KARIYA RAJ 21162101011

ALGORITHM ANALYSIS AND DESIGN

PRACTICAL -7

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

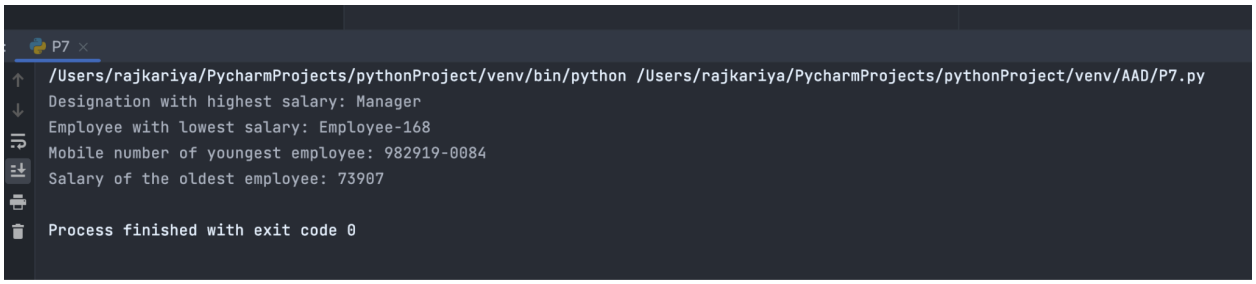
CODE:

```
import random
import time
import matplotlib.pyplot as plt


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

```
self.mobile = mobile


def linear_search(employees, key):
for i, emp in enumerate(employees):
if emp.emp_id == key:
return i
return -1


def binary_search(employees, key):
left, right = 0, len(employees) - 1
count = 0 # Initialize the count
while left <= right:
mid = (left + right) // 2
count += 1 # Increment the count
if employees[mid].emp_id == key:
return count # Return count when element is found
elif employees[mid].emp_id < key:
left = mid + 1
else:
right = mid - 1
return 0 # Return 0 when element is not found


n = 200
employees = []
for i in range(n):
emp = Employee(i, f"Employee-{i}", random.randint(30000, 100000),
"Developer" if i % 2 == 0 else "Manager",
random.randint(20, 60), f"982919-{i:04}")
employees.append(emp)

employees.sort(key=lambda x: x.emp_id)
search_keys = [random.randint(0, n - 1) for _ in range(100)]
binary_counts = []
linear_counts = []

highest_salary = float('-inf')
lowest_salary = float('inf')
youngest_age = float('inf')
oldest_age = float('-inf')
name_lowest_salary = None
mobile_youngest_employee = None

for key in search_keys:
binary_count = binary_search(employees, key)
if binary_count > 0:
binary_counts.append(binary_count)
```

```
linear_index = linear_search(employees, key)
linear_counts.append(linear_index + 1 if linear_index != -1 else 0)

if binary_count > 0:
emp = employees[key] # Assuming key is the found index
if emp.salary > highest_salary:
highest_salary = emp.salary
highest_salary_designation = emp.designation

if emp.salary < lowest_salary:
lowest_salary = emp.salary
name_lowest_salary = emp.name

if emp.age < youngest_age:
youngest_age = emp.age
mobile_youngest_employee = emp.mobile

if emp.age > oldest_age:
oldest_age = emp.age
salary_oldest_employee = emp.salary

print(f"Designation with highest salary: {highest_salary_designation}")
print(f"Employee with lowest salary: {name_lowest_salary}")
print(f"Mobile number of youngest employee: {mobile_youngest_employee}")
print(f"Salary of the oldest employee: {salary_oldest_employee}")
print(binary_counts)
print(linear_counts)
plt.plot(search_keys, binary_counts, label='Binary Search')
plt.plot(search_keys, linear_counts, label='Linear Search')
plt.xlabel('Search Key')
plt.ylabel('Count Returned')
plt.legend()
plt.title('Search Count Comparison')
plt.show()
```

OUTPUT:



KARIYA RAJ 21162101011