Exp. 2A

1.) Quick Sort

arr = [8, 4, 1, 6, 9, 3]

Initial call : quick_sort(arr, 0, 5)

Exec. of partition (arr, 0, 5):

pivot = arr[0] = 8

i = low + 1 = 0 + 1 = 1

For j = 1, arr[1] = 4 < 8 ⟹ swap(arr[i], arr[j])

∵ i = j ⟹ no change; i++ ⟹ i = 2

For j = 2, arr[2] = 1 < 8 ⟹ swap; i = j ⟹ no change, i++

For j = 3, arr[3] = 6 < 8 ⟹ " " " "

For j = 4, arr[4] = 9 > 8 ⟹ no swap ⟹ i remains 4.

For j = 5, arr[5] = 3 < 8 ⟹ swap(arr[4], arr[5])

⟹ arr = [8, 4, 1, 6, 3, 9] ⟹ i = **5**.

Final swap ⟹ swap (arr[low = 0], arr[i-1=4])

⟹ arr = [3, 4, 1, 6, 8, 9] ⟹ return i - 1 = 4

new pivot

Recursive Calls: quick_sort(arr, 0, 3)

quick_sort(arr, 5, 5)

right subarray [9], single element implies it's already sorted.

exec. of quick_sort (arr, 0, 4):

partition(arr, 0, 5):

pivot = arr[0] = 3

i = 1

For j = 1, arr[1] = 4 > 3 ⟹ no swap; i remains 1.

For j = 2, arr[2] = 1 < 3 ⟹ swap(arr[**1**], arr[2])

⟹ arr = [3, 1, 4, 6, 8, 9] ⟹ i++ ⟹ i = 2

For j = 3, arr[3] = 6 > 3 ⟹ no swap, i remains 2

Final swap: swap(arr[0], arr[i-1])

$\Rightarrow$ arr = [1, 3, 4, 6, 8, 9] $\Rightarrow$ return pivot_index = 1

Recursive calls: qs(arr, 0, 0) $\Rightarrow$ single element; base case

qs(arr, 2, 3) $\Rightarrow$ right subarray [4, 6]

exec. of qs(arr, 2, 3):

partition(arr, 2, 3):

pivot = 4

i = 3

For j = 3: arr[3] = 6 > 4 $\Rightarrow$ no swap $\Rightarrow$ i remains 3

Final swap: (arr[2], arr[i - 1]) $\Rightarrow$ no change

return pivot_index = 2

recursive calls: qs(arr, 2, 1) $\Rightarrow$ empty; base case

qs(arr, 3, 3) $\Rightarrow$ single el; " "

$\therefore$ Final sorted array = [1, 3, 4, 6, 8, 9]

2.) Merge Sort

arr = [8, 4, 1, 6, 9, 3]

initial call: m_s(arr, 0, 5)

mid = (0+5)/2 = 2

m_s(arr, 0, 2) ⟹ [8, 4, 1]

m_s(arr, 3, 5) ⟹ [6, 9, 3]

[8], [4, 1]

m_s(arr, 0, 1)

m_s(arr, 2, 2)

[6], [9, 3]

m_s(arr, 3, 4)

m_s(arr, 4, 4)

[8]

[4], [1]

[9], [3]

[6]

1 < 4

⟹ [1, 4]

Merge: 3 < 9

∴ [3, 9]

1 < 8 ⟹ advance right ptr; [1]

4 < 8 ⟹ [1, 4]; advance right ptr.

8 ⟹ [1, 4, 8]; advance left ptr.

Merge

3 < 6 ⟹ [3]; advance left ptr.

6 < 9 ⟹ [3, 6]; " right ptr.

9 ⟹ [3, 6, 9]; " left ptr.

$1 < 3 \Longrightarrow [1]$, adv. left ptr.

$3 < 4 \Longrightarrow [1, 3]$, " right "

$4 < 6 \Longrightarrow [1, 3, 4]$, " left ptr.

$6 < 8 \Longrightarrow [1, 3, 4, 6, \cancel{8}]$, " right "

$8 < 9 \Longrightarrow [1, 3, 4, 6, 8]$, " left "

$9 \Longrightarrow [1, 3, 4, 6, 8, 9]$

* Merge Sort

Divide :

4, 2, 1, 6, 7, 5, 9, 8

4, 2, 1, 6   7, 5, 9, 8

4, 2   1, 6   7, 5   9, 8

Conquer & Merge :

4   2   1   6   7   5   9   8

2, 4   1, 6   5, 7   8, 9

1, 2, 4, 6   5, 7, 8, 9

1, 2, 4, 5, 6, 7, 8, 9

* Quick Sort

7, 6, 10, 5, 9, 2, 1, 15, 7

⇓

2, 6, 1, 5, 7, $\boxed{7}$, 10, 9, 15

2, 6, 1, 5, 7   10, 9, 15

⇓   ⇓

1, $\boxed{2}$, 6, 5, 7   9, $\boxed{10}$, 15

$\boxed{1}$ 6, 5, 7 $\boxed{9}$ $\boxed{15}$

⇓

5, $\boxed{6}$, 7

$\boxed{5}$ $\boxed{7}$

- **Master's Method for solving recurrences.**

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ where $a \geq 1$, $b > 1$ & $f(n) > 0$

Idea: Compare $f(n)$ with $n^{\log_b(a)}$

Case 1: If $f(n) = O\left(n^{(\log_b a) - \varepsilon}\right)$ for some $\varepsilon > 0$
then: $T(n) = \Theta\left(n^{\log_b a}\right)$

Case 2: If $f(n) = \Theta\left(n^{\log_b a}\right)$ then $T(n) = \Theta\left(n^{\log_b a} \cdot \lg n\right)$

Case 3: If $f(n) = \Omega\left(n^{(\log_b a) + \varepsilon}\right)$ for some $\varepsilon > 0$ & if
$af(n/b) \leq cf(n)$ for some $c < 1$ & all
sufficiently large n then $T(n) = \Theta(f(n))$

eg. $T(n) = 2T(n/2) + n$
$a = 2$, $b = 2$, $\log_2 2 = 1$
$f(n) = n$
$n^{\log_b a} = n^1 = n$
$\Rightarrow$ Case 2: $f(n) = \Theta(n)$
$\therefore$ $T(n) = \Theta(n \lg n)$