Name : Harpalsinh Bhati

Enr. No.: 22162171010

Sub: Algorithm Analysis & Design

Branch: CS

Batch: 54

Sem: 5-B

Practical-4

Trigent is an early pioneer in IT outsourcing and offshore software evelopment 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
from matplotlib.figure import Figure

class Employee:
    def __init__(self, emp_id, name, designation, salary, age, mobile):
        self.emp_id = emp_id
        self.name = name
```

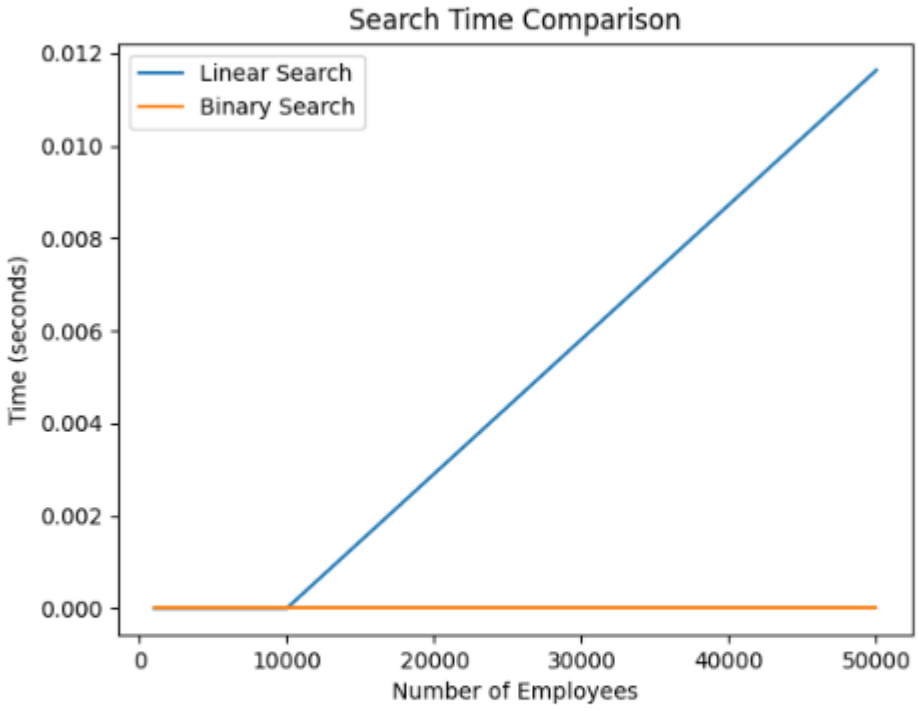
```
        self.designation = designation
        self.salary = salary
        self.age = age
        self.mobile = mobile

def generate_random_employee(emp_id):
    name = ''.join(random.choices(string.ascii_uppercase, k=5))
    designation = random.choice(['Engineer', 'Manager', 'Analyst', 'Clerk'])
    salary = random.randint(30000, 150000)
    age = random.randint(22, 60)
    mobile = ''.join(random.choices(string.digits, k=10))
    return Employee(emp_id, name, designation, salary, age, mobile)

def generate_employee_list(n):
    return [generate_random_employee(i) for i in range(1, n + 1)]

def linear_search(employees, key, attribute):
    for employee in employees:
        if getattr(employee, attribute) == key:
            return employee
    return None

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

def find_highest_salary_employee(employees):
    return max(employees, key=lambda x: x.salary)

def find_lowest_salary_employee(employees):
    return min(employees, key=lambda x: x.salary)

def find_youngest_employee(employees):
    return min(employees, key=lambda x: x.age)

def find_oldest_employee(employees):
    return max(employees, key=lambda x: x.age)

def measure_search_times(employee_list, key, attribute):

    start_time = time.time()
```

```
    linear_result = linear_search(employee_list, key, attribute)
    linear_search_time = time.time() - start_time

    sorted_employees = sorted(employee_list, key=lambda x: getattr(x,
attribute))
    start_time = time.time()
    binary_result = binary_search(sorted_employees, key, attribute, 0,
len(sorted_employees) - 1)
    binary_search_time = time.time() - start_time

    return linear_search_time, binary_search_time

app = Flask(__name__)

@app.route('/')
def index():
    sizes = [1000, 5000, 10000, 50000]
    times = {'linear': [], 'binary': []}

    employee_list = generate_employee_list(max(sizes))

    for size in sizes:
        subset = employee_list[:size]
        key = random.choice(subset).salary
        linear_time, binary_time = measure_search_times(subset, key, 'salary')
        times['linear'].append(linear_time)
        times['binary'].append(binary_time)

    highest_salary_employee = find_highest_salary_employee(employee_list)
    lowest_salary_employee = find_lowest_salary_employee(employee_list)
    youngest_employee = find_youngest_employee(employee_list)
    oldest_employee = find_oldest_employee(employee_list)

    fig = Figure()
    ax = fig.subplots()
    ax.plot(sizes, times['linear'], label='Linear Search')
    ax.plot(sizes, times['binary'], label='Binary Search')
    ax.set_xlabel('Number of Employees')
    ax.set_ylabel('Time (seconds)')
    ax.set_title('Search Time Comparison')
    ax.legend()

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

    return render_template(
```

```
        'Prac_4.html',
        plot_url=plot_url,
        highest_salary_employee=highest_salary_employee,
        lowest_salary_employee=lowest_salary_employee,
        youngest_employee=youngest_employee,
        oldest_employee=oldest_employee
    )

app.run(debug=True)
```

Output:

Search Time Comparison



Search Results

Highest Salary: Analyst - Rs.149997 (ZMZQT)

Lowest Salary: HAOJW - Rs.30001

Youngest Employee: OHGJU - Age 22, Mobile: 2845761072

Oldest Employee: NWZDM - Age 60, Salary: Rs.118455