Recurrence Relation → to get the time complexity

Substitution Method

(1)

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

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

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

$$\Rightarrow T(n-1) = T(n-2) + (n-1)$$

$$T(n) = T(n-2) + (n-1) + n \quad \text{---} (2)$$

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

$n - k = 1$ ↓ k times = (n-1) times

$n - 1 = k$

$$T(n) = T(n-k) + (n-k+1) + (n-k+2) + \cdots$$

$$\cdots + (n-2) + (n-1) + n$$

$$= T\left(n-(n-1)\right) + \left(n-(n-1)+1\right) + \left(n-(n-1)+2\right) +$$

$$\cdots + (n-2) + (n-1) + n$$

$$= \cancel{T(1)} + 2 + 3 + 4 + \cdots + (n-2) + (n-1) + n$$

$$= 1 + 2 + 3 + 4 + \cdots + (n-2) + (n-1) + n$$

Sum of n natural numbers

$$= \frac{n(n+1)}{2} \Rightarrow \frac{n^2+n}{2}$$

$$\Rightarrow O(n^2)$$

↳ Quadratic Time Complexity

②

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

factorial → $O(n)$

$$T(n) = T(n-1) + c \quad \text{— ①}$$

$$T(n) = T(n-2) + c + c$$

$$T(n) = T(n-2) + 2c \quad \text{— ②}$$

$$T(n) = T(n-3) + 3c \quad \text{— ③}$$

$n - K = 1$

$K = n - 1$

K times = $(n-1)$ time

$$T(n) = T(n-K) + K \cdot c$$

$T(1) = 1$

$$= T(n-(n-1)) + (n-1) \cdot c$$

$$= T(\overset{1}{1}) + (n-1) \cdot c$$

$$= 1 + (n-1) \cdot c$$

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

③

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

Log n

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

$$T(n) = T(n-2) + \frac{1}{n-1} + \frac{1}{n} \quad \text{—— ②}$$

③

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

$n - k = 1$

$k = n - 1$

↓ k times

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

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

$$= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \_\_ + \frac{1}{n-1} + \frac{1}{n}$$

$\Rightarrow$ $\log n$ $\frown$ $O(\log n)$

↓

Logarithmic time complexity

Sum of digits

num = 1234

output = 1+2+3+4

= 10

sumofDigits(1234)

1234%10 + sumofDigits(123)

10 = 4 + sumofDigits(123)

6

3 + sumofDigits(12)

3

= 6

2 + sumofDigits(1)

2+1 = 3

Space complexity

Stack space = O(d)

↳ # digits

```
T(d)
sum(n):
    if n < 10
        return n
    else return n%10 + sum(n//10)
```

c

T(d-1)

Substitution method

O(d) / O(n)

$T(d) = T(d-1) + c$

$T(n) = T(n-1) + c$

Power of a given number

powerfind(a,n) { a = 2, n = 16

$2^{16} = 65536$

$2^{16} = 2 * 2^{15}$ ← a * powerfind(a, n-1)

① Base case condition

→ n == 0:

↳ return 1

powerfind(a,n)

{ ② Recursive function call

else: return a * powerfind(a, n-1)

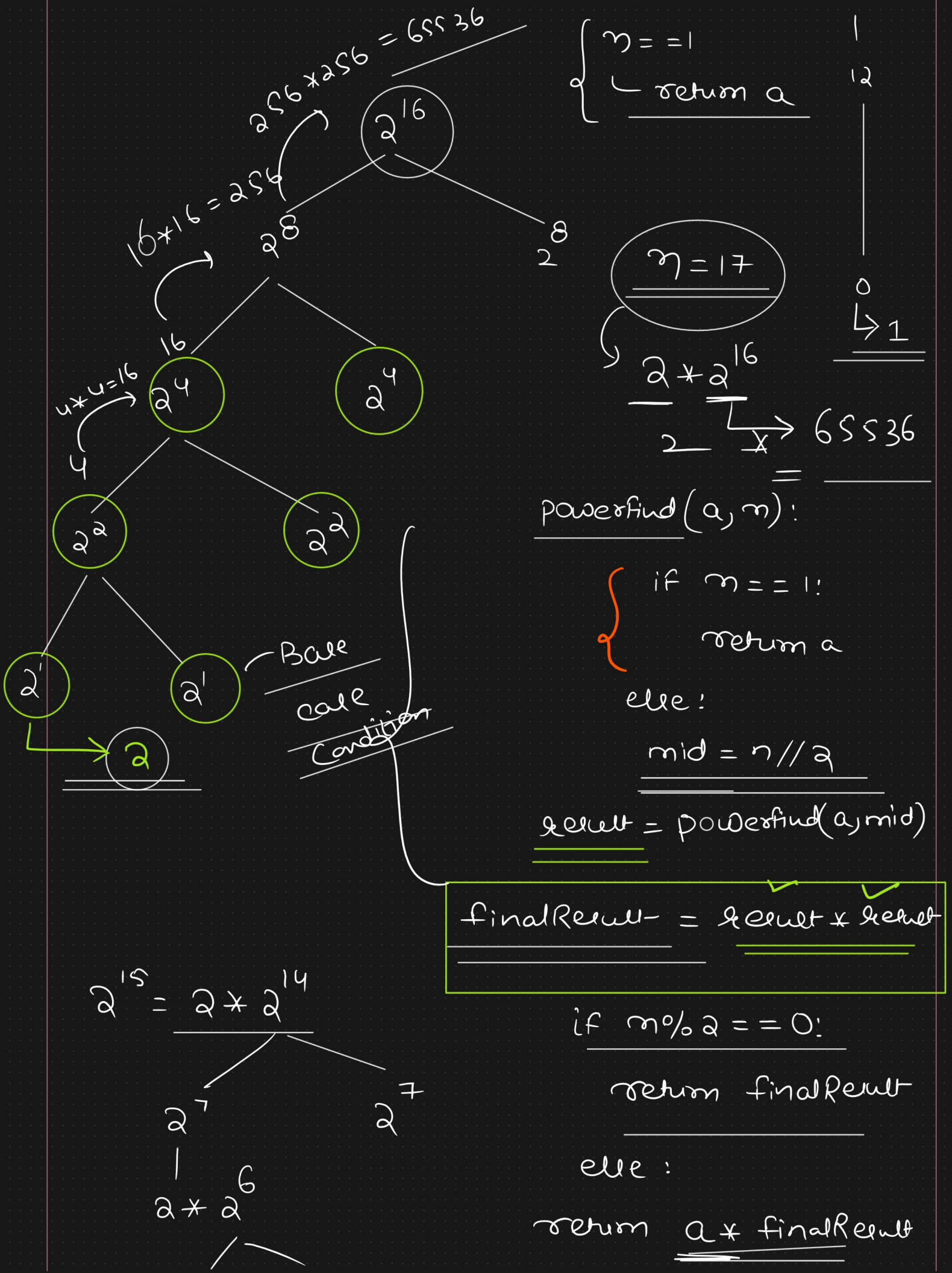
powerfind(2, 10000)

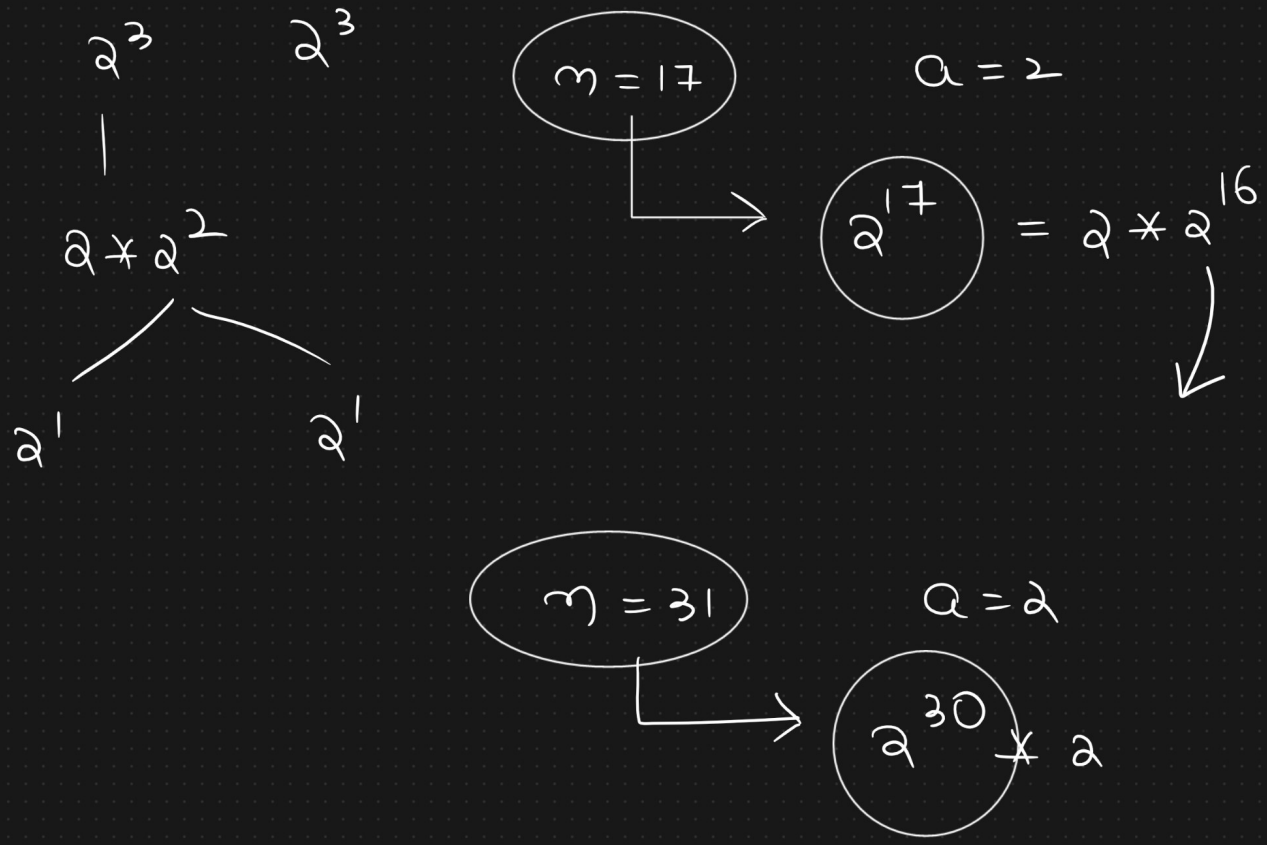
↳ time complexity = O(n)

↳ 2 * powerfind(2, 15)

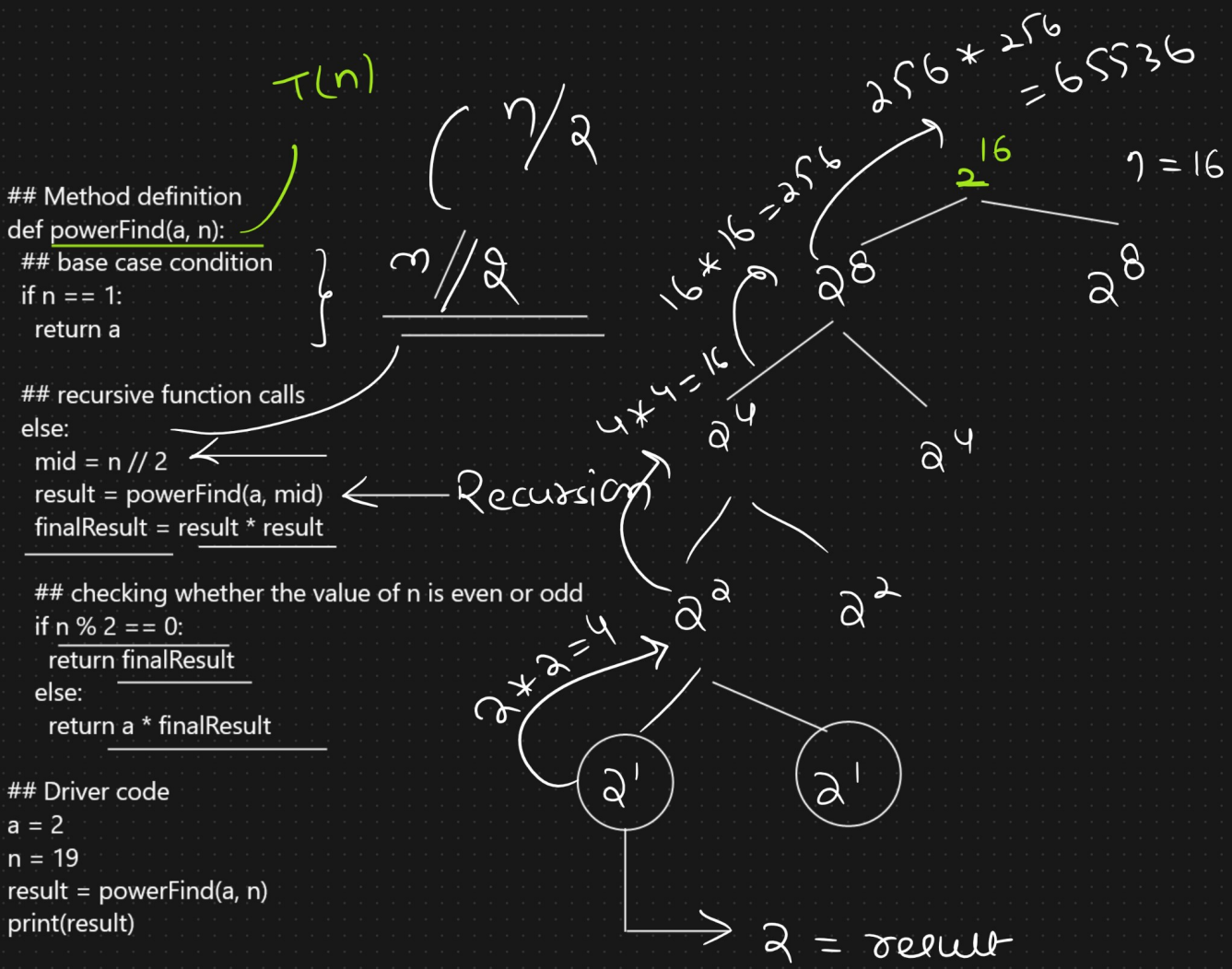
2 * powerfind(2, 14)

Divide & Conquer


256*256 = 65536
2^{16}
16*16 = 256
2^8
2^8
16
4*4=16
2^4
2^4
4
2^2
2^2
2^1
2^1
Base case Condition
2
n==1
return a
1
12
n=17
0
1
$2*2^{16}$
2
65536
powerfind(a,n):
if n==1:
return a
else:
mid = n//2
result = powerfind(a,mid)
finalResult = result * result
$2^{15} = 2*2^{14}$
2^7
2^7
$2*2^6$
if n%2==0:
return finalResult
else:
return a * finalResult


2^3
2^3
$2 * 2^2$
2^1
2^1
$m = 17$
$a = 2$
2^{17}
$= 2 * 2^{16}$
$m = 31$
$a = 2$
$2^{30} * 2$

Divide & conquer



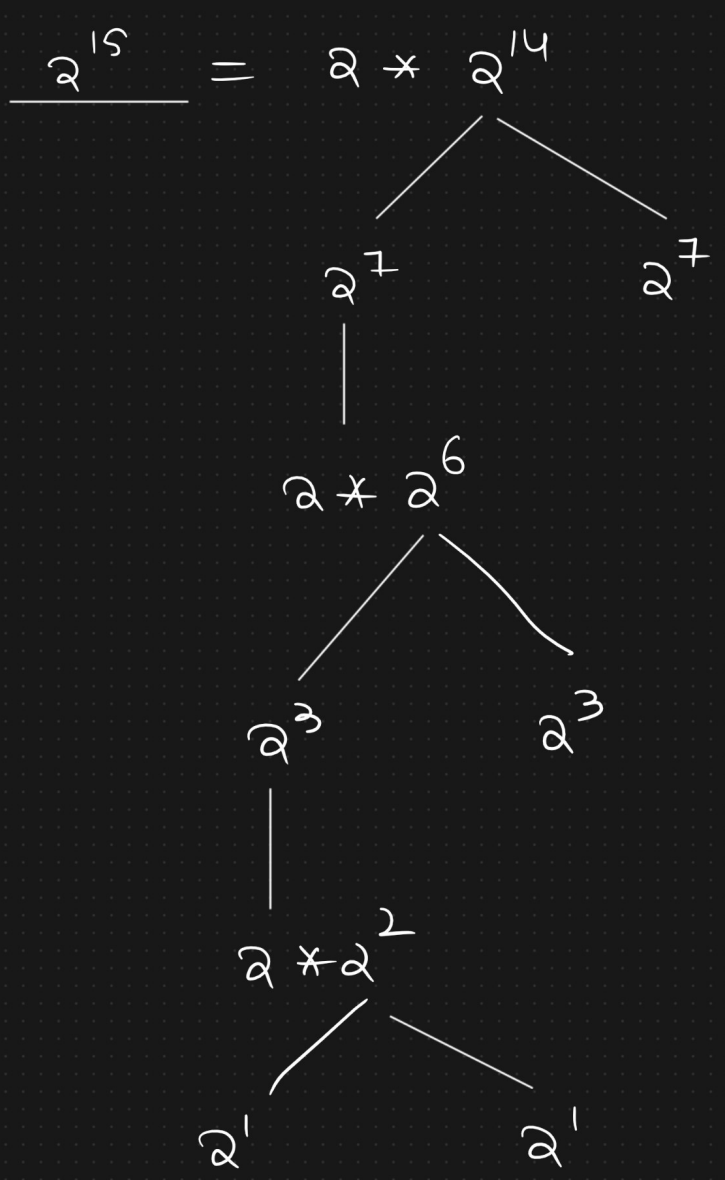

```
## Method definition
def powerFind(a, n):
  ## base case condition
  if n == 1:
    return a

  ## recursive function calls
  else:
    mid = n // 2
    result = powerFind(a, mid)
    finalResult = result * result

    ## checking whether the value of n is even or odd
    if n % 2 == 0:
      return finalResult
    else:
      return a * finalResult

## Driver code
a = 2
n = 19
result = powerFind(a, n)
print(result)
```

$$2^{17} = 2 * 2^{16}$$


$2^{15} = 2 * 2^{14}$
2^7
2^7
$2 * 2^6$
2^3
2^3
$2 * 2^2$
2^1
2^1

Most optimized

$T(1) = a$

```
## Method definition
def powerFind(a, n):
  ## base case condition
  if n == 1:
    return a
  ## recursive function calls
  else:
    mid = n // 2
    result = powerFind(a, mid)
    finalResult = result * result
    ## checking whether the value of n is even or odd
    if n % 2 == 0:
      return finalResult
    else:
      return a * finalResult

## Driver code
a = 2
n = 19
result = powerFind(a, n)
print(result)
```

$T(n)$

c

c

$T(n/2)$

c

c

Recursion

$$T(n) = \begin{cases} a & n = 1 \\ T(n/2) + c & n > 1 \end{cases}$$

Logarithmic time complexity

$O(\log n)$

$$T(n) = T(n/2) + c \quad \text{---} \quad (1)$$

$$T(n/2) = T(n/2^2) + c$$

$$T(n) = T\left(\frac{n}{2^2}\right) + 2c \quad \text{---} \quad (2)$$

$$T(n) = T\left(\frac{n}{2^3}\right) + 3c \quad \text{---} \quad (3)$$

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

↓ k times

$n = 2^k$

$\log_2 n = k \log_2 2$

$k = \log_2 n$

$$T(n) = T\left(\frac{n}{2^k}\right) + k \cdot c$$

$$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot c$$

$$= T\left(\frac{n}{n^{\log_2 2}}\right) + c \cdot \log_2 n$$

$$= a + c \cdot \log_2 n$$

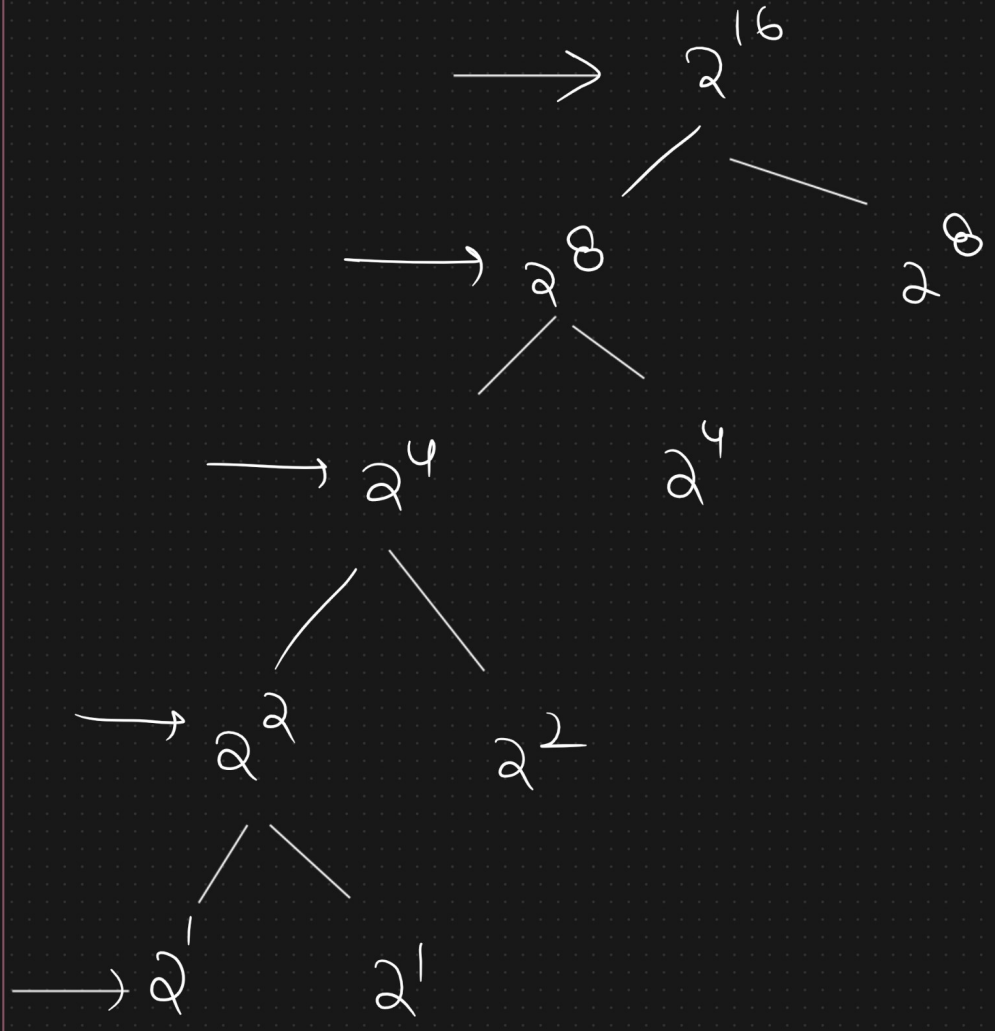
$$\Rightarrow O(\log_2 n)$$

Space complexity $\longrightarrow O(\log_2 n)$

$\log_2 2^4 \approx 4$

$\log_2 16 \approx 4/5$

16 → $n/2$

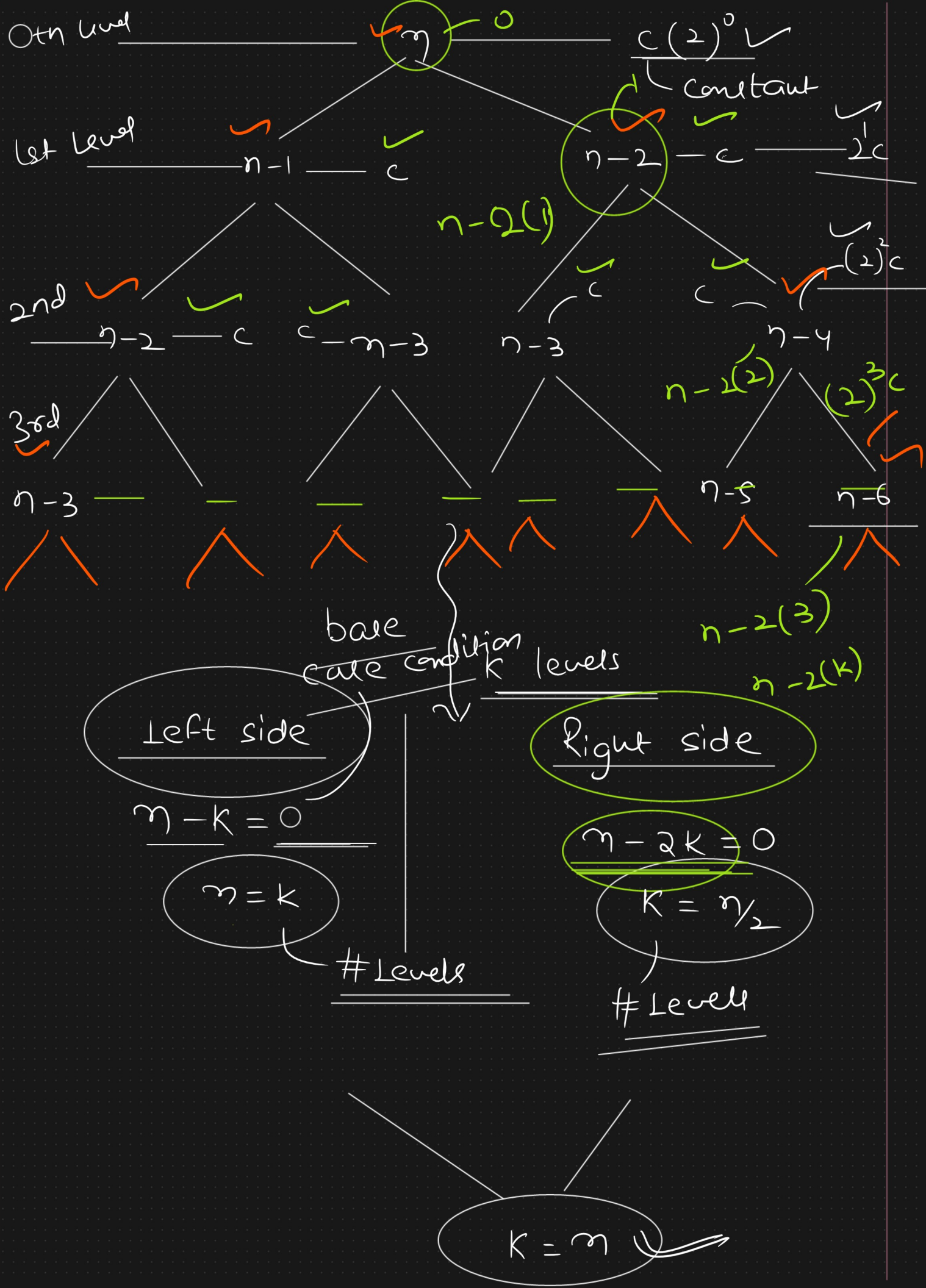


Recursive tree Approach

$$T(n) = \begin{cases} c & ; n = 0/1 \\ T(n-1) + T(n-2) + c; & n > 1 \end{cases}$$

↳ More than one recursive term, then that recurrence relation can be solved via Recursive tree approach

$$T(n) = T(n-1) + T(n-2) + c$$


0th Level
n
0
c(2)^0
constant
1st Level
n-1
c
n-2
c
2^1 c
n-2(1)
2nd
n-2
c
c
n-3
n-3
c
c
n-4
(2)^2 c
n-2(2)
3rd
n-3
n-5
n-6
(2)^3 c
n-2(3)
n-2(k)
base case condition
k levels
Left side
n-k=0
n=k
#Levels
Right side
n-2k=0
k=n/2
#Levels
k=n

$$2^0c + 2^1c + 2^2c + \cdots + 2^kc$$

$$c(2^0 + 2^1 + 2^2 + \cdots + 2^k)$$

$$c(2^0 + 2^1 + 2^2 + \cdots + 2^n)$$

$r = b/a$

$$r = \frac{2^1}{2^0} = 2 \quad \text{GP series}$$

$$\frac{\text{Sum of}}{\text{GP series}} = \frac{a(r^n - 1)}{r - 1} \qquad r > 1$$

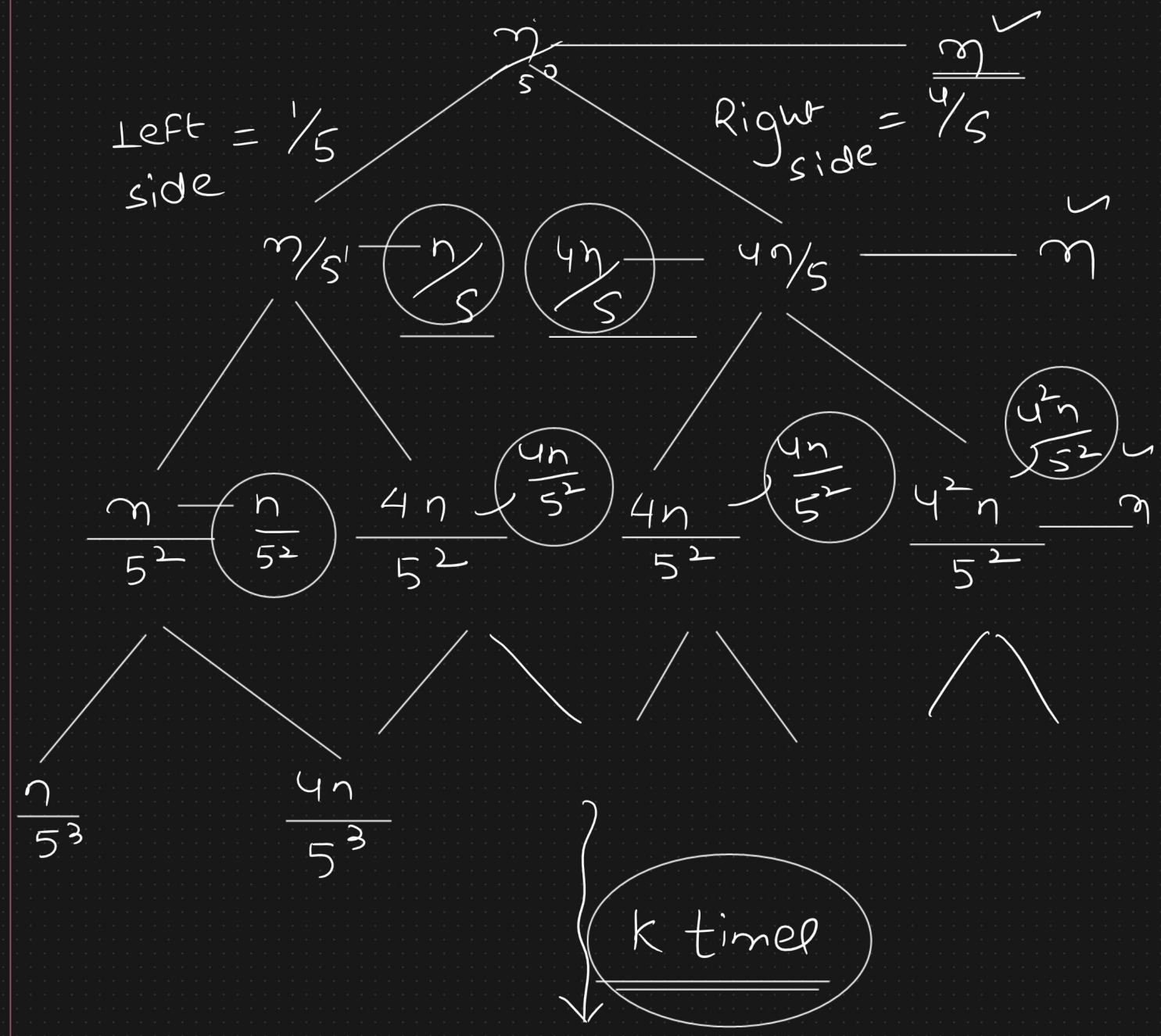
$$= \frac{1 \cdot (2^n - 1)}{2 - 1}$$

$$= O(2^n)$$

Exponential Time Complexity

$$T(n) = \begin{cases} 1 & n=1 \\ T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n & n>1 \end{cases}$$

Recursive tree approach



Left side

$$\frac{n}{5^k} = 1$$

$$n = 5^k$$

$$k = \log_5 n$$

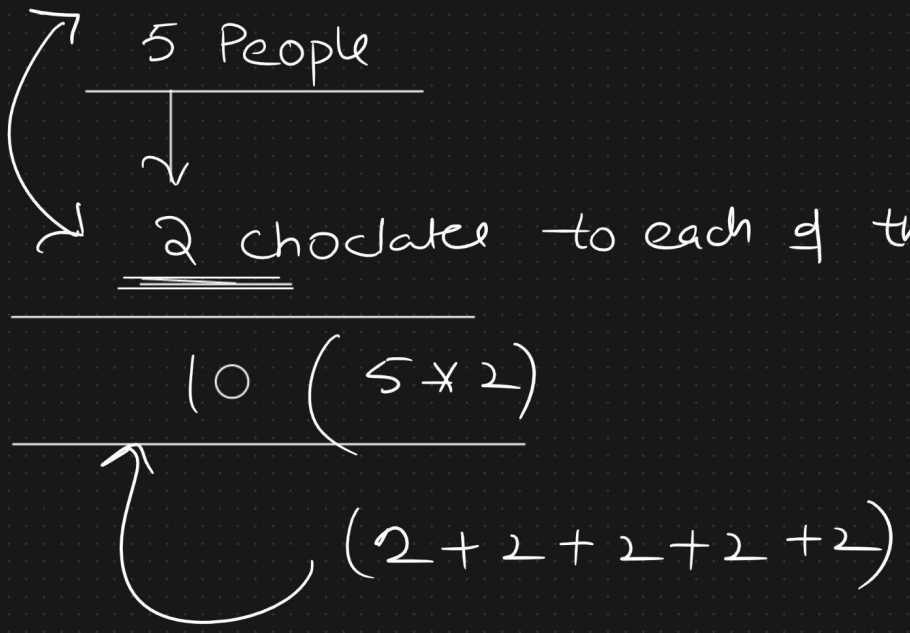
Right side

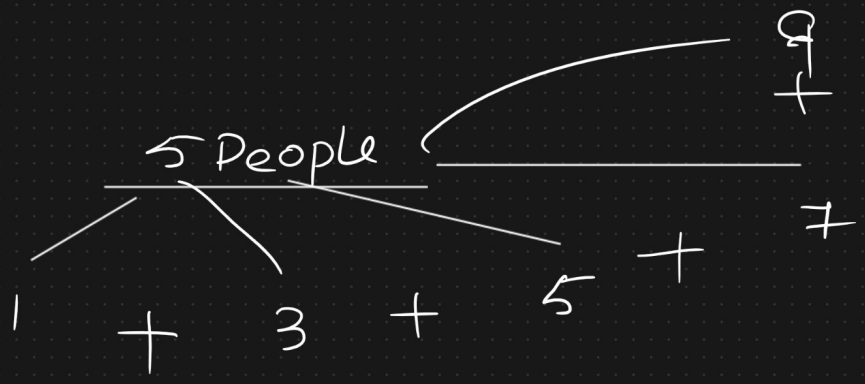
$$\frac{n}{(5/4)^k} = 1$$

$$k = \log_{5/4} n$$

$$O(m \cdot k)$$

$$O(n \log_{5/4} n)$$


5 People
2 chocolates to each of them
10 (5 * 2)
(2 + 2 + 2 + 2 + 2)


9
5 People
7
1 + 3 + 5 +