Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n > 5000 and record the time taken to sort. plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator. Demonstrate how the divide and conquer method works along with its time complexity analysis: worst case, average case and best case.

ALGORITHM & CODE :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void swap(int arr[], int i, int j)
{ int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
}
int partition(int arr[], int low, int high)
{ int pivot = arr[high];
  int i = low - 1;
  for (int j = low; j < high; j++)
  { if (arr[j] < pivot)
    { i++;
      swap(arr, i, j);
    }
  }
  swap(arr, i+1, high);
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty / Date : — Signature of Student / Date :

	Subject :	Software :
		Hardware :
	Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
    return (i+1);
}
void quicksort (int arr[], int low, int high)
{ if (low < high)
    { int pi = partition (arr, low, high);
      quicksort (arr, low, pi-1);
      quicksort (arr, pi+1, high);
    }
  }
int main()
{ int n;
  srand (time(NULL));
  printf ("Enter the size of the array:");
  scanf ("%d", &n);
  int arr[n];
  for (int i=0; i<n; i++)
{ arr[i] = rand() % 100000;
}
clock_t start_time = clock();
quicksort (arr, 0, n-1);
clock_t end_time = clock();
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student
Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
double time_taken = ((double)(end_time - start_time))/
                                    CLOCKS_PER_SEC;
Printf ("Time taken to sort the array : %f Seconds\n",
        time_taken);
Printf("Sorted Array : (first 10 elements\n");
 for (int i=0; i<10; i++)
 { printf ("%d", arr[i]);
 }
 printf("\n");
 return 0;
 }
```

Output obtained

enter the size of the array : 5000
Time taken to sort the array : 0.000764 Seconds
Sorted array:

enter the size of the array : 10000
Time taken to sort the array : 0.001628 seconds
Sorted array :

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

STATEMENT

IM & CODE :

enter the size of the array: 15000
Time taken to sort the array: 0.002398 seconds
Sorted array:

enter the size of the array : 20000
Time taken to sort the array: 0.003401 seconds.
Sorted array:

enter the sizeof the array : 25000
Time taken to sort the array: 0.004151 seconds.
Sorted array:

IVEN

OBTAINED

KS

Signature of Student

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

Time Complexity Analysis

- Worst case : $O(n^2)$ (occurs when the pivot is the smallest or largest element repeatedly, leading to unbalanced partitions).
- Average case : $O(n \log n)$ (when partition are fairly balanced)
- Best case : $O(n \log n)$ (perfectly balanced partitions).

INPUT GIVEN

OUTPUT OBTAINED