Assignment No:-8

Q.1 Apply following sorting algorithms on below input list.
$5^*, 17, 3^*, 9, 5\#, 3\#, 10, 12$

1. Bubble sort 2. selection sort 3. Insertion sort 4. Quick sort 5. shell sort 6. Radixsort 7. Bucket sort 8. counting sort

Comment on time analysis at the end.

Ans→ 1. Bubble sort
$5^*, 17, 3^*, 9, 5\#, 3\#, 10, 12$

pass 0	5^*	17	3^*	9	$5\#$	$3\#$	10	12	unsorted.
	0	1	2	3	4	5	6	7	
pass 1	5^*	17	3^*	9	$5\#$	$3\#$	10	12	$5^* < 17$ don't swap
pass 2	5^*	17	3^*	9	$5\#$	$3\#$	10	12	$17 > 3^*$ swap
Pass 3	5^*	3^*	17	9	$5\#$	$3\#$	10	12	$17 > 9$ swap
Pass 4	5^*	3^*	9	17	$5\#$	$3\#$	10	12	$17 > 5\#$ swap
Pass 5	5^*	3^*	9	$5\#$	17	$3\#$	10	12	$17 > 3\#$ swap

pass 6	5*	3*	9	5#	3#	17	10	12	17>10 swap
pass 7	5*	3*	9	5#	3#	10	17	12	17>12 swap
pass 8	5*	3*	9	5#	3#	10	12	17	5*>3* swap
pass 9	3*	5*	9	5#	3#	10	12	17	5<9 don't swap 5*>5# swap
pass 10	3*	5*	9	5#	3#	10	12	17	9>5# swap
pass 11	3*	5*	5#	9	3#	10	12	17	9>3# swap
pass 12	3*	5*	5#	3#	9	10	12	17	5#>3# swap
pass 13	5*	5*	3#	5#	9	10	12	17	5*>3# swap
pass 14	3*	3#	5*	5#	9	10	12	17	sorted.

Time analysis

Worst case: $O(n^2)$
Best case: $O(n)$
Average case: $O(n^2)$

2. selection sort

Right side element $J = i+1$ to $n-1$ $n=6$

5#	17	3*	9	5#	3#	10	12	unsorted
0	1	2	3	4	5	6	7	

Select each element & compare with it's all right side element find smallest of all & swap with current element.

Pass 1:	3#	17	3*	9	5#	5*	10	12
	0	1	2	3	4	5	6	7
pass 2:	3#	3*	17	9	5#	5*	10	12
pass 3:	3#	3*	5#	9	17	5*	10	12
pass 4:	3#	3*	5#	5*	17	9	10	12
Pass 5:	3#	3*	5#	5*	9	17	10	12
pass 6:	3#	3*	5#	5*	9	10	17	12
Pass 7:	3#	3*	5#	5*	9	10	12	17

sorted.

Time analysis: worst case: $O(n^2)$

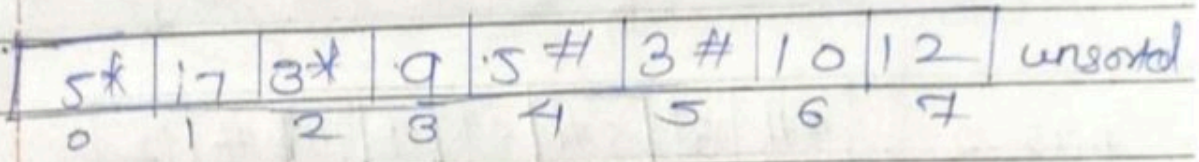
Best case: $O(n^2)$

Average case: $O(n^2)$

Time Analysis

Worst case : $O(n^2)$
Best case : $O(n)$
Average Case : $O(n^2)$

4. Quick Sort.



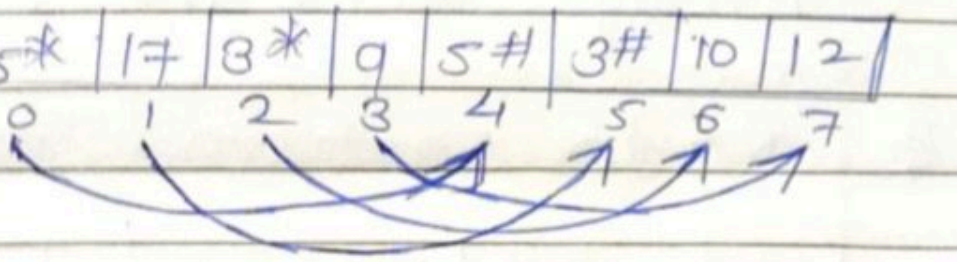
- these are two indices i & j & at the very beginning of the partition algorithm i points to the first element in the array & j points to the last one.
- Then i moves forward, until an element with value greater or equal to me pivot is found.
- Index j is moved backward until an element with value lesses or equal to the pivot is found. If i ≤ j then they swapped & i steps to the next position (i+1) is steps to the previous one (j-1).
- It stops when i becomes greater than j.
- After position, all values before ith element are less or equal than the pivot

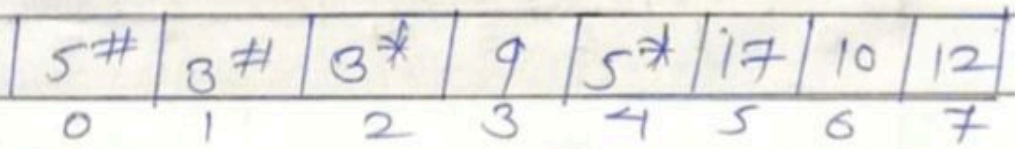
element | gap = $\frac{n}{2} = \frac{8}{2} = 4$ |

[0,4], [1,5], [2,6], [3,7]

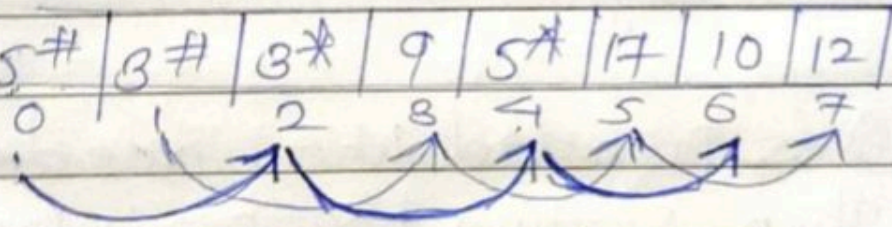
i.e element of subtists are:-

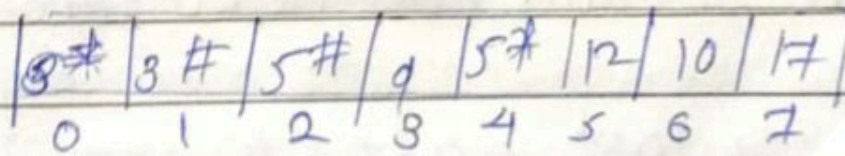
[5*, 5#], [17, 3#], [3*, 10], [9, 12]





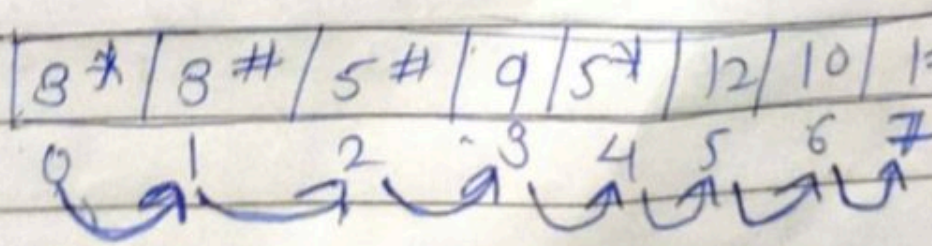
ss 2. gap = $\frac{n}{4} = \frac{8}{4} = 2$





7 / 11

ss 3 gap = $\frac{n}{8} = \frac{8}{8} = 1$



8#	3*	5#	5*	9	10	12	17	sorted

Time analysis

Worst case : $O(n^2)$
Best case : $O(n \log n)$
Average case : $O(n^2)$.

Radix sort

5*	17	3*	9	5#	3#	10	12
0	1	2	3	4	5	6	7

- starting from the rightmost (last) digit, sort the numbers based on that digit.
- 0 is prended when needed.
- then sorting the next left digit

pass 1: 05*, 17, 03*, 09, 05#, 03#, 10, 12

~~10~~	12	03*	05*	05#	17	09
0	1	2	4	5	6	~~7~~

10	12	3*	3#	5*	5#	17	09
0	1	2	3	4	5	6	7

Pass 2: 10, 12, 03*, 03#, 05*, 05#, 17, 09

03*	03#	05*	05#	09	10	12	17
0	1	2	3	4	5	6	7

3*	3#	5*	5#	9	10	12	17	sorted
0	1	2	3	4	5	6	7	

Time analysis:
Worst case: O(n+k)
Best case: O(n+k)
Average case: O(n+k)

7 Bucket sort

5*	17	3*	9	5#	3#	10	12	unsorted

- Set up an array of initially empty "buckets".
- Go over the original array, putting each object in its bucket.
- Visit the buckets in order & put all elements back into the original array.

Pass 1:

5*, 3*, 5#, 3#	9, 10, 12	17
0 - 7	8 - 14	15 - 21

Arrange:

$3^*, 3^\#, 5^*, 5^\#$	9, 10, 12	17
0 - 7	8 - 14	15 - 21

3^*	$3^\#$	5^*	$5^\#$	9	10	12	17
0	1	2	3	4	5	6	7

Time analysis:
Worst case:- $O(n^2)$
Best case: $O(n)$
Average case: $O(n)$

8. Counting sort:

5^*	17	3^*	9	$5^\#$	$3^\#$	10	12

unsorted

- Take a count array to store the count.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
count	0	0	0	2	0	2	0	0	0	1	1	0	1	0	0

- Modify the count array such that each element at each index store the sum of previous counts.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0	0	0	2	2	4	4	4	4	5	6	6	7	7	7	7	7