Worksheet No.: 1

Student Name: Harsh	**UID:** 24MCA20045
Branch: MCA (General)	**Section/Group:** 1-A
Semester: 2nd	**Date of Performance:**
Subject Name: DESIGN AND ANALYSIS OF ALGORITHM LAB	**Subject Code:** 24CAP-612

Q1. Implement following:
Question of Worksheet-1 (DAA)

Sort a given set of elements using quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot the graph of time taken vs n. The element can be read number generator from a file or can be generated using a random number generator.

1. Aim of the practical:

- **The practical aims to evaluate the performance of the quicksort algorithm by measuring the time required to sort lists of varying sizes. By conducting this experiment, you will understand how the sorting time scales with the number of elements and visualize the algorithm's efficiency.**

2. Task to be done:

- Implement the Quick Sort Algorithm: Develop a function to sort a list of elements using quicksort.
- Generate or Read Lists: Create lists of different sizes either by generating random numbers or reading from a file.
- Measure Sorting Time: Record the time taken to sort each list.
- Repeat for Different Sizes: Perform the sorting and timing experiment for various sizes of lists.
- Plot Results: Create a graph showing the relationship between list size and time taken to sort.

3. Algorithm/Flowchart:

➢ **Quicksort Algorithm:**

- If the list has one or no elements, it is already sorted.
- Choose a pivot element from the list.
- Partition the list into three sublists: elements less than the pivot, elements equal to the pivot, and elements greater than the pivot.
- Recursively apply the quicksort algorithm to the sublists.
- Concatenate the sorted sublists.

➢ **Timing and Plotting:**

- For each list size n:
- Generate a list of n random integers.
- Record the start time.
- Sort the list using quicksort.
- Record the end time.
- Calculate the elapsed time.
- Store the time taken.
- Plot the graph of list size vs. time taken to sort.

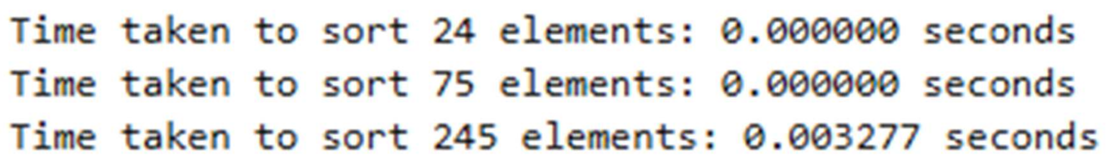
➢ **Flowchart:**

- Start: Begin the program.
- Define Quick Sort Functions: Set up the Quick Sort and Partition functions.
- Define Time Measurement Function: Create a function to measure sorting time.
- For Each Value of n:
- Generate or Read List: Create or load the list of elements.
- Record Start Time: Capture the time before sorting.
- Apply Quick Sort: Sort the list using the Quick Sort algorithm.
- Record End Time: Capture the time after sorting.
- Calculate Elapsed Time: Compute the duration of sorting.
- Store n and Time: Save the number of elements and the sorting time.
- Plot Time Taken vs. Size: Create a plot of the number of elements versus time taken.
- End: Finish the process.

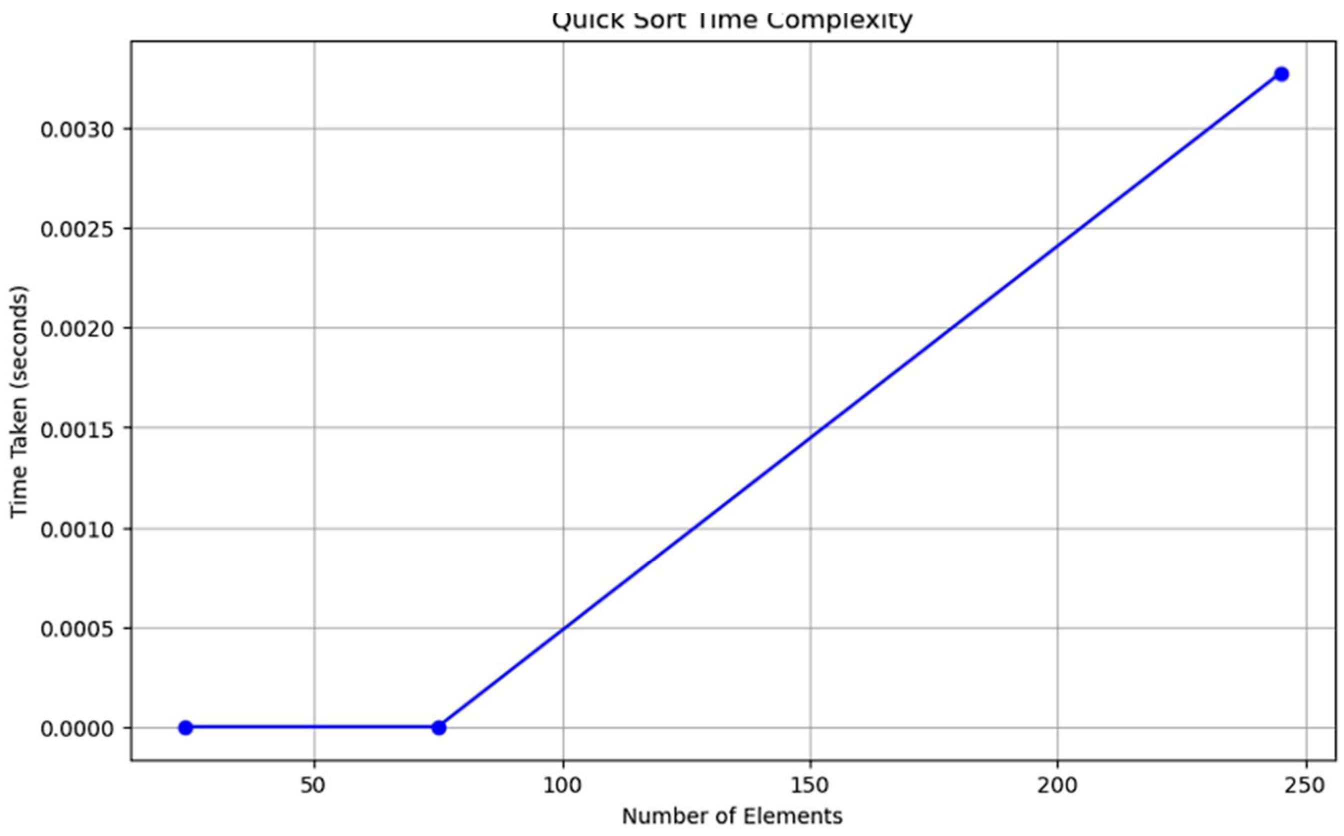
4. Code for experiment/practical:

```
import random
import time
import matplotlib.pyplot as plt

def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

def generate_random_list(size):
    # Generate a random list of integers
    return [random.randint(0, 300) for _ in range(size)]

def time_quick_sort(size):
    data = generate_random_list(size)
    start_time = time.time()
    quick_sort(data)
    end_time = time.time()
    return end_time - start_time

def run_experiment(sizes):
    times = [ ]
    for size in sizes:
        duration = time_quick_sort(size)
        times.append(duration)
        print(f"Time taken to sort {size} elements: {duration:.6f} seconds")
    return times

def plot_time_vs_size(sizes, times):
    plt.figure(figsize=(10, 6))
    plt.plot(sizes, times, marker='o', linestyle='-', color='b')
    plt.xlabel('Number of Elements')
    plt.ylabel('Time Taken (seconds)')
    plt.title('Quick Sort Time Complexity')
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    sizes = [24, 75, 245]                 # Different values of n
    times = run_experiment(sizes)
    plot_time_vs_size(sizes, times)
```

5. **Result/Output/Writing Summary:**



```
Time taken to sort 24 elements: 0.000000 seconds
Time taken to sort 75 elements: 0.000000 seconds
Time taken to sort 245 elements: 0.003277 seconds
```



6. Learning outcomes (What I have learnt):

- Understanding Quicksort: Learn how quicksort operates and it efficiency compared to other sorting algorithms.
- Performance Measurement: Gain experience in measuring algorithm performance and understanding time complexity.
- Data Visualization: Develop skills in plotting and interpreting performance data.
- Experimentation: Experience designing and conducting experiments to evaluate algorithm efficiency.
- Analysis and Interpretation: Improve the ability to analyze experimental results and draw meaningful conclusions from them.

Sr. No.	**Parameters**	**Marks Obtained**	**Maximum Marks**
1.	Worksheet		8 Marks
2.	Viva		10 Marks
3.	Simulation		12 Marks
	Total		30 Marks

Teacher Signature