**Experiment 6**

**Aim:** To analyse the time complexity of Bubble Sort, Merge Sort, Quick Sort and Insertion Sort algorithms using emprical method and plotting the graph for the same.

**Procedure:** This experiment is performed as follows:

1) Implementing all the algorithms using random integers from 1 to 100 and measuring the time taken for all the sorting algorithms implemented in experiment 1, 2, 3 and 4 using clock function in C++ for the entries between the size 10 - 100000 in the.csv file.

2) Plotting the graph using numpy, pandas and matplotlib library in Python.

**Code and Output:**

**#Importing Necessary Libraries**

import numpy as np

**[1]** import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

**#Take the Input from the Sorting Algorithms.csv**

**[2]** algo = pd.read\_csv('C://Users//hp//Desktop//Shaina//Sorting Algorithms.csv')

**[3]** algo

**Out [3]**

Table

Description automatically generated

**[4]** labels = algo['Entries']

**[5]** labels = list(np.array(labels))

**[6]** labels

**Out [6]**



**[7]** B\_Sort = list(np.array(algo['Bubble Sort']))

I\_Sort = list(np.array(algo['Insertion Sort']))

M\_Sort = list(np.array(algo['Merge Sort']))

Q\_Sort = list(np.array(algo['Quick Sort']))

**# Plotting the Graph**

**[8]** plt.plot(labels,B\_Sort,'--b')

plt.plot(labels,I\_Sort,'-og')

plt.plot(labels,M\_Sort,'oy')

plt.plot(labels,Q\_Sort,'--r')

plt.xlabel('Time in Milliseconds')

plt.ylabel('Input Size / Number of Elements in Array')

plt.title('Sorting Algorithms Comparisons')

plt.legend(['Bubble','Insertion','Merge','Quick'])

**Out [8]**

Chart, line chart

Description automatically generated

**Results and Conclusion:** The analysis of the time complexity of Bubble Sort, Merge Sort, Quick Sort and Insertion Sort algorithms using empirical method and graph plotting has been done successfully.