Name:- Marvin Patel
ER bo :-22162101013
Batch:- 51
Sub :- AAD

Practical-3

NextMid Technology is an American food company that manufactures, markets, and distributes spices, seasoning mixes, condiments, and other flavoring products for the industrial, restaurant, institutional, and home markets, they are having some number quantity of different categories item food, kindly help them to sort datausing any three sorting methods 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 a graph of the comparison between them. Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

Questions:

What is the best, average and worst case analysis of algorithms?

Which are different asymptotic notations? What is their use?

What is the time complexity of above 3 sorting algorithms in all cases?

Code :-

Name:- Marvin Patel
ER bo :-22162101013
Batch:- 51
Sub :- AAD

Practical-3

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

# Bubble Sort def bubble_sort(arr):
n = len(arr)     for i in range(n):
for j in range(0, n-i-1):            if arr[j] >
arr[j+1]:                arr[j], arr[j+1] =
arr[j+1], arr[j]     return arr

# Merge Sort def
merge_sort(arr):
if len(arr) > 1:
        mid = len(arr) // 2
left = arr[:mid]         right
= arr[mid:]
```

Name:- Marvin Patel
ER bo :-22162101013
Batch:- 51
Sub :- AAD

```
        merge_sort(left)
        merge_sort(right)
        i = j = k = 0         while i < len(left)
and j < len(right):             if left[i] <
right[j]:                 arr[k] = left[i]
i         += 1
else:
                arr[k] = right[j]
j         += 1
k += 1

        while i < len(left):
arr[k] = left[i]
i         += 1
k += 1

        while j < len(right):
arr[k] = right[j]
j         += 1
k += 1
    return arr

# Quick Sort def
quick_sort(arr):
if len(arr) <= 1:
return arr     else:
        pivot = arr[0]        left = [x for x in
arr[1:] if x < pivot]        right = [x for x in
arr[1:] if x >= pivot]
        return quick_sort(left) + [pivot] + quick_sort(right)

# Time Comparison def
compare_sorting_algorithms(n_values):
    bubble_times = []
```

Name:- Marvin Patel
ER bo :-22162101013
Batch:- 51
Sub :- AAD

Practical-3

```
    merge_times = []
    quick_times = []

    for n in n_values:
        arr = random.sample(range(n * 10), n)

        # Bubble Sort
start_time = time.time()
bubble_sort(arr.copy())
        bubble_times.append(time.time() - start_time)

        # Merge Sort
start_time = time.time()
merge_sort(arr.copy())
        merge_times.append(time.time() - start_time)

        # Quick Sort
start_time = time.time()
quick_sort(arr.copy())
        quick_times.append(time.time() - start_time)

    return bubble_times, merge_times, quick_times


def plot_comparison(n_values, bubble_times, merge_times, quick_times):
plt.plot(n_values, bubble_times, label="Bubble Sort")
plt.plot(n_values, merge_times, label="Merge Sort")     plt.plot(n_values,
quick_times, label="Quick Sort")     plt.xlabel('Number of Elements')
plt.ylabel('Time (seconds)')     plt.legend()
    plt.show()


n_values = [100, 574, 1368, 2192, 5424, 10430] bubble_times,
merge_times, quick_times =
compare_sorting_algorithms(n_values)
plot_comparison(n_values, bubble_times, merge_times, quick_times)
```

Name:- Marvin Patel
ER bo :-22162101013
Batch:- 51
Sub :- AAD

Practical-3

Output: