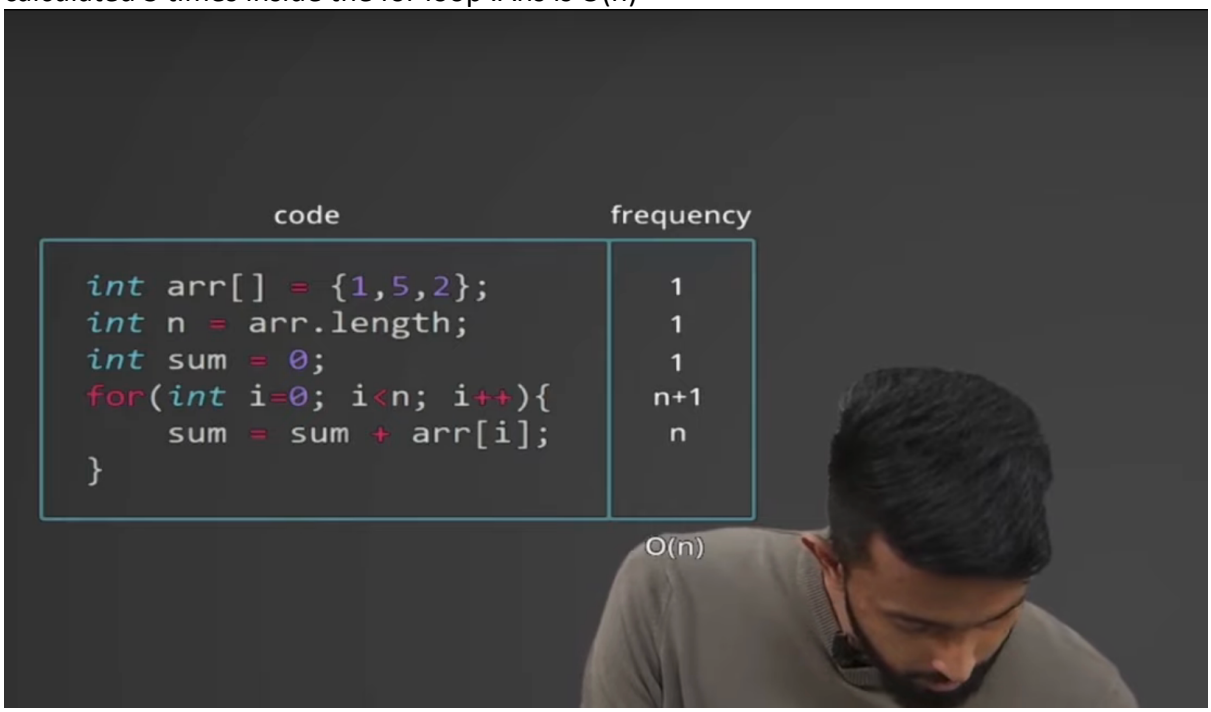
Learned from tap academy
https://www.youtube.com/watch?v=IyfMNHoOwJ0&list=PLU83Ru7iGtAvP1rqt65MCDSBPFCzUKGXG&index=5

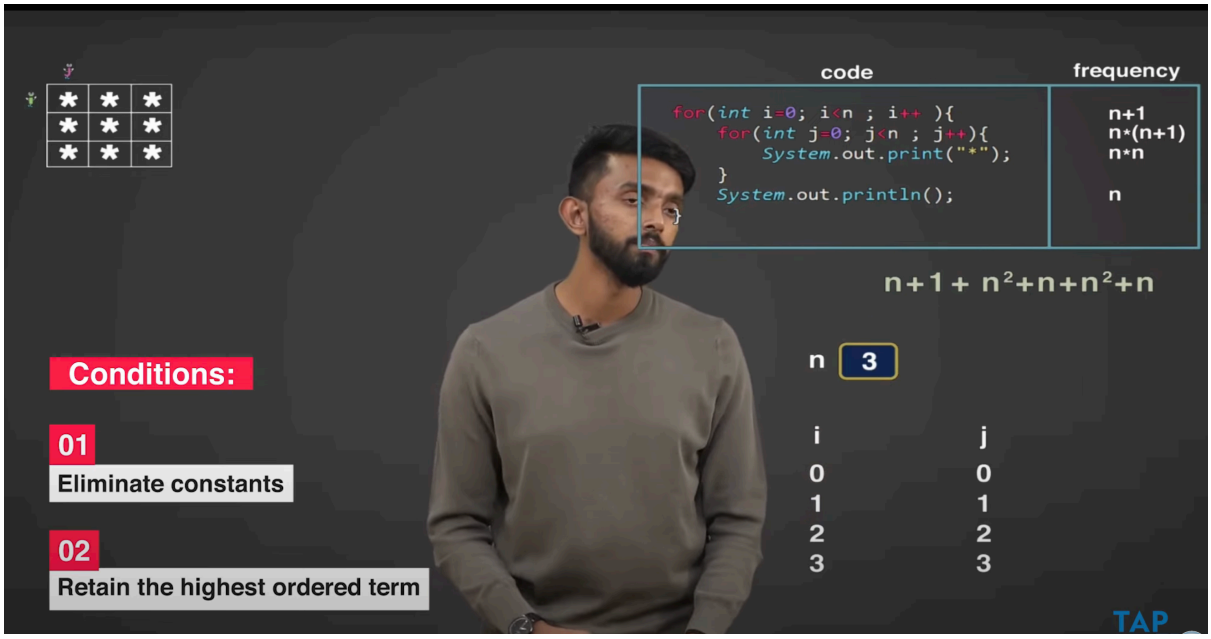
We are finding time complexity using a method called “FREQUENCY COUNT METHOD”

This is based on condition that
1)Eliminate constants
2)Retain highest order term

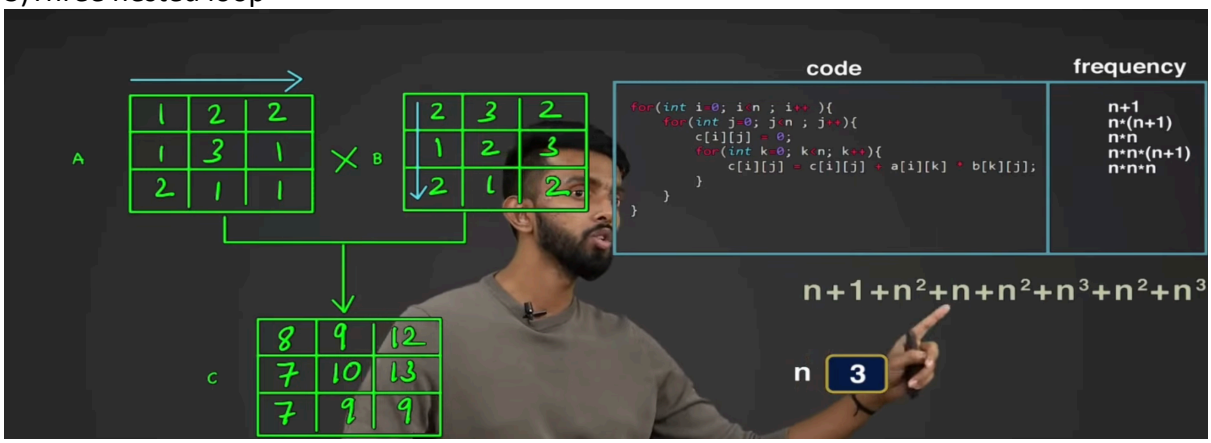
1)
Here for statement frequency is n+1 because on n+1 only the termination condition is reached. But the inner statements inside for loop will run only for n time.
Eg: n=3. So I will reach till 0,1,2,3[ie 4 times] for the termination condition,but sum will be calculated 3 times inside the for loop .Ans is O(n)



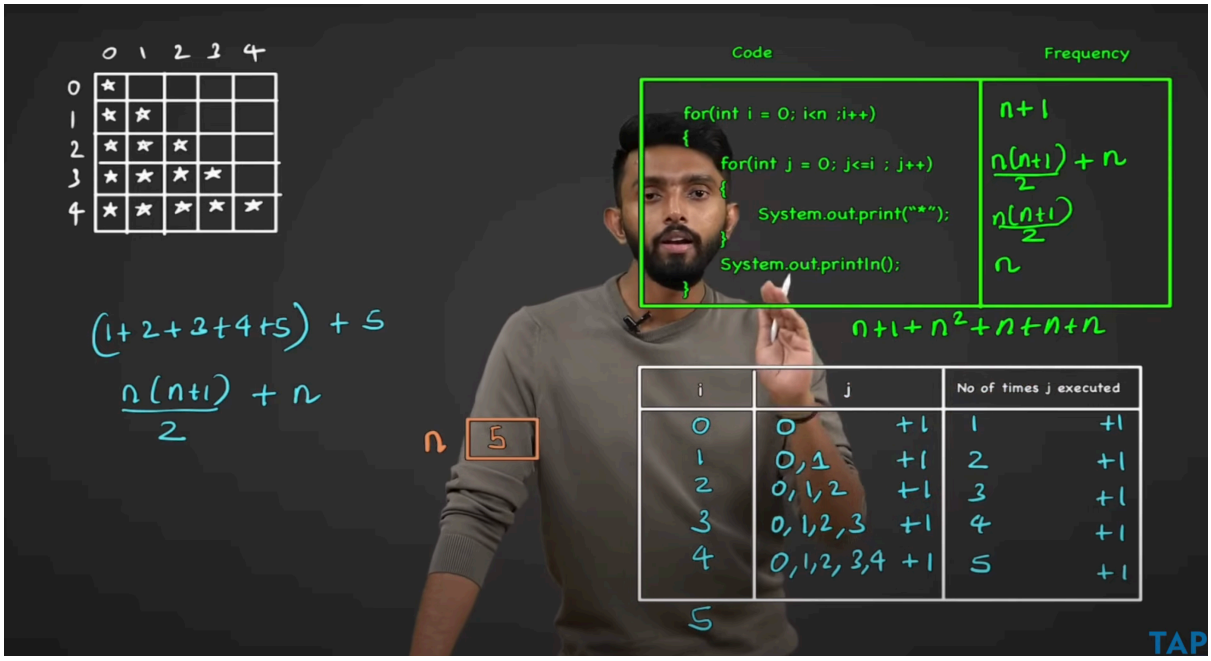
2) Next loop example,here inner for loop is n(n+1) is because due to outer for loop it will run n times and the inner for loop itself will execute for n+1 times ,hence n*(n+1).Answer is O(n^2)



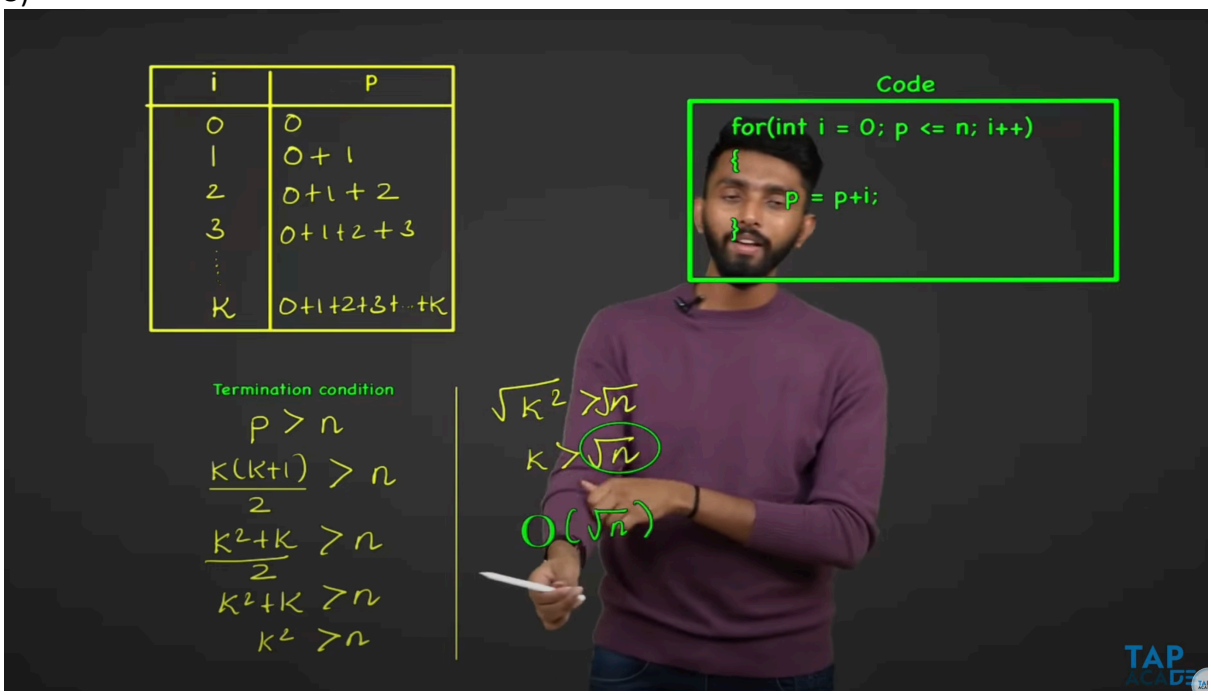
3)Three nested loop



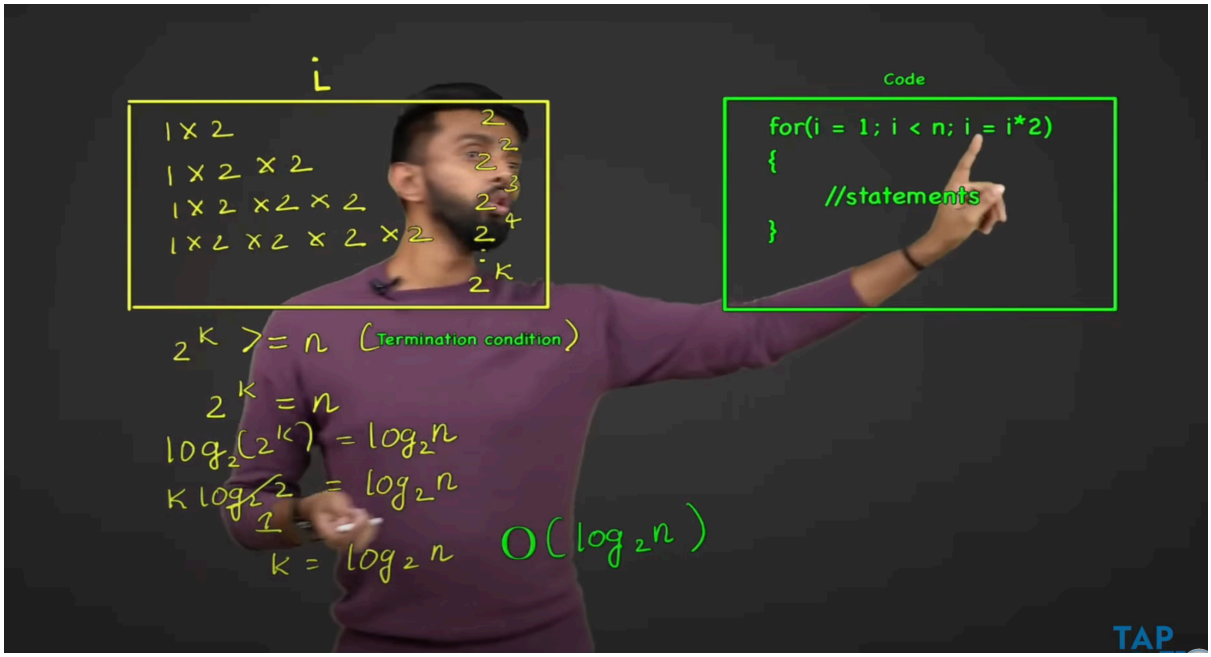
4)Ans O(n). +1 in each row is to denote the termination condition. Summation of n natural number is written as n(n+1)/2


Code
Frequency
for(int i = 0; i<n ;i++)
{
for(int j = 0; j<=i ; j++)
{
System.out.print("*");
}
System.out.println();
}
n+1
n(n+1)/2 + n
n(n+1)/2
n
n+1+n²+n+n+n
(1+2+3+4+5) + 5
n(n+1)/2 + n
n 5
i
j
No of times j executed
0 0 +1 1 +1
1 0,1 +1 2 +1
2 0,1,2 +1 3 +1
3 0,1,2,3 +1 4 +1
4 0,1,2,3,4 +1 5 +1
5
TAP

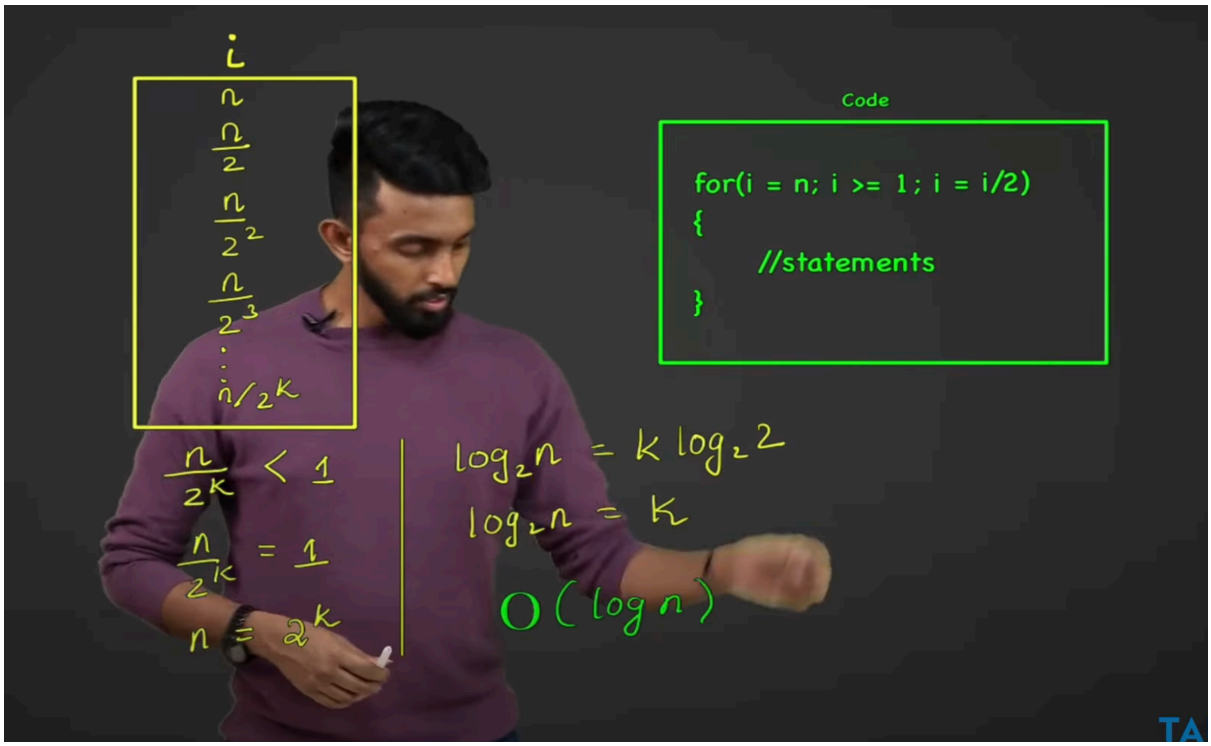
5)


i
P
0 0
1 0+1
2 0+1+2
3 0+1+2+3
K 0+1+2+3+...+K
Code
for(int i = 0; p <= n; i++)
{
p = p+i;
}
Termination condition
p > n
K(K+1)/2 > n
K²+K/2 > n
K²+K > n
K² > n
√K² > √n
K > √n
O(√n)
TAP

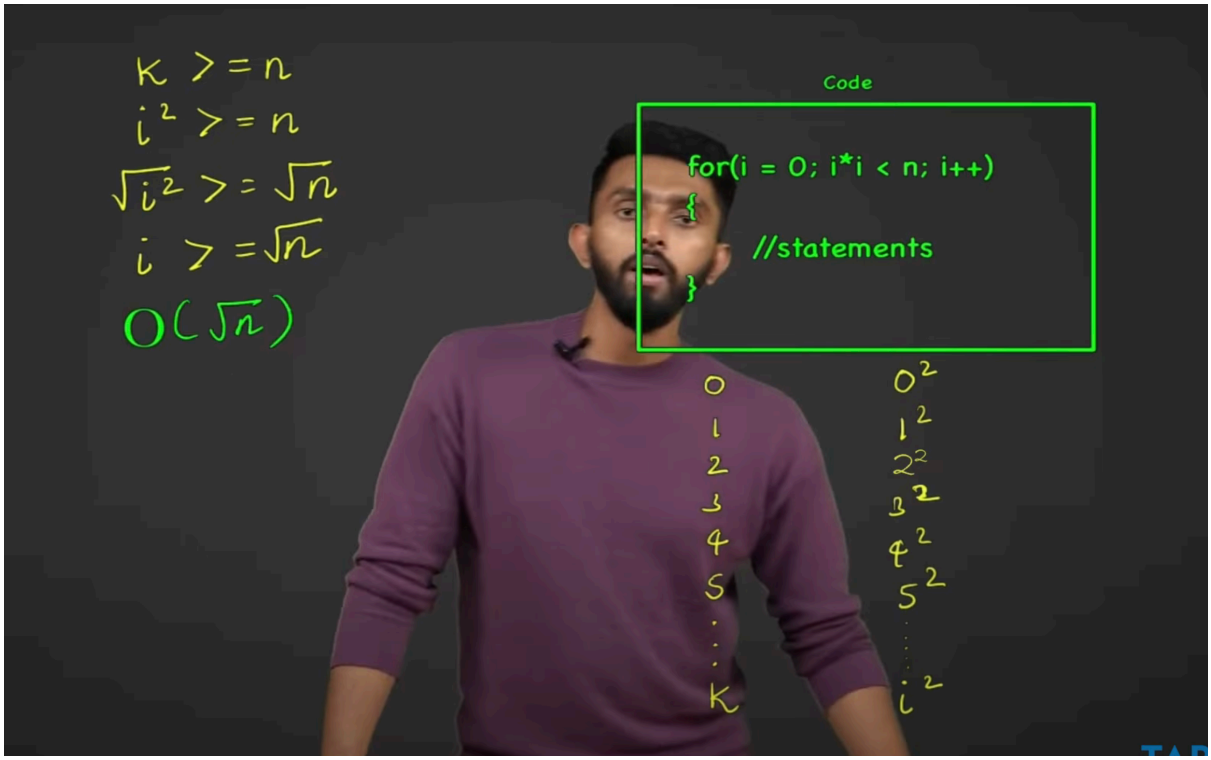
6)Ans O(logn)


i
1 x 2
1 x 2 x 2
1 x 2 x 2 x 2
1 x 2 x 2 x 2 x 2
2
2^2
2^3
2^4
2^k
Code
for(i = 1; i < n; i = i*2)
{
//statements
}
2^k >= n (Termination condition)
2^k = n
log2(2^k) = log2 n
k log2 2 = log2 n
1
k = log2 n
O(log2 n)

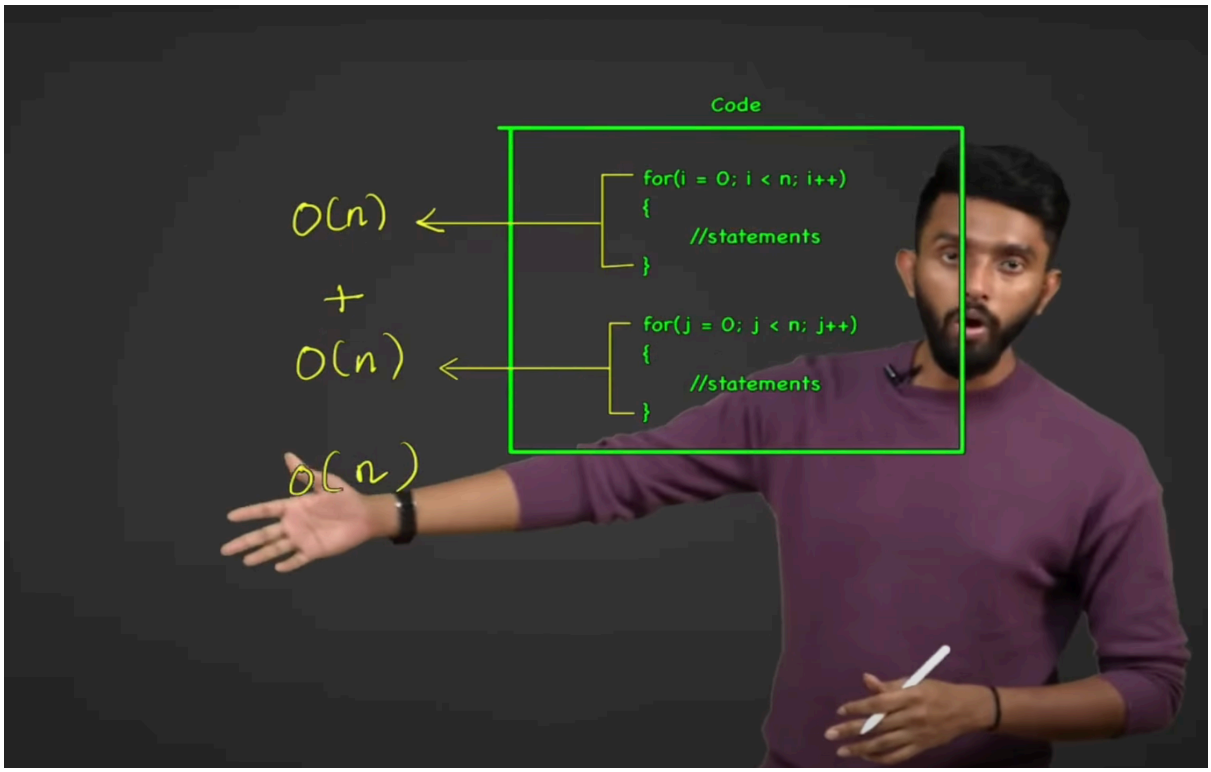
7)


i
n
n/2
n/2^2
n/2^3
n/2^k
Code
for(i = n; i >= 1; i = i/2)
{
//statements
}
n/2^k < 1
n/2^k = 1
n = 2^k
log2 n = k log2 2
log2 n = k
O(log n)

8)

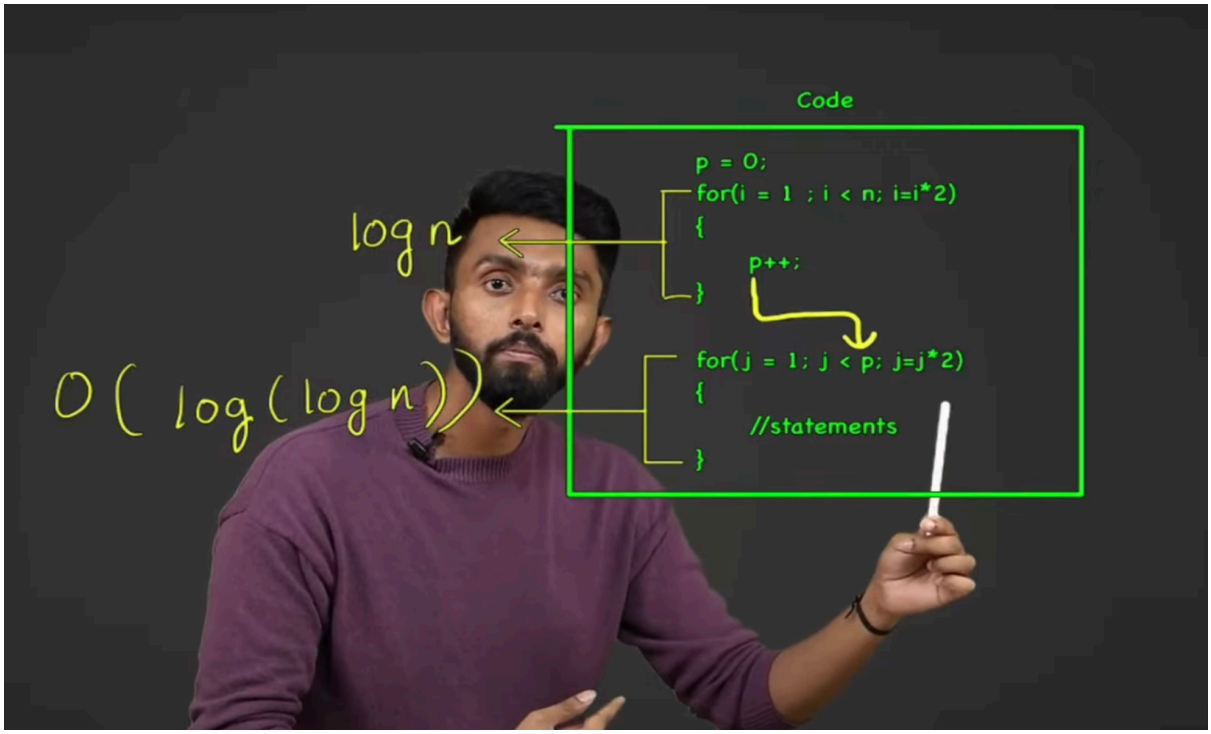

K >= n
i^2 >= n
√i^2 >= √n
i >= √n
O(√n)
Code
for(i = 0; i*i < n; i++)
{
//statements
}
0
1
2
3
4
5
K
0^2
1^2
2^2
3^2
4^2
5^2
i^2

9)ANS :O(n)

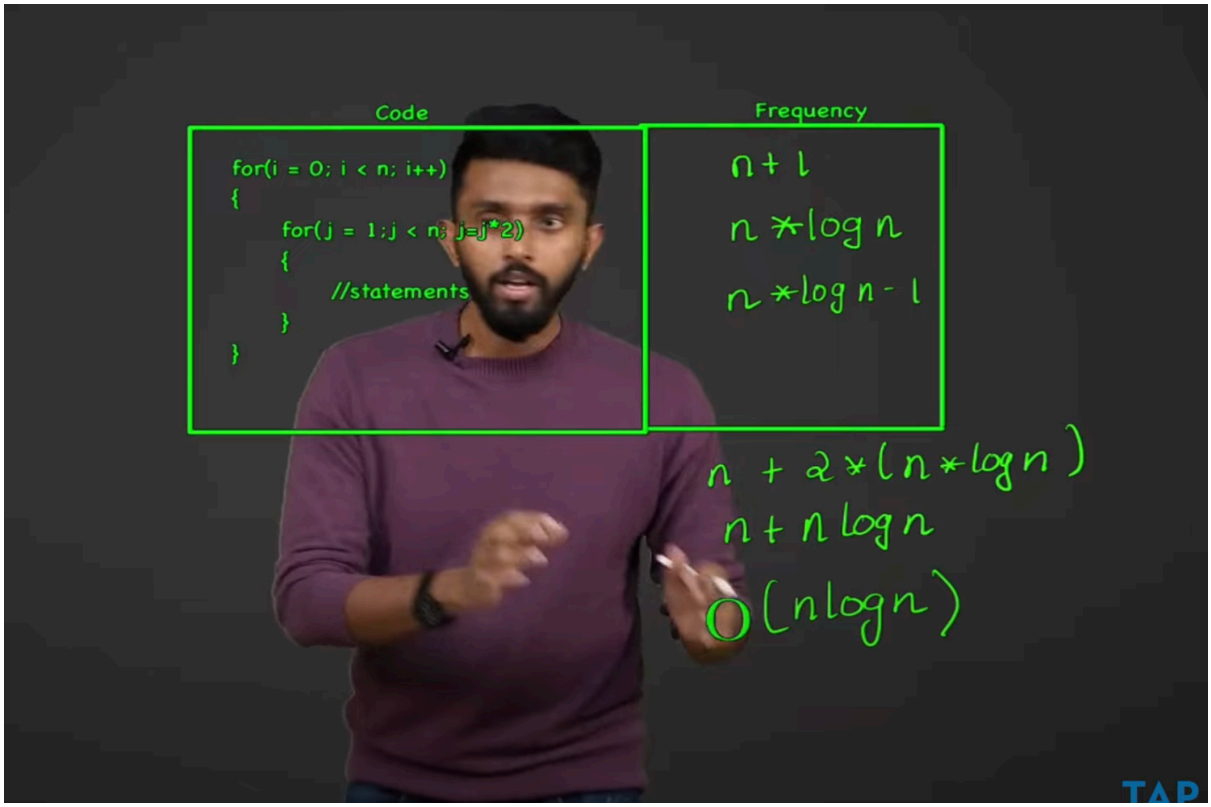

Code
for(i = 0; i < n; i++)
{
//statements
}
for(j = 0; j < n; j++)
{
//statements
}
O(n)
+
O(n)
O(n)

10) Ans O(log(logn)).First for loop Log n. and second for loop can be represented as log p. Since p itself take log n we are coming to conclusion of log(log n)

P=log n
Log p=log(log n)



11)


Code
Frequency
for(i = 0; i < n; i++)
{
for(j = 1;j < n; j=j*2)
{
//statements
}
}
n + 1
n *log n
n *log n - 1
n + 2*(n*log n)
n + n log n
O(n log n)