cases} 2 & \text{if } n=0 \text{ (or) } 1 \\ 3+T(n-1)+T(n-2) & \text{if } n \geq 2 \end{cases}$$

Ex:- Fib(3)

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

$= 3+3+T(1)+T(0)+T(1)$

$= 3+3+2+2+2$

$= 12$

```
for i=1 to n          → (n+1)
for J=1 to n          → i=1: (n+1) + i=2: (n+1) + ---- + i=n: (n+1)
                        = n(n+1)
    print i, J        → i=1: J=1 -- J=n  (n)
                        i=2: J=1 --- J=n  (n)
                        ----
                        i=n: J=1 --- J=n  (n)
```

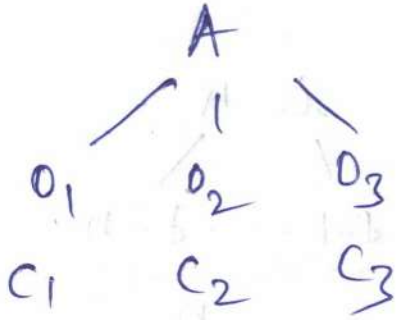
$= n^2$

Step Count $= n+1 + n(n+1) + n^2$

Order of Growth

problem $\begin{cases} A_1 \sim & \text{Step Count method} = 2n+3 \text{ (Comp, Inc, Assign)} \\ A_2 \sim & = 3n-1 \end{cases}$

$n^2 + 5n - 2$ [Comp, Inc, Cost of ret / recursive call]



Step Count = $\underbrace{C_1 + C_2 + C_3}$
Compare this fundamental ops

Ex:- ① $4n + n + 100$

② $3n^2 + n^3 + n - 5$
Quad Cubic Linear

which op is frequent
" primitive "
" dominant "

= Frequent term
primitive term
dominant term
} "Cubic"

Order of Growth; Focus is on "Significant terms"

↳ Asymptotic Analysis
- ↳ upper bound Analysis (Bg-oh Notation 'O')
- ↳ Lower bound (Big-Omega 'Ω')
- ↳ Tight bound (Theta Notation 'Θ')

P → A → Step Count = $3n^2+5n-2$

$3n^2+5n-2 \leq 5n^2 \quad \forall n \geq 4$ ('O')

$3n^2+5n-2 \geq 100n \quad \forall n \geq 100$ ('Ω')

$n^2 \leq 3n^2+5n-2 \leq 5n^2$ ('Θ')