Rate of Growth

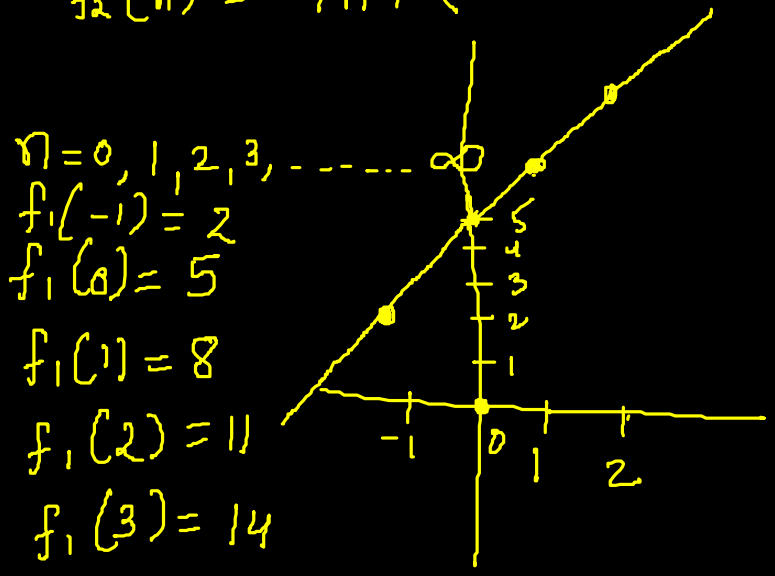
For n → infinity

Linear

$f(n) = an + b$

$f_1(n) = 3n + 5$

$f_2(n) = 4n + 2$



Quadratic :-

$f(n) = an^2 + bn + c$

$f_1(n) = 2n^2 - 5n + 2$

$f_2(n) = 3n^2 - 2$

$n = 0, 1, 2, 3, \ldots \infty$

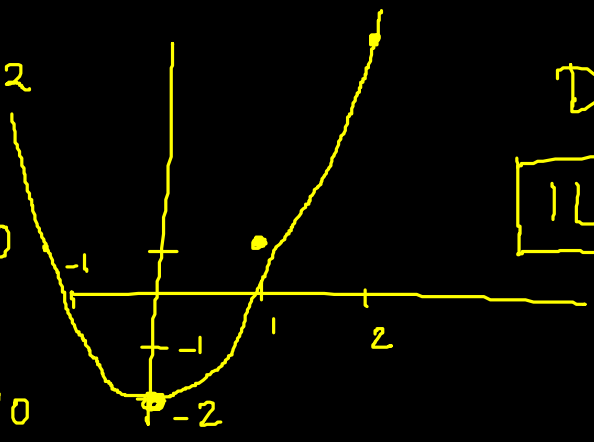
$f_2(0) = -2$

$f_2(1) = 1$

$f_2(2) = 10$

$f_2(-1) = 1$

$f_2(-2) = 10$



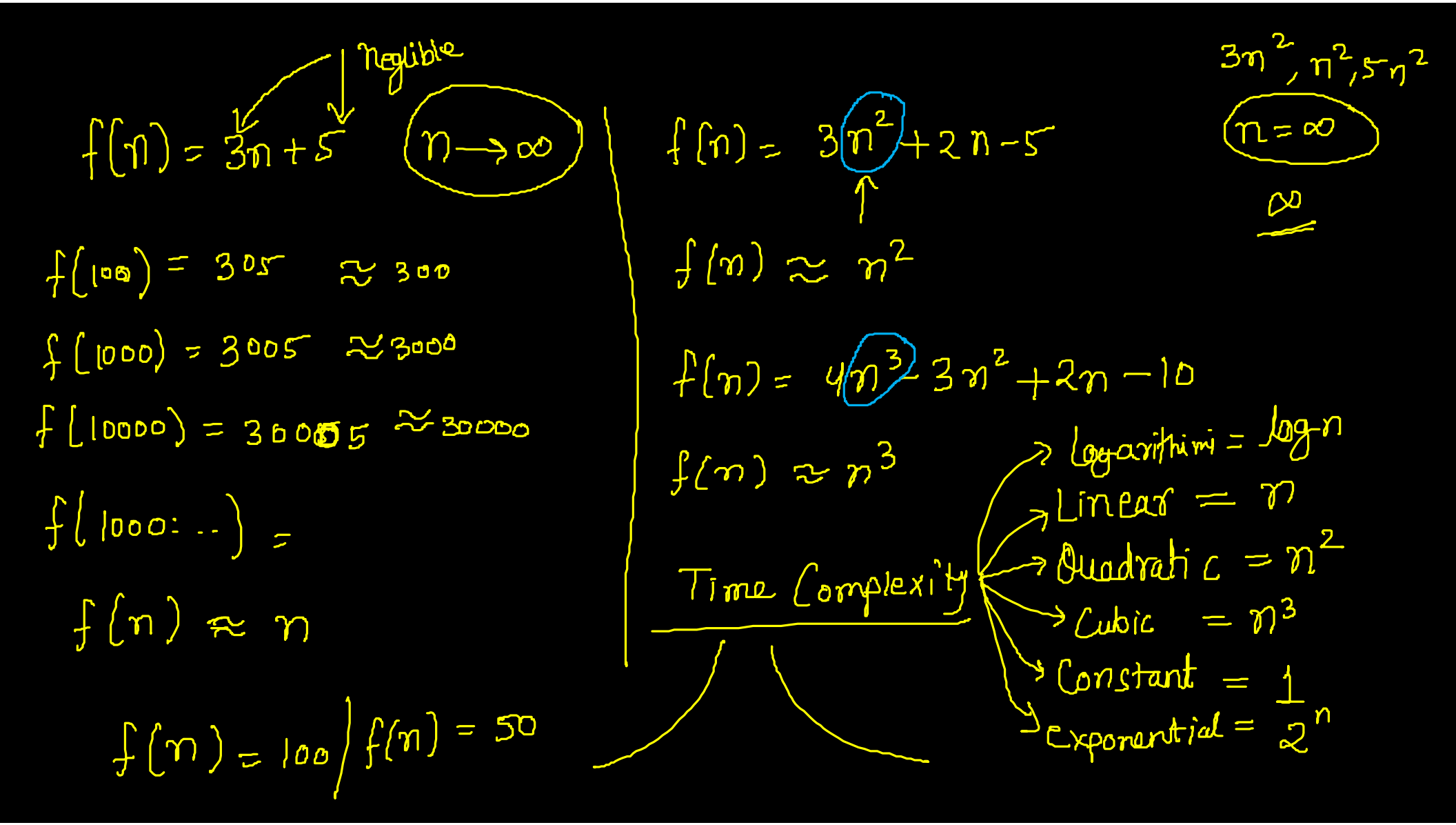
Cubic :-

$f(n) = an^3 + bn^2 + cn + d$

1) fridge 2) pen 3) Pencil 4) eraser

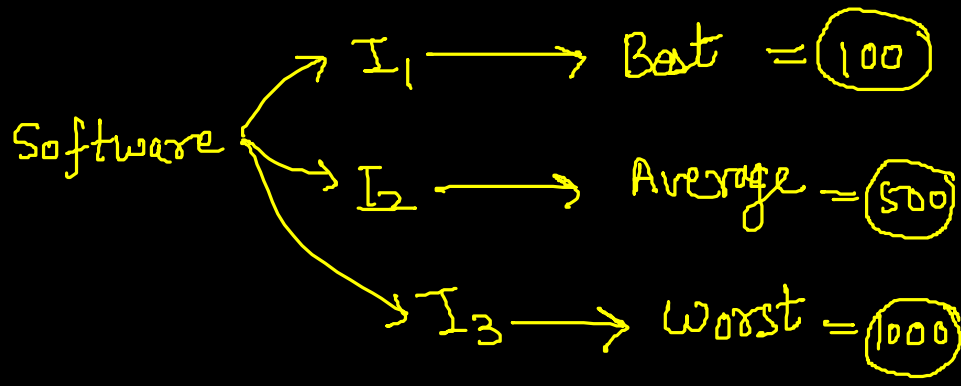
Dost

1 Lakh + 10


Negligible
$f(n) = 3n + 5$
$n \to \infty$
$f(100) = 305 \approx 300$
$f(1000) = 3005 \approx 3000$
$f(10000) = 30005 \approx 30000$
$f(1000\ldots) =$
$f(n) \approx n$
$f(n) = 100$ / $f(n) = 50$
$f(n) = 3n^2 + 2n - 5$
$f(n) \approx n^2$
$f(n) = 4n^3 - 3n^2 + 2n - 10$
$f(n) \approx n^3$
Time Complexity
Logarithimi = $\log n$
Linear = n
Quadratic = n^2
Cubic = n^3
Constant = 1
Exponential = 2^n
$3n^2, n^2, 5n^2$
$n = \infty$
∞

Rate of Growth

1) Best Case → Lower Bound
2) Worst Case → upper Bound
3) Average Case → Tight Bound

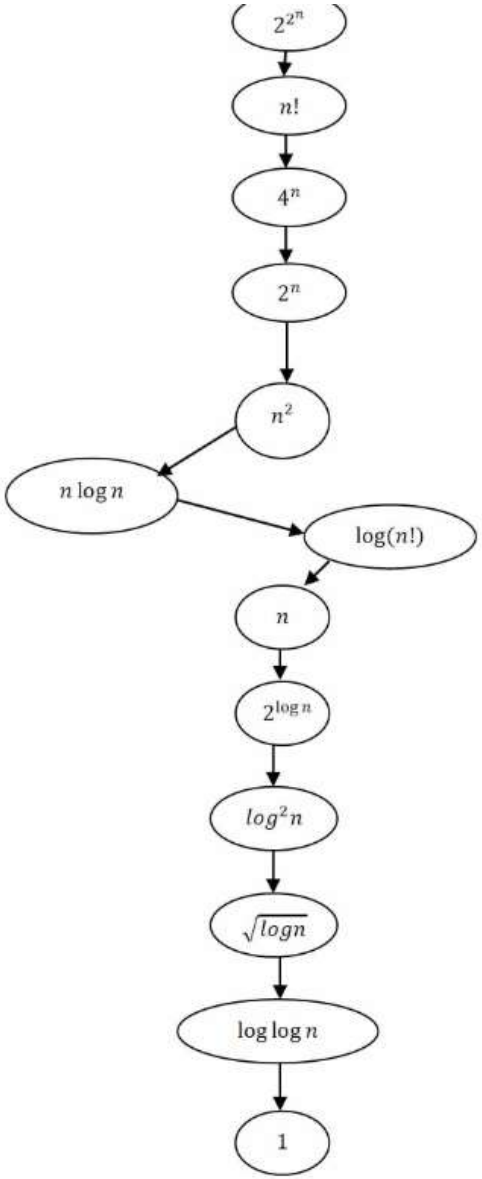


fastest → 1 month → Best Case
Slowest → 6 month → worst case
approx → 3 month → Average Case

Lower Bound <= Tight Bound <= Upper Bound

Time Complexity & Its Name

- Constant ⟶ 1
- Logarithmic ⟶ $\log n$
- Linear Logarithmic ⟶ $n \log n$
- Linear ⟶ n
- Quadratic ⟶ n^2
- Cubic ⟶ n^3
- Exponential ⟶ $2^n, 3^n, a^n$



Decreasing Rates Of Growth

Comparison of Time Complexity

worst → fastest

$1 < \log\log n < \sqrt{\log n} < \log^2 n < 2^{\log n} < n$

$< \log(Ln)$

$n = 2$

slow

n^2 n^3

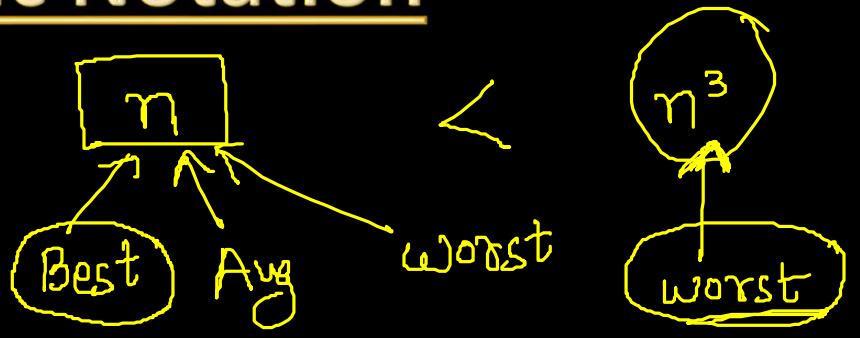
4 < 8

fast

← Best

Asymptotic Notation

1) Worst Case → Big – O Notation
2) Best Case → Omega Notation
3) Average Case → Theta Notation



1 cm
1 m
1 km
1 litre
1 kg
1 ml
1 sec

Worst Case → $O(n^3)$

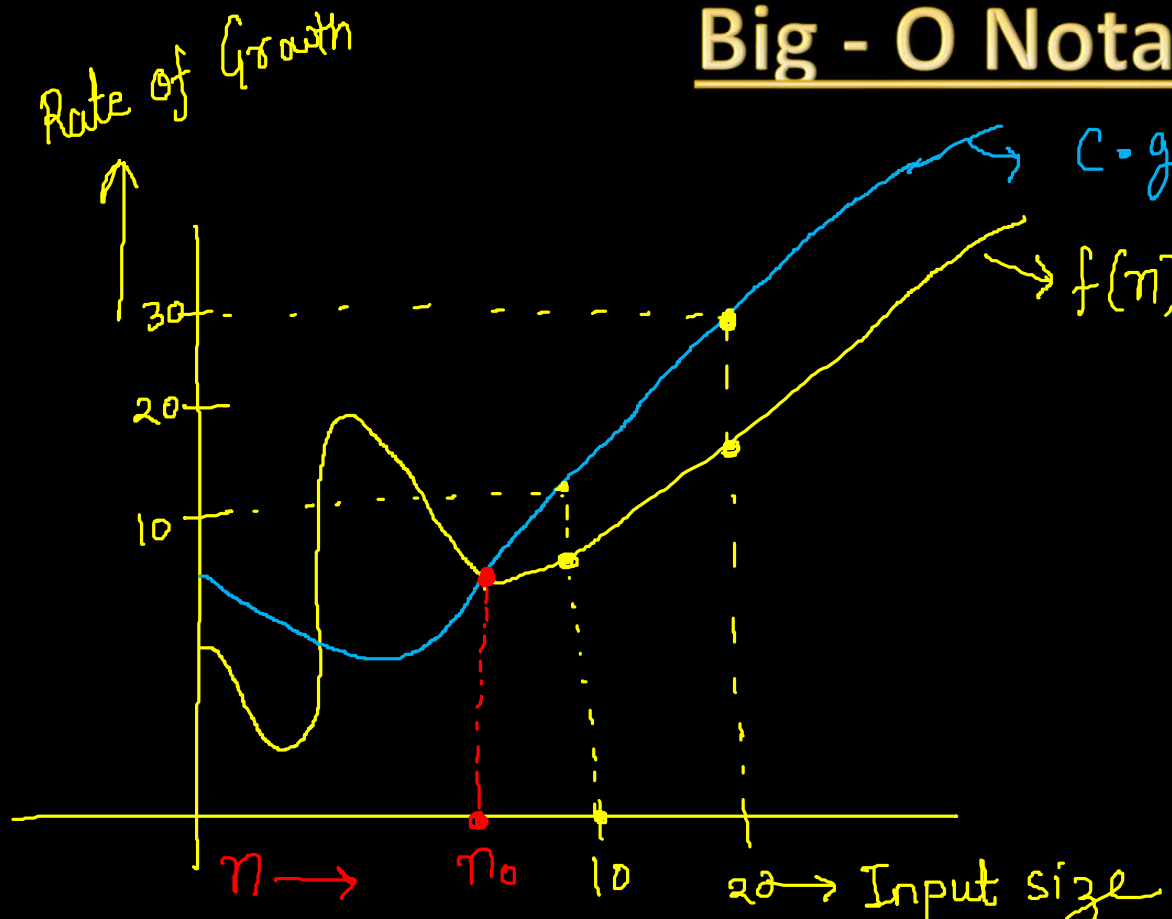
Best Case → $\Omega(n)$

Average Case → $\Theta(n^2)$

Prateek Jain
9555031137

for Guidance

Big - O Notation



$\forall \Rightarrow$ for all

$f(n) = 3n + 5$

$= O(n)$

$f(n) = 2n^2 - 2n + 10$

$= O(n^2)$

$\{ 0 \leq f(n) \leq C \cdot g(n) :$

$C =$ constant, $\forall\ n > n_0 \}$

$\Rightarrow \boxed{O(g(n))}$

$f(n) = 5n + 1$

$= O(n)$

$c \cdot g(n) = c \cdot n$

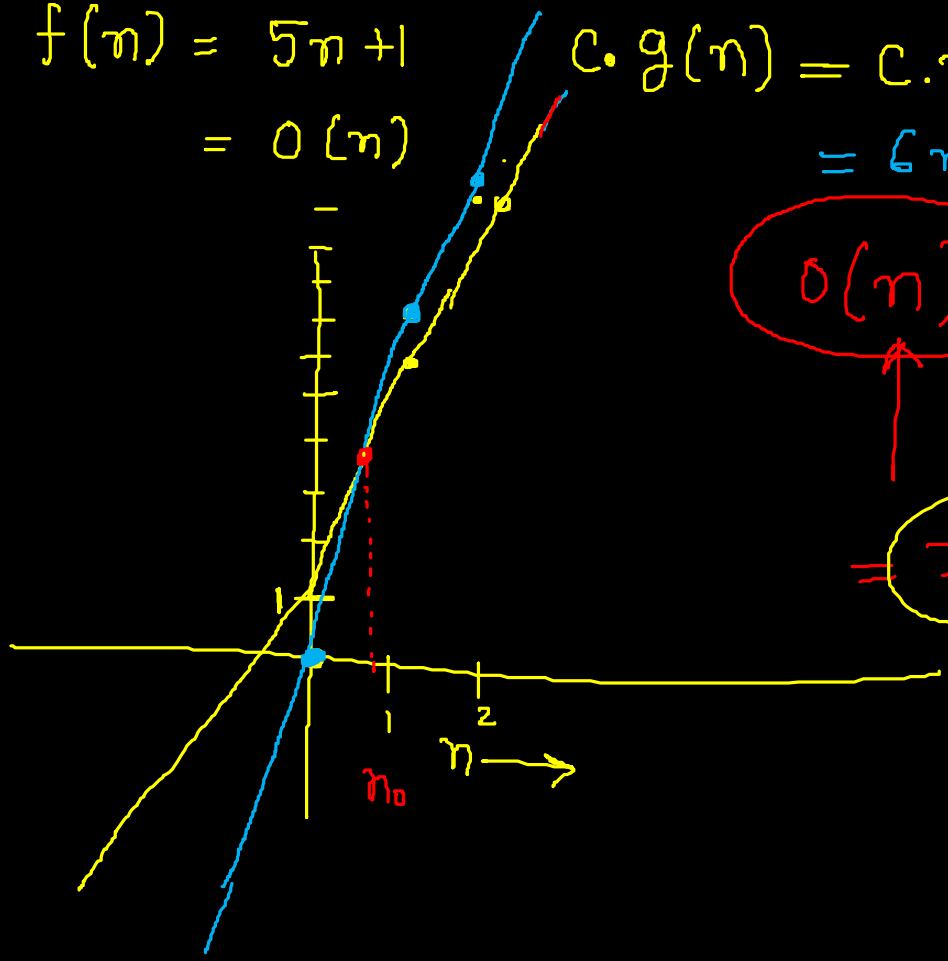
$= 6n$

$c = 6$

$n_0 = 1$

$O(n)$

$c = 6, \; n_0 \geq 1$



$= 7n$ n_0

$= 8n$ n_0

$= n^2$

$5n+1$

$O(n^2)$ $O(n^3)$

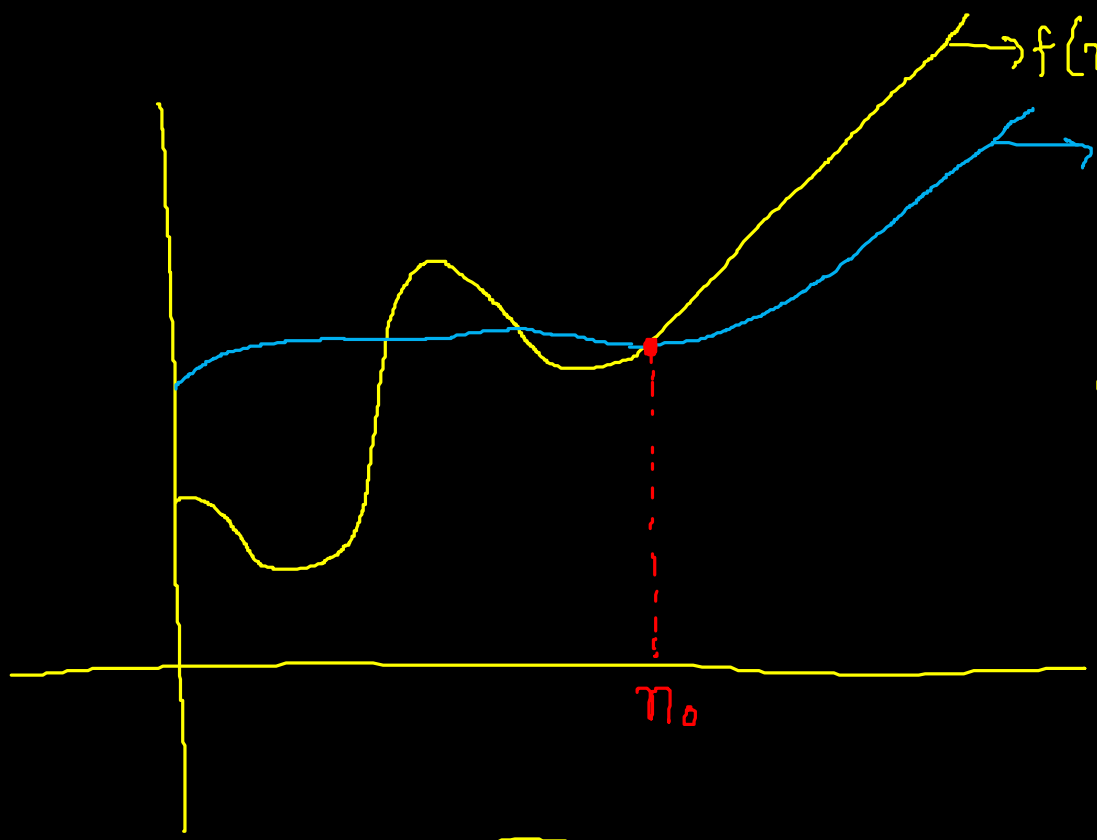
5 din

50 din

100%

45%

Omega Notation



$f(n) = 5n^2$

$= \Omega(n^2)$

$\neq \Omega(n^3)$?

$= \Omega(n)$ ✓

$\{ 0 \leq c \cdot g(n) \leq f(n) : c = \text{const}$

$\forall\ n \geq n_0 \}$

$\Omega(g(n))$

$C \leq 5$

$n_0 = 1, \quad C = 5$

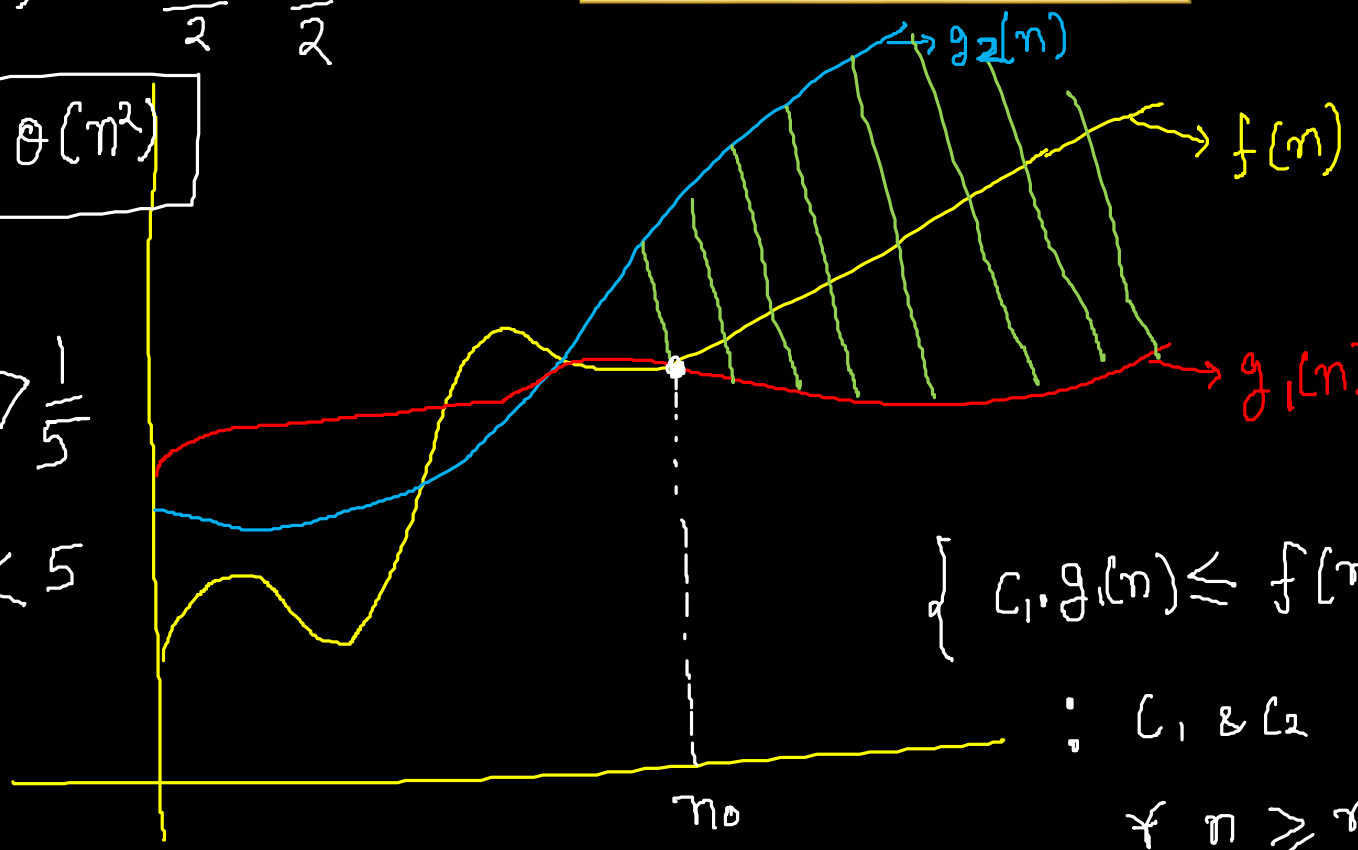
[$\Omega(n^2)$ so, you can also write it as $\Omega(n)$, $\Omega($

Theta Notation

$f(n) = \frac{n^2}{2} - \frac{n}{2}$

$\theta(n^2)$

$\theta(g(n))$



$\frac{1}{2} > \frac{1}{5}$

$2 < 5$

$\{ c_1 g_1(n) \leq f(n) \leq c_2 g_2(n)$

: c_1 & c_2 are constant

$\forall\ n \geq n_0 \}$

$c_1 g_1(n) = \frac{n^2}{5}$, $c_2 g_2(n) = n^2$,

$c_1 = \frac{1}{5}$, $c_2 = 1$

$n_0 = 2$

1) Constant Time :-

```
int main()
{
    int x, y, z;      ---> 3
    z = x + y;        ---> 2
}
```

5 sec

O(1)

```
int main()
{
    int a, b, c, d;       ---> 4
    a = b + c;            ---> 2
    c = a + d * 2;        ---> 3
    d = a - b + c / 3;    ---> 4
}
```

13

O(1)

2) Linear Time:-

$$f(n) = 6 + 2 + 2n + 3 + n$$

$$f(n) = 3n + 11 = O(n)$$

```
int main()
{
    int i, a, b, c, n = 10;
    a = b + c;
    for(i = 0; i < n; i++)
    {
        b++;
    }
}
```

`int i, a, b, c, n = 10;` → 6

`a = b + c;` → 2

Constant

`for(i = 0; i < n; i++)` → $1 + (n+1) + (n+1)$ → $2n+3$

`b++;` → n

n times → $O(n)$

```
int main()
{
    int a, b, c, d, e, i, j, n;
    c = a + b;
    d = e + c - a * b;
    for (i = 0; i < n/2; i++)
    {
        c++;
    }
    for (j = n; j > 0; j--)
    {
        a++;
    }
}
```

constant + $\frac{n}{2} \approx n$ + n

$$f(n) = n + \frac{n}{2} + c$$

$$f(n) = 2n + c$$

$$= O(n)$$ ✓

3) Quadratic Time:-

$$f(n) = 2n^2 + n + C$$
$$= O(n^2)$$

```
int main()
{
    int a, b, c, d, i, j, n;          ⟶ Constant
    for( i = 0; i<n; i++)             ⟶ (n+1) ≈ n
    {
        for (j = 0 ; j<n ; j++)       ⟶ (n+1)^2 ≈ n^2
        {
            a++;                      ⟶ ≈ n^2
        }
    }
}
```

+ + +

Best / Avg

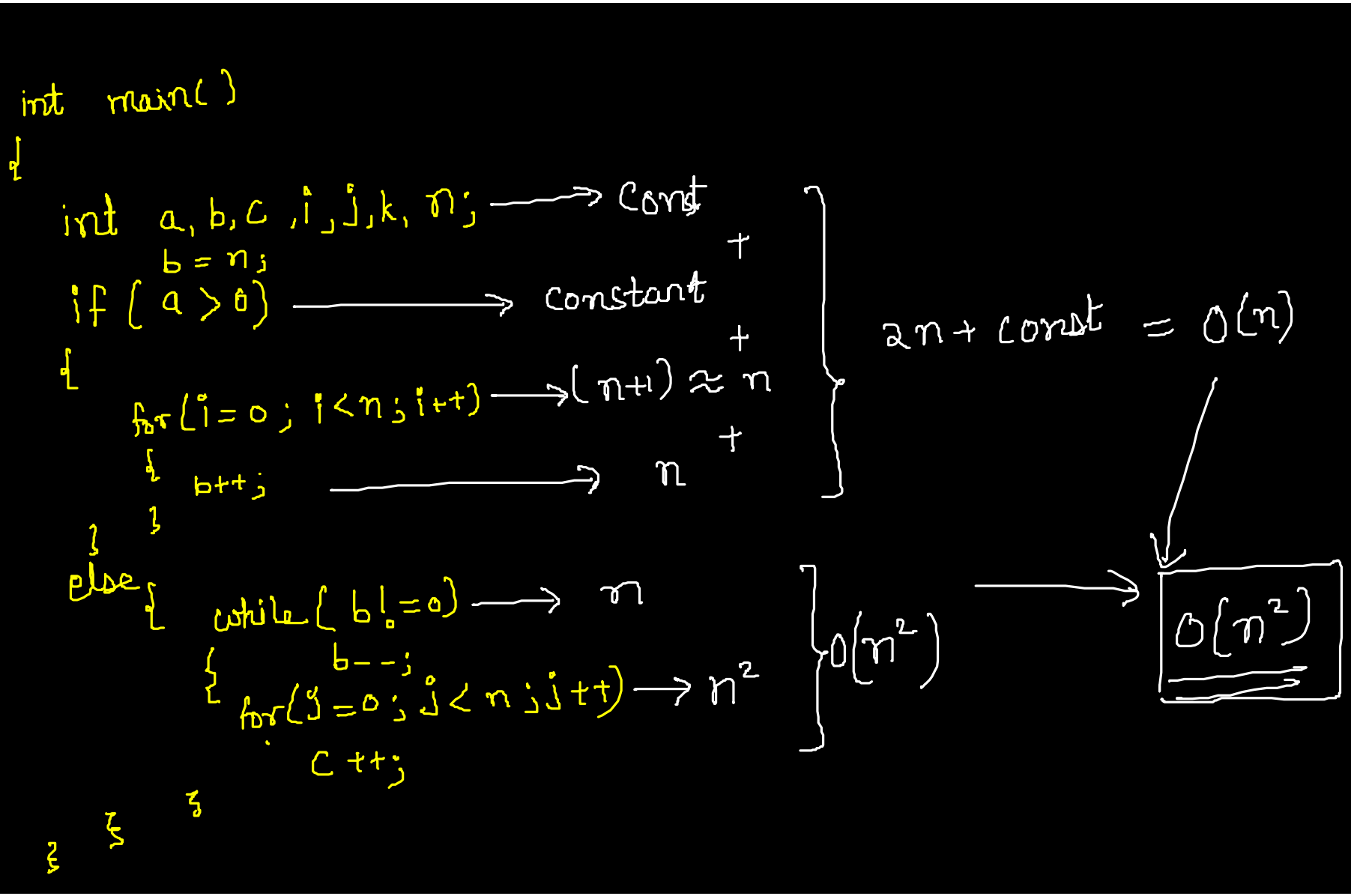
i = 0 j = (0 − n) = (n+1)
i = 1 j = (0 − n) = (n+1)

4) Cubic Time :-

$$f(n) = an^3 + bn^2 + cn + d$$
$$= O(n^3)$$

```
int main()
{
    int a, b, c, i, j, k, n;        ---> constant

    for(i=0; i<n; i++)              ---> (n+1) ≃ n
    {
        for(j=0; j<n; j++)          ---> (n+1)^2 ≃ n^2
        {
            for(k=0; k<n; k++)      ---> (n+1)^3 ≃ n^3
            {
                a++;
            }
        }
    }
```


int main( )
{
int a, b, c, i, j, k, n; ⟶ Const
b = n;
if (a > 0) ⟶ constant
{
for(i = 0; i < n; i++) ⟶ (n+1) ≈ n
{
b++; ⟶ n
}
}
else
{
while (b != 0) ⟶ n
b--;
{
for(j = 0; j < n; j++) ⟶ n²
c++;
}
}
}
+
+
+
2n + const = O(n)
O(n²)
O(n²)