SP 2024

HOMEWORK - 2A

CS 2500

Prob. 1

Given $A = \{1, 5, 2, 3, 0, 2, 2, 1, 4, 5\}$.

let the repeated elements be given a subscript based on the order of their occurrence.

In other words, let

$$A = \{1_a, 5_a, 2_a, 3_a, 0_a, 2_b, 2_c, 1_b, 4_a, 5_b\}$$

Insertion Sort (A):

Iteration 1 :

1_a	5_a	2_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b
	j								

Iteration 2 :

1_a	5_a	2_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b
		j							

Iteration 3 :

1_a	2_a	5_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b
			j						

Iteration 4 :

1_a	2_a	3_a	5_a	0_a	2_b	2_c	1_b	4_a	5_b
				j					

Iteration 5 :

0_a	1_a	2_a	3_a	5_a	2_b	2_c	1_b	4_a	5_b
					j				

Iteration 6 :

0_a	1_a	2_a	2_b	3_a	5_a	2_c	1_b	4_a	5_b
						j			

Iteration 7 :

0_a	1_a	2_a	2_b	2_c	3_a	5_a	1_b	4_a	5_b
							j		

Iteration 8 :

0_a	1_a	1_b	2_a	2_b	2_c	3_a	5_a	4_a	5_b
								j	

Iteration 9 :

0_a	1_a	1_b	2_a	2_b	2_c	3_a	4_a	5_a	5_b
									j

Note that Insertion Sort (A) returns a Stable output.

Merge Sort (A) :

Iteration 1 :

1_a	5_a	2_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b
1	2	3	4	5	6	7	8	9	10

Stack : { Merge Sort (A[1:5]), Merge Sort (A[6:10]), Merge (A[1:5], A[6:10]) }

"top"

P.T.O.

#3

Iteration 2 :

1	2	3	4	5	6	7	8	9	10
1_a	5_a	2_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b

Stack : { Merge Sort (A[1:2]), Merge Sort (A[3:5]),
Merge (A[1:2], A[3:5]),
Merge Sort (A[6:10]), Merge (A[1:5], A[6:10]) }

Iteration 3 :

1	2	3	4	5	6	7	8	9	10
1_a	5_a	2_a	3_a	0_a	2_b	2_c	1_b	4_a	5_b

sorted

Stack : { Merge Sort (A[3]), Merge Sort (A[4:5]),
Merge (A[3], A[4:5]), Merge (A[1:2], A[3:5]),
Merge Sort (A[6:10]), Merge (A[1:5], A[6:10]) }

Iteration 4 :

1	2	3	4	5	6	7	8	9	10
1_a	5_a	2_a	0_a	3_a	2_b	2_c	1_b	4_a	5_b

sorted sorted sorted

Stack : { Merge (A[3], A[4:5]), Merge (A[1:2], A[3:5]),
Merge Sort (A[6:10]),
Merge (A[1:5], A[6:10]) }

#4

Iteration 5:

1	2	3	4	5	6	7	8	9	10
1_a	5_a	0_a	2_a	3_a	2_b	2_c	1_b	4_a	5_b
sorted		sorted							

Stack: { Merge (A[1:2], A[3:5]),
Merge Sort (A[6:10]),
Merge (A[1:5], A[6:10]) }

Iteration 6:

1	2	3	4	5	6	7	8	9	10
0_a	1_a	2_a	3_a	5_a	2_b	2_c	1_b	4_a	5_b
sorted.									

Stack: { Merge Sort (A[6:10]),
Merge (A[1:5], A[6:10]) }

Iteration 7:

1	2	3	4	5	6	7	8	9	10
0_a	1_a	2_a	3_a	5_a	2_b	2_c	1_b	4_a	5_b
sorted									

Stack: { Merge Sort (A[6:7]), Merge Sort (A[8:10]),
Merge (A[6:7], ~~Merge~~ A[8:10]),
Merge (A[1:5], A[6:10]) }

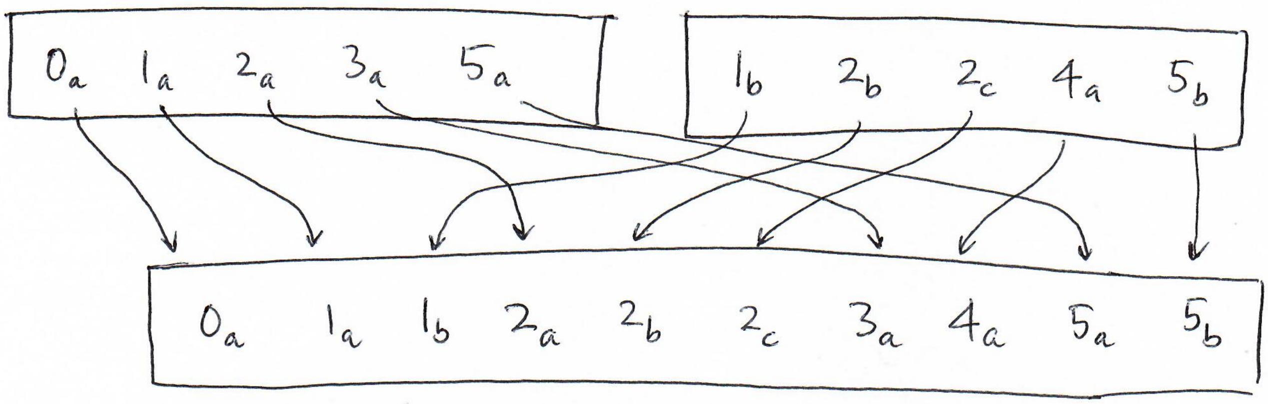
P.T.O.

#5

Iteration 8 :

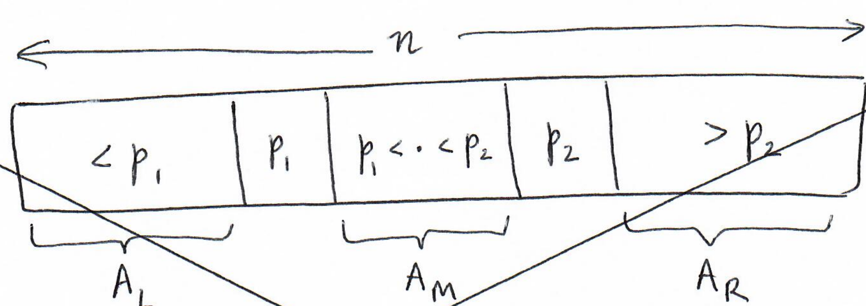
1	2	3	4	5	6	7	8	9	10
0_a	1_a	2_a	3_a	5_a	1_b	2_b	2_c	4_a	5_b
sorted					sorted				

Stack : { Merge (A[1:5], A[6:10]) }



Note that MergeSort (A) also returns a stable outcome.

Prob. 2



~~Note that, if~~ $\cancel{|A| = n}$,

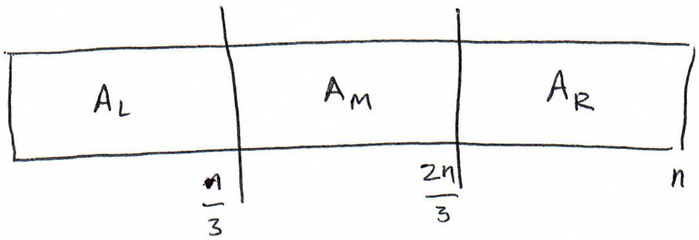
$$\cancel{|A_L| = |A_M| = |A_R| = \frac{n-2}{3}}.$$

~~Running QuickSort-3way (A_L), QuickSort-3way (A_M)~~

P.T.O.

#6

Prob. 2



Note that, if $|A| = n$,

$$|A_L| = |A_M| = |A_R| = \frac{n}{3}.$$

Running MergeSort-3way(A_L), MergeSort-3way(A_M) and MergeSort-3way(A_R), and merging them as $C = \text{Merge}(A_L, A_M)$, returning Merge$(C, A_R)$,

we have

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \underbrace{O(n)}$$

due to running Merge function twice.

$\therefore$ By Master method,

Since $a = 3$, $b = 3$ and $d = 1$,

we are under case-1 $(a = b^d = 3)$

i.e. $T(n) = O(n^d \log n) = \underline{\underline{O(n \log n)}} \Rightarrow$ Ans:

$\Rightarrow$ No gain by further splitting the array.