­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Контрольна робота №1

з курсу «Теорія Алгоритмів»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 14

Виконав студент гр. КНз-2

Чалий Михайло

­­

Львів 2014

# Завдання №1

Здійснити розпізнавання образів із застосуванням евристичного алгоритму порогової величини. Порогова величина Т=2.

X1(1,2), X2(1,3), X3(1,4), X4(1,5), X5(2,2), X6(2,3), X7(2,4), X8(3,7), X9(3,8), X10(3,9)

## Рішення

import sys, math, random

class Point(object):

def \_\_init\_\_(self, x, y):

self.\_x = x

self.\_y = y

def distanceTo(self, other):

return math.sqrt(math.pow(self.\_x - other.\_x, 2) + math.pow(self.\_y - other.\_y, 2))

def \_\_repr\_\_(self):

return "(" + str(self.\_x) + "; " + str(self.\_y) + ")"

class Clusterizable(object):

def \_\_init\_\_(self, points):

self.\_points = points

self.\_centers = []

self.\_clusters = []

self.\_t = 0;

def dump\_points(self):

print(self.\_points)

def dump\_cluster(self):

print("Clusted with T=" + str(self.\_t) + " :")

for cluster\_index in range(0, len(self.\_clusters)):

center = self.\_centers[cluster\_index]

cluster = self.\_clusters[cluster\_index]

print("#" + str(cluster\_index + 1) + ": " + str(center) + " [" + str(cluster) + "]")

def cluster(self, t):

self.\_t = t

for next in self.\_points:

next\_handled = False

for cluster\_index in range(0, len(self.\_clusters)):

center = self.\_centers[cluster\_index]

distance = center.distanceTo(next)

if distance < t:

# Create new cluster

self.\_clusters[cluster\_index].append(next)

next\_handled = True;

if next\_handled == False:

# Create new cluster

self.\_centers.append(next)

self.\_clusters.append([next])

from clust import \*

s = Clusterizable([

Point(1,2),

Point(1,3),

Point(1,4),

Point(1,5),

Point(2,2),

Point(2,3),

Point(2,4),

Point(3,7),

Point(3,8),

Point(3,9)

])

s.dump\_points()

s.cluster(2)

s.dump\_cluster()

## Результат

[(1; 2), (1; 3), (1; 4), (1; 5), (2; 2), (2; 3), (2; 4), (3; 7), (3; 8), (3; 9)]

Clusted with T=2 :

#1: (1; 2) [[(1; 2), (1; 3), (2; 2), (2; 3)]]

#2: (1; 4) [[(1; 4), (1; 5), (2; 3), (2; 4)]]

#3: (3; 7) [[(3; 7), (3; 8)]]

#4: (3; 9) [[(3; 9)]]

# Завдання №2

Здійснити розпізнавання образів із застосуванням максимінного алгоритму.

## Рішення

import sys, math, random

class Point(object):

def \_\_init\_\_(self, x, y):

self.\_x = x

self.\_y = y

def distanceTo(self, other):

return math.sqrt(math.pow(self.\_x - other.\_x, 2) + math.pow(self.\_y - other.\_y, 2))

def \_\_repr\_\_(self):

return "(" + str(self.\_x) + "; " + str(self.\_y) + ")"

class Clusterizable(object):

def \_\_init\_\_(self, points):

self.\_points = points

self.\_z\_points = []

def dump\_points(self):

print(self.\_points)

def dump\_cluster(self):

print("Cluster centers with 0.5 thresold:")

print(self.\_z\_points)

def cluster(self):

self.\_z\_points = [self.\_points[0]]

max\_distances = []

while True:

other\_points = filter(lambda x: x not in self.\_z\_points, self.\_points)

def row\_min\_distance(other\_point):

min\_distance = 10000 # Should be max int

min\_point = None

for z\_point in self.\_z\_points:

item\_distance = other\_point.distanceTo(z\_point)

if item\_distance < min\_distance:

min\_distance = item\_distance

min\_point = other\_point

return (min\_point, min\_distance)

def col\_max\_distance(other\_points):

max\_distance = 0

max\_point = None

for other\_point in other\_points:

row\_distance = row\_min\_distance(other\_point)

item\_distance = row\_distance[1]

if item\_distance > max\_distance:

max\_distance = item\_distance

max\_point = row\_distance[0]

return (max\_point, max\_distance)

result = col\_max\_distance(other\_points)

max\_distance = result[1]

max\_point = result[0]

if len(max\_distances) == 0 or (max\_distance > min(max\_distances) / 2):

max\_distances.append(max\_distance)

self.\_z\_points.append(max\_point)

else:

return

## Результат

[(1; 2), (1; 3), (1; 4), (1; 5), (2; 2), (2; 3), (2; 4), (3; 7), (3; 8), (3; 9)]

Cluster centers with 0.5 thresold:

[(1; 2), (3; 9)]