Bahria University
Karachi Campus

Bahria University
Discovering Knowledge

COURSE: DATA STRUCTURES AND ALGORITHMS
TERM: SPRING 2020, CLASS: BCE- 4(A)

Submitted By:

NAME: **QASIM HASSAN** REG NO: **57485**

Assignment: 02

Submitted To:

Engr. Sidra Mudassar

Signed **Remarks:** **Score**_____

Question no 1*:*

Explain the following sorting algorithms with example.

a) Quick Sort

b) Radix Sort.

Answer:

a) QUICK SORT:

Quick Sort method is an efficient sorting method for larger List. It works fine for the list having large number of elements. It uses Divide and conquers strategy in which a list is divided into two smaller lists.

Procedure:

First initialize LOW with index of the first element and HIGH with index of last element.
Now find the pivot element from the list of elements using the equation:
PIVOT = (LOW + HIGH) / 2
Now scan elements from left to right and compare each element with PIVOT element. If the scanned element is less than the PIVOT element, we scan next element and increment the value of LOW. Repeat same procedure until we found the element which is greater than the PIVOT element and vice versa.

Compare the value of LOW and HIGH. If LOW is less then HIGH then we interchange the element which are at the index LOW and HIGH.
Increment the value of LOW and decrement the value of HIGH. Repeat above procedure while value of LOW less than or equal to value of HIGH.
After the completion of first PASS the entire list of elements is divided in to two lists. First list contains elements which are less than the PIVOT element and second list contains elements which are greater than the PIVOT element.

The above procedure is repeated until all the elements of the array are sorted. Its complexity is O (nlogn).

Example:

For example, the given array is:

Element	5	2	35	9	3	-2	52	0
Index	0	1	2	3	4	5	6	7

Select high and low values from it and mark the pivot.

Pass 1:

Low	High	Pivot
0	7	(0+7)/2=3.5=3

Index	0	1	2	3	4	5	6	7
	5	2	35	9p	3	-2	52	0
	5	2	0	9p	3	-2	52	35
	5	2	0	-2	3	**9p**	52	35
	List1						List2	

In the above image, there are sub-lists marked of the elements that are less than the pivot and greater than the pivot (List1 & List2) respectively.

Repeat this process along with the passes until all elements are sorted.

Pass 2:

Low	High	Pivot
0	4	(0+4)/2=2

	List 1						List 2	
Index	0	1	2	3	4	5	6	7
	5	2	0p	-2	3	9	52	35
	-2	2	0p	5	3	9	52	35
	-2	0p	2	5	3	9	52	35
	List1.1		List1.2				List 2	

Pass 3:

Low	High	Pivot
2	4	(2+4)/2=6/2=3

	List1.1		List 1.2				List 2	
Index	0	1	2	3	4	5	6	7
	-2	0	2	5p	3	9	52	35
	-2	0	2	3	5p	9	52	35

Pass 4:

Low	High	Pivot
6	7	(6+7)/2=13/2=6.5=6

	List1.1		List 1.2				List 2	
Index	0	1	2	3	4	5	6	7
	-2	0	2	3	5	9	52p	35
	-2	0	2	3	5	9	35	52p

The above image shows that all the elements are sorted now.

b) RADIX SORT:

Radix Sort is an example of External Sort.
The number of pass required to sort element in Radix sort depends on the number of digits in the Maximum number among list. If the list having a maximum number with 4 digits then 4 passes have to be performed.
In Radix Sort technique we use ten buckets for the digits 0 to 9.
During First PASS we scan each element from the list and separate its unit digit and according to its unit digit the number is placed in appropriate bucket. After the completion of first pass we arrange the numbers from bucket 0 to 9.
During second PASS, scan each element from the newly arranged list and separate next higher digit and according to that digit the number is placed in appropriate bucket. After the completion of second PASS, arrange the numbers from bucket 0 to 9. This continues until all elements are sorted.

Example:

Consider the given array:

42 23 74 11 65 57 94 36 99 87 70 81 61

In the above set of data, the maximum number is 99 which is of 2 digits, so we have to perform two passes.

In the first pass, separate the unit digit of the number and place the number in to appropriate bucket according to its unit digit.

1st Pass:

		61							
		81			94			87		
	70	11	42	23	74	65	36	57		99
Pocket	0	1	2	3	4	5	6	7	8	9

The updated array is:

70 11 81 61 42 23 74 94 65 36 57 87 99

Now separate the hundredth digit of the number and place the number in to appropriate bucket according to its hundredth digit.

2nd Pass:

							65	74	87	99
		11	23	36	42	57	61	70	81	94
Pocket	0	1	2	3	4	5	6	7	8	9

Above image shows that all elements of the given array are sorted.