DAA Assignment-3

Write an algorithm to find gross and net salary of employees.

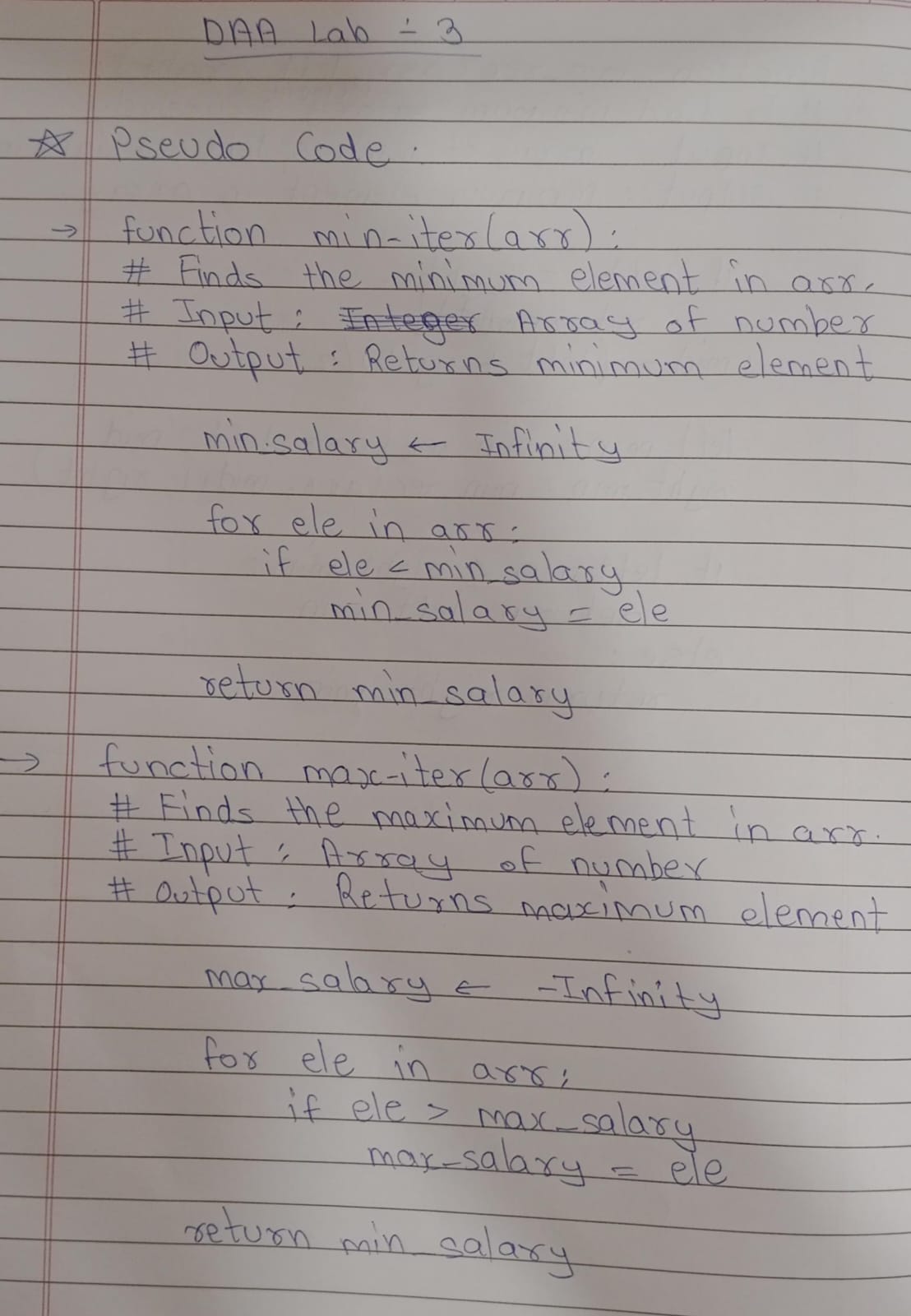
ABC co. ltd. has 2000 employees.

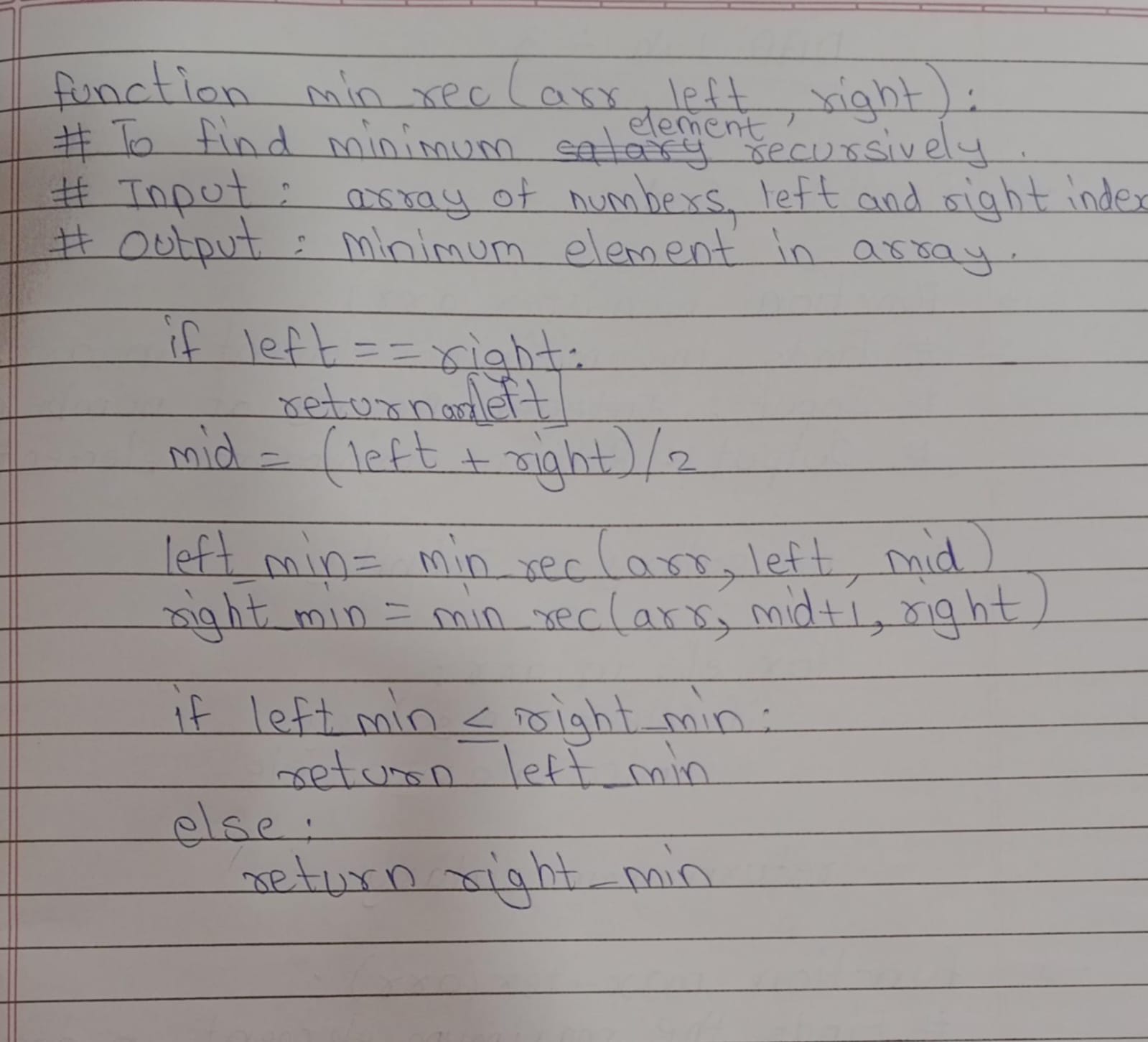
Your task is to calculate each employee's salary and find employees with minimum salary and maximum salary.

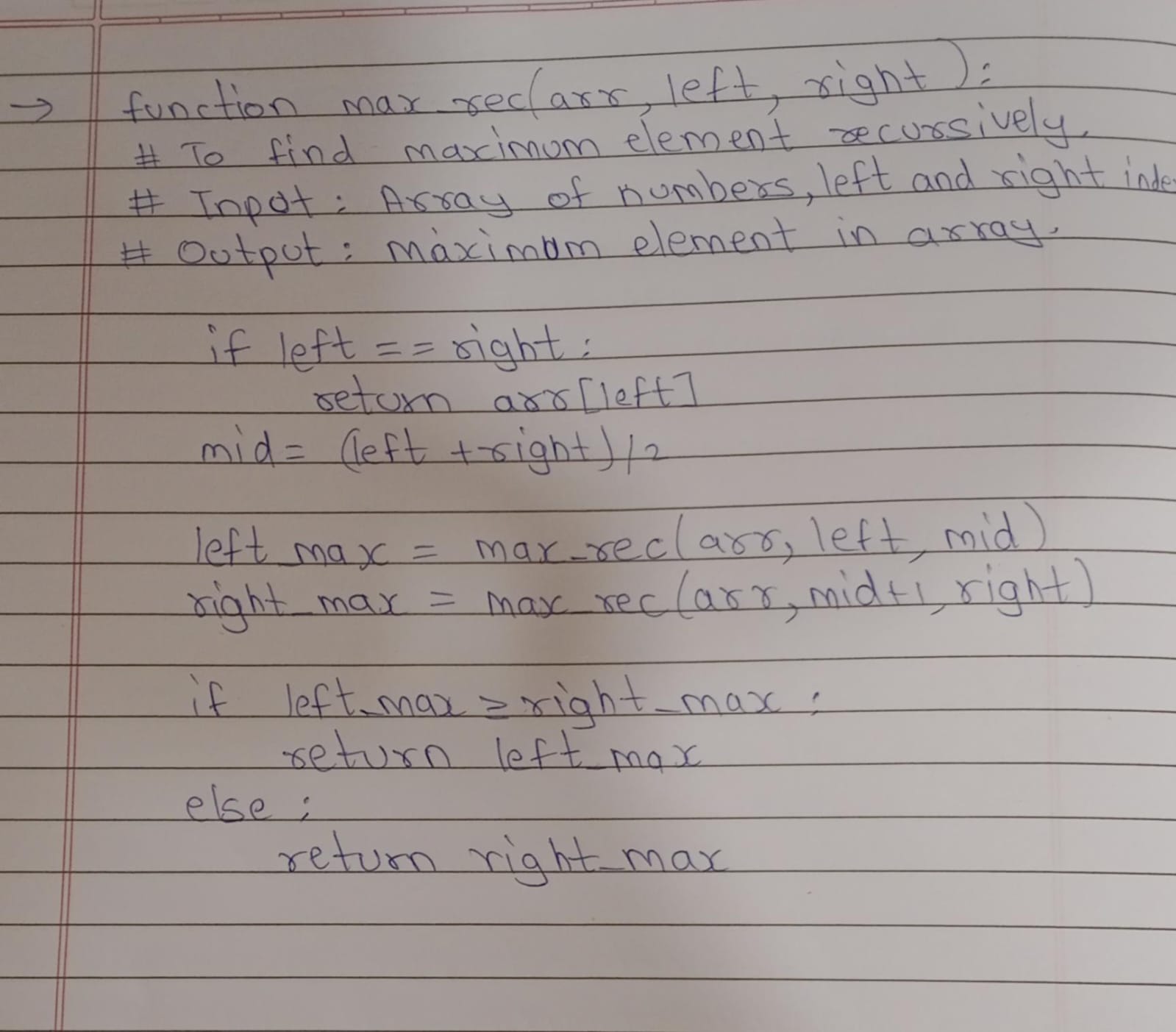
Do the above task using divide and conquer technique.

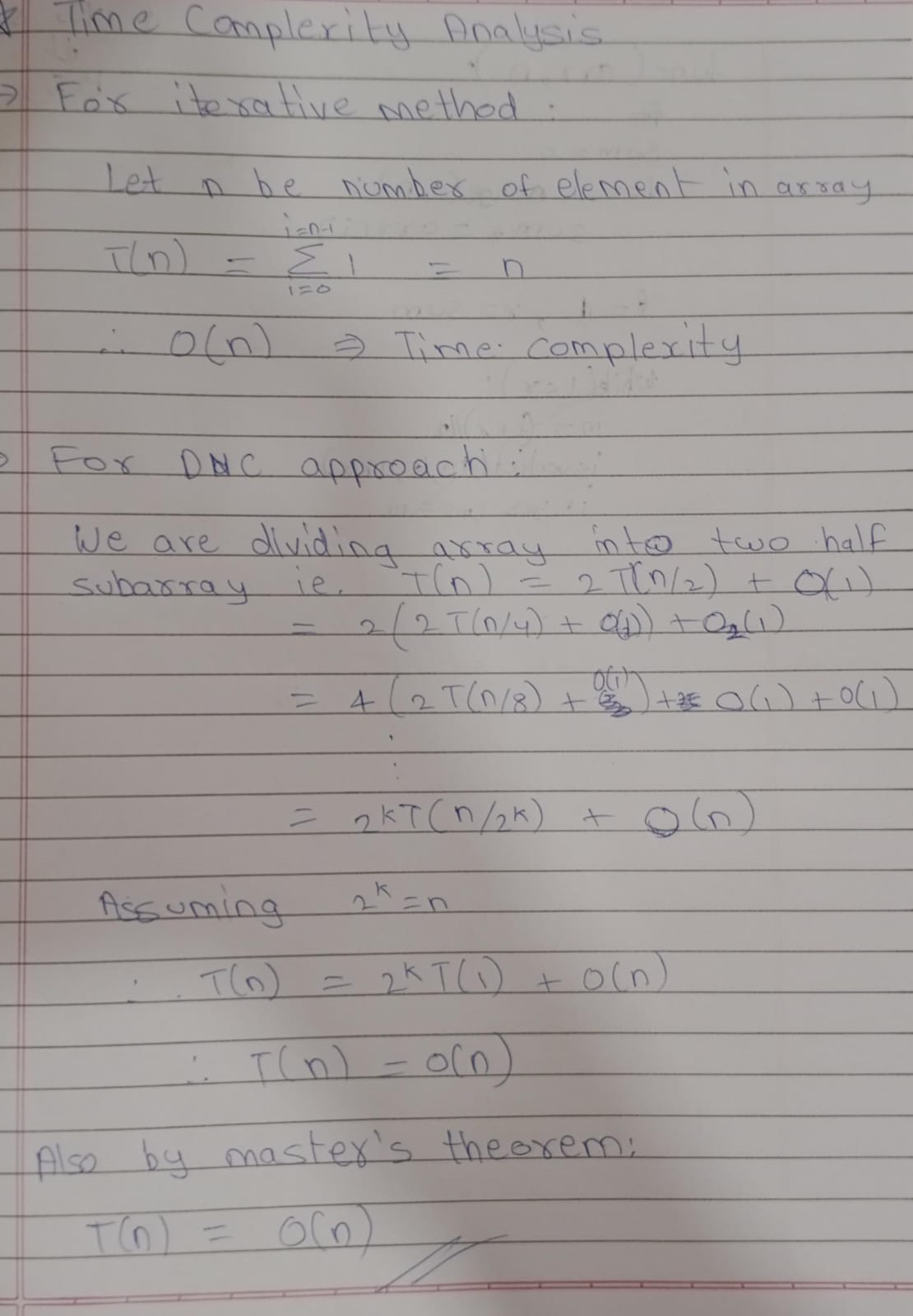
Find the improvement in the complexity using divide and conquer method.

**Pseudo code:**









**Program:**

To generate the employee data:

import pandas as pd

import random

from employee import Employee

def gen\_emp\_data():

*# generating random data*

names = [f"Employee\_{i}" for i in range(2000)]

base\_salaries = [random.randint(100000, 3000000) for \_ in range(2000)]

HRAs = [random.randint(10000, 30000) for \_ in range(2000)]

Bonuses = [random.randint(0, 10000) for \_ in range(2000)]

deductions = [random.randint(1000, 10000) for \_ in range(2000)]

*# Data structure(array) to store data*

employees = []

for i in range(2000):

*# creating employee object of class Employee*

employee = Employee(

names[i],

base\_salaries[i],

HRAs[i],

Bonuses[i],

deductions[i]

)

*# appending into the array*

employees.append(

{

"Name": employee.name,

"Base\_Salary": employee.base\_salary,

"HRA": employee.HRA,

"Bonus": employee.bonus,

"Deductions": employee.deductions,

"Gross\_Salary":employee.gross\_salary(),

"Net\_Salary":employee.net\_salary()

}

)

return employees

employee\_data = gen\_emp\_data()

df = pd.DataFrame(employee\_data)

df.to\_csv(f"employee\_data.csv")

print("Data saved successfully")

To find minimum and maximum salary:

import pandas as pd

import os

def tax\_to\_be\_paid(amount):

taxable\_amount=0

if amount<=300000:

return 0

if amount>300000 and amount<=700000:

return 0.05\*(amount-300000)

taxable\_amount+=0.05\*400000

if amount>700000 and amount<=1000000:

return 0.1\*(amount-700000)+taxable\_amount

taxable\_amount+=0.1\*300000

if amount>1000000 and amount<1200000:

return 0.15\*(amount-1000000)+taxable\_amount

taxable\_amount+=0.15\*200000

if amount>1200000 and amount<1500000:

return 0.20\*(amount-1200000)+taxable\_amount

taxable\_amount+=0.20\*300000

if amount>1500000:

return 0.3\*(amount-1500000)+taxable\_amount

class Employee:

def \_\_init\_\_(self, name, base, HRA, bonus, deductions):

*self*.name = name

*self*.base\_salary = base

*self*.HRA = HRA

*self*.bonus = bonus

*self*.deductions = deductions

def gross\_salary(self):

*# calculating Grossalary of the employee*

GrossSalary = *self*.base\_salary + *self*.bonus + *self*.HRA

return GrossSalary

def net\_salary(self):

*# calculating NetSalary of the employee*

NetSalary=*self*.gross\_salary()-tax\_to\_be\_paid(*self*.gross\_salary()-*self*.HRA)-*self*.deductions

return NetSalary;

def min\_iter(df):

m = df.shape[0]

min\_salary = float("inf")

for i in range(m):

if df.iloc[i]["Net\_Salary"] < min\_salary:

min\_salary=df.iloc[i]["Net\_Salary"]

id = i

return id

def max\_iter(df):

m = df.shape[0]

max\_salary = float("-inf")

for i in range(m):

if df.iloc[i]["Net\_Salary"] > max\_salary:

max\_salary = df.iloc[i]["Net\_Salary"]

id = i

return id

def min\_rec(df,left, right):

if left == right:

return left

mid = left + (right - left)//2

left\_id= min\_rec(df,left, mid)

right\_id= min\_rec(df,mid + 1, right)

if df.iloc[left\_id]["Net\_Salary"] < df.iloc[right\_id]["Net\_Salary"]:

return left\_id

else:

return right\_id

def max\_rec(df,left, right):

if left == right:

return left

mid = left + (right - left) // 2

left\_id = max\_rec(df,left, mid)

right\_id = max\_rec(df,mid + 1, right)

if df.iloc[left\_id]["Net\_Salary"] > df.iloc[right\_id]["Net\_Salary"]:

return left\_id

else:

return right\_id

def main():

file\_size = os.path.getsize('employee\_data.csv')

if file\_size == 0:

print('File is Empty')

return

if(not os.path.isfile("employee\_data.csv")):

print("Employee\_data missing! Please ensure that the employee data is stored in file named 'employee\_data.csv'.")

return

df = pd.read\_csv("employee\_data.csv")

print("Employees with Minimum Net Salaries: ")

print(f"Iterative Result: ",end="")

id = min\_iter(df)

name=df.iloc[id]["Name"]

min\_salary=df.iloc[id]["Net\_Salary"]

print(f"Id: {id}; Name: {name}; Salary: {min\_salary}")

print(f"Divide and Conquer Result: ",end="")

id = min\_rec(df,0,m-1)

name=df.iloc[id]["Name"]

min\_salary=df.iloc[id]["Net\_Salary"]

print(f"Id: {id}; Name: {name}; Salary: {min\_salary}")

print("Employees with Maximum Net Salaries: ")

print(f"Iterative Result: ",end="")

id = max\_iter(df)

name=df.iloc[id]["Name"]

max\_salary=df.iloc[id]["Net\_Salary"]

print(f"Id: {id}; Name: {name}; Salary: {max\_salary}")

print(f"Divide and Conquer Result: ",end="")

id = max\_rec(df,0,m-1)

name=df.iloc[id]["Name"]

max\_salary=df.iloc[id]["Net\_Salary"]

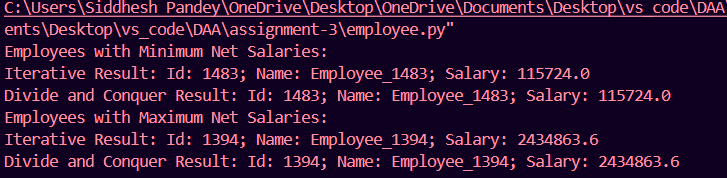
print(f"Id: {id}; Name: {name}; Salary: {max\_salary}")

if \_\_name\_\_ == "\_\_main\_\_":

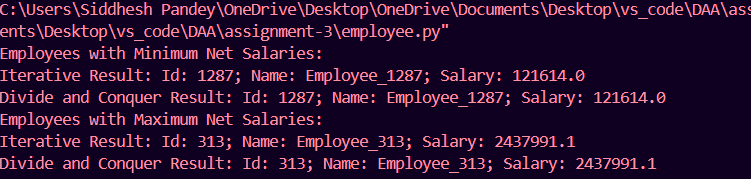
main()

**Test Cases:**

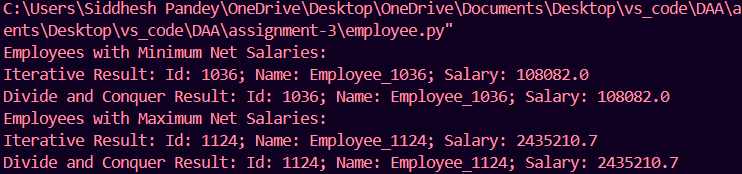
Test 1:



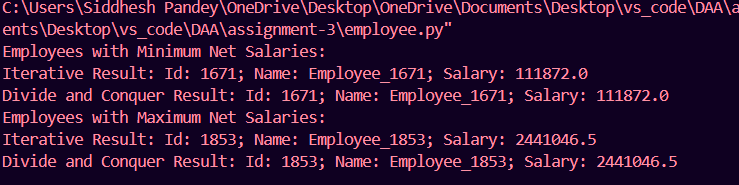
Test 2:



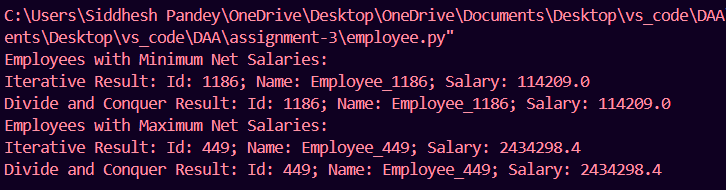
Test 3:



Test 4:



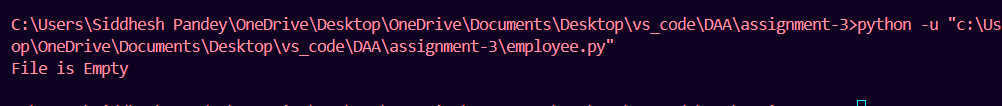
Test 5:



Negative Test cases:

Test 6: If file does not exist: 

Test 7: If file is empty:



**Conclusion:**

It can be seen that for finding maximum or minimum value from the given data the time complexity of linear search and divide and conquer algorithm is same (linear time) i.e. O(n).

Therefore for finding maximum or minimum data from the data set we should use linear search due to its simple and easy to understand algorithm and also the space complexity for linear search is O(1) whereas for DnC algorithm it is O(log n).