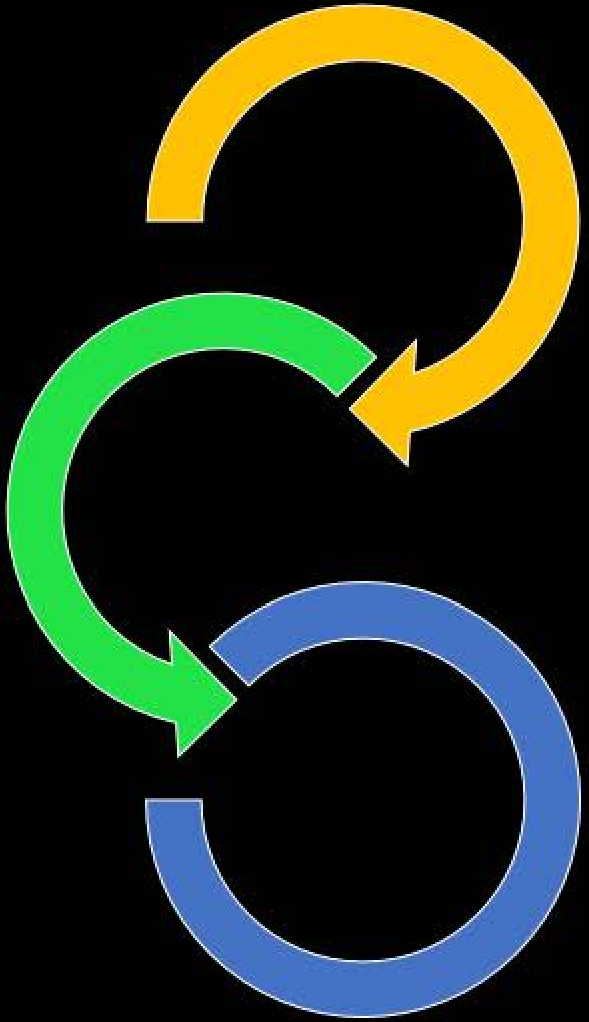


Quick Sort

Quick Sort Algorithm -

Description of Quick Sort

- Quicksort, is based on the divide-and-conquer paradigm.

- Divide &
  Conquer

- Quick sort

Three-step divide-and-conquer process

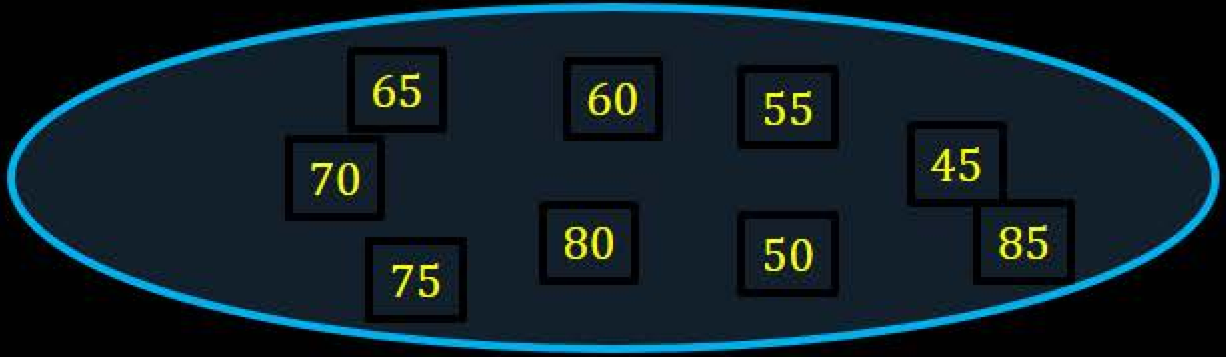
- **Divide:** Partition (rearrange) the array $A[p \quad q]$ into two (possibly empty) subarrays $A[p \ \ j - 1]$ and $A[j + 1 \ \ q]$ such that each element of $A[p \ \ j - 1]$ is less than or equal to $A[j]$, which is, in turn, less than or equal to each element of $A[j + 1 \quad q]$. Compute the index j as part of this partitioning procedure.

Three-step divide-and-conquer process

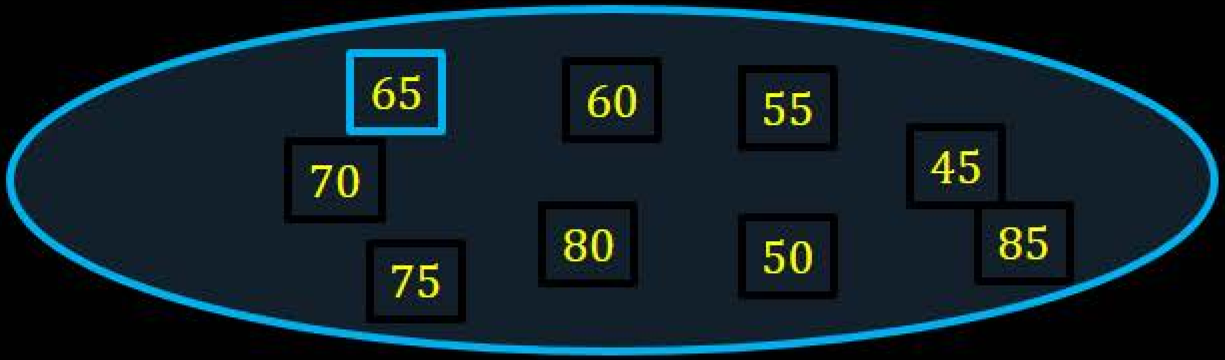
- **Conquer:** Sort the two subarrays $A[p\ \ j-1]$ and $A[j+1\ \ q]$ by recursive calls to quicksort.

- **Combine:** Since the subarrays are sorted in place, no work is needed to combine them: the entire array $A[p\ \ q]$ is now sorted.

Quick Sort: Example



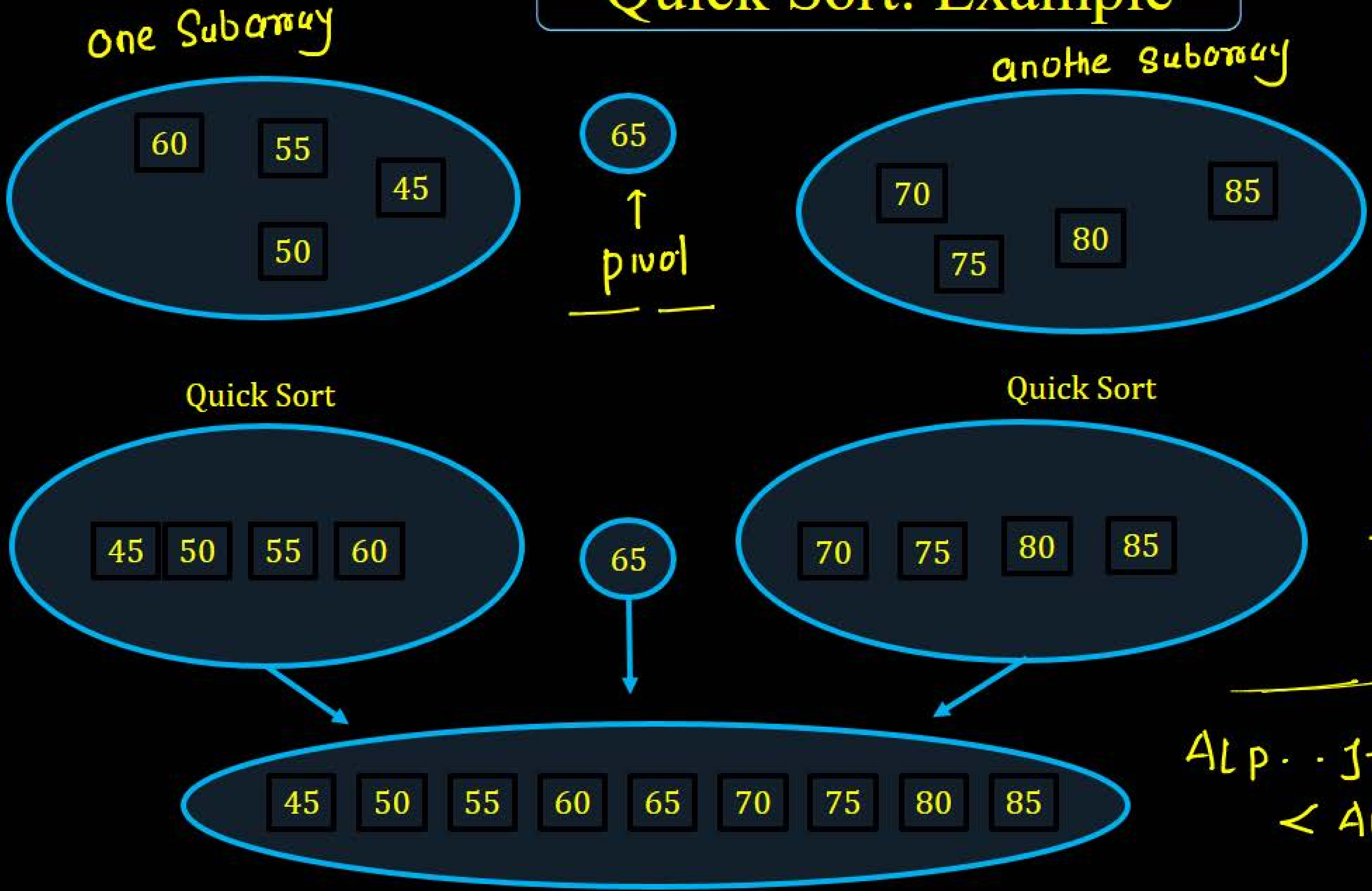
Select Pivot



Selection of pivot element

Hoare partitioning

Quick Sort: Example



After partitioning

A[j] is in its correct position

A[p..q]

A[j]

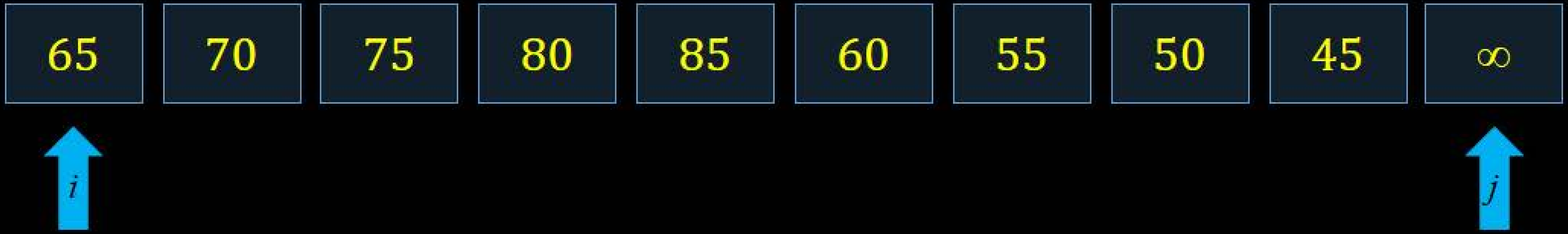
A[p..j-1] < A[j]

A[j+1..q]

Issues To Consider

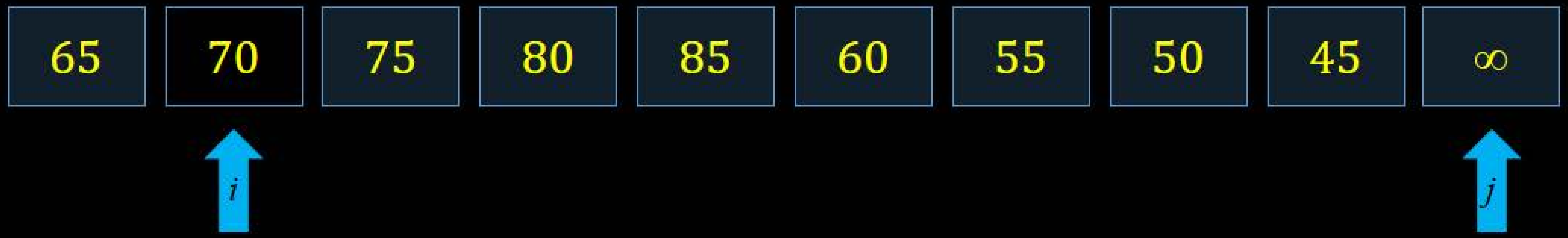
- How to pick the pivot?
  - Many methods
- How to partition?
  - Several methods exist.
  - The one we consider is known to give good results and to be easy and efficient.
  - We discuss the partition strategy first.

Hoare Partitioning



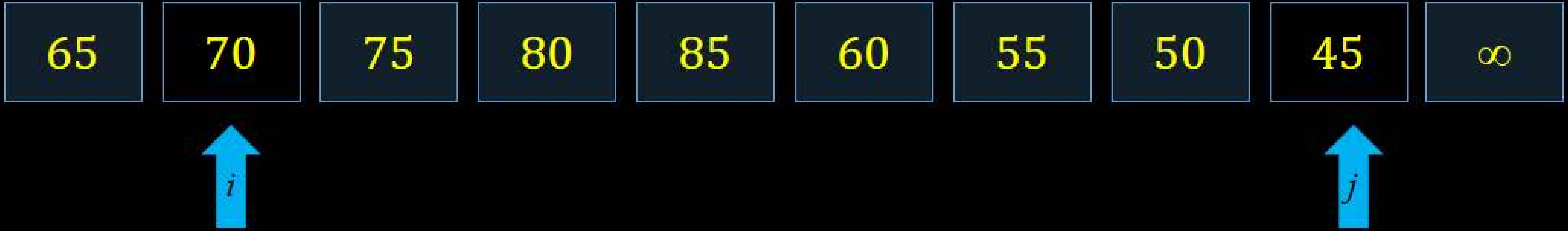
65	70	75	80	85	60	55	50	45	∞
i									j

Hoare Partitioning



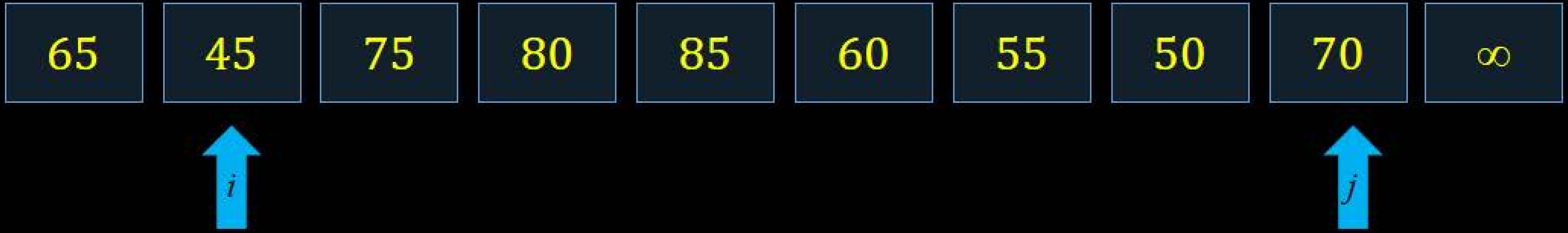
65	70	75	80	85	60	55	50	45	∞
	i								*j*

Hoare Partitioning



65	70	75	80	85	60	55	50	45	∞

Hoare Partitioning



65	45	75	80	85	60	55	50	70	∞

Partition Happen around pivot

(1) two Subarray

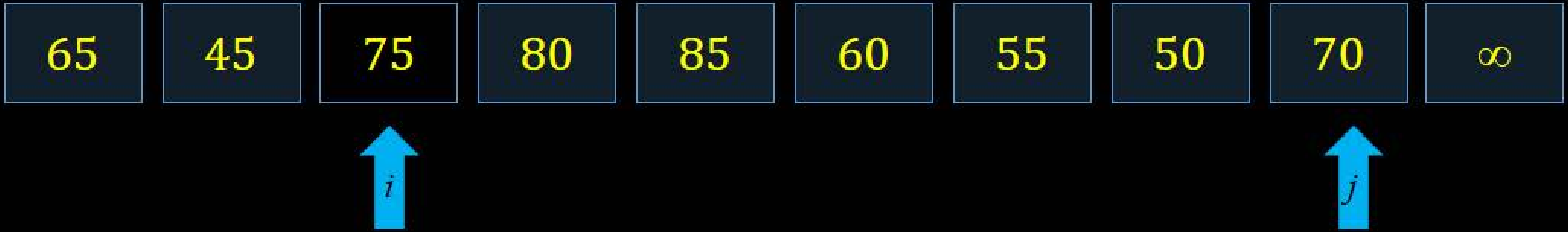
(2) pivot element will be in correct position

(3) A[p..q] – A[j]

A[p... j-1] – lesser value A[j]

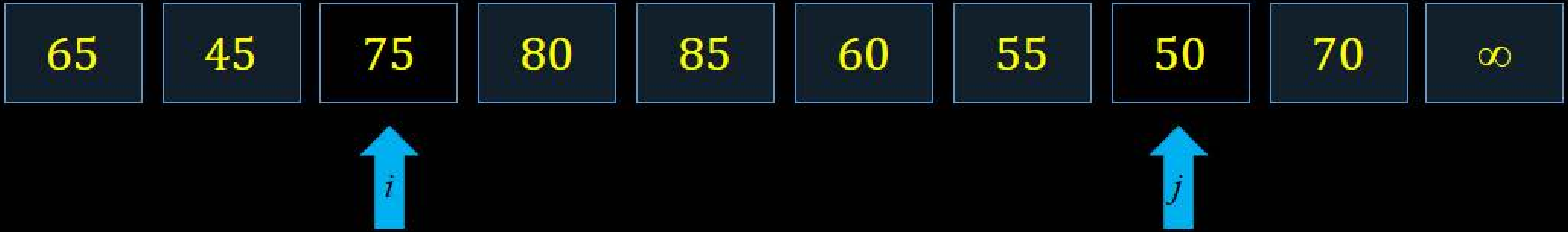
A[j+1 -- q] – greate value A[j]

Hoare Partitioning



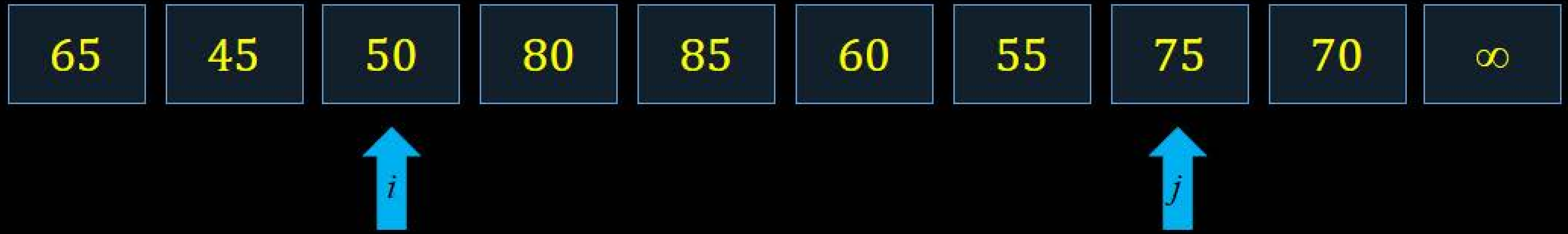
65	45	75	80	85	60	55	50	70	∞

Hoare Partitioning



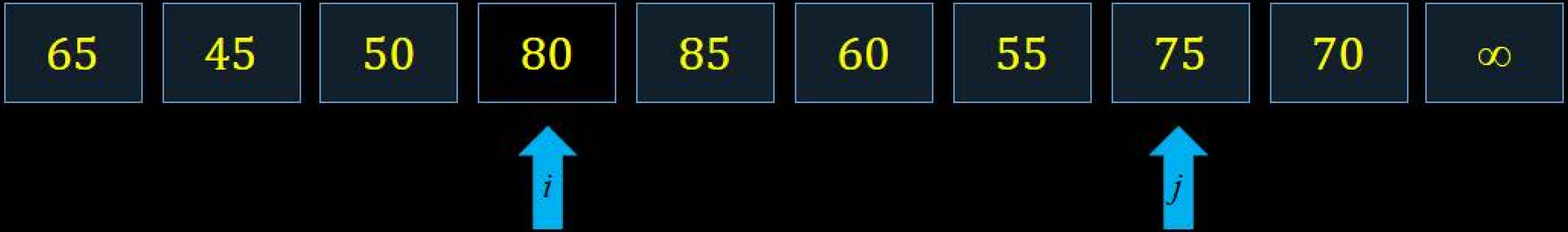
65	45	75	80	85	60	55	50	70	∞

Hoare Partitioning



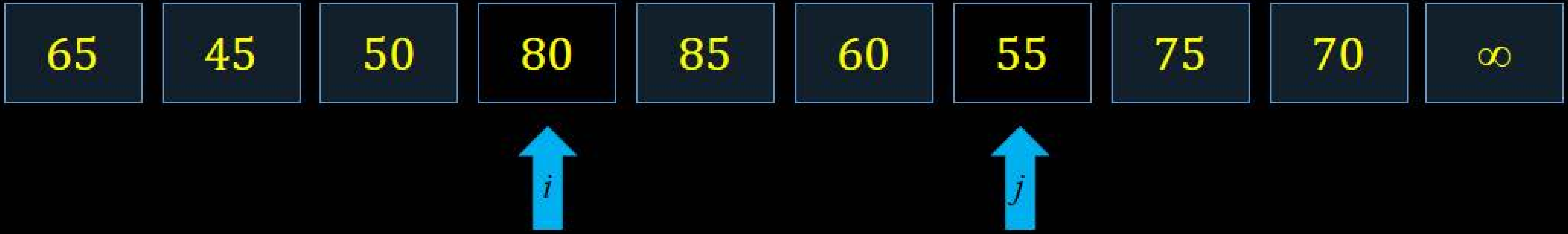
65	45	50	80	85	60	55	75	70	∞

Hoare Partitioning



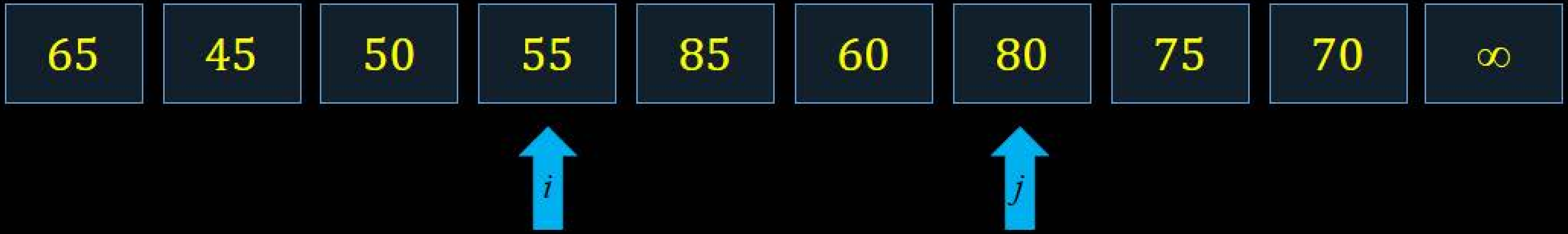
65	45	50	80	85	60	55	75	70	∞

Hoare Partitioning



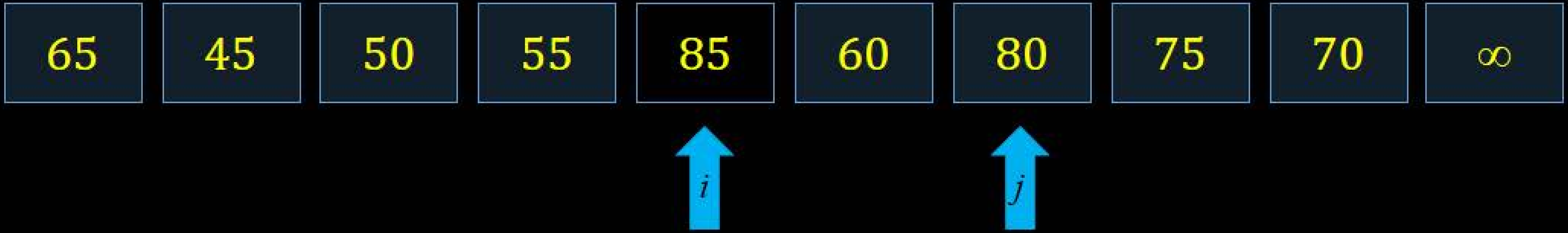
65	45	50	80	85	60	55	75	70	∞
			i			j			

Hoare Partitioning



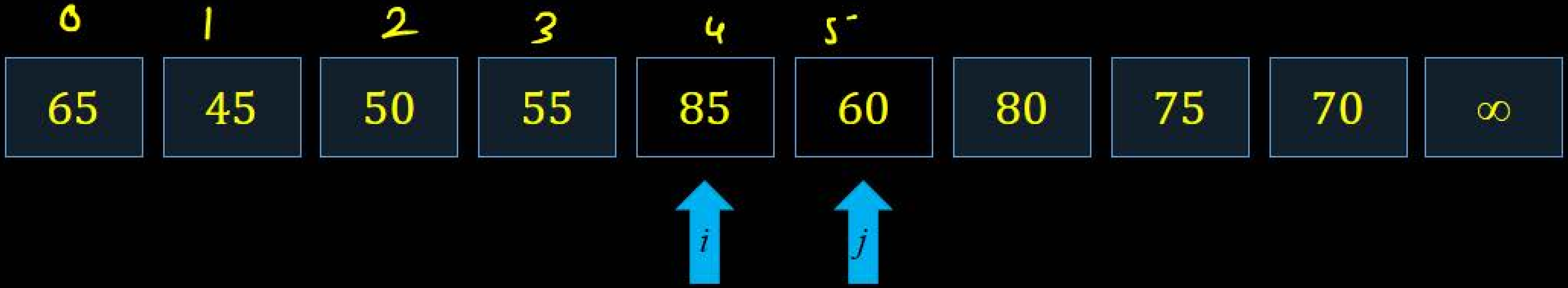
65	45	50	55	85	60	80	75	70	∞
			i			*j*			

Hoare Partitioning



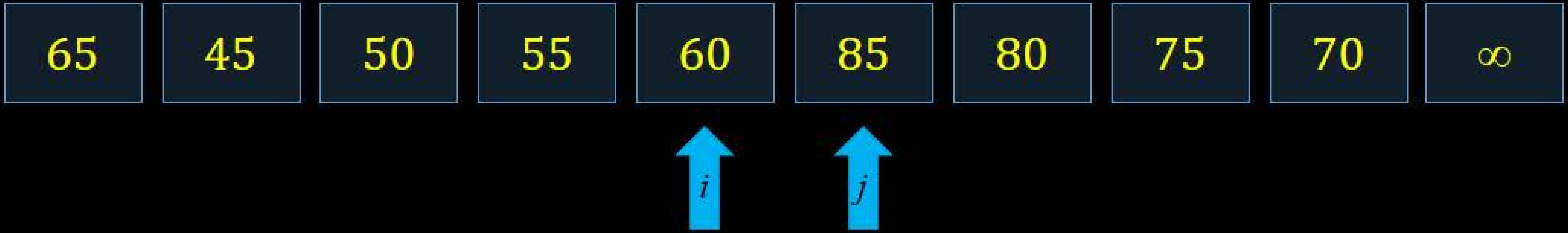
65	45	50	55	85	60	80	75	70	∞

Hoare Partitioning



0	1	2	3	4	5				
65	45	50	55	85	60	80	75	70	∞
				i	*j*				

Hoare Partitioning



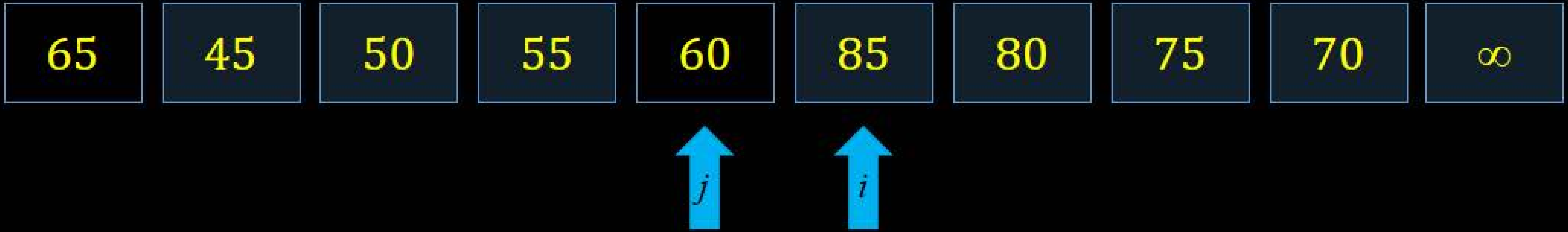
65	45	50	55	60	85	80	75	70	∞

Hoare Partitioning

65	45	50	55	60	85	80	75	70	∞
				↑ j	↑ i				

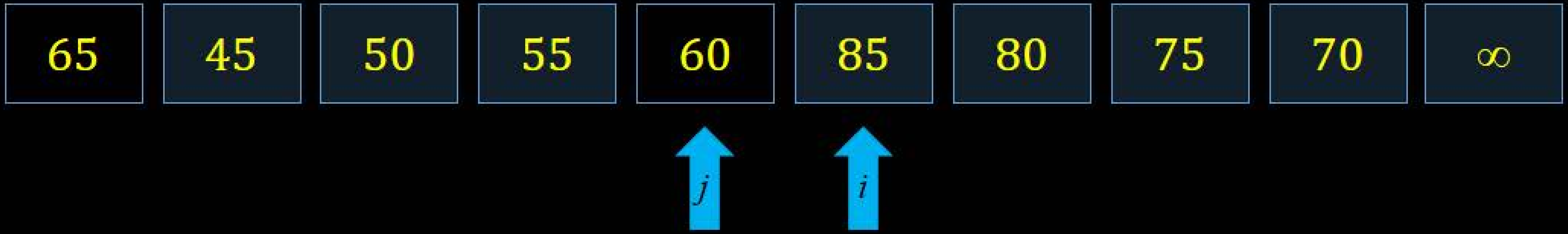
Swap A[j] with pivot

Hoare Partitioning



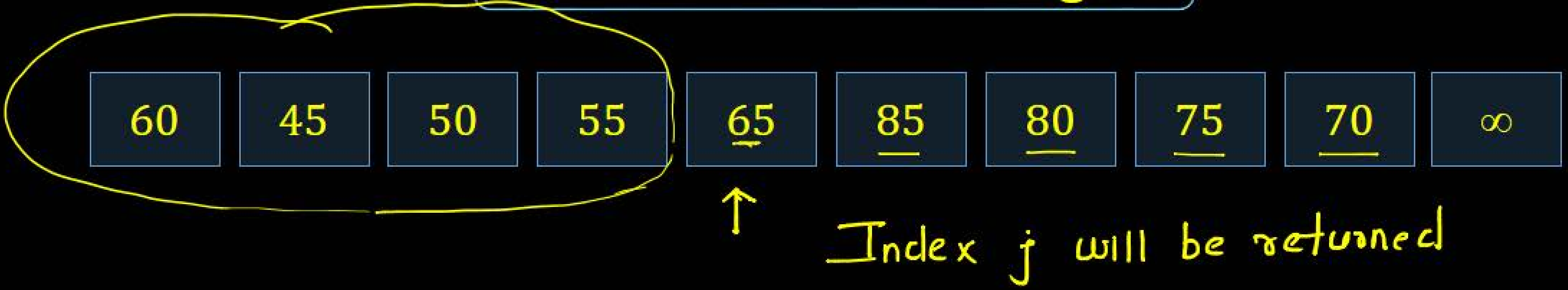
65	45	50	55	60	85	80	75	70	∞

Hoare Partitioning



65	45	50	55	60	85	80	75	70	∞

Hoare Partitioning



60	45	50	55	65	85	80	75	70	∞

Index j will be returned

The QuickSort Method

```
Algorithm QuickSort(p, q)
   if (p < q) then{
      j := Partition(a, p, q + 1);
      QuickSort(p, j - 1);
      QuickSort(j + 1, q);
   }
}
```

The Partition Subroutine

```
Algorithm Partition(a, m, p){
v := a[m]; i := m; j := p;
repeat {
  repeat
    i := i + 1;
  until (a[i]  >= v);
   repeat
    j=j- 1;
   until (a[j]<= v);
   if (i < j) then Interchange(a,i,j);
} until (i >= j);
a[m] := a[j]; a[j] := v; return j;
}
```

65	~~70~~	75	80	85	60	55	50	45	∞	~~2~~	~~9~~
	45	50					75	70			
		i					j	j	j		

partition algorithm

A[m] - pivot

The Partition Subroutine

Use the initial value of A[p] as the “pivot,” in the sense that the keys are compared against it. Scan the keys A[p..q − 1] and rearranges them.

The Partition Subroutine

The Partition Subroutine

Given a subarray $A[p..q]$ such that $p \leq q - 1$, this subroutine rearranges the input subarray in to two subarrays, $A[p..j - 1]$ and $A[j+1..q]$, so that

- each element in $A[p..j - 1]$ is less than or equal to $A[j]$ and
- each element in $A[j+1..q]$ is greater than or equal to $A[j]$

Then the subroutine outputs the value of j.

partition subroutine

Partition the following

Index

7	2	1	6	8	5	3	4
				i			j

7	2	1	6	4	5	3	8
						j	i

3	2	1	6	4	5	7	8
						↑	

partition this around pivot

pivot is a[1]

Subroutine is a procedure or function

Partition the following

pivot = 17

10		11	13			17		31				22
~~17~~	9	~~22~~	31	7	12	10	21	13	29	18	20	11
		i	i			j	i	j				j

partition the array using quick Sort

10, 9, 11, 13, 7, 12, $\boxed{17}$ 21, 31, 29, 18, 20, 22

Answer

pivot = 17

10	9	11	13	7	12	17	21	31	29	18	20	22

Partition the following

$i = i+1$

$\}(A[i] >= v)$

$K=1$

0

$T(n) = n + \underline{T(0)} + T(n-1)$

pivot = 1

j

1	2	3	4	5	6	7	8	9	10	11	12	13

p i j

partition (a, p, q+1)

pivot value, j

13 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 $\boxed{\infty}$ ← partition it

j i

while ( ) { } repeat { }

1 12 11 10, 9, 8, 7 6 5, 4, 3 2, $\boxed{13}$ __

↑ 0

n - 1

worst case partitioning

Answer

pivot = 17

1	2	3	4	5	6	7	8	9	10	11	12	13

Question:MSQ

Consider the following elements (in order) of an array A[9] just after partitioning it by first step of quick sort.

4,1,3,5,6,9,8,7,10

Which of the following is TRUE

(A) 5 is chosen as pivot value ✓

(B) 6 is chosen as pivot value ✓

(C) 7 is chosen is as pivot value ✗

(D) 10 is chosen as pivot value

a, b, d

Multiple ~~Sec~~ select choice

more than one

Other than the ~~Last~~ first element as Pivot

Any other element other than last element is taken as pivot then first we will perform the exchange and then we start working on the same algorithm

first element is considered as pivot

- fist perform exchange operation with ~~fi~~ first position and then Same partition algorithm will be followed

Time Complexity of Quick Sort

Depends upon - Selection of <u>pivot</u>

pivot elem is in k^{th} position $\quad A[1 \ldots n]$

Subarray - Left subarray $= 1 .. k-1 \qquad \underline{k-1}-1+1$

Right subarray $= k+1 - n \quad - \quad n-k-\cancel{1}+\cancel{1}$

$\underline{n-k}$

$$T(n) = \underbrace{T(k-1)}_{\uparrow\ \text{Left subarray}} + \underbrace{T(n-k)}_{\text{Right subarray}} + \underset{\uparrow\ \text{cost of partitioning}}{\underline{n}}$$

Time Complexity of Quick Sort

$k = \textit{final position of pivot element}$

Time Complexity of Quick Sort

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

$$T(0) = T(1) = 0$$

partition Method

either increment $\underline{i}$

or

decrementing $\underline{j}$

1 $\underline{2}$ 3 4, 5 ∞

k = *final position of pivot element*

n = *Partition*

$T(k-1)$ = *sorting the left subarray using quick sort*

$T(n-k)$ = *sorting the right subarray using quicksort*

Worst Case Time Complexity of Quick Sort

Quick Sort worst case will occur when elements are alredy sorted.

$$T(n) = T(n-1) + T(0) + n$$

$$T(n) = T(n-1) + \underline{n} \Rightarrow T(n) = \Theta(n^2) \text{ (for worst case)}$$

worst case time complexity for quick sort occurs when elements already <u>sorted</u> — $\underline{O(n^2)}$

Worst Case Time Complexity of Quick Sort

A bad case (actually the worst case): At every step.

Partition( ) splits the array as unequally as possible

(k=1 or k = n). Then our recurrence becomes

Worst Case Time Complexity of Quick Sort

$T(n) = \underline{n} + \underline{T(n\text{-}1)}$, $\underline{T(0)} = \underline{T(1) = 0}$

This is easy to solve.

$$\left\{\begin{aligned}
&T(n) = n + T(n\text{-}1) \\
&= n + n\text{-}1 + T(n\text{-}2) \\
&= n + n\text{-}l + n\text{-}2 + T(n\text{-}3) \\
&= n + n\text{-}l + n\text{-}2 + \ldots + 3 + 2 + T(0) \\
&= (n + n\text{-}l + n\text{-}2 + \ldots + 3 + 2 + 1) - 1 \\
&= \frac{n(n+1)}{2} - 1
\end{aligned}\right.$$

$\approx n^2/2$

$\boxed{\mathrm{O}(n^2)}$

Best Case Time Complexity

Best case for quick sort : Equal partitioning

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

$T(n) = 2T\left(\frac{n}{2}\right) + n$ (for analysis)

$a = 2$

$b = 2$

$n^{\log_2 2} = n$

$f(n)$ is $\theta(n^{\log_2 2} \log^0 n)$

$T(n) = \theta(n \log n)$

The Best case when equal partition occurs in every step and hence $\theta(n \log n)$

Lower bound $-\Omega(n \log n)$

upper bound $-O(n^2)$

Best Case Time Complexity

$$T(n) = \underline{2\,T\left(\frac{n}{2}\right) + n}$$

$$T(n) = \Theta(n \mathrm{Log} n)$$

GATE 1987 | 2 Marks Question

Let P be a quicksort program to sort numbers in ascending order. Let t1 and t2 be the time taken by the program for the inputs [1 2 3 4] and [5 4 3 2 1], respectively. Which of the following holds?

(A) $t1 = t2$

(B) $t1 > t2$

[✓ (C) $t1 < t2$]

(D) $t1 = t2 + 5\log 5$

pivot will be compared ~~un~~ against each value

t_1 1, 2, 3, 4 ascending

$t_1 < t_2$

5 4 3, 2, 1 t_2

descending

worst case quick sort

$t_1 < t_2$

worst case for quick sort

1, 2, 3, 4 — 3
i

1 [2 3 4] — n - 1

2 3 4 — 2

[3 . 4] — 1

4

(3+2+1) total comparison

4 3, 2, 1 ∞ — n-1
↑

1 3 2 4 — 3

[1 3, 2] [4] — 2

[3 2] — 1

[2] 3

GATE 1992 | 2 Marks Question

Assume that the last element of the set is used as partition element in Quicksort. If n distinct elements from the set $[1.....n]$ are to be sorted, give an input for which Quicksort takes maximum time.

Theoritical

5 4 3 2 1

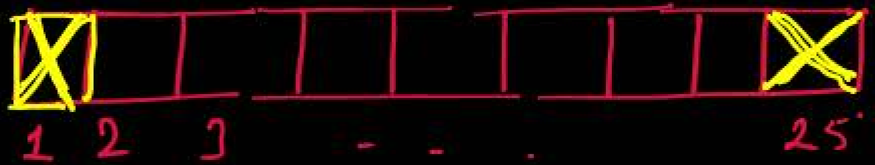
1, 2, 3, 4, 5

GATE 2019, Question Number 20

An array of 25 distinct elements is to be sorted using quick sort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is 0.08__.

is –

first



what is the wast possible posithons – 1, and 25

$$\text{probability} = \frac{2}{25} = \boxed{0.08}$$

GATE 1996 | 2 Marks Question

Quick-sort is run on two inputs shown below to sort in ascending order

(i) $1,2,3 \ldots .n$

(ii) $n, n - 1, n - 2, \ldots .2, 1$

Let C_1 and C_2 be the number of comparisons made for the inputs (i) and (ii) respectively. Then,

(A) $C_1 < C_2$

(B) $C_1 > C_2$

(C) $C_1 = C_2$

(D) we cannot say anything for arbitrary n.

GATE 2019, Question Number 20

An array of 25 distinct elements is to be sorted using quick sort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is _____.

GATE 2014, Question Number 14, 1-Mark

Let P be a quicksort program to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparisons made by P for the inputs [12345] and [41532] respectively. Which one of the following holds?

(A) $t_1 = 5$ (B) $t_1 < t_2$ (C) $t_1 > t_2$ (D) $t_1 = t_2$

GATE 2014| 1-Mark Question

Step by step also

No. of comparison

Let P be a quicksort program to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparisons made by P for the inputs [12345] and [41532] respectively. Which one of the following holds?

(A) $t_1 = 5$ (B) $t_1 < t_2$ [✓ (C) $t_1 > t_2$] (D) $t_1 = t_2$

[1 2 3 4 5] — 4

[1] [2, 3, 4 5] — 3

[3, 4, 5] — 2

[4, 5] - 1

[4 1 5 3 2] — 4

i j

4 1, 2 3 5

i j j i

[3 1 2] 4 [5]

[3 1, 2] → 2

i 2

2 1 3 — 2

[2, 1] - 1

GATE 2014| 1-Mark Question

Question is asking about number of comparisons.

The splitting occurs as
[1][2345]
[2][345]
[3][45]
[4][5]]

and

[123][45]
[1][23][4][5]
[2][3]

Hence, in second case number of comparisons is less. => t1 > t2.

Recurrence Tree

Consider the following Recurrence Relation

$$T(n) = \underline{3T(\lfloor n/4 \rfloor)} + \underline{\Theta(n^2)}$$

$a = 3$

$b = 4$

$n^{\log_4 3}$

$f(n) = \underline{n^2}$

(A) Draw the Recurrence Tree

(B) Cost at each level

(C) Height of the Recurrence Tree

(D) Number of leaves

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$ —— I

$T(n/4) = 3T(n/16) + \frac{n^2}{16}$ —— II

$T(n) = 3^2 T(n/4^2) + \frac{3}{16}n^2 + n^2$ ←

$T(n/4^2) = 3T(n/4^3) + \frac{n^2}{16^2}$ —— III

$= 3^3 T(n/4^3) + \left(\frac{3}{16}\right)^2 n^2 + \left(\frac{3}{16}\right) n^2 + n^2$

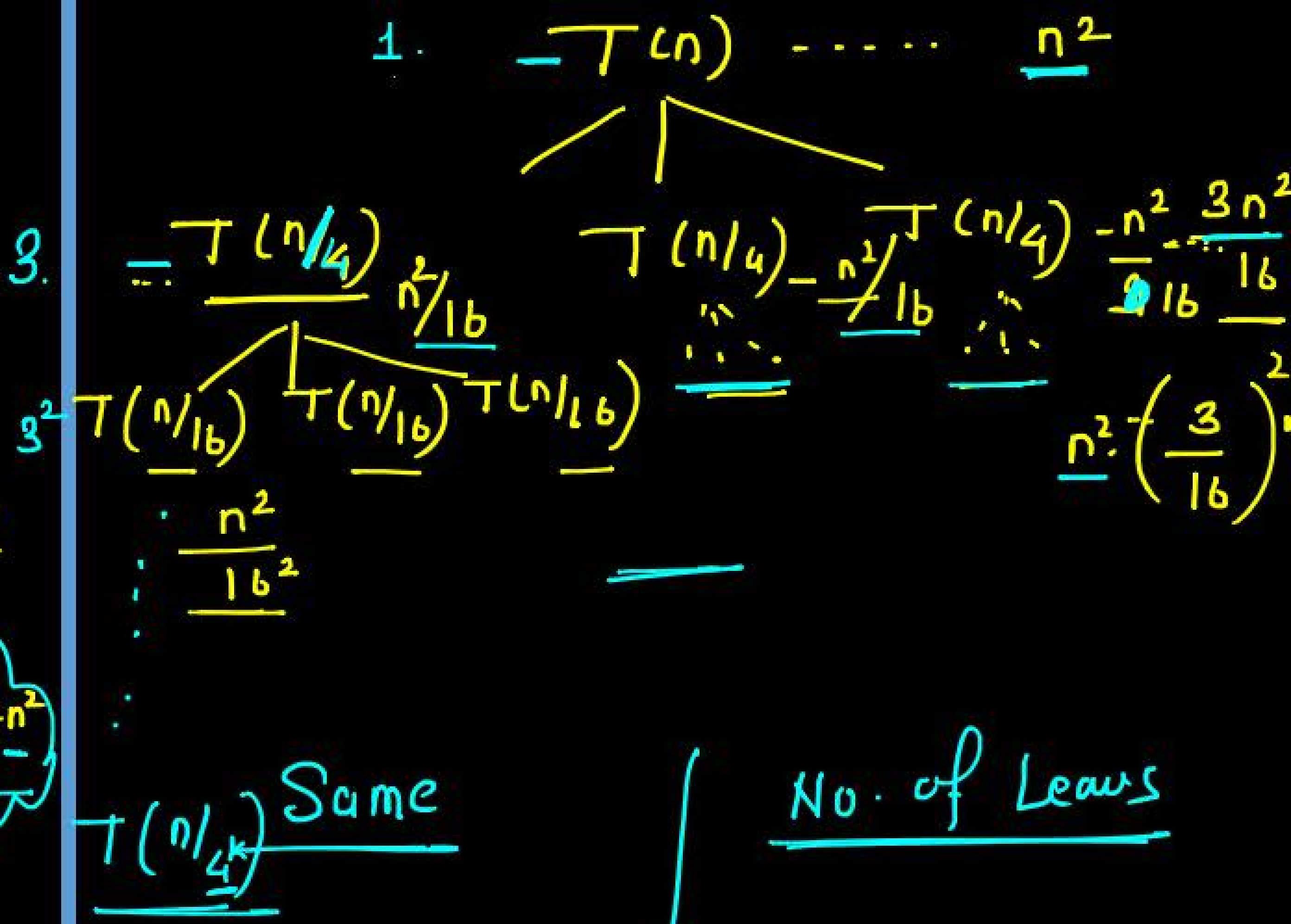
No. of Leaves

Series

$\boxed{3^k} T(n/4^k)$

$n/4^k = 1$ (heigh)

$3^k = 3^{\log_4 n} = n^{\log_4 3}$



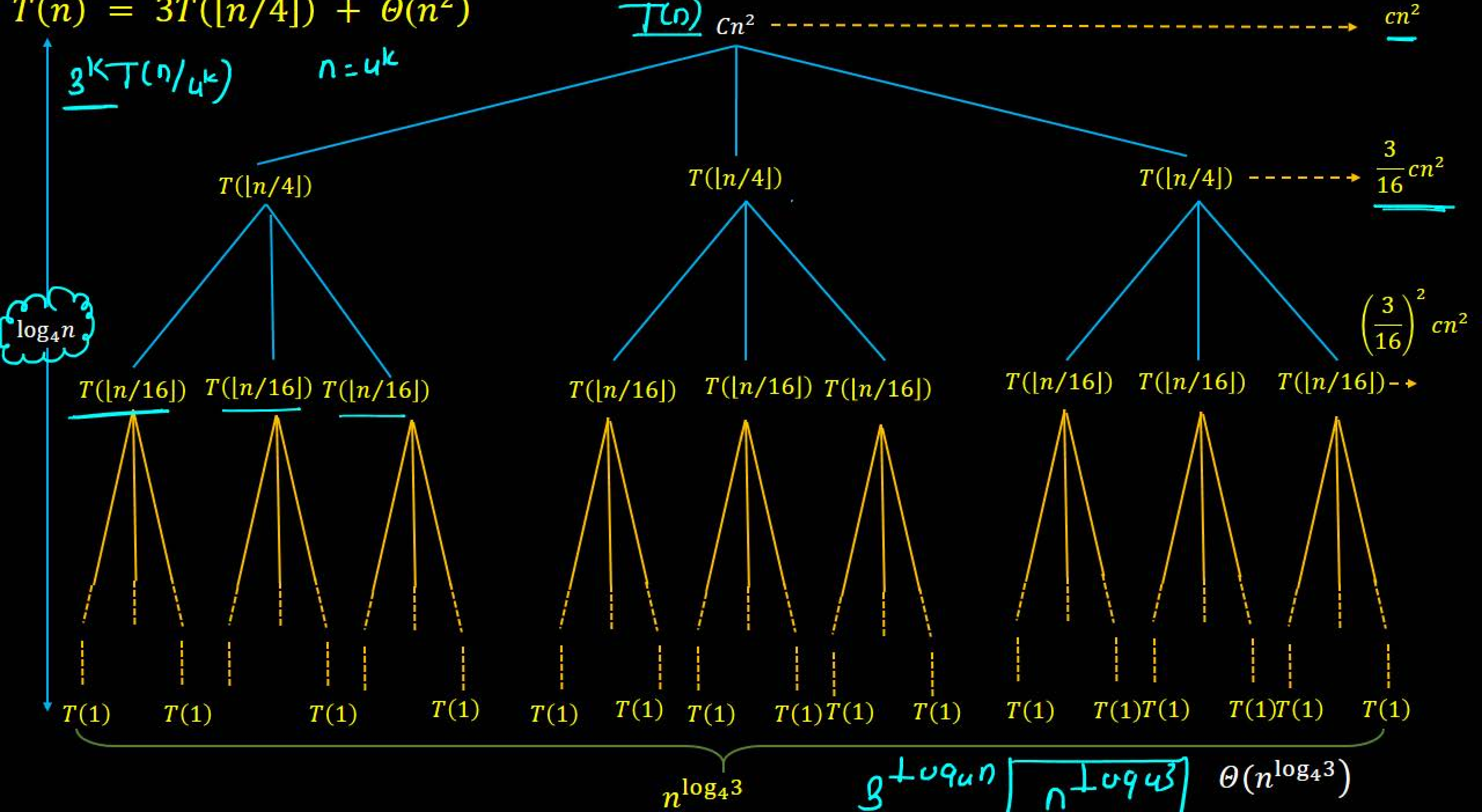
for $n = 4^k$

$k = \log_4 n$

← No. of leaves

No. of Leaws

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$



$\underline{T(n)}$ Cn^2 → $\underline{cn^2}$

$\underline{3}^K T(n/4^k)$

$n = 4^k$

$T(\lfloor n/4 \rfloor)$ $T(\lfloor n/4 \rfloor)$ $T(\lfloor n/4 \rfloor)$ → $\underline{\frac{3}{16} cn^2}$

$\left(\frac{3}{16}\right)^2 cn^2$

$\log_4 n$

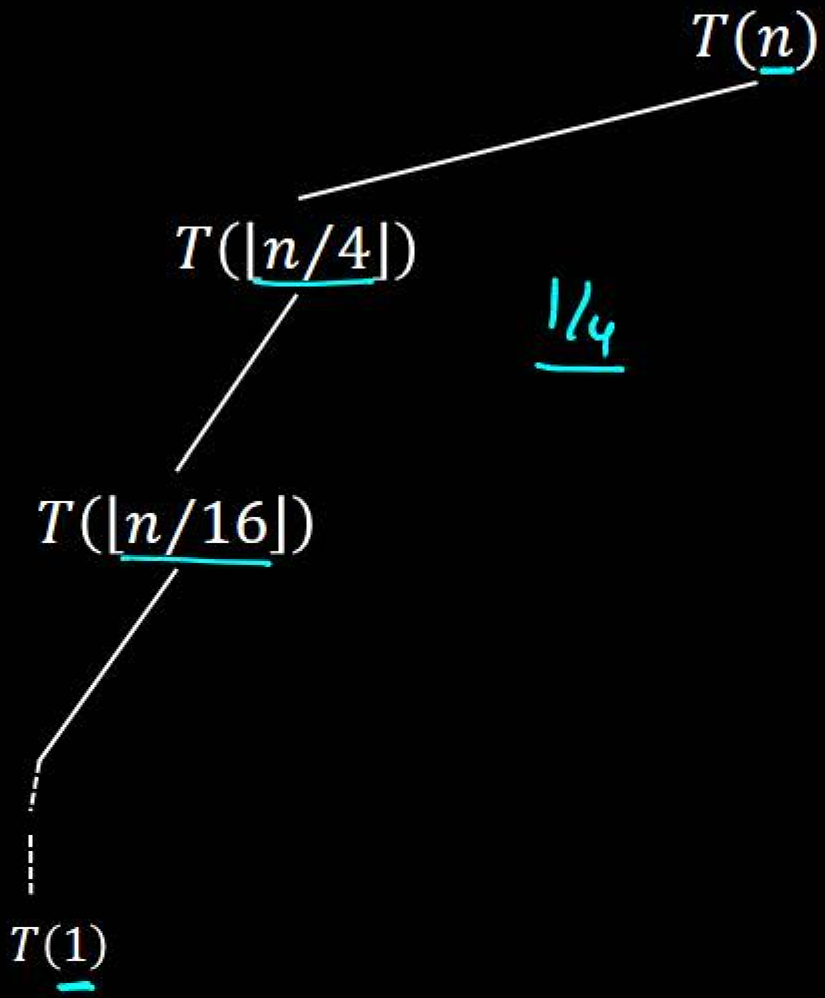
$\underline{T(\lfloor n/16 \rfloor)}$ $\underline{T(\lfloor n/16 \rfloor)}$ $\underline{T(\lfloor n/16 \rfloor)}$ $T(\lfloor n/16 \rfloor)$ $T(\lfloor n/16 \rfloor)$ $T(\lfloor n/16 \rfloor)$ $T(\lfloor n/16 \rfloor)$ $T(\lfloor n/16 \rfloor)$ $T(\lfloor n/16 \rfloor)$ →

$T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$ $T(1)$

$n^{\log_4 3}$

$3^{\log_4 n}$ $\boxed{n^{\log_4 3}}$ $\Theta(n^{\log_4 3})$

Height of the ~~Recursion~~ Recurrence Tree



$T(n)$

$T(\lfloor n/4 \rfloor)$

$T(\lfloor n/16 \rfloor)$

$T(1)$

$1/4$

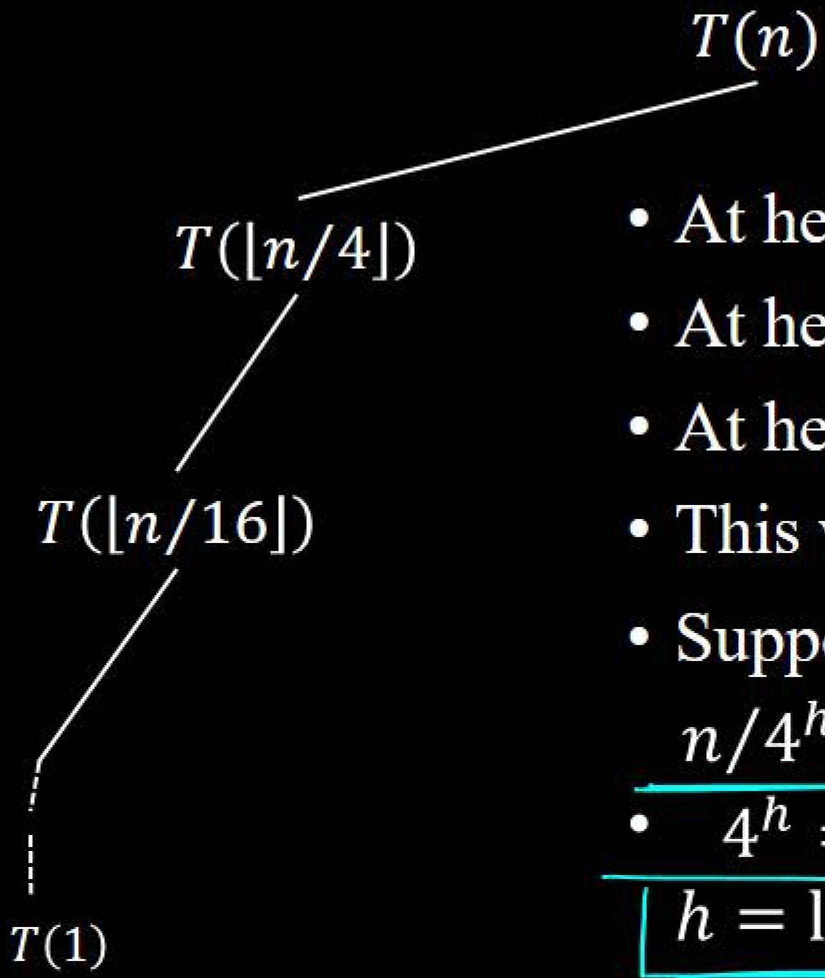
till — $T(n/4^k)$

$$\frac{n}{4^k} = 1$$

$$n = 4^k$$

$$k = \log_4 n$$

Height of the Recursion Tree



$T(n)$

$T(\lfloor n/4 \rfloor)$

$T(\lfloor n/16 \rfloor)$

$T(1)$

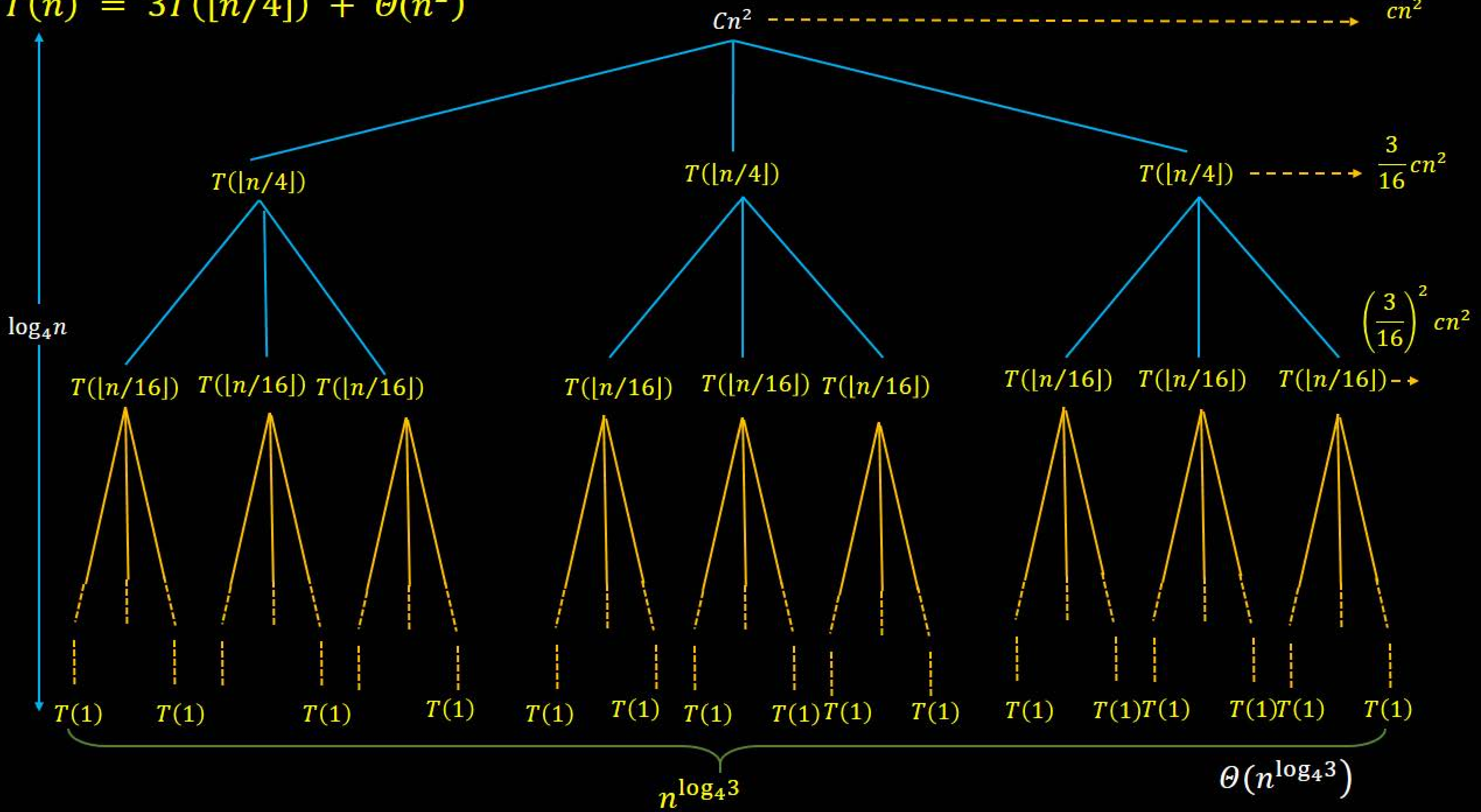
- At height $h = 0$, term $\underline{n}$
- At height $h = 1$, term $\underline{n/4}$
- At height $h = 2$, term $\underline{n/16}$
- This will continue till the term becomes 1.
- Suppose at height h, $T(n/4^h) = T(1)$

$$\underline{n/4^h = 1}$$

- $4^h = n$

$$\boxed{h = \log_4 n}$$

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$



Cn^2
cn^2
$T(\lfloor n/4 \rfloor)$
$T(\lfloor n/4 \rfloor)$
$T(\lfloor n/4 \rfloor)$
$\frac{3}{16}cn^2$
$\left(\frac{3}{16}\right)^2 cn^2$
$\log_4 n$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(\lfloor n/16 \rfloor)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$T(1)$
$n^{\log_4 3}$
$\Theta(n^{\log_4 3})$

Number of Leaves

$3T(n/4)$

height – 0 – 1

height 1 – 3

height 2 – 3^2

height h – $\underline{3^h} \Rightarrow 3^{\log_4 n} \Rightarrow \underline{n^{\log_4 3}}$

Number of Leaves

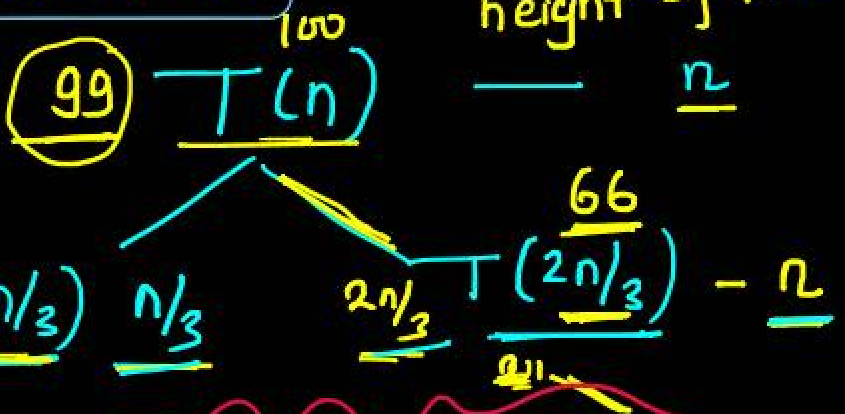
- At height $h = 0$, number of nodes $\underline{1}$
- At height $h = 1$, number of nodes $\underline{3}$
- At height $h = 2$, number of nodes $\underline{3}^2$
- At height $\underline{h} = \underline{\log_4 n}$, number of nodes $\underline{3^{\log_4 n}} \leftarrow$

- $3^{\log_4 n} = \underline{n^{\log_4 3}}$

Asymmetric Recurrence Relation

which side will decide height of tree

$T(n) = 3T(n/4) + n^2$ ← Master Method - 3



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

$T(n/3) = T(n/9) + T(2n/9) + n/3$

$T(2n/n) = T(2n/9) + T\left(\frac{4n}{9}\right) + 2n/3$

Imp *

\$ the side which Reduce n value user $T(an/b)$

Height $\log_{b/a} n$

$n \quad 2n/3 - 4n/9$

$\left(\frac{2}{3}\right)^0 n + \left(\frac{2}{3}\right)^1 n + \left(\frac{2}{3}\right)^2 n$

$\left(\frac{2}{3}\right)^k n = 1$

$n = \left(\frac{3}{2}\right)^k \quad k = \log_{3/2} n$

Asymmetric Recurrence Relation

$T(n) = T(n/4) + T(3n/4) + n$

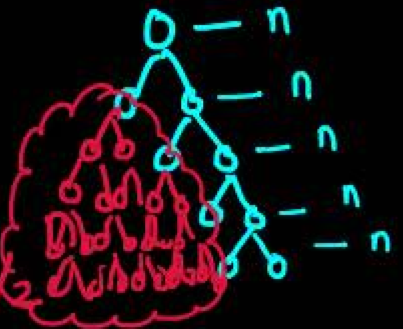
$T(n) = T(n/3) + T(2n/3) + n$

$n/9 + 2\cdot\frac{2n}{9} + \frac{4n}{9} = \frac{9n}{9} = \underline{n}$

$T(n) = T(n/4) + T(3n/4) + \underline{n}$ ✓

will the cost will same at every height

No. of

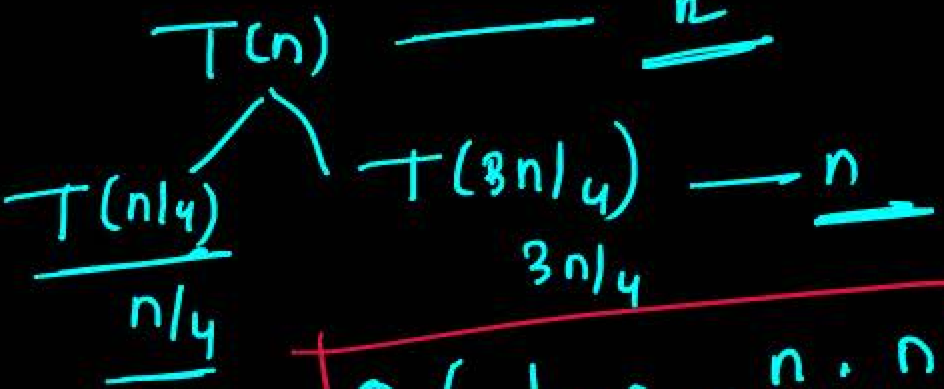


Height — $\log_{3/2}$

Cost at every level is still $\underline{n}$

Complexity is $\underline{\log_{3/2} n} \cdot \underline{n}$

$= \underline{\Theta(n\log n)}$



$\boxed{\Theta(\log_{4/3} n \cdot \underline{n})}$

GATE 2021 Set-I| 2 Mark Question

Consider the following recurrence relation

*

$$T(n) = \begin{cases} \underline{T(n/2)} + \underline{T(2\,n/5)} + \boxed{7n} & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

* only this

Which one of the following options is correct?

(a) $T(n) = \Theta\left(n^{5/2}\right)$

(b) $T(n) = \Theta(n\log n)$ ✓

[✓ (c) $T(n) = \Theta(n)$]

(d) $T(n) = \Theta\left((\log n)^{5/2}\right)$

(1) cost at first 3 level heign — 7n

h=1

h=1 —

h=2 —

(2) Height of the tree

$n.\log_{3/2} n$

(I) $T(n/3) + T(2n/3) + n$ —

$n/3$ $\quad$ $2n/3$

(II) $T(n/4) + T(3n/4) + n$ —

$n.\log_{4/3} n$

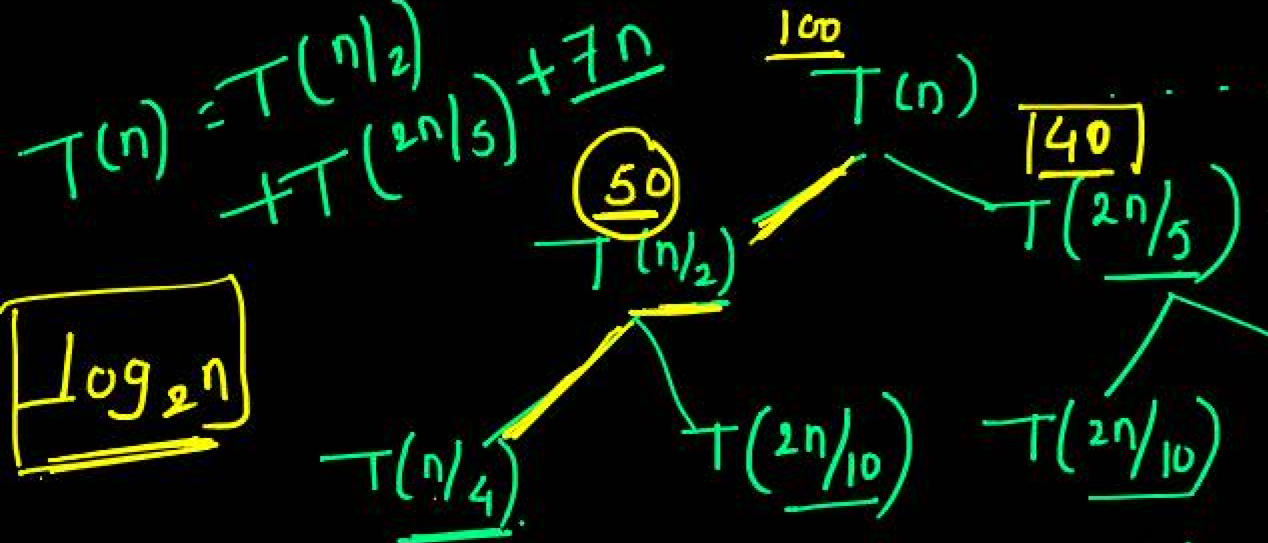
cost at every level same

cost at every level same

$T(n) = T(n/2) + T(2n/5) + 7n$

Series — Summation of series

$T(n) = T(n/2) + T(2n/5) + 7n$



$\frac{7n}{2} + 7 \times \frac{2n}{5} = 7n\left(\frac{1}{2} + \frac{2}{5}\right)$

$= 7n\left(\frac{9}{10}\right)$

$7n\left(\frac{9}{10}\right)^2$

$7\frac{n}{4} + 7\cdot\frac{2n}{10} + 7\cdot\frac{2n}{10} + 7\cdot 4n/25 =$

$7n\left(\frac{25+20+20+16}{100}\right) = 7n\left(\frac{9}{10}\right)^2$

$$7n + 7n\left(\frac{9}{10}\right) + 7(n)\left(\frac{9}{10}\right)^2 + \cdots + \left(\frac{9}{10}\right)^h \cdot 7n$$

$$7n\left(1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \cdots \left(\frac{9}{10}\right)^h\right)$$

GP

$r < 1$

Infinitesum

$\frac{a}{1-r}$

$$7n\left(\frac{1}{1 - 9/10}\right) = 70n < \Theta(n)$$

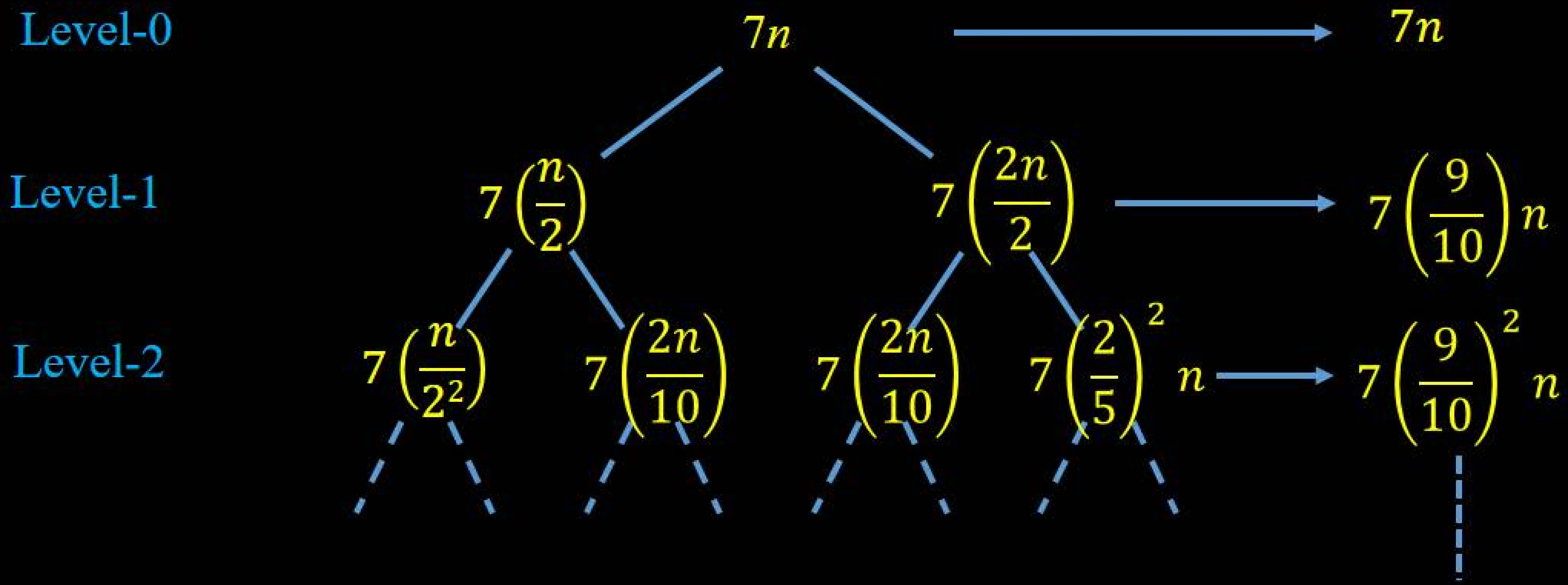
$$\underline{7n} + \underline{7n\left(\frac{9}{10}\right)} + 7n\left(\frac{9}{10}\right)^2 + \cdots + 7n\left(\frac{9}{10}\right)^h$$

$$\underline{7n}\left(1+\frac{9}{10}+\left(\frac{9}{10}\right)^2+\cdots\left(\frac{9}{10}\right)^h\right)$$

$$\frac{7n\left(1-\left(9/10\right)^h\right)}{1-9/10}$$

$$\underline{70n}\left(1-\frac{9^{\log_2 n}}{10^{\log_2 n}}\right) = \underline{70n}\left(1-\frac{n^{\log_2 9}}{n^{\log_2 10}}\right.$$

The Recurrence Tree


Level-0
$7n$
$7n$
Level-1
$7\left(\frac{n}{2}\right)$
$7\left(\frac{2n}{2}\right)$
$7\left(\frac{9}{10}\right)n$
Level-2
$7\left(\frac{n}{2^2}\right)$
$7\left(\frac{2n}{10}\right)$
$7\left(\frac{2n}{10}\right)$
$7\left(\frac{2}{5}\right)^2 n$
$7\left(\frac{9}{10}\right)^2 n$