**Institute of Computer Technology**

**B. Tech Computer Science and Engineering**

**Sub: Algorithm Analysis and Design**

**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?

Which are different asymptotic notations? What is their use?

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

Code:

App.py

from flask import Flask, render\_template, request, jsonify, send\_file

import time

import matplotlib.pyplot as plt

import io

import random

app = Flask(\_\_name\_\_)

*# Sorting algorithms*

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

def merge\_sort(arr):

    if len(arr) > 1:

        mid = len(arr) // 2

        L = arr[:mid]

        R = arr[mid:]

        merge\_sort(L)

        merge\_sort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):

            if L[i] < R[j]:

                arr[k] = L[i]

                i += 1

            else:

                arr[k] = R[j]

                j += 1

            k += 1

        while i < len(L):

            arr[k] = L[i]

            i += 1

            k += 1

        while j < len(R):

            arr[k] = R[j]

            j += 1

            k += 1

    return arr

def quick\_sort(arr):

    if len(arr) <= 1:

        return arr

    else:

        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)

@app.route('/')

def index():

    return render\_template('index.html')

*# Task 1 page and functionality*

@app.route('/task1')

def task1():

    return render\_template('task1.html')

*# Sorting logic associated with task1*

@app.route('/sort', methods=['POST'])

def sort():

    data = request.json

    arr = data['arr']

    start\_time = time.time()

    bubble\_sorted = bubble\_sort(arr.copy())

    bubble\_time = time.time() - start\_time

    start\_time = time.time()

    merge\_sorted = merge\_sort(arr.copy())

    merge\_time = time.time() - start\_time

    start\_time = time.time()

    quick\_sorted = quick\_sort(arr.copy())

    quick\_time = time.time() - start\_time

    return jsonify({

        'bubble\_sorted': bubble\_sorted,

        'merge\_sorted': merge\_sorted,

        'quick\_sorted': quick\_sorted,

        'bubble\_time': bubble\_time \* 1000,

        'merge\_time': merge\_time \* 1000,

        'quick\_time': quick\_time \* 1000

    })

*# Plotting logic associated with task1*

@app.route('/plot', methods=['GET'])

def plot():

    sizes = [10, 50, 100, 500, 1000]

    bubble\_times = []

    merge\_times = []

    quick\_times = []

    for size in sizes:

        arr = [random.randint(0, 1000) for \_ in range(size)]

        start\_time = time.time()

        bubble\_sort(arr.copy())

        bubble\_times.append((time.time() - start\_time) \* 1000)

        start\_time = time.time()

        merge\_sort(arr.copy())

        merge\_times.append((time.time() - start\_time) \* 1000)

        start\_time = time.time()

        quick\_sort(arr.copy())

        quick\_times.append((time.time() - start\_time) \* 1000)

    plt.figure(figsize=(12, 6))

    plt.plot(sizes, bubble\_times, label='Bubble Sort', marker='o')

    plt.plot(sizes, merge\_times, label='Merge Sort', marker='o')

    plt.plot(sizes, quick\_times, label='Quick Sort', marker='o')

    plt.xlabel('Number of Elements')

    plt.ylabel('Time (ms)')

    plt.title('Sorting Algorithm Performance Comparison')

    plt.legend()

    plt.grid(True)

    img = io.BytesIO()

    plt.savefig(img, format='png')

    img.seek(0)

    plt.close()

    return send\_file(img, mimetype='image/png')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Task1.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sorting Algorithm Comparison</title>

    <style>

        body {

            font-family: *Arial*, *sans-serif*;

            background-color: #3e3e3e;

            display: *flex*;

            justify-content: *center*;

            align-items: *center*;

            height: 100vh;

            margin: 0;

            color: *aliceblue*;

            font-weight: 100%;

        }

        .container {

            background-color: rgb(33, 33, 33);

            padding: 20px;

            border-radius: 10px;

            box-shadow: 0 0 10px rgba(255, 255, 255, 0.866);

        }

        .form-group {

            margin-bottom: 15px;

        }

        label {

            display: *block*;

            margin-bottom: 5px;

            color: *antiquewhite*;

        }

        input[type="text"] {

            width: 100%;

            padding: 8px;

            box-sizing: *border-box*;

            background-color: #3e3e3e;

            color: *aliceblue*;

        }

        button {

            padding: 10px 15px;

            background-color: #007BFF;

            color: *white*;

            border: *none*;

            border-radius: 4px;

            cursor: *pointer*;

        }

        button:hover {

            background-color: #0056b3;

        }

        .result {

            margin-top: 20px;

        }

        .chart {

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Sorting Algorithm Comparison</h1>

        <form id="sortForm">

            <div class="form-group">

                <label for="arr">Enter Array Elements (comma separated):</label>

                <input type="text" id="arr" name="arr" required>

            </div>

            <button type="submit">Sort</button>

        </form>

        <div class="result" id="result"></div>

        <button id="viewPlot">View Execution Time Plot</button>

        <img id="plotImage" class="chart" style="display:none;" src="/plot" alt="Execution Time Plot">

    </div>

    <script>

        document.getElementById('sortForm').addEventListener('submit', function(event) {

            event.preventDefault();

            const arr = document.getElementById('arr').value.split(',').map(Number);

            fetch('/sort', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json'

                },

                body: JSON.stringify({ arr: arr })

            })

            .then(response => response.json())

            .then(result => {

                document.getElementById('result').innerHTML = `

                    <p>Bubble Sort Result: ${result.bubble\_sorted} (Time: ${result.bubble\_time.toFixed(2)} ms)</p>

                    <p>Merge Sort Result: ${result.merge\_sorted} (Time: ${result.merge\_time.toFixed(2)} ms)</p>

                    <p>Quick Sort Result: ${result.quick\_sorted} (Time: ${result.quick\_time.toFixed(2)} ms)</p>

                `;

            })

            .catch(error => {

                console.error('Error:', error);

            });

        });

        document.getElementById('viewPlot').addEventListener('click', function() {

            document.getElementById('plotImage').*style*.display = 'block';

        });

    </script>

</body>

</html>

Output:





