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?

App.py

```
from flask import Flask, request, jsonify, render_template, send_file
import time
import matplotlib.pyplot as plt
import io
```

```
import random


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]
    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

# Quick Sort
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')
```

```
@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
    })

@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)
```

Index.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: #f0f0f0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .container {
            background-color: white;
```

```
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        width: 90%;
        max-width: 600px;
    }
    .form-group {
        margin-bottom: 15px;
    }
    label {
        display: block;
        margin-bottom: 5px;
    }
    input[type="text"] {
        width: 100%;
        padding: 8px;
        box-sizing: border-box;
    }
    button {
        padding: 10px 15px;
        background-color: #007BFF;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }
    button:hover {
        background-color: #0056b3;
    }
```

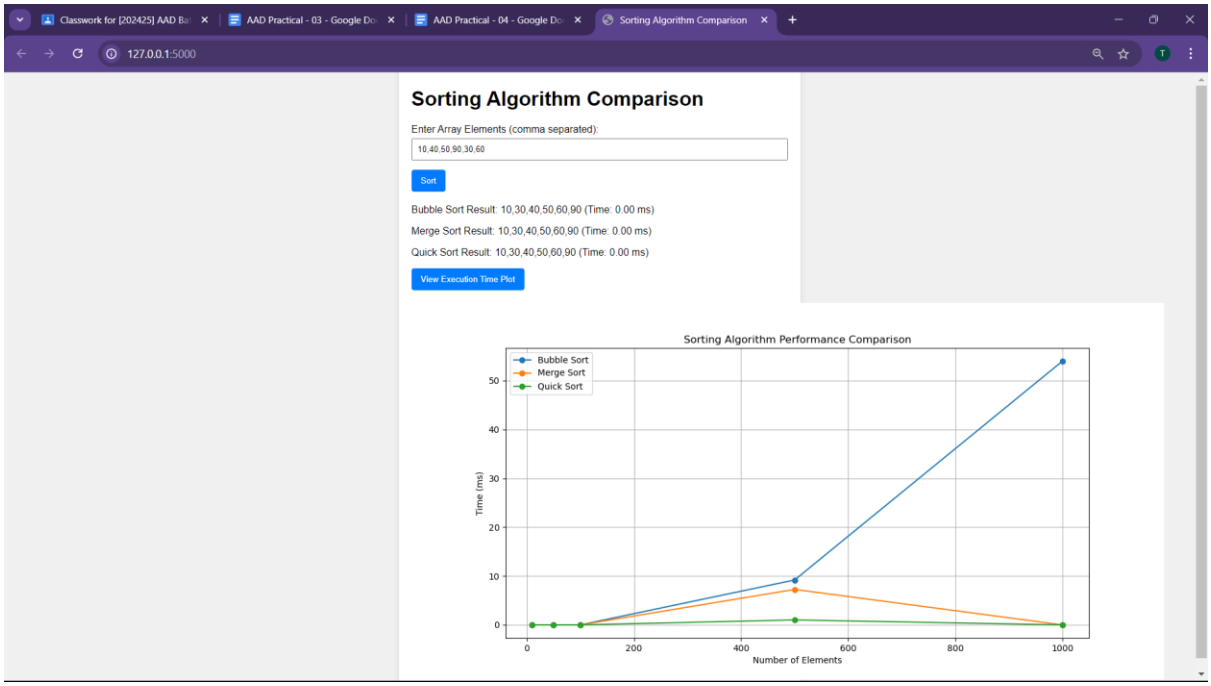
```
        .result {
            margin-top: 20px;
        }
        .chart {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Sorting Algorithm Comparison</h1>
        <form id="sortForm">
            <div class="form-group">
                <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>
```


Classwork for [202425] AAD Ba
AAD Practical - 03 - Google Do
AAD Practical - 04 - Google Do
Sorting Algorithm Comparison
127.0.0.1:5000
Sorting Algorithm Comparison
Enter Array Elements (comma separated):
10,40,50,90,30,60
Sort
Bubble Sort Result: 10,30,40,50,60,90 (Time: 0.00 ms)
Merge Sort Result: 10,30,40,50,60,90 (Time: 0.00 ms)
Quick Sort Result: 10,30,40,50,60,90 (Time: 0.00 ms)
View Execution Time Plot
Sorting Algorithm Performance Comparison
Bubble Sort
Merge Sort
Quick Sort
Time (ms)
0
10
20
30
40
50
0
200
400
600
800
1000
Number of Elements