# 05_Recurrence Relation

## Methods:

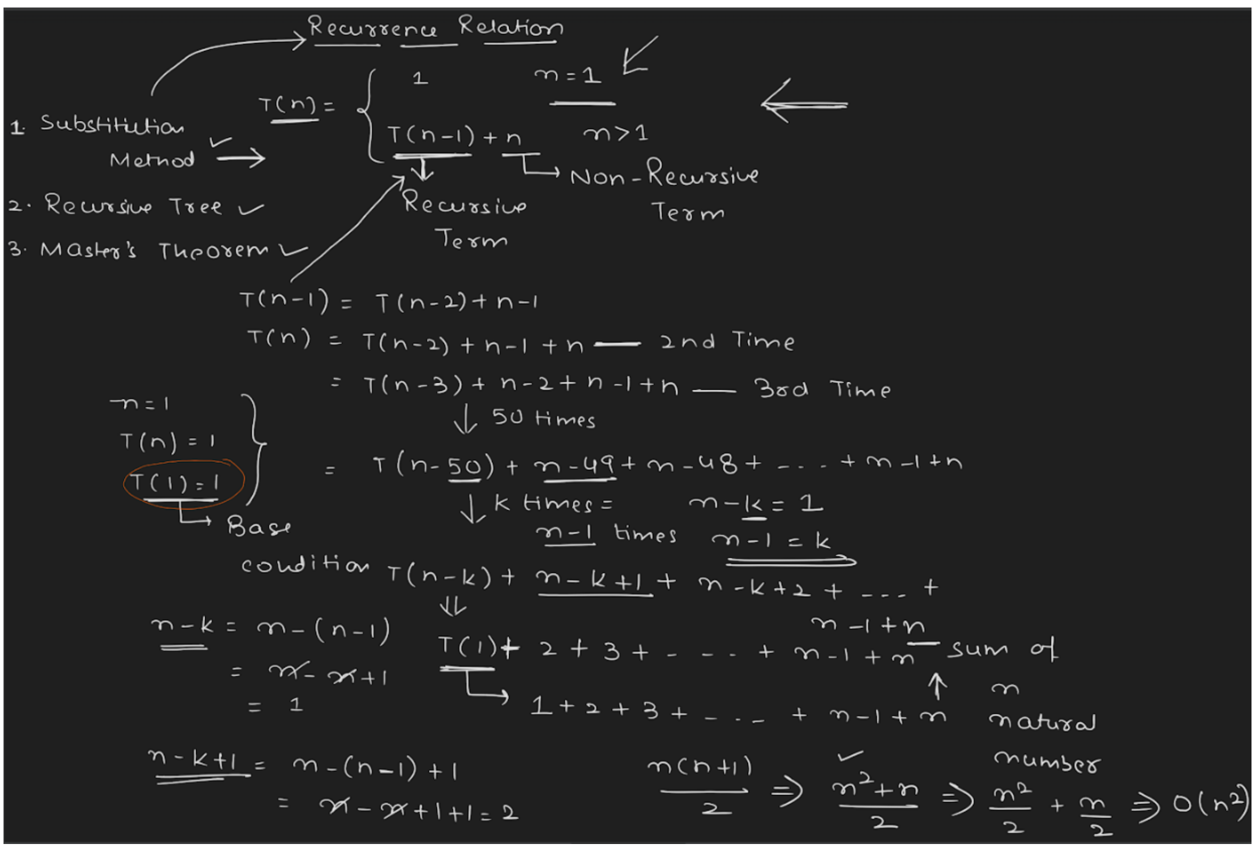
1) Substitution Method

2) Recursive Tree

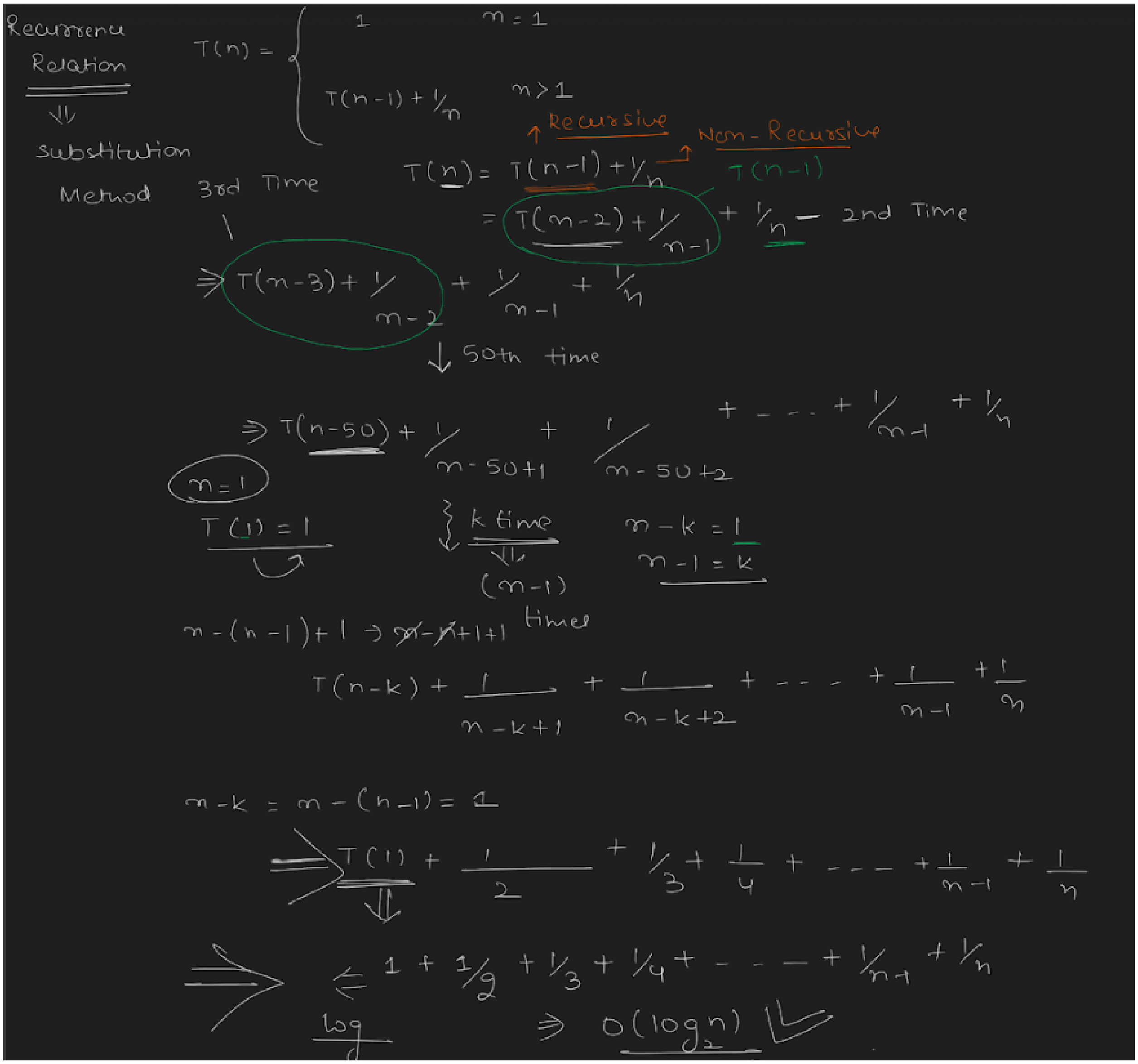
3) Masters Method

1. Substitution Method:

eg 1)




eg 2)



Recurrence Relation

⇓

Substitution Method

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + \frac{1}{n} & n > 1 \end{cases}$$

Recursive      Non-Recursive

$T(n) = T(n-1) + \frac{1}{n}$      $T(n-1)$

$= T(n-2) + \frac{1}{n-1} + \frac{1}{n}$ — 2nd Time

3rd Time

$\Rightarrow T(n-3) + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n}$

↓ 50th time

$\Rightarrow T(n-50) + \frac{1}{n-50+1} + \frac{1}{n-50+2} + \cdots + \frac{1}{n-1} + \frac{1}{n}$

$n = 1$

$T(1) = 1$

k time ⇓ $(n-1)$ times

$n - k = 1$

$n - 1 = k$

$n-(n-1)+1 \Rightarrow \cancel{n}-\cancel{n}+1+1$

$T(n-k) + \frac{1}{n-k+1} + \frac{1}{n-k+2} + \cdots + \frac{1}{n-1} + \frac{1}{n}$

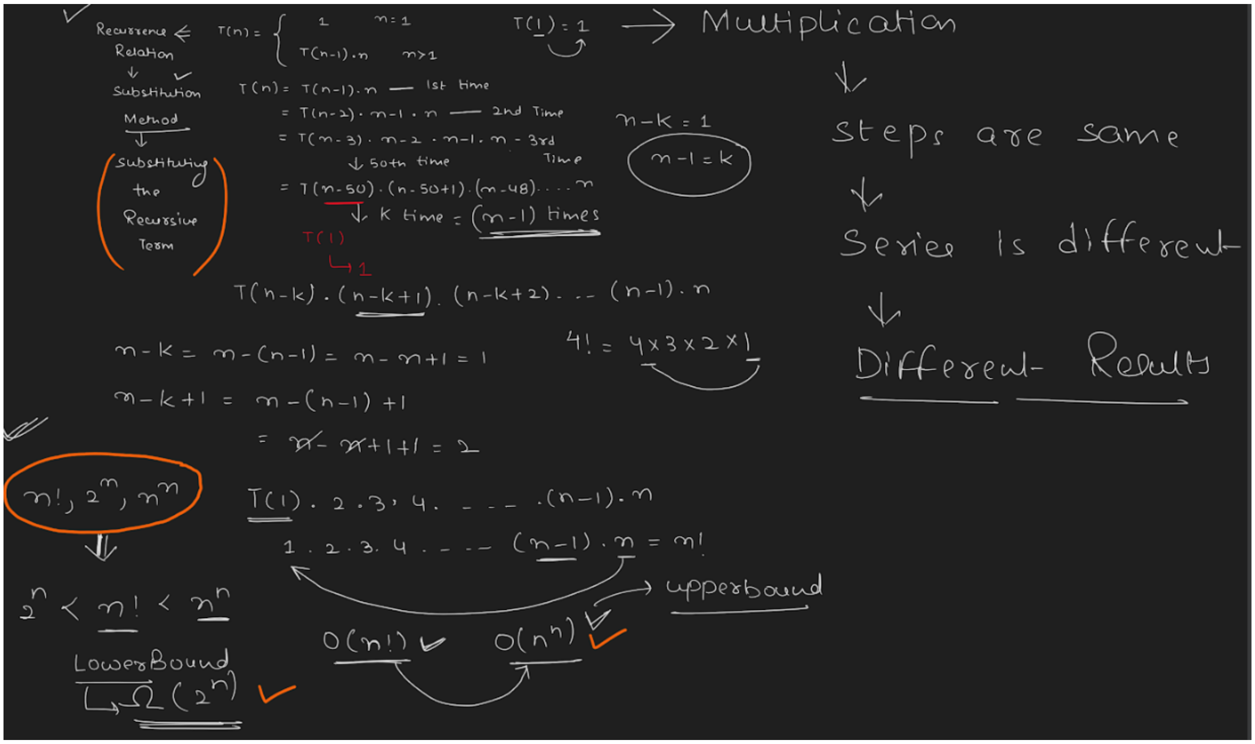
$n - k = n - (n-1) = 1$

$\Rightarrow T(1) + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n-1} + \frac{1}{n}$

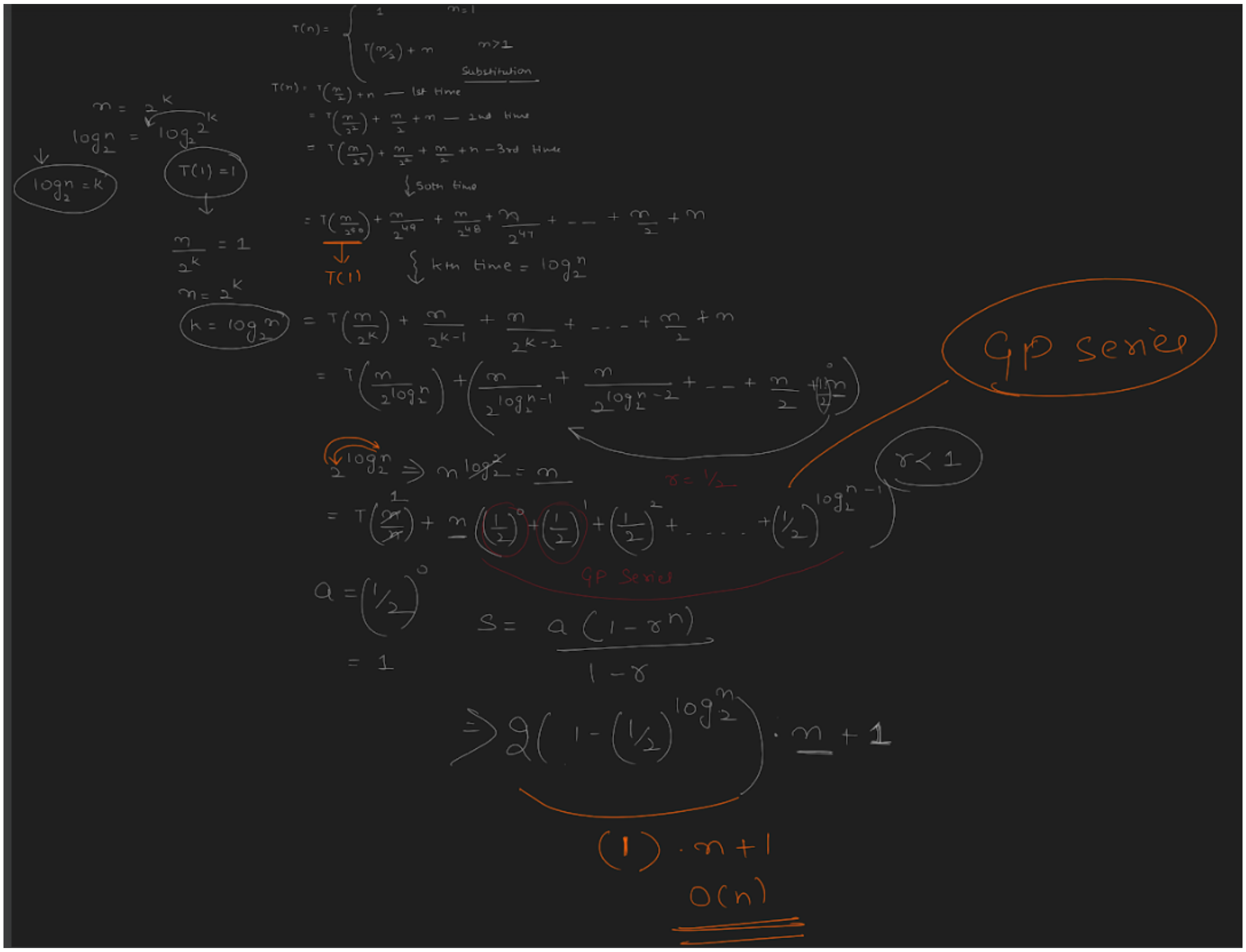
$\Rightarrow \Leftarrow 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n-1} + \frac{1}{n}$

log      $\Rightarrow O(\log_2 n)$ ✓

eg3)

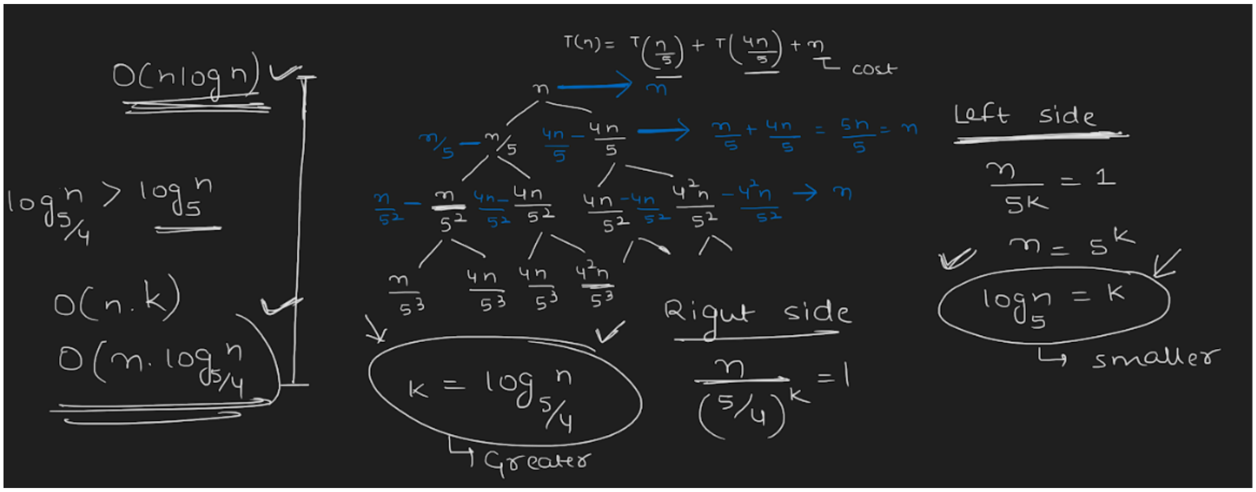

Recurrence Relation
Substitution Method
Substituting the Recursive Term
T(n) = 1 n=1
T(n-1).n n>1
T(1) = 1
Multiplication
T(n) = T(n-1).n — 1st time
= T(n-2).n-1.n — 2nd Time
= T(n-3).n-2.n-1.n - 3rd Time
50th time
= T(n-50).(n-50+1).(n-48)....n
k time = (n-1) times
T(1)
1
n-k = 1
n-1 = k
steps are same
Series is different
Different Results
T(n-k).(n-k+1).(n-k+2)...(n-1).n
n-k = n-(n-1) = n-n+1 = 1
4! = 4x3x2x1
n-k+1 = n-(n-1)+1
= n-n+1+1 = 2
n!, 2^n, n^n
T(1).2.3.4.....(n-1).n
1.2.3.4....(n-1).n = n!
upperbound
2^n < n! < n^n
O(n!)
O(n^n)
LowerBound
Ω(2^n)

eg4)

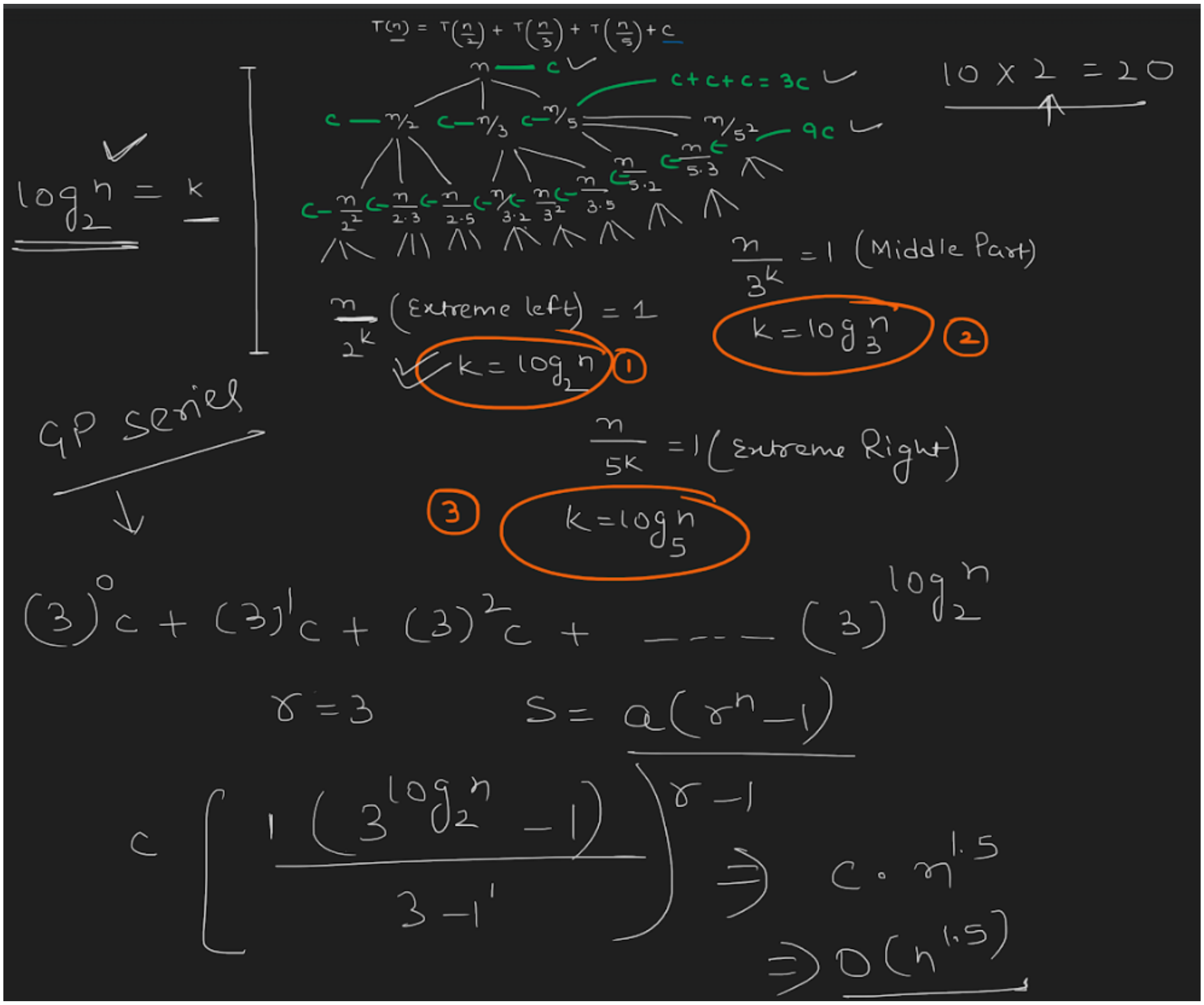



2) Recursive Tree

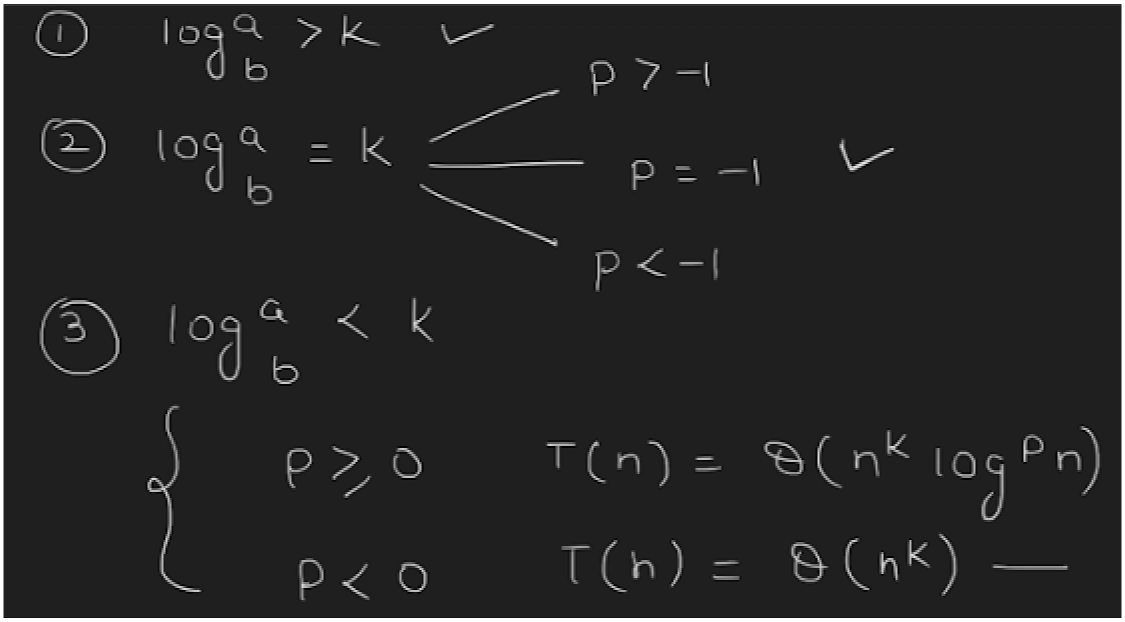
eg1)


T(n) = T(n/5) + T(4n/5) + n
cost
Left side
n/5^k = 1
n = 5^k
log_5 n = k
smaller
Right side
n/(5/4)^k = 1
k = log_{5/4} n
Greater
O(n log n)
log_{5/4} n > log_5 n
O(n.k)
O(n. log_{5/4} n)

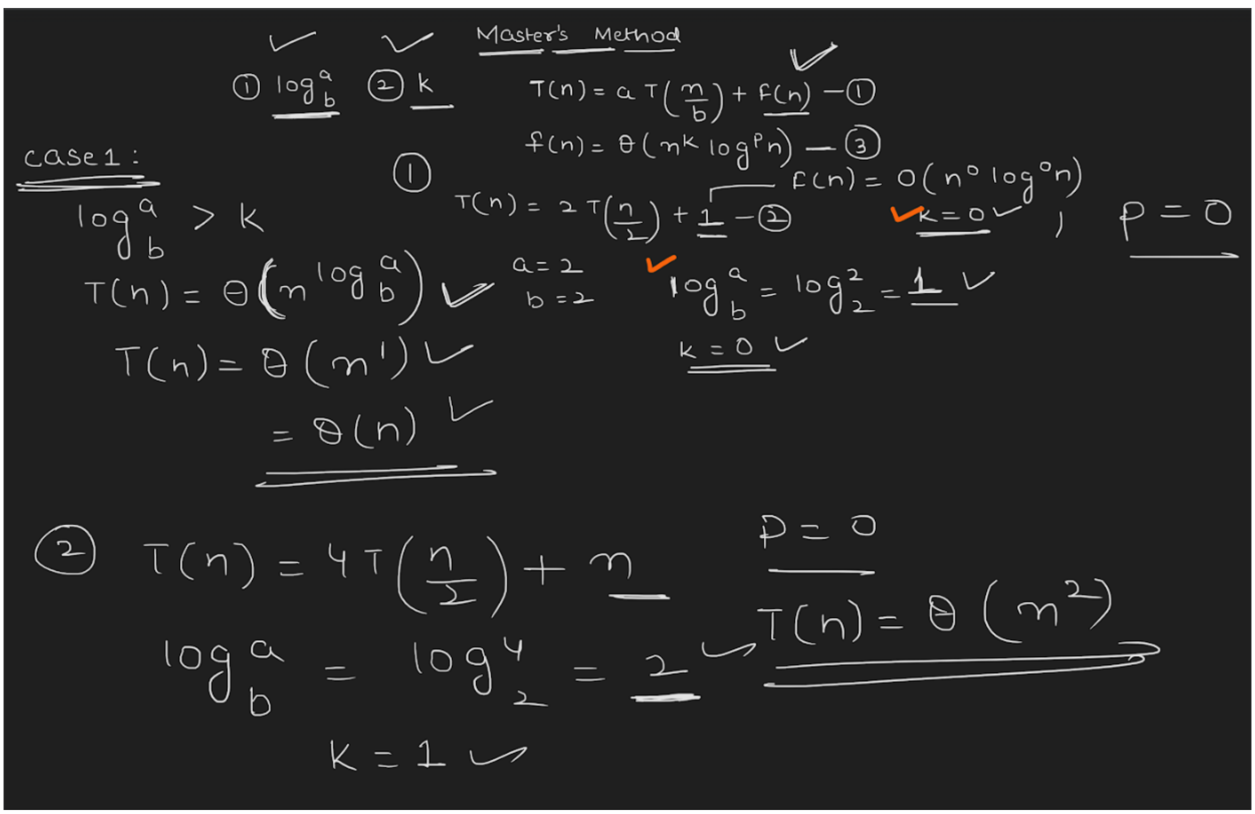
eg2)



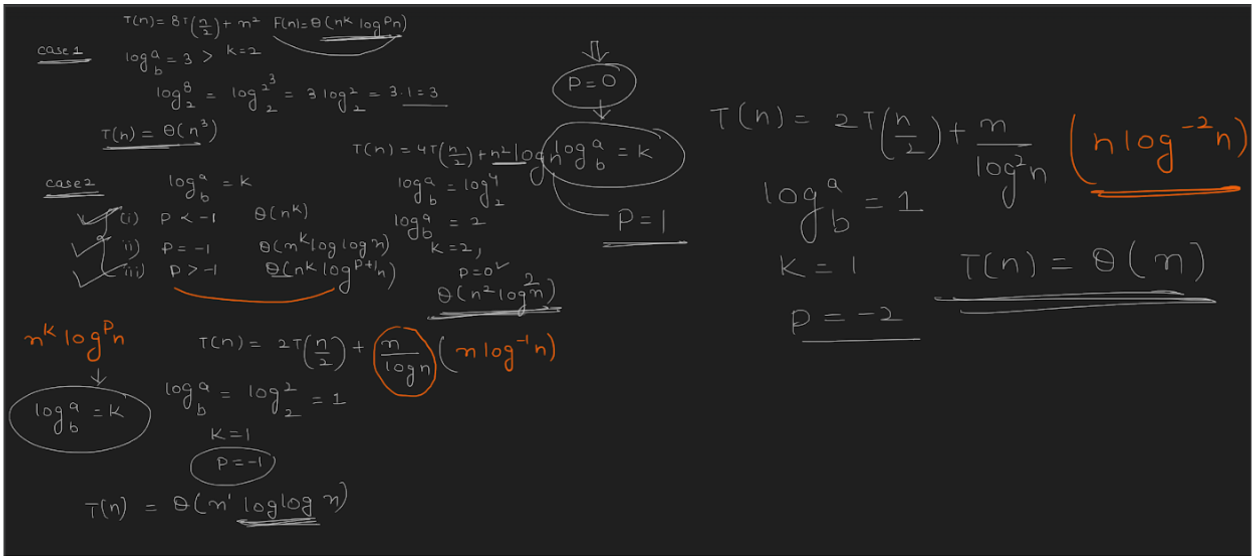

3) Masters Method


① log_b a > k
② log_b a = k
p > -1
p = -1
p < -1
③ log_b a < k
p ≥ 0 T(n) = θ(n^k log^p n)
p < 0 T(n) = θ(n^k)

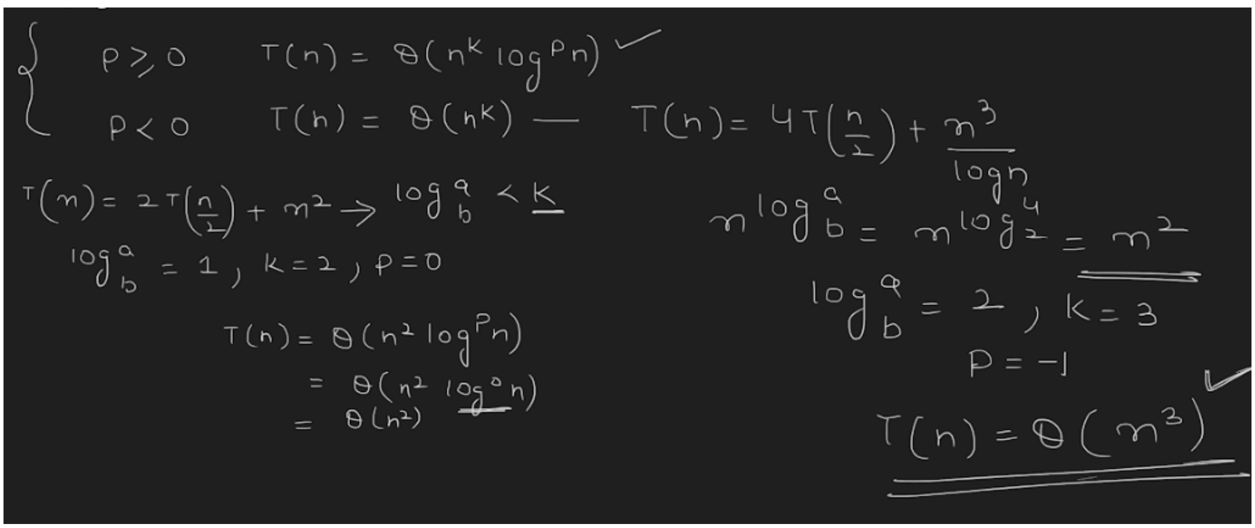
case1)


Master's Method
① log_b a ② k
T(n) = a T(n/b) + f(n) — ①
f(n) = θ(n^k log^p n) — ③
case1 :
log_b a > k
T(n) = θ(n^(log_b a))
T(n) = θ(n^1)
= θ(n)
①
T(n) = 2T(n/2) + 1 — ②
f(n) = O(n^0 log^0 n)
k = 0 , p = 0
a = 2
b = 2
log_b a = log_2 2 = 1
k = 0
② T(n) = 4T(n/2) + n
log_b a = log_2 4 = 2
k = 1
p = 0
T(n) = θ(n^2)

Case 2)


T(n) = 8T(n/2) + n^2   F(n) = θ(n^k log^p n)
case 1
log_b a = 3 > k = 2
log_2 8 = log_2 2^3 = 3 log_2 2 = 3·1 = 3
T(n) = θ(n^3)
P = 0
T(n) = 4T(n/2) + n^2 log n
log_b a = k
P = 1
case 2
log_b a = k
(i) P < -1   θ(n^k)
ii) P = -1   θ(n^k log log n)
iii) P > -1   θ(n^k log^(P+1) n)
log_b a = log_2 4
log_b a = 2
k = 2,
P = 0
θ(n^2 log^2 n)
T(n) = 2T(n/2) + n/log^2 n   (n log^-2 n)
log_b a = 1
K = 1
P = -2
T(n) = θ(n)
n^k log^p n
log_b a = k
T(n) = 2T(n/2) + n/log n   (n log^-1 n)
log_b a = log_2 2 = 1
K = 1
P = -1
T(n) = θ(n^1 loglog n)

Case 3)


P ≥ 0   T(n) = θ(n^k log^P n)
P < 0   T(n) = θ(n^k)
T(n) = 2T(n/2) + n^2 → log_b a < K
log_b a = 1, K = 2, P = 0
T(n) = θ(n^2 log^P n)
= θ(n^2 log^0 n)
= θ(n^2)
T(n) = 4T(n/2) + n^3/log n
n^(log_b a) = n^(log_2 4) = n^2
log_b a = 2, K = 3
P = -1
T(n) = θ(n^3)