Itisha Desai
Branch – Cyber Security
Sem – 5
Batch – CSE54
Enrollment No. – 22162171006

Subject: Algorithm Analysis and Design

Practical 4

Trigent is an early pioneer in IT outsourcing and offshore software development business. Thousands of employees working in this company kindly help to find out the employee's details (i.e employee ID, employee salary etc) to implement Recursive Binary search and Linear search (or Sequential Search) and determine the time taken to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

Using the algorithm search for the following

1. The designation which has highest salary package
2. The Name of the Employee who has the lowest salary
3. The Mobile number who is youngest employee
4. Salary of the employee who is oldest in age

Code :-

Python File – main.py

```
from flask import Flask, render_template, request
import time
import matplotlib.pyplot as plt
import io
import base64
import numpy as np
from scipy.interpolate import interp1d
from scipy.interpolate import make_interp_spline

app = Flask(__name__)

employees = [
    {"id": 1, "name": "Mansi", "salary": 63000, "age": 55,
        "mobile": "9876543210", "designation": "Analyst"},
    {"id": 2, "name": "Princy", "salary": 66000, "age": 27,
        "mobile": "8765432109", "designation": "Senior Developer"},
    {"id": 3, "name": "Itisha", "salary": 80000, "age": 20,
        "mobile": "8394632754", "designation": "Project Manager"},
    {"id": 4, "name": "Trishla", "salary": 50000, "age": 40,
        "mobile": "6543210987", "designation": "Product Manager"},
    {"id": 5, "name": "Vaidehi", "salary": 46000, "age": 50,
        "mobile": "5432109876", "designation": "Marketing Manager"},
]

def linear_search(employees, key, value):
    start_time = time.time()
    for idx, emp in enumerate(employees):
        if emp[key] == value:
            time_taken = time.time() - start_time
            return emp, time_taken, idx + 1
    time_taken = time.time() - start_time
    return None, time_taken, len(employees)

def binary_search(employees, key, value, low, high, iterations=0):
    if low <= high:
        mid = (low + high) // 2
        iterations += 1
        if employees[mid][key] == value:
            return employees[mid], iterations
        elif employees[mid][key] < value:
            return binary_search(employees, key, value, mid + 1, high, iterations)
        else:
            return binary_search(employees, key, value, low, mid - 1, iterations)
    return None, iterations
```

```
def measure_time_binary_search(employees, key, value):
    start_time = time.time()
    result, iterations = binary_search(employees, key, value, 0, len(employees) - 1)
    time_taken = time.time() - start_time
    return result, time_taken, iterations


def plot_linear_graph(n_values, times, label, color):
    plt.figure(figsize=(5, 5))

    if len(n_values) >= 4:
        try:
            spl = make_interp_spline(n_values, times, k=3)
            x_smooth = np.linspace(min(n_values), max(n_values), 300)
            y_smooth = spl(x_smooth)
        except ValueError as e:
            print(f"Error with cubic spline interpolation: {e}")
            spl = interp1d(n_values, times, kind='linear')
            x_smooth = np.linspace(min(n_values), max(n_values), 300)
            y_smooth = spl(x_smooth)
    else:
        spl = interp1d(n_values, times, kind='linear')
        x_smooth = np.linspace(min(n_values), max(n_values), 300)
        y_smooth = spl(x_smooth)

    plt.plot(x_smooth, y_smooth, label=label, marker='', linewidth=4, color=color)
    plt.xlabel("Number of Elements (n)")
    plt.ylabel("Time Taken (seconds)")
    plt.legend()
    plt.title(f"{label} Time Complexity")
    plt.xlim(0, max(n_values) + 1)
    plt.ylim(0, max(times) + 0.1)

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()
    plt.close()
    return plot_url


def plot_smooth_binary_graph(n_values, times, label, color):
    plt.figure(figsize=(5, 5))

    if len(n_values) >= 4:
        try:
            spl = make_interp_spline(n_values, times, k=3)
            x_smooth = np.linspace(min(n_values), max(n_values), 300)
            y_smooth = spl(x_smooth)
```

```
        except ValueError as e:
            print(f"Error with cubic spline interpolation: {e}")
            spl = interp1d(n_values, times, kind='linear')
            x_smooth = np.linspace(min(n_values), max(n_values), 300)
            y_smooth = spl(x_smooth)
    else:
        spl = interp1d(n_values, times, kind='linear')
        x_smooth = np.linspace(min(n_values), max(n_values), 300)
        y_smooth = spl(x_smooth)

    plt.plot(x_smooth, y_smooth, label=label, marker='', linewidth=2, color=color)
    plt.xlabel("Number of Elements (n)")
    plt.ylabel("Time Taken (seconds)")
    plt.legend()
    plt.title(f"{label} Time Complexity")
    plt.xlim(0, max(n_values) + 1)
    plt.ylim(0, max(times) + 0.1)

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()
    plt.close()
    return plot_url


@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        key = request.form['key']
        value = request.form['value']

        if key in ['id', 'salary', 'age']:
            value = int(value)

        linear_result, linear_time, linear_iterations = linear_search(employees, key, value)

        sorted_employees = sorted(employees, key=lambda x: x[key])
        binary_result, binary_time, binary_iterations =
measure_time_binary_search(sorted_employees, key, value)

        n_values = list(range(0, len(employees) + 1))

        linear_times = [(linear_iterations / len(employees)) * linear_time for _ in n_values]

        binary_times = [(binary_iterations / len(employees)) * np.log2(n) if n > 0 else 0 for
n in n_values]
```

```
    linear_graph_url = plot_linear_graph(n_values, linear_times, "Linear Search (O(n))",
'red')
    binary_graph_url = plot_smooth_binary_graph(n_values, binary_times, "Binary
Search (O(log n))", 'blue')

    return render_template('index.html', linear_result=linear_result,
binary_result=binary_result,
                           linear_graph_url=linear_graph_url, binary_graph_url=binary_graph_url)
  return render_template('index.html')


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

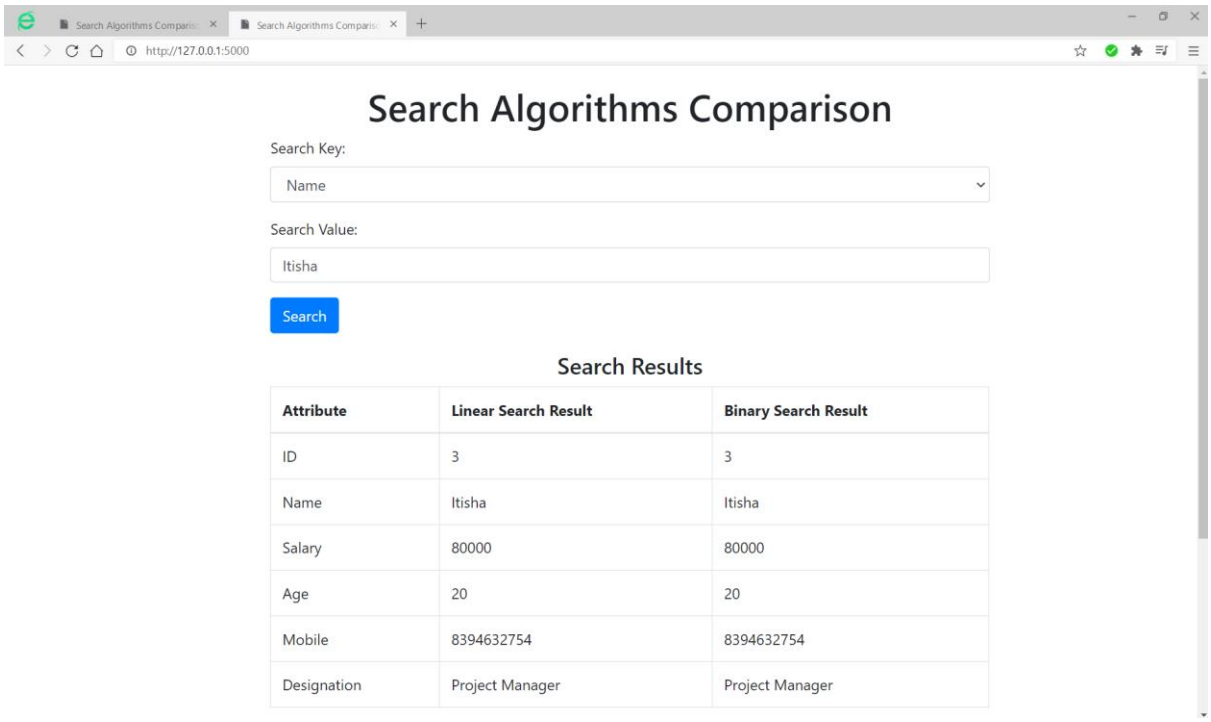
HTML File - index.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Search Algorithms Comparison</title>
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    body {
      padding-top: 20px;
    }
    .container {
      max-width: 800px;
    }
    .result-table {
      margin-top: 20px;
    }
    .graph-container {
      margin-top: 20px;
    }
    .graph-container img {
      max-width: 100%;
    }
  </style>
</head>
<body>
  <div class="container">
```

```
<h1 class="text-center">Search Algorithms Comparison</h1>

<form method="POST" action="/">
  <div class="form-group">
    <label for="key">Search Key:</label>
    <select class="form-control" id="key" name="key" required>
      <option value="id">ID</option>
      <option value="name">Name</option>
      <option value="salary">Salary</option>
      <option value="age">Age</option>
      <option value="mobile">Mobile</option>
      <option value="designation">Designation</option>
    </select>
  </div>
  <div class="form-group">
    <label for="value">Search Value:</label>
    <input type="text" class="form-control" id="value" name="value" required>
  </div>
  <button type="submit" class="btn btn-primary">Search</button>
</form>

{% if linear_result or binary_result %}
<div class="result-table">
  <h4 class="text-center">Search Results</h4>
  <table class="table table-bordered">
    <thead>
      <tr>
        <th>Attribute</th>
        <th>Linear Search Result</th>
        <th>Binary Search Result</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>ID</td>
        <td>{{ linear_result.id if linear_result else 'Not Found' }}</td>
        <td>{{ binary_result.id if binary_result else 'Not Found' }}</td>
      </tr>
      <tr>
        <td>Name</td>
        <td>{{ linear_result.name if linear_result else 'Not Found' }}</td>
        <td>{{ binary_result.name if binary_result else 'Not Found' }}</td>
      </tr>
      <tr>
        <td>Salary</td>
        <td>{{ linear_result.salary if linear_result else 'Not Found' }}</td>
        <td>{{ binary_result.salary if binary_result else 'Not Found' }}</td>
      </tr>
```

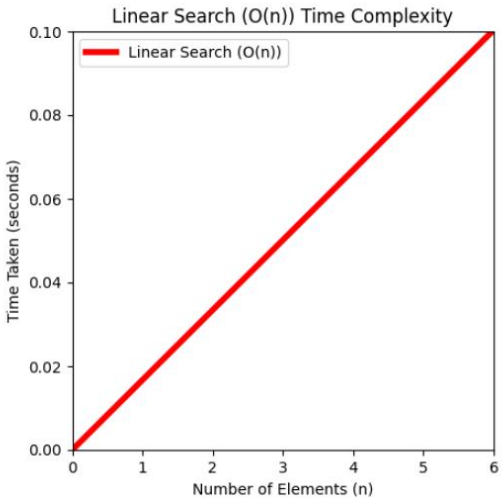
```
            <tr>
              <td>Age</td>
              <td>{{ linear_result.age if linear_result else 'Not Found' }}</td>
              <td>{{ binary_result.age if binary_result else 'Not Found' }}</td>
            </tr>
            <tr>
              <td>Mobile</td>
              <td>{{ linear_result.mobile if linear_result else 'Not Found' }}</td>
              <td>{{ binary_result.mobile if binary_result else 'Not Found' }}</td>
            </tr>
            <tr>
              <td>Designation</td>
              <td>{{ linear_result.designation if linear_result else 'Not Found' }}</td>
              <td>{{ binary_result.designation if binary_result else 'Not Found' }}</td>
            </tr>
          </tbody>
        </table>
      </div>

      <div class="graph-container">
        <h4 class="text-center">Graphs</h4>
        <div class="row">
          <div class="col-md-6">
            <h5>Linear Search</h5>
            <img src="{{ linear_graph_url }}" alt="Linear Search
Time Complexity">
          </div>
          <div class="col-md-6">
            <h5>Binary Search</h5>
            <img src="{{ binary_graph_url }}" alt="Binary Search
Time Complexity">
          </div>
        </div>
      </div>
      {% endif %}
    </div>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js">
</script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script
>
</body>
</html>
```

Output :



Graphs

Linear Search



Binary Search