Name:-Devansh koyani

Batch:-54

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: 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:-

```
import random
import string
import time
import io
import base64
from flask import Flask, render_template_string
```

Name:-Devansh koyani

Batch:-54

```
from matplotlib.figure import Figure

class Employee:
    def __init__(self, emp_id, name, role, salary, age, phone_number):
        self.emp_id = emp_id
        self.name = name
        self.role = role
        self.salary = salary
        self.age = age
        self.phone_number = phone_number

def create_random_employee(emp_id):
    name = ''.join(random.choices(string.ascii_uppercase, k=6))
    role = random.choice(['Developer', 'Supervisor', 'Consultant', 'Assistant'])
    salary = random.randint(25000, 160000)
    age = random.randint(20, 65)
    phone_number = ''.join(random.choices(string.digits, k=10))
    return Employee(emp_id, name, role, salary, age, phone_number)

def generate_employees(num):
    return [create_random_employee(i) for i in range(1, num + 1)]

def search_linear(employees, search_key, attr_name):
    for emp in employees:
        if getattr(emp, attr_name) == search_key:
            return emp
    return None

def search_binary(employees, search_key, attr_name, low, high):
    if low > high:
        return None
    mid = (low + high) // 2
    if getattr(employees[mid], attr_name) == search_key:
        return employees[mid]
    elif getattr(employees[mid], attr_name) < search_key:
        return search_binary(employees, search_key, attr_name, mid + 1, high)
    else:
        return search_binary(employees, search_key, attr_name, low, mid - 1)

def employee_with_highest_salary(employees):
    return max(employees, key=lambda emp: emp.salary)

def employee_with_lowest_salary(employees):
    return min(employees, key=lambda emp: emp.salary)

def youngest_employee(employees):
    return min(employees, key=lambda emp: emp.age)

def oldest_employee(employees):
    return max(employees, key=lambda emp: emp.age)

def compare_search_times(employee_list, search_key, attr_name):
    start_time = time.time()
    linear_result = search_linear(employee_list, search_key, attr_name)
    linear_search_duration = time.time() - start_time

    sorted_employees = sorted(employee_list, key=lambda emp: getattr(emp, attr_name))
    start_time = time.time()
    binary_result = search_binary(sorted_employees, search_key, attr_name, 0, len(sorted_employees) - 1)
    binary_search_duration = time.time() - start_time

    return linear_search_duration, binary_search_duration

app = Flask(__name__)

@app.route('/')
def homepage():
    employee_counts = [1000, 5000, 10000, 50000]
    search_times = {'linear': [], 'binary': []}

    all_employees = generate_employees(max(employee_counts))

    for count in employee_counts:
        current_batch = all_employees[:count]
        search_key = random.choice(current_batch).salary  # Example: searching by salary
                linear_duration, binary_duration = compare_search_times(current_batch, search_key,
'salary')
        search_times['linear'].append(linear_duration)
        search_times['binary'].append(binary_duration)

    top_paid_employee = employee_with_highest_salary(all_employees)
    least_paid_employee = employee_with_lowest_salary(all_employees)
    youngest_emp = youngest_employee(all_employees)
    oldest_emp = oldest_employee(all_employees)

    fig = Figure()
    ax = fig.subplots()
    ax.plot(employee_counts, search_times['linear'], label='Linear Search')
    ax.plot(employee_counts, search_times['binary'], label='Binary Search')
    ax.set_xlabel('Employee Count')
    ax.set_ylabel('Search Duration (seconds)')
    ax.set_title('Performance: Linear vs Binary Search')
    ax.legend()

    output_image = io.BytesIO()
    fig.savefig(output_image, format='png')
    output_image.seek(0)
    plot_image = base64.b64encode(output_image.getvalue()).decode('utf8')

    html_content = f'''
    <html>
        <head>
            <title>Employee Search Comparison</title>
        </head>
        <body>
            <h1>Employee Search: Linear vs Binary Search</h1>
            <div>
                <h2>Search Time Comparison</h2>
                <img src="data:image/png;base64,{plot_image}" alt="Search Time Comparison">
            </div>
            <div>
                <h3>Employee with Highest Salary</h3>
                <p>Name: {top_paid_employee.name}, Salary: {top_paid_employee.salary}</p>
            </div>
            <div>
                <h3>Employee with Lowest Salary</h3>
                <p>Name: {least_paid_employee.name}, Salary: {least_paid_employee.salary}</p>
            </div>
            <div>
```

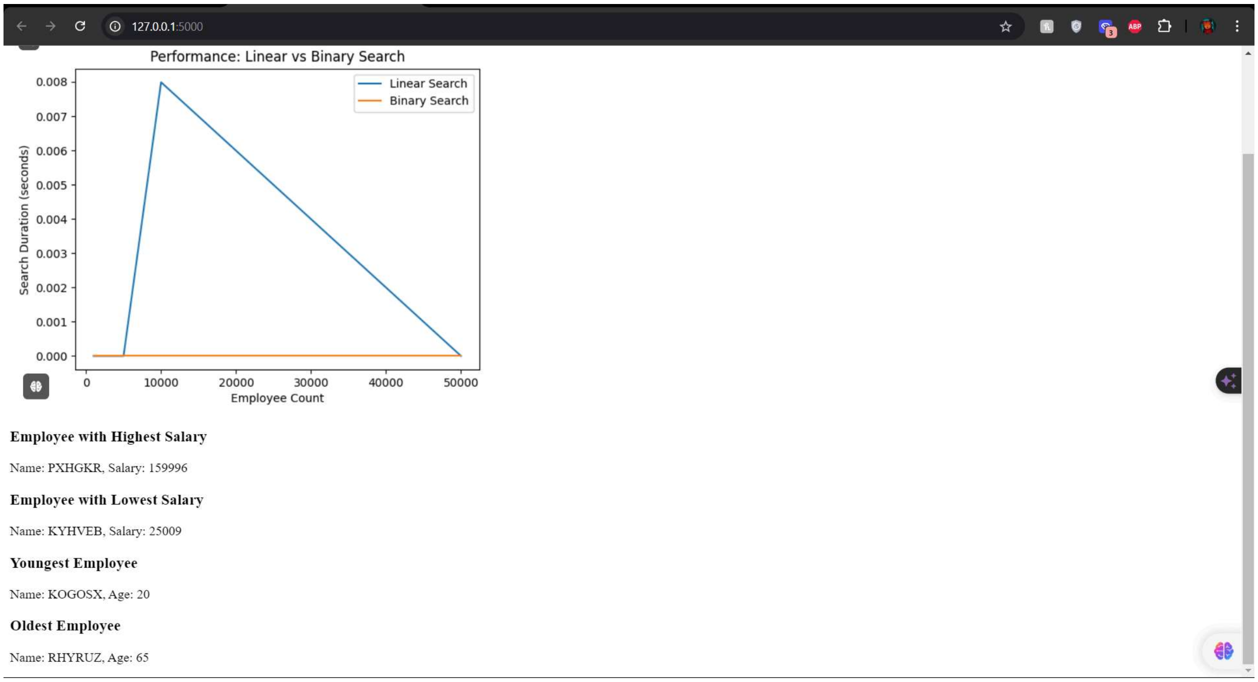
Name:-Devansh koyani

Batch:-54

```
                <h3>Youngest Employee</h3>
                <p>Name: {youngest_emp.name}, Age: {youngest_emp.age}</p>
            </div>
            <div>
                <h3>Oldest Employee</h3>
                <p>Name: {oldest_emp.name}, Age: {oldest_emp.age}</p>
            </div>
        </body>
    </html>
    '''

    return render_template_string(html_content)

app.run(debug=True)
```



Employee with Highest Salary

Name: PXHGKR, Salary: 159996

Employee with Lowest Salary

Name: KYHVEB, Salary: 25009

Youngest Employee

Name: KOGOSX, Age: 20

Oldest Employee

Name: RHYRUZ, Age: 65