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?
Best Case Analysis: The algorithm performs the minimum possible number of operations.
Worst Case Analysis: The algorithm performs the maximum possible number of operations.

Which are different asymptotic notations? What is their use?
Big O Notation: Represents the upper bound of the time complexity. It gives the worst-case scenario of an algorithm.

Theta Notation: Represents the exact bound of the time complexity. It provides both the upper and lower bounds of an algorithm.

Omega Notation: Represents the lower bound of the time complexity. It gives the best-case scenario of an algorithm.

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

Selection Sort: $O(n^2)$ (all cases)

Insertion Sort: Best Case: $O(n)$

Worst Case: $O(n^2)$

Bubble Sort: Best Case: $O(n)$

Worst Case: $O(n^2)$

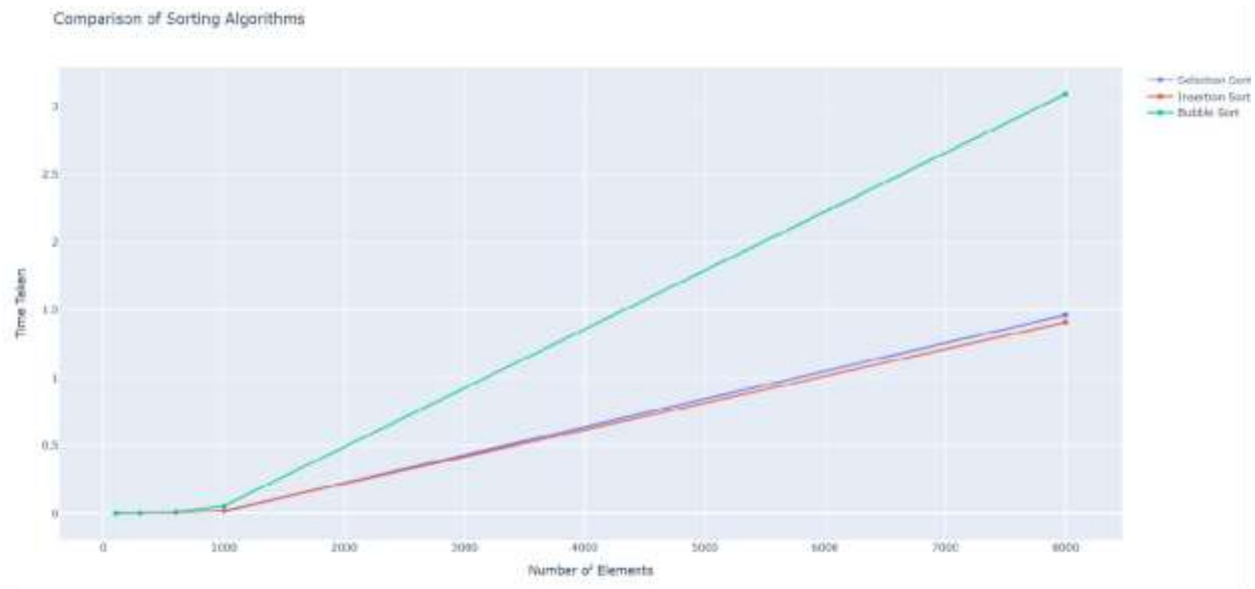
CODE:

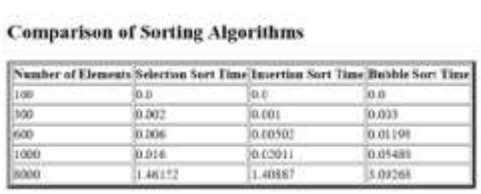
```
<html>
<body>
  {{plot|safe}}
  <h2>Comparison of Sorting Algorithms</h2>
  <table border="4">
    <thead>
      <tr>
        <th>Number of Elements</th>
        <th>Selection Sort Time</th>
        <th>Insertion Sort Time</th>
        <th>Bubble Sort Time</th>
      </tr>
    </thead>
    <tbody>
      {% for i in range(sizes|length) %}
      <tr>
        <td>{{ sizes[i] }}</td>
```

```
                <td>{{ selection_times[i] }}</td>
                <td>{{ insertion_times[i] }}</td>
                <td>{{ bubble_times[i] }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>

from flask import Flask, render_template
import time
import random
import plotly.graph_objs as go
import plotly.offline as pyo
app = Flask(_name_)

def selection(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

def insertion(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 bubble(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 measure_time(sort_function, arr):
    start_time = time.time()
    sort_function(arr.copy())
    end_time = time.time()
    return end_time - start_time
@app.route('/')
def index():
    sizes = [100,300,600,1000,8000]
    selection_times = []
    insertion_times = []
    bubble_times = []

    for size in sizes:
        arr = random.sample(range(size * 10), size)
        selection_times.append(round(measure_time(selection, arr),5))
        insertion_times.append(round(measure_time(insertion, arr),5))
        bubble_times.append(round(measure_time(bubble, arr),5))

    trace1 = go.Scatter(x=sizes, y=selection_times, mode='lines+markers', name='Selection Sort')
    trace2 = go.Scatter(x=sizes, y=insertion_times, mode='lines+markers', name='Insertion Sort')
    trace3 = go.Scatter(x=sizes, y=bubble_times, mode='lines+markers', name='Bubble Sort')
    layout = go.Layout(title='Comparison of Sorting Algorithms',xaxis=dict(title='Number of
Elements'),yaxis=dict(title='Time Taken'),)

    fig = go.Figure(data=[trace1, trace2, trace3], layout=layout)
    plot_html = pyo.plot(fig, output_type='div', include_plotlyjs=True)
```

```
returnrender_template('index.html',plot=plot_html,sizes=sizes,selection_times=selection_times,insertion_
times=insertion_times,bubble_times=bubble_times)
app.run(debug=True)
```

OUTPUT:





Comparison of Sorting Algorithms

Number of Elements	Selection Sort Time	Insertion Sort Time	Bubble Sort Time
100	0.0	0.0	0.0
300	0.002	0.001	0.003
600	0.006	0.00502	0.01198
1000	0.016	0.02011	0.05488
8000	1.46172	1.40887	3.09268