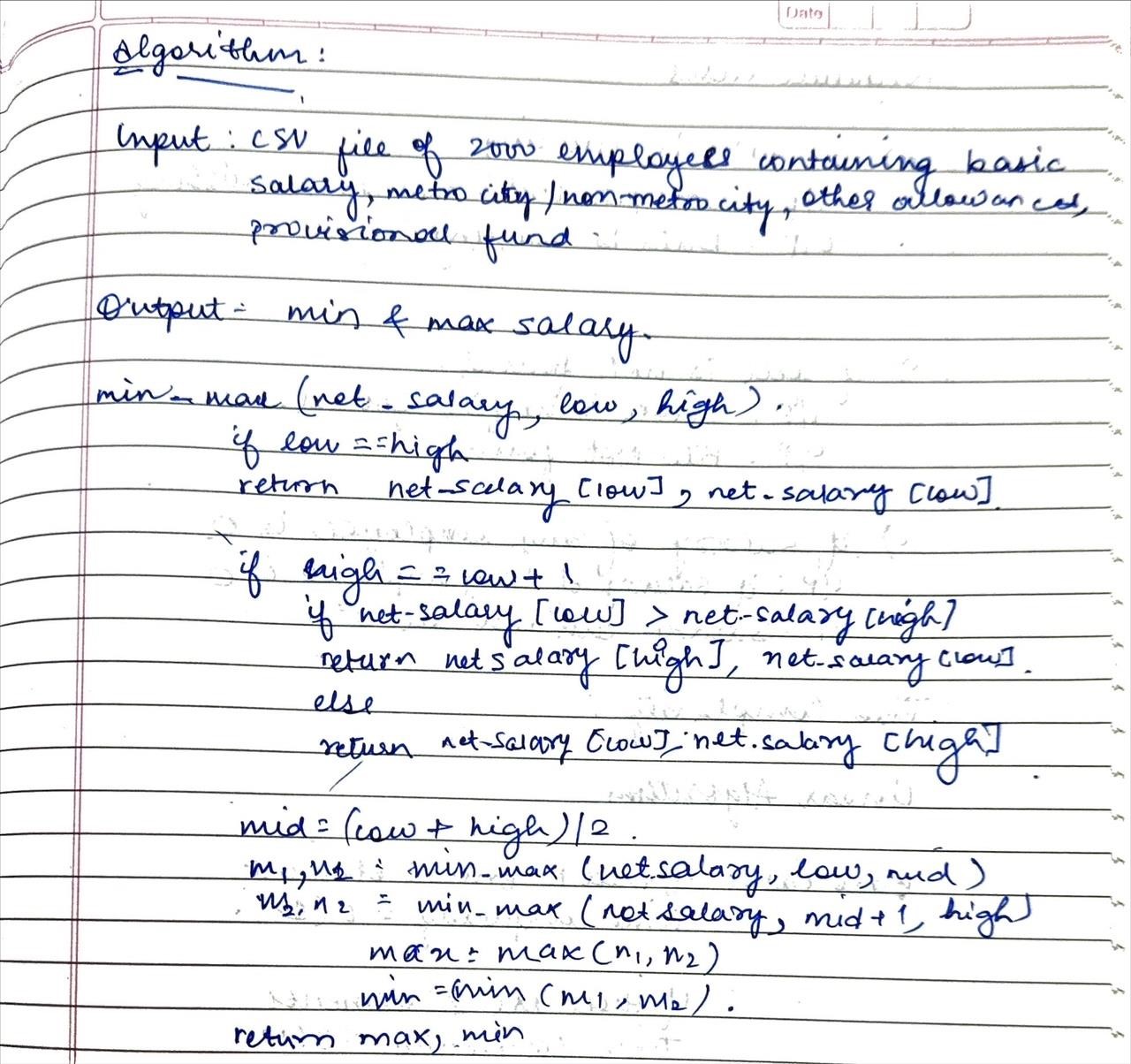
**DAA Lab 3**

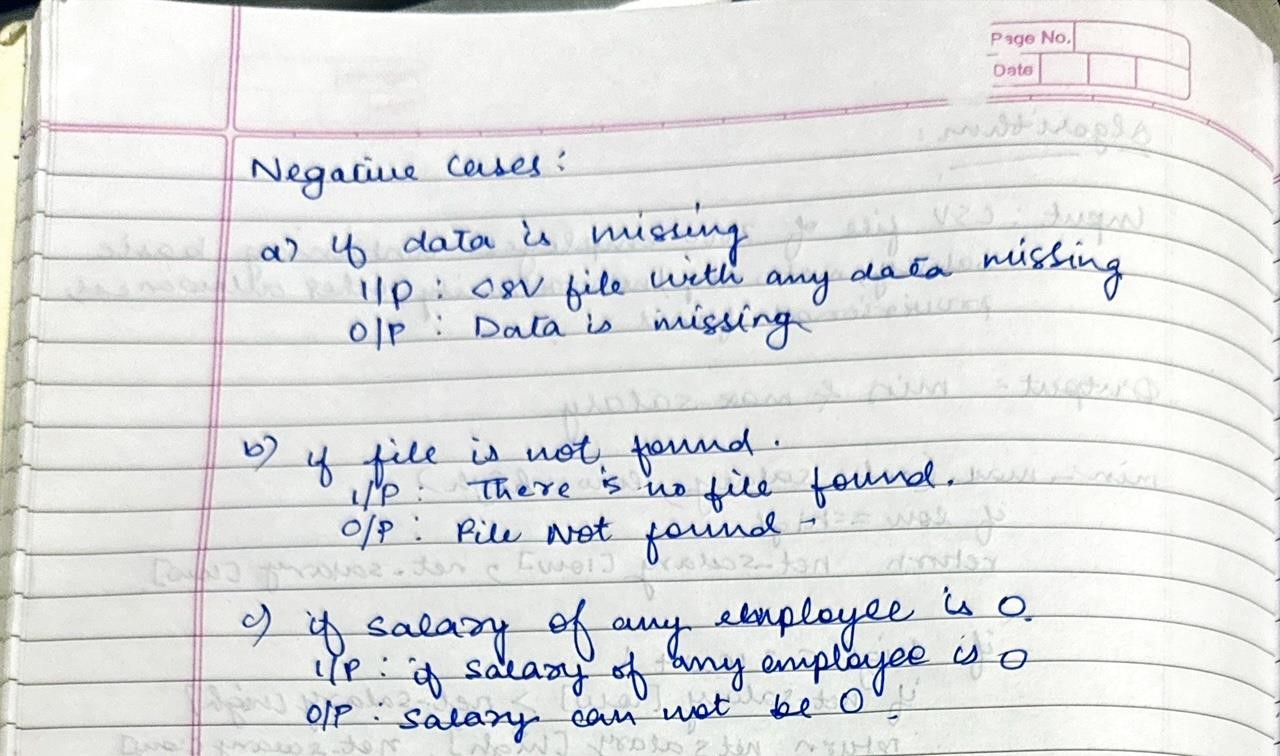
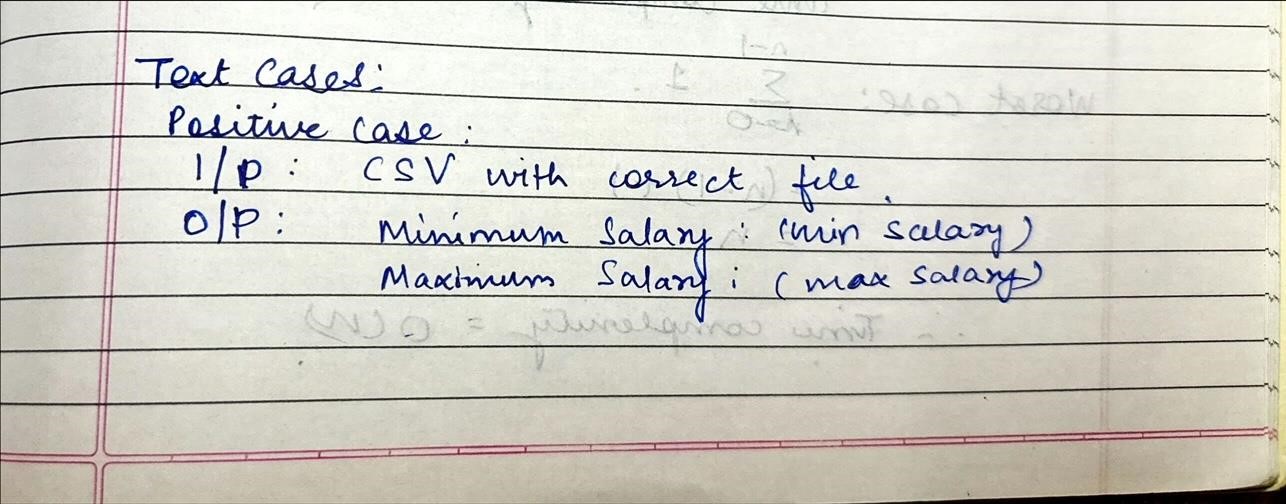
Name: Sarakshi Mamodia

Reg No: 231071056

**Aim**: To find the net and gross salary of the employees and find the minimum and maximum salary. **Algorithm:**



**Test Cases:**

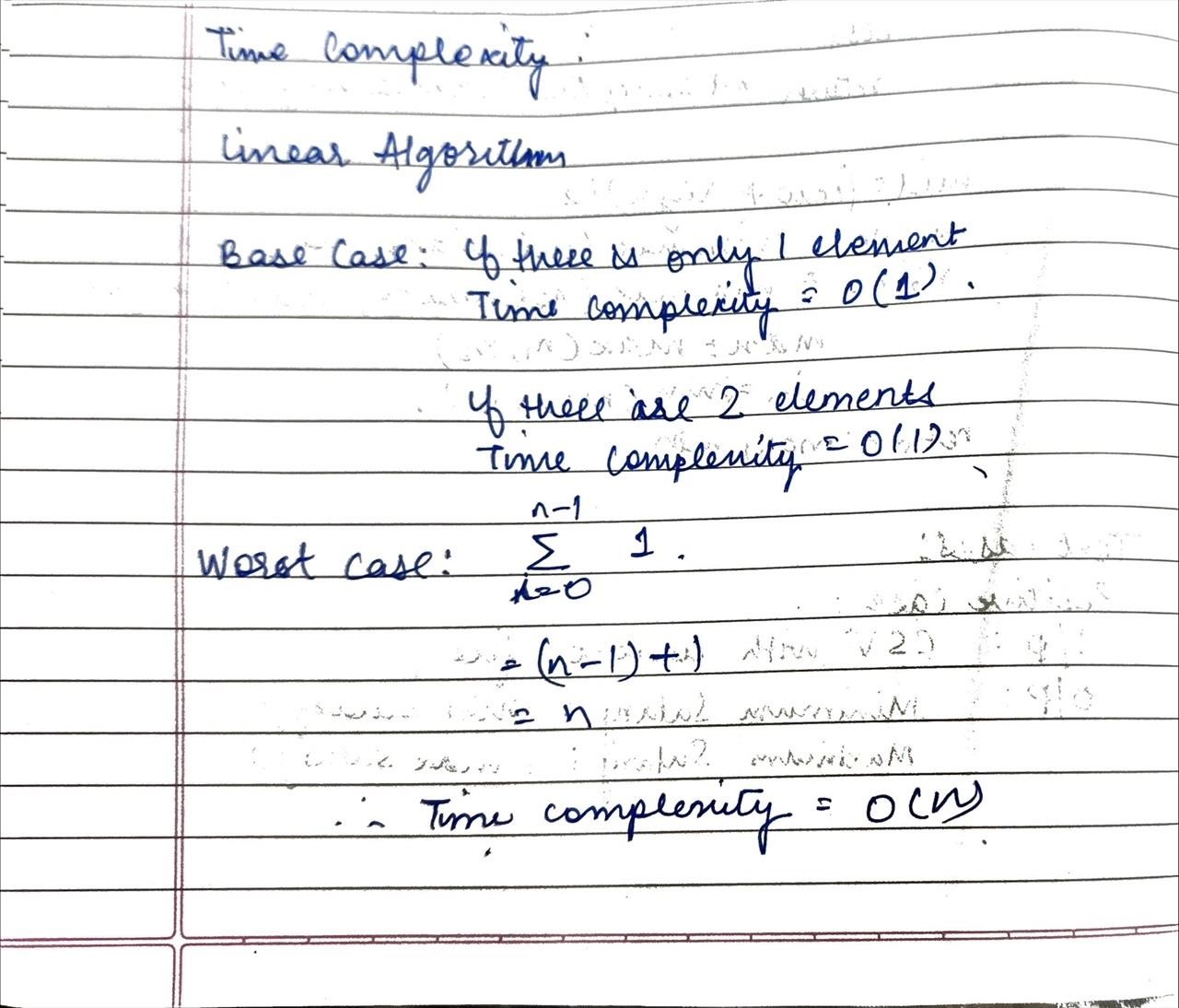


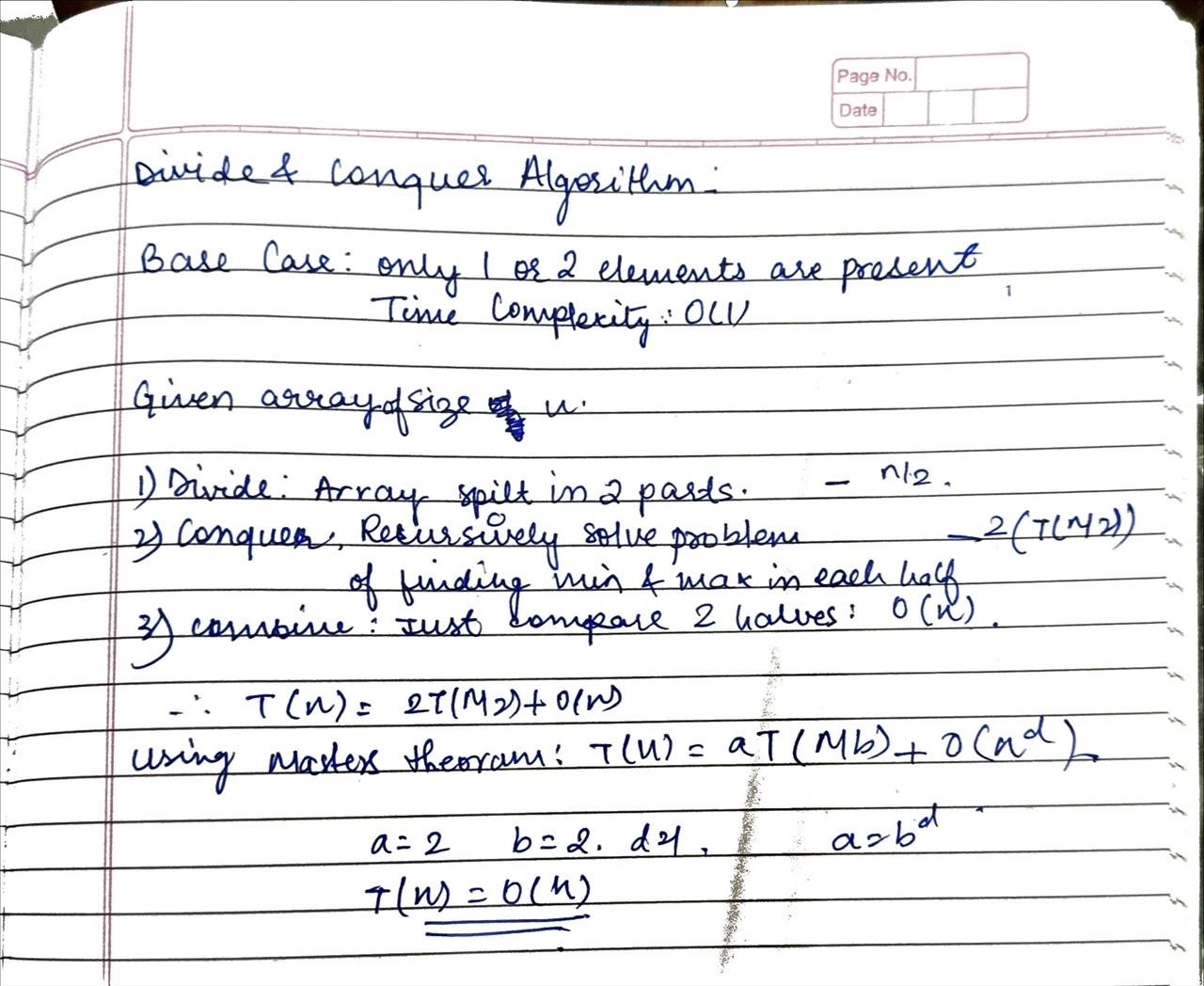
**Formulas:**

Gross Salary=Basic Salary + Allowances+ Bonuses

Net Salary: Gross Salary - Deductions

**Time Complexity:**





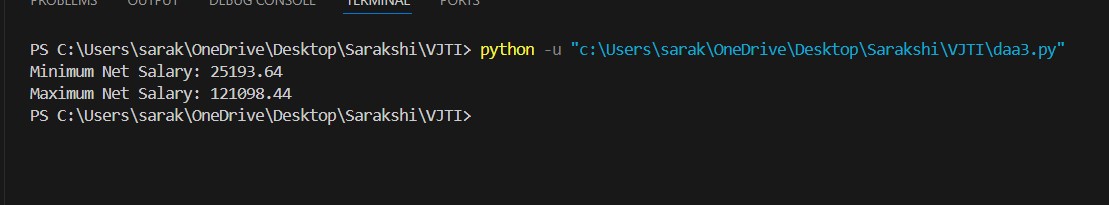
**Code:**

|  |
| --- |
| import csv  class data\_analysis:  def gross\_sal(s, basic, metro, other):  s.HRA = basic \* 0.05 if metro else basic \* 0.04  s.PF = basic \* 0.12  s.gross = basic + s.HRA + other - s.PF return s.gross  def net\_sal(s, gross, inc\_tax):  s.net = gross - inc\_tax return s.net  def extract\_minmax(s, arr):  s.min\_val = float('inf')  s.max\_val = float('-inf') for \_, net in arr: if net < s.min\_val: |

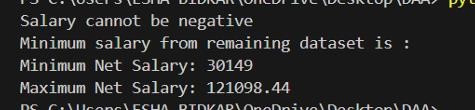
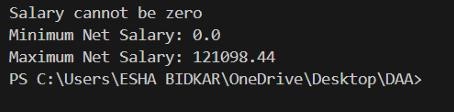
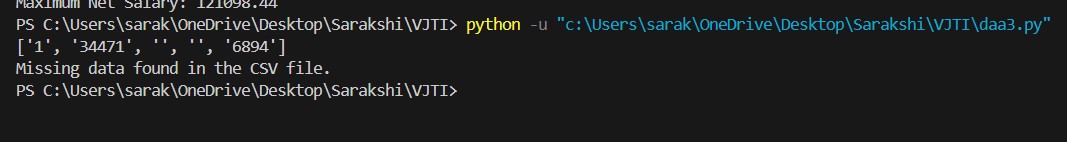
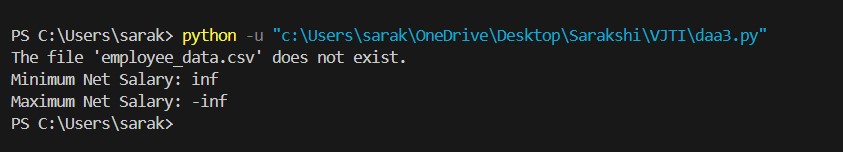
|  |
| --- |
| s.min\_val = net if net > s.max\_val:  s.max\_val = net return s.min\_val, s.max\_val  def check\_missing\_data(s, file): missing\_data\_found = False  for row in file: if any(cell == '' or cell is None for cell in row):  missing\_data\_found = True print(row) break  return missing\_data\_found  def data\_is\_zero(s, file):  for row in file : if int(row[1]) == 0:  print("Salary cannot be zero") break  def data\_is\_negative(s, file): for row in file : if int(row[1]) < 0:  print("Salary cannot be negative") break  def data\_is\_ne\_min(s, file):  non\_negative = [ int(i[1]) for i in file if int(i[1]) > 0 ] if non\_negative :  min\_non = min(non\_negative) return min\_non def main():  obj1 = data\_analysis()  input\_file = 'employee\_data.csv' output\_file = 'datatest.csv'    net\_sort = []  try: with open(input\_file, 'r') as fin:  reader = csv.reader(fin) rows = list(reader) # Check for missing data if obj1.check\_missing\_data(rows): |
| print("Missing data found in the CSV file.") return obj1.data\_is\_zero(rows) obj1.data\_is\_negative(rows) for line in rows: empID = line[0] basic = float(line[1]) metro = line[2] == 'YES' other = float(line[3]) inc\_tax = float(line[4])  gross = obj1.gross\_sal(basic, metro, other) net = obj1.net\_sal(gross, inc\_tax)  net\_sort.append((empID, net))  except FileNotFoundError:  print(f"The file '{input\_file}' does not exist.")  with open(output\_file, 'w', newline='') as fout:  writer = csv.writer(fout) for empID, net in net\_sort:  writer.writerow([empID, obj1.gross\_sal(basic, metro, other), net])  min\_net, max\_net = obj1.extract\_minmax(net\_sort)  if min\_net < 0 :  print("Minimum salary from remaining dataset is :") m = obj1.data\_is\_ne\_min(rows) print(f"Minimum Net Salary: {m}") else :  print(f"Minimum Net Salary: {min\_net}") print(f"Maximum Net Salary: {max\_net}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Output:**

**Positive:**



**Negative:**



**Conclusion:** In this experiment, we have seen that the divide and conquer algorithm has time complexity of O(n). In both linear and divide and conquer algorithm, divide and conquer algorithm is more preferable. Divide and conquer method is easy to handle in case of large dataset as it divides the problem into smaller subproblems.