**Name : Harpalsinh Bhati**

**Enr. No.: 22162171010**

**Sub: Algorithm Analysis & Design**

**Branch: CS**

**Batch: 54**

**Sem: 5-B**

## 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 data using 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?

- **Best Case:**

- The best-case scenario refers to the minimum time an algorithm can take to complete, given the most favorable conditions. It helps understand how efficient the algorithm can be.

- **Average Case:**

- The average-case scenario gives an estimate of the time taken by the algorithm for a random input. It is often the most practical measure since it accounts for the typical behavior of the algorithm.

- **Worst Case:**

- The worst-case scenario refers to the maximum time an algorithm can take to complete, given the least favorable conditions. It helps understand the upper bound on the time an algorithm might need.

## Which are different asymptotic notations? What is their use?

Asymptotic notations are mathematical tools used to describe the running time of an algorithm as the input size grows. They allow us to analyze the performance of algorithms in a way that is independent of hardware and implementation details.

- **Big O Notation (O):**
  - Represents the upper bound of the running time. It describes the worst-case scenario. For example, if an algorithm is O(n^2), its running time will not exceed a function of n^2 as the input size increases.
- **Omega Notation (Ω):**
  - Represents the lower bound of the running time. It describes the best-case scenario. For example, if an algorithm is Ω(n), its running time will be at least a function of n as the input size increases.
- **Theta Notation (Θ):**
  - Represents the tight bound of the running time. It describes the average-case scenario. If an algorithm is Θ(n log n), its running time will grow proportionally to n log n as the input size increases, both in the best and worst cases.

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

- **Bubble Sort:**

- **Best Case:** O(n) - Occurs when the array is already sorted.
- **Average Case:** O(n^2) - Generally, it has to perform a significant number of comparisons and swaps.
- **Worst Case:** O(n^2) - Occurs when the array is sorted in reverse order.

- **Insertion Sort:**

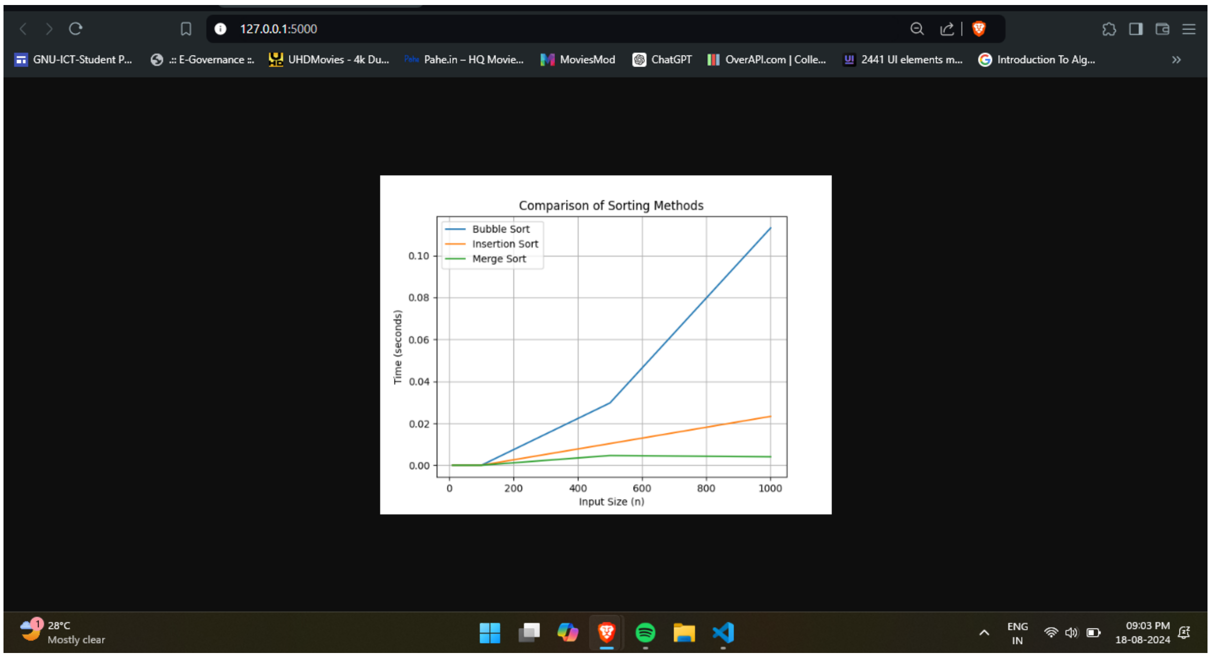
- **Best Case:** O(n) - Occurs when the array is already sorted.
- **Average Case:** O(n^2) - On average, elements have to be moved back in the list.
- **Worst Case:** O(n^2) - Occurs when the array is sorted in reverse order.

- **Merge Sort:**

- **Best Case:** O(n log n) - The best case is the same as the average and worst cases since merge sort always divides the list and merges them.
- **Average Case:** O(n log n) - It always divides the array into halves and then merges them.
- **Worst Case:** O(n log n) - Even if the array is sorted in reverse order, the time complexity remains the same.

**Output:**

```python
import random
import time
import matplotlib.pyplot as plt
from flask import Flask, render_template, Response
import io
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas

app = Flask(__name__)

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]

def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

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

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

def measure_sorting_time(sort_function, data):
    start_time = time.time()
    sort_function(data)
    end_time = time.time()
    return end_time - start_time

def generate_food_quantities(n):
    return [random.randint(1, 1000) for _ in range(n)]

@app.route('/')
def index():
    input_sizes = [10, 50, 100, 500, 1000]
    bubble_sort_times = []
    insertion_sort_times = []
    merge_sort_times = []

    for n in input_sizes:
        data = generate_food_quantities(n)
        bubble_sort_time = measure_sorting_time(bubble_sort, data.copy())
        bubble_sort_times.append(bubble_sort_time)
        insertion_sort_time = measure_sorting_time(insertion_sort,
data.copy())
        insertion_sort_times.append(insertion_sort_time)
        merge_sort_time = measure_sorting_time(merge_sort, data.copy())
        merge_sort_times.append(merge_sort_time)

    plt.figure()
    plt.plot(input_sizes, bubble_sort_times, label='Bubble Sort')
    plt.plot(input_sizes, insertion_sort_times, label='Insertion Sort')
    plt.plot(input_sizes, merge_sort_times, label='Merge Sort')
    plt.xlabel('Input Size (n)')
    plt.ylabel('Time (seconds)')
    plt.legend()
    plt.title('Comparison of Sorting Methods')
    plt.grid(True)

    # Render the plot to a PNG image and send it as a response
    canvas = FigureCanvas(plt.gcf())
    output = io.BytesIO()
    canvas.print_png(output)
    plt.close()  # Close the figure after rendering

    return Response(output.getvalue(), mimetype='image/png')

if __name__ == '__main__':
    app.run(debug=True)
```



127.0.0.1:5000
GNU-ICT-Student P...
:: E-Governance ::
UHDMovies - 4k Du...
Pahe.in – HQ Movie...
MoviesMod
ChatGPT
OverAPI.com | Colle...
2441 UI elements m...
Introduction To Alg...
Comparison of Sorting Methods
Bubble Sort
Insertion Sort
Merge Sort
Time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
0
200
400
600
800
1000
Input Size (n)
28°C
Mostly clear
ENG
IN
09:03 PM
18-08-2024