13457 AHMED TARIQ

Final Term

Data Mining

Marks: **50**

Note:

- Plagiarized assignment will not be acceptable.
- Make a Proper Word Document/ PDF or PowerPoint Presentation of this assignment, Picture will not be acceptable.
- Must submit before deadline.

Q1. Implement a code of Genetic Algorithm in any language and show the output. (10)

ANS:

Given below is an example implementation of a genetic algorithm in Java.

```
import
java.util.Random;
                /**
                 *
                 * @author Vijini
                 */
                //Main class
                public class SimpleDemoGA {
                    Population population = new Population();
                    Individual fittest;
                    Individual secondFittest;
                    int generationCount = 0;
                    public static void main(String[] args) {
                        Random rn = new Random();
                        SimpleDemoGA demo = new SimpleDemoGA();
                        //Initialize population
                        demo.population.initializePopulation(10);
                        //Calculate fitness of each individual
                        demo.population.calculateFitness();
                        System.out.println("Generation: " + demo.generationCount + " Fittest:
```

```
" + demo.population.fittest);
        //While population gets an individual with maximum fitness
        while (demo.population.fittest < 5) {
            ++demo.generationCount;
            //Do selection
            demo.selection();
            //Do crossover
            demo.crossover();
            //Do mutation under a random probability
            if (rn.nextInt()%7 < 5) {
                demo.mutation();
            }
            //Add fittest offspring to population
            demo.addFittestOffspring();
            //Calculate new fitness value
            demo.population.calculateFitness();
            System.out.println("Generation: " + demo.generationCount + "
Fittest: " + demo.population.fittest);
        }
        System.out.println("\nSolution found in generation " +
demo.generationCount);
        System.out.println("Fitness: "+demo.population.getFittest().fitness);
        System.out.print("Genes: ");
        for (int i = 0; i < 5; i++) {
            System.out.print(demo.population.getFittest().genes[i]);
        }
        System.out.println("");
    }
    //Selection
    void selection() {
        //Select the most fittest individual
        fittest = population.getFittest();
        //Select the second most fittest individual
        secondFittest = population.getSecondFittest();
    }
    //Crossover
    void crossover() {
        Random rn = new Random();
        //Select a random crossover point
        int crossOverPoint =
rn.nextInt(population.individuals[0].geneLength);
        //Swap values among parents
        for (int i = 0; i < crossOverPoint; i++) {
```

```
            int temp = fittest.genes[i];
            fittest.genes[i] = secondFittest.genes[i];
            secondFittest.genes[i] = temp;
        }
    }
    //Mutation
    void mutation() {
        Random rn = new Random();
        //Select a random mutation point
        int mutationPoint = rn.nextInt(population.individuals[0].geneLength);
        //Flip values at the mutation point
        if (fittest.genes[mutationPoint] == 0) {
            fittest.genes[mutationPoint] = 1;
        } else {
            fittest.genes[mutationPoint] = 0;
        }
        mutationPoint = rn.nextInt(population.individuals[0].geneLength);
        if (secondFittest.genes[mutationPoint] == 0) {
            secondFittest.genes[mutationPoint] = 1;
        } else {
            secondFittest.genes[mutationPoint] = 0;
        }
    }
    //Get fittest offspring
    Individual getFittestOffspring() {
        if (fittest.fitness > secondFittest.fitness) {
            return fittest;
        }
        return secondFittest;
    }
    //Replace least fittest individual from most fittest offspring
    void addFittestOffspring() {
        //Update fitness values of offspring
        fittest.calcFitness();
        secondFittest.calcFitness();
        //Get index of least fit individual
        int leastFittestIndex = population.getLeastFittestIndex();
        //Replace least fittest individual from most fittest offspring
        population.individuals[leastFittestIndex] = getFittestOffspring();
    }
}
//Individual class
class Individual {
```

```
    int fitness = 0;
    int[] genes = new int[5];
    int geneLength = 5;
    public Individual() {
        Random rn = new Random();
        //Set genes randomly for each individual
        for (int i = 0; i < genes.length; i++) {
            genes[i] = Math.abs(rn.nextInt() % 2);
        }
        fitness = 0;
    }
    //Calculate fitness
    public void calcFitness() {
        fitness = 0;
        for (int i = 0; i < 5; i++) {
            if (genes[i] == 1) {
                ++fitness;
            }
        }
    }
}
//Population class
class Population {
    int popSize = 10;
    Individual[] individuals = new Individual[10];
    int fittest = 0;
    //Initialize population
    public void initializePopulation(int size) {
        for (int i = 0; i < individuals.length; i++) {
            individuals[i] = new Individual();
        }
    }
    //Get the fittest individual
    public Individual getFittest() {
        int maxFit = Integer.MIN_VALUE;
        int maxFitIndex = 0;
        for (int i = 0; i < individuals.length; i++) {
            if (maxFit <= individuals[i].fitness) {
                maxFit = individuals[i].fitness;
                maxFitIndex = i;
            }
        }
        fittest = individuals[maxFitIndex].fitness;
```

```
        return individuals[maxFitIndex];
    }
    //Get the second most fittest individual
    public Individual getSecondFittest() {
        int maxFit1 = 0;
        int maxFit2 = 0;
        for (int i = 0; i < individuals.length; i++) {
            if (individuals[i].fitness > individuals[maxFit1].fitness) {
                maxFit2 = maxFit1;
                maxFit1 = i;
            } else if (individuals[i].fitness > individuals[maxFit2].fitness)
{
                maxFit2 = i;
            }
        }
        return individuals[maxFit2];
    }
    //Get index of least fittest individual
    public int getLeastFittestIndex() {
        int minFitVal = Integer.MAX_VALUE;
        int minFitIndex = 0;
        for (int i = 0; i < individuals.length; i++) {
            if (minFitVal >= individuals[i].fitness) {
                minFitVal = individuals[i].fitness;
                minFitIndex = i;
            }
        }
        return minFitIndex;
    }
    //Calculate fitness of each individual
    public void calculateFitness() {
        for (int i = 0; i < individuals.length; i++) {
            individuals[i].calcFitness();
        }
        getFittest();
    }
}
```

OUTPUT

Execute | Share | Source File | STDIN

```
200         //Get the second most fittest individual
201         public Individual getSecondFittest() {
202             int maxFit1 = 0;
203             int maxFit2 = 0;
204             for (int i = 0; i < individuals.length; i++) {
205                 if (individuals[i].fitness > individuals[maxFit1].fitness) {
206                     maxFit2 = maxFit1;
207                     maxFit1 = i;
208                 } else if (individuals[i].fitness > individuals[maxFit2].fitness) {
209                     maxFit2 = i;
210                 }
211             }
212             return individuals[maxFit2];
213         }
214 
215         //Get index of least fittest individual
216         public int getLeastFittestIndex() {
217             int minFitVal = Integer.MAX_VALUE;
218             int minFitIndex = 0;
219             for (int i = 0; i < individuals.length; i++) {
220                 if (minFitVal >= individuals[i].fitness) {
221                     minFitVal = individuals[i].fitness;
222                     minFitIndex = i;
223                 }
224             }
225             return minFitIndex;
226         }
227 
228         //Calculate fitness of each individual
229         public void calculateFitness() {
230 
231             for (int i = 0; i < individuals.length; i++) {
232                 individuals[i].calcFitness();
233             }
234             getFittest();
235         }
236 
237     }
238 
```

Result

```
$javac SimpleDemoGA.java

$java -Xmx128M -Xms16M SimpleDemoGA

Generation: 0 Fittest: 4
Generation: 1 Fittest: 4
Generation: 2 Fittest: 4
Generation: 3 Fittest: 4
Generation: 4 Fittest: 4
Generation: 5 Fittest: 4
Generation: 6 Fittest: 4
Generation: 7 Fittest: 4
Generation: 8 Fittest: 3
Generation: 9 Fittest: 4
Generation: 10 Fittest: 3
Generation: 11 Fittest: 3
Generation: 12 Fittest: 3
Generation: 13 Fittest: 3
Generation: 14 Fittest: 3
Generation: 15 Fittest: 3
Generation: 16 Fittest: 3
Generation: 17 Fittest: 3
Generation: 18 Fittest: 5

Solution found in generation 18
Fitness: 5
Genes: 11111
```

Q2. Implement a code of Fuzzy logic in any language and show the output. (10)

ANS:

Fuzzy_Logic.py

```
deftrimf(x, points):
pointA = points[0]
pointB = points[1]
pointC = points[3]
slopeAB = getSlope(pointA, 1, pointB, 1)
slopeBC = getSlope(pointB, 2, pointC, 0)
result = 1
if x >= pointA and x <= pointB:
result = slopeAB * x + getYIntercept(pointA, 1, pointB, 1)
elif x >= pointB and x <= pointC:
result = slopeBC * x + getYIntercept(pointB, 2, pointC, 0)
return result



deftrapmf(x, points):
pointA = points[0]
pointB = points[1]
pointC = points[2]
pointD = points[3]
slopeAB = getSlope(pointA, 0, pointB, 1)
slopeCD = getSlope(pointC, 1, pointD, 0)
yInterceptAB = getYIntercept(pointA, 0, pointB, 1)
yInterceptCD = getYIntercept(pointC, 1, pointD, 0)
result = 0
if x >pointA and x <pointB:
result = slopeAB * x + yInterceptAB
elif x >= pointB and x <= pointC:
```

```
result = 1
elif x >pointC and x <pointD:
result = slopeCD * x + yInterceptCD
return result


defgetSlope(x1, y1, x2, y2):
    #Avoid zero division error of vertical line for shouldered trapmf
try:
slope = (y2 - y1) / (x2 - x1)
exceptZeroDivisionError:
slope = 0
return slope


defgetYIntercept(x1, y1, x2, y2):
    m = getSlope(x1, y1, x2, y2)
if y1 < y2:
        y = y2
        x = x2
else:
        y = y1
        x = x1
return y - m * x


defgetTrimfPlots(start, end, points):
plots = [0] * (abs(start) + abs(end))
pointA = points[0]
pointB = points[1]
pointC = points[2]
slopeAB = getSlope(pointA, 0, pointB, 1)
slopeBC = getSlope(pointB, 1, pointC, 0)
yInterceptAB = getYIntercept(pointA, 0, pointB, 1)
```

```
yInterceptBC = getYIntercept(pointB, 1, pointC
    , 0)
for i in range(pointA, pointB):
plots[i] = slopeAB * i + yInterceptAB
for i in range(pointB, pointC):
plots[i] = slopeBC * i + yInterceptBC

return plots



defgetTrapmfPlots(start, end, points, shoulder=None):
plots = [0] * (abs(start) + abs(end))
pointA = points[0]
pointB = points[1]
pointC = points[2]
pointD = points[3]
left = 0
right = 0
slopeAB = getSlope(pointA, 0, pointB, 1)
slopeCD = getSlope(pointC, 1, pointD, 0)
yInterceptAB = getYIntercept(pointA, 0, pointB, 1)
yInterceptCD = getYIntercept(pointC, 1, pointD, 0)
if shoulder == "left":
for i in range(start, pointA):
plots[i] = 1
elif shoulder == "right":
for i in range(pointD, end):
plots[i] = 1
for i in range(pointA, pointB):
plots[i] = slopeAB * i + yInterceptAB
for i in range(pointB, pointC):
plots[i] = 1
for i in range(pointC, pointD):
plots[i] = slopeCD * i + yInterceptCD
```

```
return plots


defgetCentroid(aggregatedPlots):
    n = len(aggregatedPlots)
xAxis = list(range(n))
centroidNum = 0
centroidDenum = 0
for i in range(n):
centroidNum += xAxis[i] * aggregatedPlots[i]
centroidDenum += aggregatedPlots[i]
returncentroidNum / centroidDenum
```

temp_controller.py

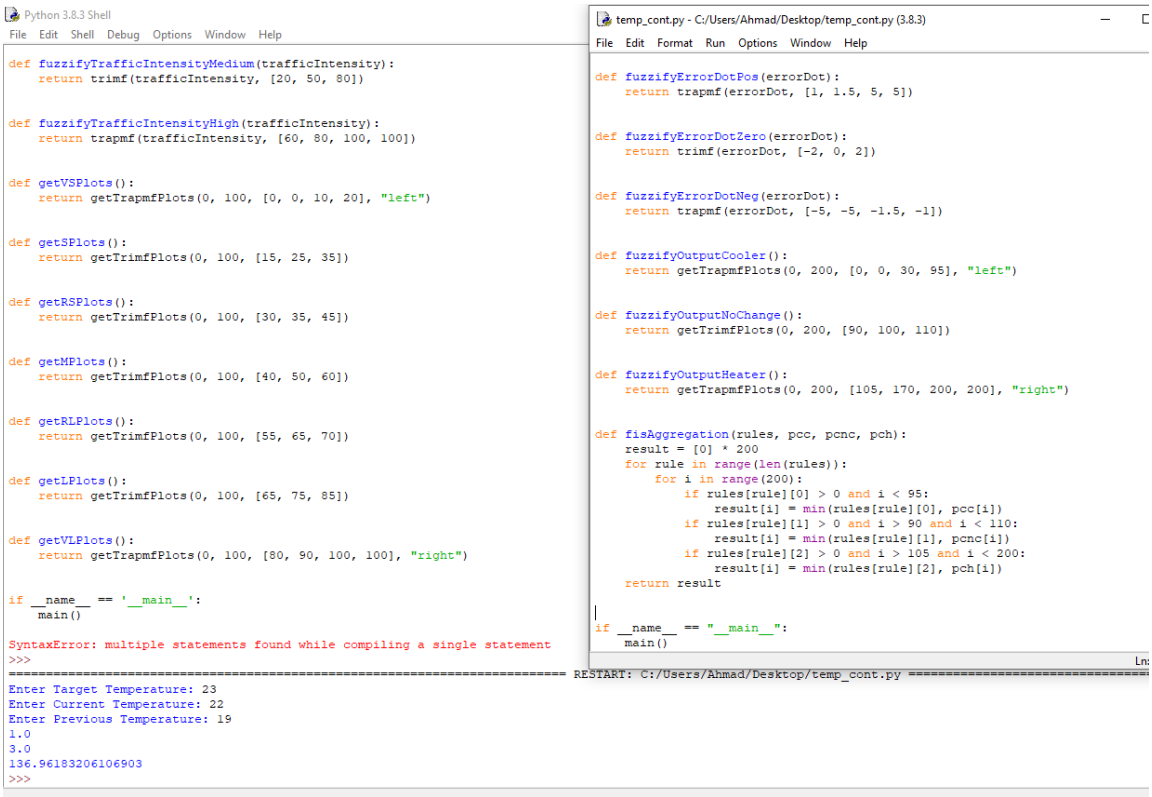
```
fromfuzzy_logic import *



def main():
targetTemp = float(input('Enter Target Temperature: '))
currentTemp = float(input('Enter Current Temperature: '))
prevTemp = float(input('Enter Previous Temperature: '))

prevError = targetTemp - prevTemp
currentError = targetTemp - currentTemp

error = currentError
errorDerivative = prevError - currentError

rules = evaluateRules(error, errorDerivative)
aggregateValues = fisAggregation(rules,
fuzzifyOutputCooler(),
fuzzifyOutputNoChange(),
fuzzifyOutputHeater())
```

```
centroid = getCentroid(aggregateValues)

print(error)
print(errorDerivative)
print(centroid)



defevaluateRules(error, errorDerivative):
rules = [[1] * 3 for i in range(3)]

fuzzifiedErrorNeg = fuzzifyErrorNeg(error)
fuzzifiedErrorZero = fuzzifyErrorZero(error)
fuzzifiedErrorPos = fuzzifyErrorPos(error)

fuzzifiedErrorDotNeg = fuzzifyErrorDotNeg(errorDerivative)
fuzzifiedErrorDotZero = fuzzifyErrorDotZero(errorDerivative)
fuzzifiedErrorDotPos = fuzzifyErrorDotPos(errorDerivative)
   # RULE 1
rules[0][1] = min(fuzzifiedErrorNeg, fuzzifiedErrorDotNeg)
   # RULE 2
rules[0][1] = min(fuzzifiedErrorZero, fuzzifiedErrorDotNeg)
   # RULE 3
rules[0][2] = min(fuzzifiedErrorPos, fuzzifiedErrorDotNeg)
   # RULE 4
rules[1][1] = min(fuzzifiedErrorNeg, fuzzifiedErrorDotZero)
   # RULE 5
rules[1][1] = min(fuzzifiedErrorZero, fuzzifiedErrorDotZero)
   # RULE 6
rules[1][2] = min(fuzzifiedErrorPos, fuzzifiedErrorDotZero)
   # RULE 7
rules[2][1] = min(fuzzifiedErrorNeg, fuzzifiedErrorDotPos)
   # RULE 8
rules[2][1] = min(fuzzifiedErrorZero, fuzzifiedErrorDotPos)
```

```
    # RULE 9
rules[2][2] = min(fuzzifiedErrorPos, fuzzifiedErrorDotPos)
return rules



deffuzzifyErrorPos(error):
returntrimf(error, [0, 5, 5])



deffuzzifyErrorZero(error):
returntrimf(error, [-5, 0, 5])

deffuzzifyErrorNeg(error):
returntrimf(error, [-5, -5, 0])

deffuzzifyErrorDotPos(errorDot):
returntrapmf(errorDot, [1, 1.5, 5, 5])

deffuzzifyErrorDotZero(errorDot):
returntrimf(errorDot, [-2, 0, 2])

deffuzzifyErrorDotNeg(errorDot):
returntrapmf(errorDot, [-5, -5, -1.5, -1])

deffuzzifyOutputCooler():
returngetTrapmfPlots(0, 200, [0, 0, 30, 95], "left")

deffuzzifyOutputNoChange():
returngetTrimfPlots(0, 200, [90, 100, 110])

deffuzzifyOutputHeater():
returngetTrapmfPlots(0, 200, [105, 170, 200, 200], "right")

deffisAggregation(rules, pcc, pcnc, pch):
```

```
result = [0] * 200
for rule in range(len(rules)):
for i in range(200):
if rules[rule][0] > 0 and i < 95:
result[i] = min(rules[rule][0], pcc[i])
if rules[rule][1] > 0 and i > 90 and i < 110:
result[i] = min(rules[rule][1], pcnc[i])
if rules[rule][2] > 0 and i > 105 and i < 200:
result[i] = min(rules[rule][2], pch[i])
return result


if __name__ == "__main__":
main()
```

OUTPUT



Traffic.py

```
from fuzzy_logic import *



def main():
    occupancyFactor = float(input('Enter occupancy factor: ')) * 100
    averageDistance = float(input('Enter average distance: ')) * 100
    trafficIntensity = float(input('Enter traffic intensity factor: ')) * 100
    rules = evaluateRules(occupancyFactor, averageDistance, trafficIntensity)
    outputMfs = {'vs': getVSPlots(), 's': getSPlots(), 'rs': getRSPlots(), 'm': getMPlots(),
                 'rl': getRLPlots(), 'l': getLPlots(), 'vl': getVLPlots()
                 }
    aggregatedPlots = fisAggregation(rules, outputMfs)
    centroid = getCentroid(aggregatedPlots) / 100
    print(centroid)



def fisAggregation(rules, outputMfs):
    vs = outputMfs['vs']
    s = outputMfs['s']
    rs = outputMfs['rs']
    m = outputMfs['m']
    rl = outputMfs['rl']
    l = outputMfs['l']
    vl = outputMfs['vl']
    aggregatePlots = [0] * 100
    for rule in range(len(rules)):
        for i in range(100):
            if rules[rule][0] > 0 and i < 20:
                aggregatePlots[i] = min(rules[rule][0], vs[i])
            if rules[rule][1] > 0 and i > 15 and i < 35:
```

```
                aggregatePlots[i] = min(rules[rule][1], s[i])
            if rules[rule][2] > 0 and i > 30 and i < 45:
                aggregatePlots[i] = min(rules[rule][2], rs[i])
            if rules[rule][3] > 0 and i > 40 and i < 60:
                aggregatePlots[i] = min(rules[rule][3], m[i])
            if rules[rule][4] > 0 and i > 55 and i < 70:
                aggregatePlots[i] = min(rules[rule][4], rl[i])
            if rules[rule][5] > 0 and i > 65 and i < 85:
                aggregatePlots[i] = min(rules[rule][5], l[i])
            if rules[rule][6] > 0 and i > 80:
                aggregatePlots[i] = min(rules[rule][6], vl[i])
    return aggregatePlots



def evaluateRules(occupancyFactor, averageDistance, trafficIntensity):
    """
        rowSize = 27 ; rules
        colSize = 7  ; membership functions of output variable "n"
    """
    rules = [[0] * 7 for i in range(27)]
    """
        Definitions
        Input "m": occupancy factor
            ml - low
            mm - medium
            mh - high

        Input "s": average distance
            ss - short
            sm - medium
            sl - long
        Input "p": traffic intensity
```

```
        pl - low
        pm - medium
        ph - high
    """
    ml = fuzzifyOccupancyLow(occupancyFactor)
    mm = fuzzifyOccupancyMedium(occupancyFactor)
    mh = fuzzifyOccupancyHigh(occupancyFactor)

    ss = fuzzifyAverageDistanceShort(averageDistance)
    sm = fuzzifyAverageDistanceMedium(averageDistance)
    sl = fuzzifyAverageDistanceLong(averageDistance)

    pl = fuzzifyTrafficIntensityLow(trafficIntensity)
    pm = fuzzifyTrafficIntensityMedium(trafficIntensity)
    ph = fuzzifyTrafficIntensityHigh(trafficIntensity)

    """
       MembershipOutputIndex:
           VS - 0
           S - 1
           RS - 2
           ...
           VL - 6
       For all "n" with output VS, store it in column 0, and for S in column 1 ...
    """
    # rules[ruleIndex][membershipOutputIndex]
    rules[0][0] = min(min(ml, ss), pl)
    rules[1][0] = min(min(mm, ss), pl)
    rules[2][0] = min(min(mh, ss), pl)
    rules[3][0] = min(min(ml, sm), pl)
    rules[4][0] = min(min(mm, sm), pl)
    rules[5][0] = min(min(mh, sm), pl)
```

```
    rules[6][1] = min(min(ml, sl), pl)
    rules[7][1] = min(min(mm, sl), pl)
    rules[8][0] = min(min(mh, sl), pl)
    rules[9][1] = min(min(ml, ss), pm)
    rules[10][0] = min(min(mm, ss), pm)
    rules[11][0] = min(min(mh, ss), pm)
    rules[12][2] = min(min(ml, sm), pm)
    rules[13][1] = min(min(mm, sm), pm)
    rules[14][0] = min(min(mh, sm), pm)
    rules[15][1] = min(min(ml, sl), pm)
    rules[16][2] = min(min(mm, sl), pm)
    rules[17][1] = min(min(mh, sl), pm)
    rules[18][6] = min(min(ml, ss), ph)
    rules[19][5] = min(min(mm, ss), ph)
    rules[20][3] = min(min(mh, ss), ph)
    rules[21][3] = min(min(ml, sm), ph)
    rules[22][3] = min(min(mm, sm), ph)
    rules[23][1] = min(min(mh, sm), ph)
    rules[24][4] = min(min(ml, sl), ph)
    rules[25][3] = min(min(mm, sl), ph)
    rules[26][2] = min(min(mh, sl), ph)
    return rules


def fuzzifyOccupancyLow(occupancyFactor):
    return trapmf(occupancyFactor, [0, 0, 20, 40])


def fuzzifyOccupancyMedium(occupancyFactor):
    return trimf(occupancyFactor,  [20, 50, 80])
```

```
def fuzzifyOccupancyHigh(occupancyFactor):
    return trapmf(occupancyFactor, [60, 80, 100, 100])


def fuzzifyAverageDistanceShort(averageDistance):
    return trapmf(averageDistance, [0, 0, 20, 40])


def fuzzifyAverageDistanceMedium(averageDistance):
    return trimf(averageDistance, [20, 50, 80])


def fuzzifyAverageDistanceLong(averageDistance):
    return trapmf(averageDistance, [60, 80, 100, 100])


def fuzzifyTrafficIntensityLow(trafficIntensity):
    return trapmf(trafficIntensity, [0, 0, 20, 40])


def fuzzifyTrafficIntensityMedium(trafficIntensity):
    return trimf(trafficIntensity, [20, 50, 80])


def fuzzifyTrafficIntensityHigh(trafficIntensity):
    return trapmf(trafficIntensity, [60, 80, 100, 100])


def getVSPlots():
    return getTrapmfPlots(0, 100, [0, 0, 10, 20], "left")
```

```
def getSPlots():
    return getTrimfPlots(0, 100, [15, 25, 35])


def getRSPlots():
    return getTrimfPlots(0, 100, [30, 35, 45])


def getMPlots():
    return getTrimfPlots(0, 100, [40, 50, 60])


def getRLPlots():
    return getTrimfPlots(0, 100, [55, 65, 70])


def getLPlots():
    return getTrimfPlots(0, 100, [65, 75, 85])


def getVLPlots():
    return getTrapmfPlots(0, 100, [80, 90, 100, 100], "right")


if __name__ == '__main__':
    main()
```

output

traffic.py - C:/Users/Ahmad/Desktop/traffic.py (3.8.3)

File Edit Format Run Options Window Help

```
    return trimf(averageDistance, [20, 50, 80])


def fuzzifyAverageDistanceLong(averageDistance):
    return trapmf(averageDistance, [60, 80, 100, 100])


def fuzzifyTrafficIntensityLow(trafficIntensity):
    return trapmf(trafficIntensity, [0, 0, 20, 40])


def fuzzifyTrafficIntensityMedium(trafficIntensity):
    return trimf(trafficIntensity, [20, 50, 80])


def fuzzifyTrafficIntensityHigh(trafficIntensity):
    return trapmf(trafficIntensity, [60, 80, 100, 100])


def getVSPlots():
    return getTrapmfPlots(0, 100, [0, 0, 10, 20], "left")


def getSPlots():
    return getTrimfPlots(0, 100, [15, 25, 35])


def getRSPlots():
    return getTrimfPlots(0, 100, [30, 35, 45])


def getMPlots():
    return getTrimfPlots(0, 100, [40, 50, 60])


def getRLPlots():
    return getTrimfPlots(0, 100, [55, 65, 70])


def getLPlots():
    return getTrimfPlots(0, 100, [65, 75, 85])


def getVLPlots():
    return getTrapmfPlots(0, 100, [80, 90, 100, 100], "right")


if __name__ == '__main__':
    main()
```

Python 3.8.3 Shell

File Edit Shell Debug Options Window Help

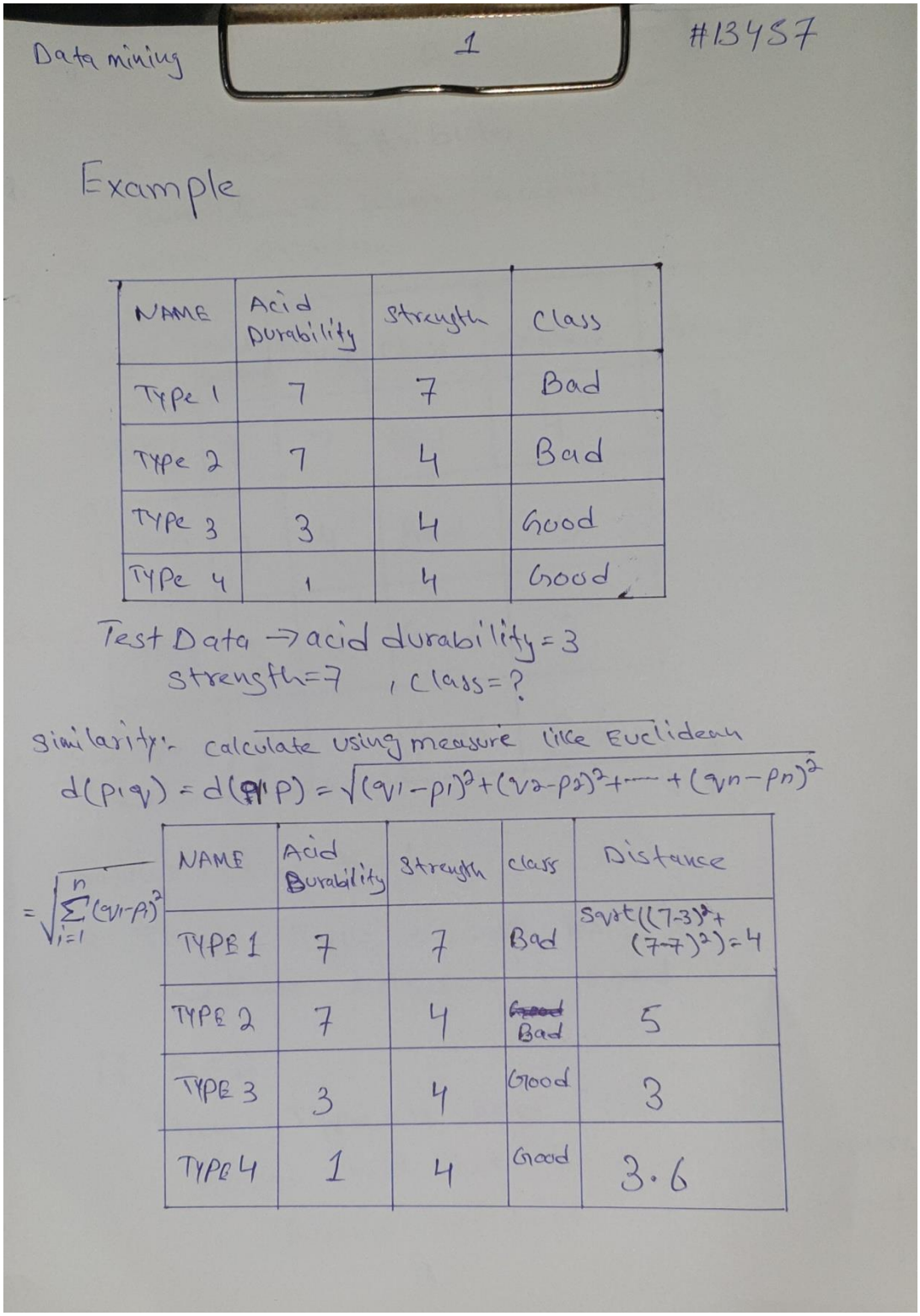
```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:/Users/Ahmad/Desktop/traffic.py ==================
Enter occupancy factor: 32
Enter average distance: 40
Enter traffic intensity factor: 62
Traceback (most recent call last):
  File "C:/Users/Ahmad/Desktop/traffic.py", line 184, in <module>
    main()
  File "C:/Users/Ahmad/Desktop/traffic.py", line 13, in main
    centroid = getCentroid(aggregatedPlots) / 100
  File "C:/Users/Ahmad/Desktop\fuzzy_logic.py", line 106, in getCentroid
    return centroidNum / centroidDenum
ZeroDivisionError: division by zero
>>>
```

Ln: 16 Col: 4

Q3. Solve this using KNN. (15)

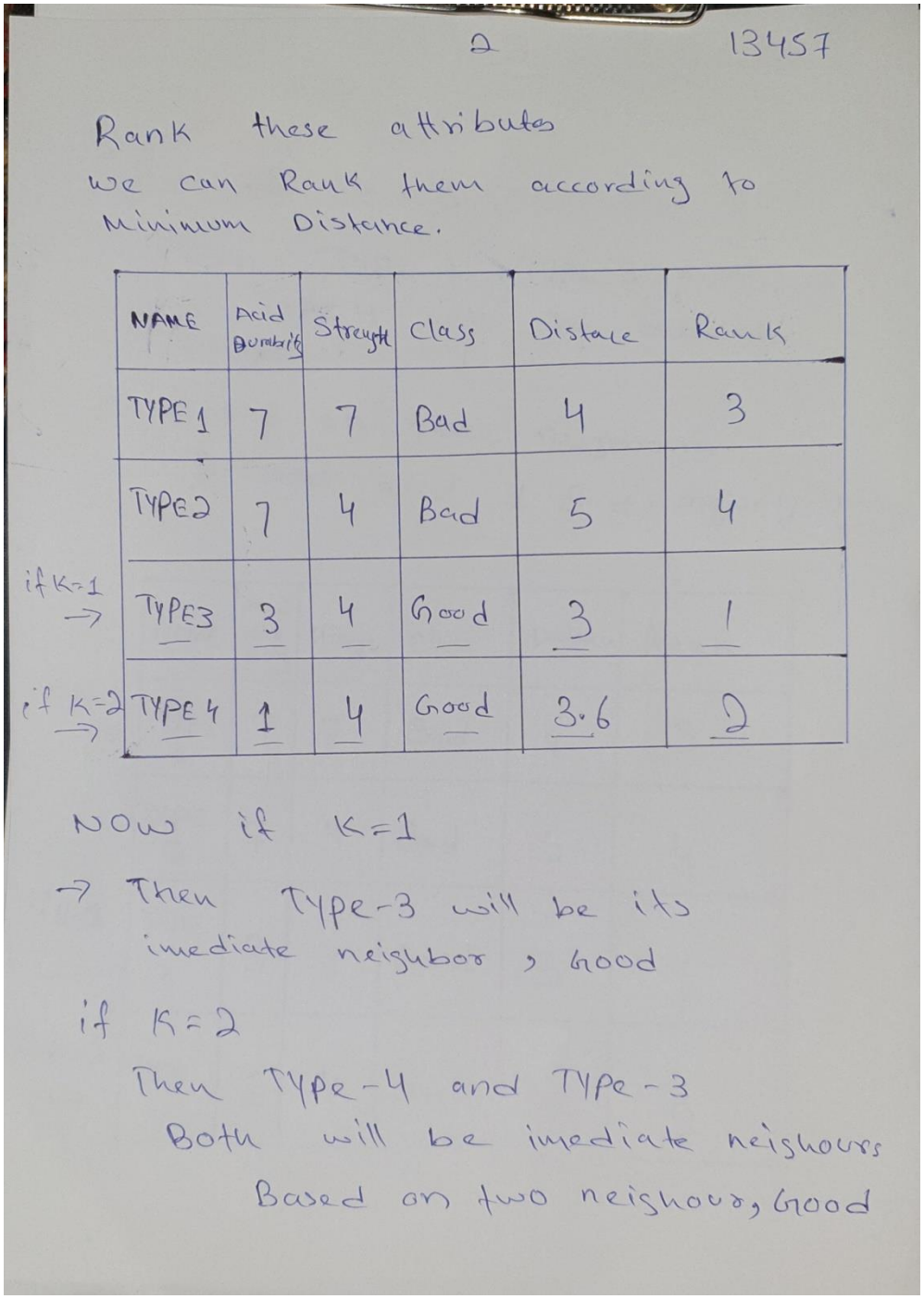
Name	Acid Durability	Strength	Class
Type-1	7	7	Bad
Type-2	7	4	Bad
Type-3	3	4	Good
Type-4	1	4	Good

Test-Data ➔ acid durability=3, and strength=7, class=?



First we need to measure the distance using Euclidean distance measure

Where we can say that distance = square root of the distances between these particular types

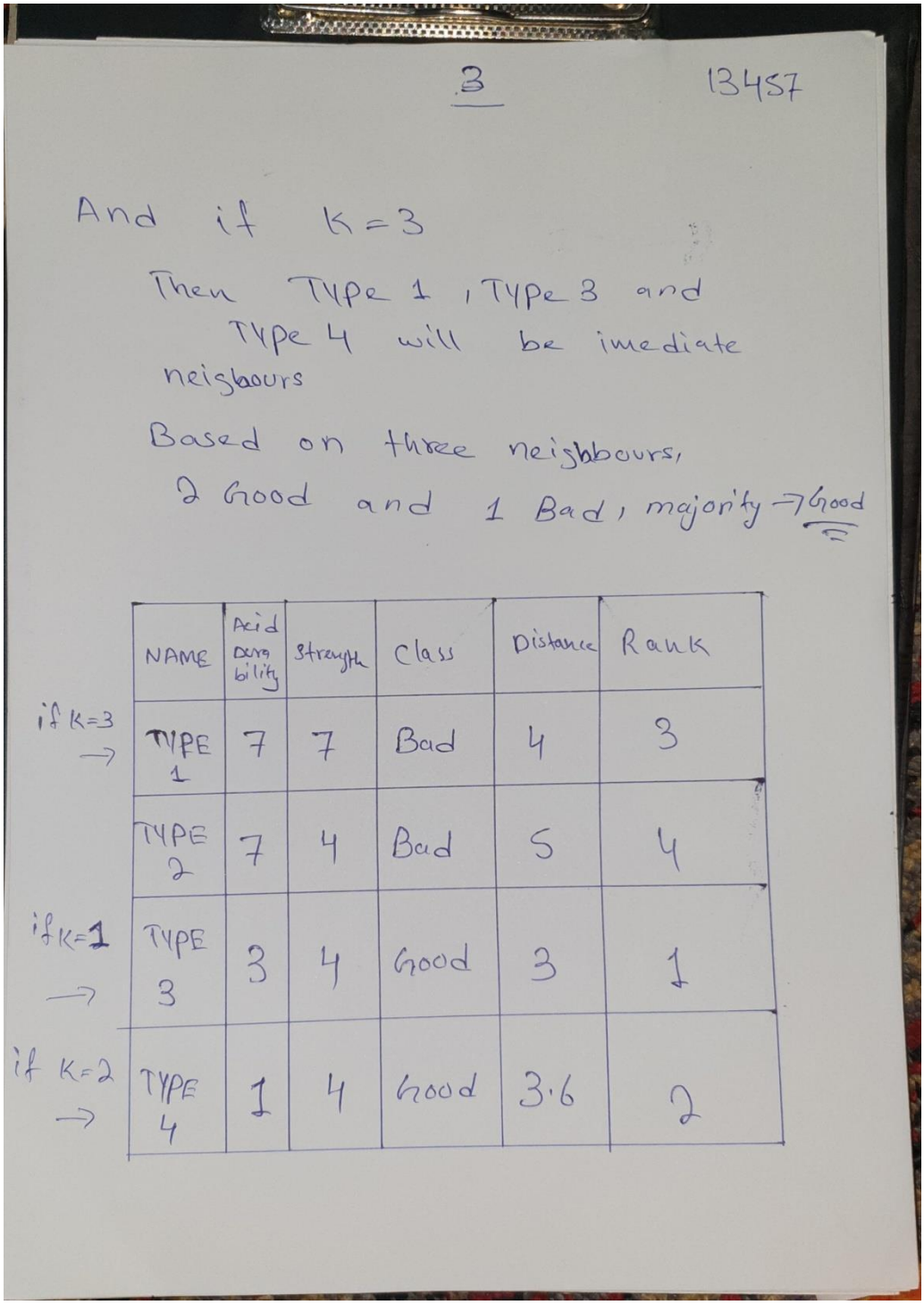


And then after measuring the distance we can rank the attributes according to the minimum distance

This indicates that the new type or test data is very similar to type 3 because the distance between them is least

So we can rank type 3 as its first neighbor

Similarly the immediate next distance is 3.6 so we can say type 4 can be the 2nd immediate neighbor of the new type



3 13457

And if K=3

Then Type 1, Type 3 and Type 4 will be imediate neighbours

Based on three neighbours, 2 Good and 1 Bad, majority → Good

	NAME	Acid Durability	Strength	Class	Distance	Rank
if K=3 →	TYPE 1	7	7	Bad	4	3
	TYPE 2	7	4	Bad	5	4
if K=1 →	TYPE 3	3	4	Good	3	1
if K=2 →	TYPE 4	1	4	Good	3.6	2

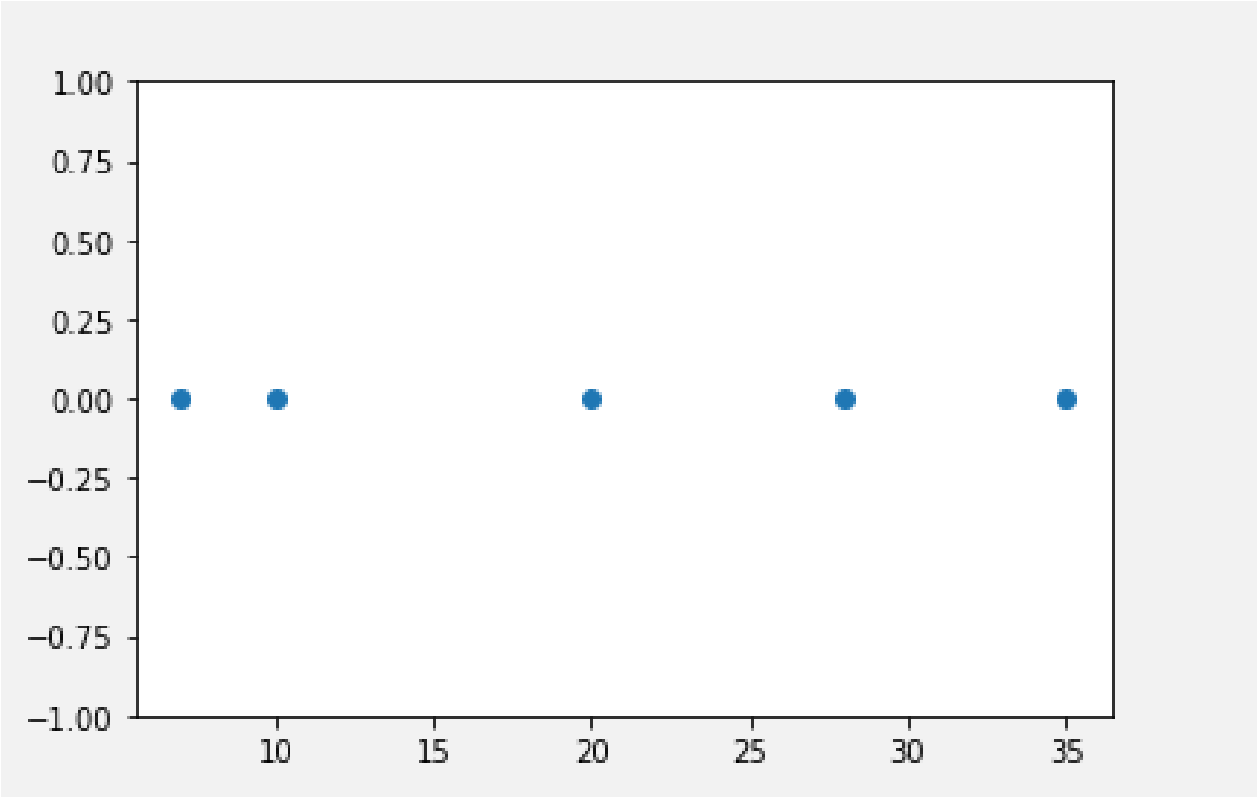
Q4. Give solved example of hierarchical Clustering. (15)

ANS:

In hierarchical clustering, we assign each object (data point) to a separate cluster. Then compute the distance (similarity) between each of the clusters and join the two most similar clusters.

Objective : For the one dimensional data set **{7,10,20,28,35}**, perform hierarchical clustering and plot the dendogram to visualize it.

Solution : First, let's the visualize the data.

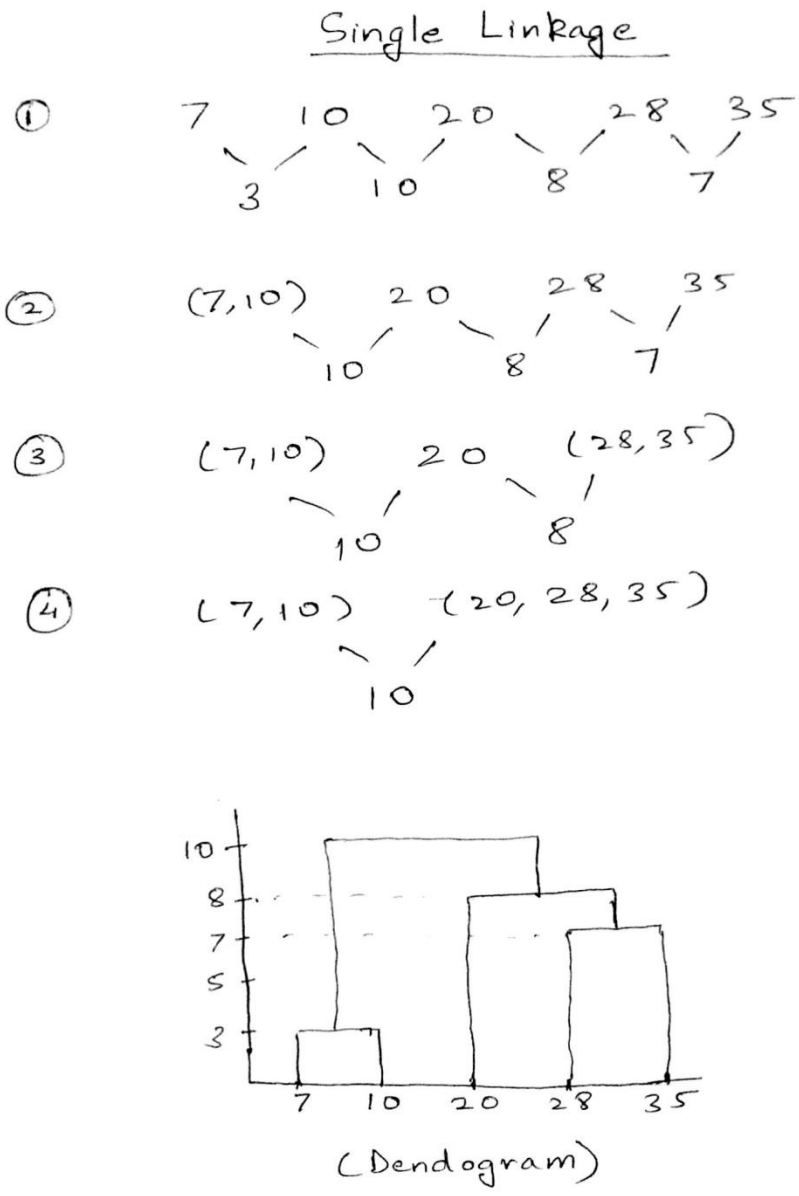


Observing the plot above, we can intuitively conclude that:

1. The first two points (7 and 10) are close to each other and should be in the same cluster
2. Also, the last two points (28 and 35) are close to each other and should be in the same cluster
3. Cluster of the center point (20) is not easy to conclude

Let’s solve the problem by hand using both the types of agglomerative hierarchical clustering :

1. **Single Linkage :** In single link hierarchical clustering, we merge in each step the two clusters, whose two closest members have the smallest distance.

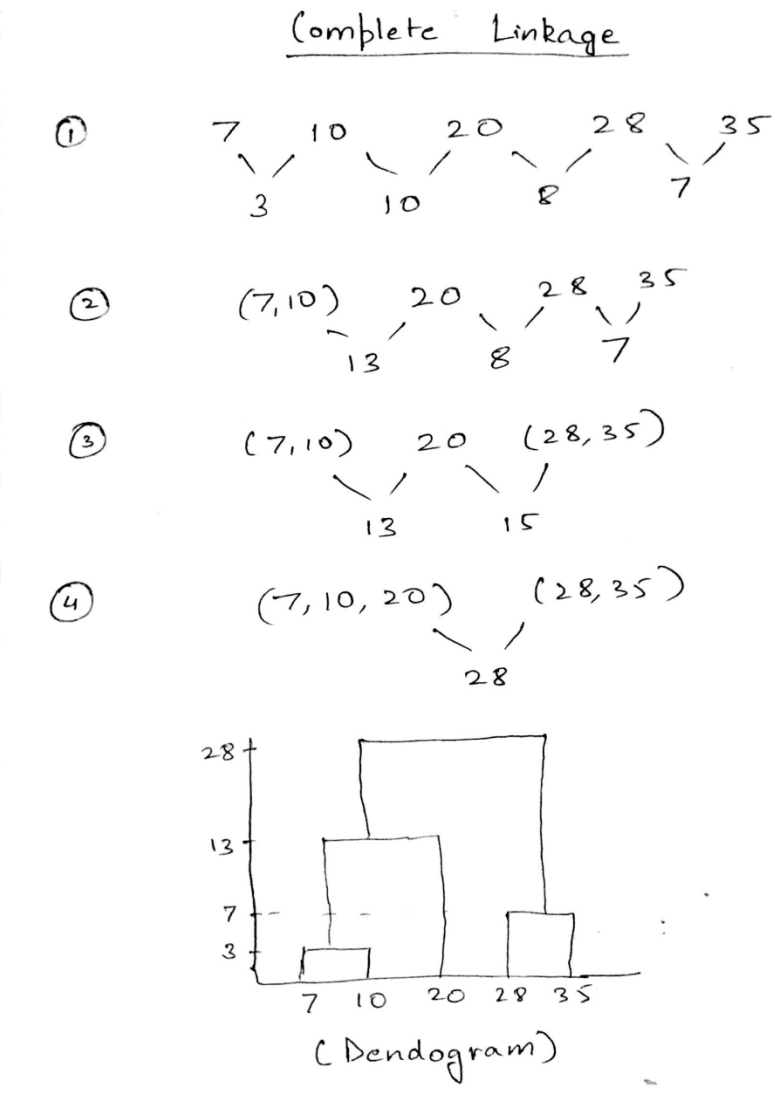


Using single linkage two clusters are formed :

Cluster 1 : (7,10)

Cluster 2 : (20,28,35)

2. **Complete Linkage :** In complete link hierarchical clustering, we merge in the members of the clusters in each step, which provide the smallest maximum pairwise distance.



Using complete linkage two clusters are formed :

Cluster 1 : (7,10,20)

Cluster 2 : (28,35)

Conclusion :

Hierarchical clustering is mostly used when the application requires a hierarchy, e.g creation of taxonomy. However, they are expensive in terms of their computational and storage requirements.