time for → dif values g n.–

//static → need not create objects
(available to all the objects.

(Either we can create one more class, Explain Quicksort & partition in the other class & create objects & create the method.)

P4) A) Sort the given set of n integer elements using quick sort method and compute its time complexity. Run the program for varied values of n > 5000 & record the time taken to sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read from a file & can be generated using the random number generator. Demonstrate using ~~the~~ Java how the divide & conquer method works along with its time complexity analysis : worst, avg, best cases.

```
import java.util.Random;   for generating no.'s randomly
import java.util.Scanner;

public class quick {
 static int a[] = new int[100000];
public static void main(String[] args) throws
ArrayIndexOutOfBoundsException
{
Scanner in = new Scanner(System.in);
long start, end;
System.out.println("-- Quick Sort --");
System.out.println("Enter the number of
        Elements to be sorted");
```

//nanoTime → func.
vs
returnTime.

```
int n = in.nextInt();
Random rand = new Random();
for (int i=0;i<n; i++)
    a[i]=rand.nextInt(100);
System.out.println("Array Elements to be sorted:");
for (int i=0; i<n; i++)
  System.out.println(a[i]+" ");
  int low=0 , high=n-1;
  a[n]=999;
  start = System.nanoTime();
  int m= partition (a,low,high)
  quicksort (a,low,high);
  end = System.nanoTime();
  System.out.println("The sorted array is:");
  for (int i=0;i<n;i++)
  System.out.println(a[i]+" ");
System.out.println("\n The time taken to sort
    is: "+(end-start)+"ns");
System.out.println("***\n\n");
}
static void quicksort (int a[], int low, high)
{
    int mid;
    if (low<high)
    {
        mid = partition (a,low,high);
        quicksort (a,low, mid-1);
        quicksort (a, mid+1,high);
    }
}
static int partition( int a[], int low, int high)
{
    int pivot=a[low])
    int i = low+1;
    int j = high;
    while(i <=j)
   {
    while (a[i] <=pivot)
     {
        i++;
     }
    while(a[j] >pivot)
     {
        j--;
     }
   if (i<j)
   {
     swap(a, i, j)
   }
}
```

```
swap(a, low, j);
return j;

}
static   void swap (int a[], int i, int j)
{
    int temp;
    temp = a[i];
    a[i] = a[j];
    a[j] = temp;
  }
}
```

<u>output:</u>

```
--Quick Sort --
Enter the number of Elements to be sorted:
9
Array Elements to be sorted are:
55
10
77
12
78
16
17
```

Date
Page

38
14
The sorted array is:
10
12
14
16
17
38
55
77
78
the time taken to sort the array is: 15100ns
***

	0	1	2	3	4	5	6	7	8	
Array:	55	10	77	12	78	16	17	38	14	999
	↑pivot	↑i							↑j	

i <= j ⇒ true.

1) a[i] <= 55
   10 ≤ 55
   i++

2) a[2] <= 55
   77 <= 55 → false
   a[j] > ~~55~~ 55
   14 > 55 → false

i = 2 & j = 8
  2 < 8
swap a[i] & a[j]

	0	1	2	3	4	5	6	7	8	
array ⇒	55	10	14	12	78	16	17	38	77	99
	↑p		i						j	

i <= j ⇒ 2 <= 8 ⇒ true

3) a[i] <= 55   4) a[3] <= 55   5) a[4] <= 55
14 <= 55   12 <= 55   78 <= 55
i++   i++   ↳ false
$\boxed{i=4}$

⇒ a[j] > 55   6) a[7] > 55
77 > 55   38 > 55
j--   ↳ false.
i = 4 j = 7

→ i < j ⇒ 4 < 7
swap a[j] & a[i]

	0	1	2	3	4	5	6	7	8	
array ⇒	55	10	14	12	38	16	17	78	77	999
	p↑				i			j		

i <= j ⇒ 4 <= 7 → true

7) a[4] <= 55   8) a[5] <= 55   9) a[6] <= 55
38 <= 55   16 <= 55   17 <= 55
i++   i++   i++

10) a[7] <= 55
78 <= 55
↳ false.
i = 7
i < j
7 < 6
return

→ a[j] > 55
78 > 55
j--
j = 6

a[6] > 55
17 > 55
↳ false.

swap pivot / low & a[j] , return j

	0	1	2	3	4	5	6	7	8	
array ⇒	17	10	14	12	38	16	55	78	77	999
	l					mid(	j ↑ i)		h	

l<h

~~*A~~ quicksort(a, l, mid-1) *B qs (a, mid+1, h)

0	1	2	3	4	5
17	10	14	12	38	16
p↑	i↑				j

i <= j ⇒ 1<=5 ✓

1) a[i] ≤ 17   2) a[2] ≤ 17   3) a[3] ≤ 17
10 ≤ 17   14 ≤ 17   12 ≤ 17
i++   i++   i++

4) a[4] ≤ 17   i=4
38 ≤ 17 → false
a[j] > 17
16 > 17
↳ false   j = 5

→ i<j
4<5
swap a[i] & a[j]

	0	1	2	3	4	5
array ⇒	17	10	14	12	16	38
	p				i	j

5) a[i] ≤ 17   6) a[i] ≤ 17   a[j] > 17
16 ≤ 17   a[5] ≤ 17   38 > 17
i++   38 ≰ 17 $\boxed{i=5}$ j--

7) a[j] > 17   i < j
$16 \not> 17$ $5 \not< 4$
j = 4   or

swap pivot & a[j], return 4

* array ⇒ 16   10   14   12   17   38
l↑   mid↑   h↑
l<h

B*a quicksort (a, l, mid-1) | *B.b qs (a, mid+1, h)

0	1	2	3
16	10	14	12
p↑	i↑		j↑

999.   38
l↑ ↑h
l ≮ h.

1) a[i] ≤ 16   2) a[2] ≤ 16   3) a[3] ≤ 16
10 ≤ 16   14 ≤ 16   12 ≤ 16
i++   i++   i++

i <= j
a[j] > 16
$12 \not> 16$
i = 3   j = 3

→ $i \not< j$
Swap pivot & a[j]
return 3.

array ⇒ 12   10   14   16
l   mid (h)
l<h

Aa→a qs(a, l, mid-1)

0	1	2
12	10	14
p↑	i↑	j↑

$a[1] \le 12$

2) $10 \le 12$ ~~a)~~ $a[2] \le 12$ 3) $a[j] > 12$

i++ $14 \nleq 12$ $P = 2$ $10 \ngtr 12$

$a[j] > 12$

$14 > 12$ $P = 2$ & $j = 1$

j-- $j = 1$ $P < j$

swap a[j] & P , <u>return 1</u>

~~F~~c) array → 10   12   14

l $\overset{\uparrow}{mid}$ h

c.a) qs(a, l, mid-1)   c.b) qs(a, mid+1, h)

10   14

l↑ h↑   l↑ h↑

$l \nless h$ $l \nless h$.

B → 10   12   14   16   17   38   <u>55</u>

7 8

*B 78 77

p↑ mid↑ l↑ h↑. i↑ j↑

1) $a[i] \le 77$ 2) $i \nless j$

78 ≰ 78 swap $a[j]$ & pivot

$a[j] \ngtr 77$ 77 .

$77 \le 78$

$i+1$

$p =$

array 77 78.

final array:

10 12 14 16 17 38 55 77 78