**Πολυδιάστατες Δομές Δεδομένων**

**ΘΕΜΑ 14 : Voronoi Clustering**

Ονοματεπώνυμο : Αδαμόπουλος Κωνσταντίνος

ΑΜ : 236270 (1043760)

Έτος : 7ο

**Σημείωση:** Στο τέλος αναφέρονται **οδηγίες** σχετικά με τα modules που πρέπει να εγκατασταθούν πριν την εκτέλεση του προγράμματος καθώς και τον τρόπο με τον οποίο θα μπορείτε να εκτελέσετε το πρόγραμμα.

**Υλοποίηση 2D Voronoi clustering /diagram**

Για να δημιουργήσω το διάγραμμα ***Voronoi*** Ν σημείων στον δισδιάστατο χώρο υπολόγισα την τριγωνοποίηση ***Delaunay*** των σημείων χρησιμοποιώντας τον αλγόριθμο των ***Bowyer - Watson*** και έπειτα γνωρίζοντας ότι η τριγωνοποίηση ***Delaunay*** είναι ο δυϊκός γράφος του διαγράμματος ***Voronoi*** δημιουργώ το διάγραμμα με κορυφές τα κέντρα των τριγώνων και ακμές μεταξύ αυτών δημιουργούνται μόνο όταν κάποιο τρίγωνο μοιράζεται κάποια ακμή με κάποιο άλλο τρίγωνο(γειτονικά τρίγωνα). Παρακάτω παρουσιάζονται οι πηγαίοι κώδικες για την υλοποίηση του αλγορίθμου της τριγωνοποίησης ***Delaunay*** καθώς και της δημιουργίας των διαγραμμάτων ***Voronoi*** μαζί με τα απαραίτητα σχόλια .

Η τριγωνοποίηση βρίσκεται στο αρχείο ***triangulation.py*** και ο κώδικας παρουσιάζεται παρακάτω :

'''

Υλοποίηση του Bowyer–Watson incremental algorithm ωστε να

υπολογίζω την Delaunay Triangulation για Ν σημεία στον 2D χώρο

'''

**import** **matplotlib.pyplot** **as** **plt**

**import** **matplotlib**

**from** **math** **import** inf, sqrt

**import** **numpy** **as** **np**

**class** **Delaunay**:

**def** **\_\_init\_\_**(self,center=(**0**,**0**),radius=**1000000**):

center = np.asarray(center)

self.coords = [center+radius\*np.array((-**1**,-**1**)),

center+radius\*np.array((**1**,-**1**)),

center+radius\*np.array((**1**,**1**)),

center+radius\*np.array((-**1**,**1**))]

self.triangles = {}

self.circles = {}

# counter clock wise τριγωνα

# δύο υπερ-τρίγωνα

T1 = (**0**,**1**,**3**)

T2 = (**2**,**3**,**1**)

self.triangles[T1] = [T2,**None**,**None**]

self.triangles[T2] = [T1,**None**,**None**]

**for** triangle **in** self.triangles:

self.circles[triangle] = self.circumcenter(triangle)

**def** **circumcenter**(self,triangle):

points = np.asarray([self.coords[v] **for** v **in** triangle]) #3x2 μητρωο σημεια των κορυφών του τριγωνου

# 3x3 μητρωο εσωτερικου γινομένου μεταξυ των κορυφών η διαγωνιος δινει το μετρο και οι γραμμεσ,στηλεσ με i!=j δινουν τις σχεσεις καθετοτητας ή μη των κορυφων.

points2 = np.dot(points,points.T)

A = np.bmat([[**2**\*points2,[[**1**],

[**1**],

[**1**]]],

[[[**1**,**1**,**1**,**0**]]]]) # μπλοκ μητρωο του μητρωου εσωτερικου γινομένου και των κορυφών

b = np.hstack((np.sum(points\*points,axis=**1**),[**1**])) # διανυσμα b που περιέχει το μέτρο και το 1 για την τελευταια προσθετη στήλη στο μοκ μητρωο

x = np.linalg.solve(A,b) # υπολογισμος βαρυκεντρων συντεταγμένων λ .

bcoords = x[:-**1**]

center = np.dot(bcoords,points) # μετατροπή βαρυκεντρων συντεταγμένων σε καρτεσιανές

radius = np.linalg.norm(points[**0**] - center) # ακτινα

**return** (center, radius)

**def** **inCircle**(self,triangle,p):

center, radius = self.circles[triangle]

**return** np.linalg.norm(center - p) <= radius

**def** **addPoint**(self,point):

point = np.asarray(point)

idx = len(self.coords)

self.coords.append(point)

bad\_triangles = []

**for** triangle **in** self.triangles: # αν υπαρχουν σημεια μεσα στον κυκλο τοτε είναι "κακό" τρίγωνο

**if** self.inCircle(triangle,point):

bad\_triangles.append(triangle)

boundary = []

triangle = bad\_triangles[**0**] # "τυχαίο τρίγωνο"

**if** bad\_triangles[**0**]==**None**:

triangle = bad\_triangles[**1**]

edge = **0** # "τυχαία" edge

**while** **True**:

triangle\_opposite = self.triangles[triangle][edge] # γειτονικο τρίγωνο

**if** triangle\_opposite **not** **in** bad\_triangles:

boundary.append((triangle[(edge+**1**) % **3**],triangle[(edge-**1**) % **3**],triangle\_opposite)) # ακμη και το τριγωνο στο οποιο "συνορευει- είναι κοινή"

edge = (edge+**1**) % **3**

**if** boundary[**0**][**0**] == boundary[-**1**][**1**]:

**break**

**else**:

# Μετακινηση στην επόμενη CCW ακμή στο απέναντι τρίγωνο

edge = (self.triangles[triangle\_opposite].index(triangle)+**1**) % **3**

triangle = triangle\_opposite # επόμενος γείτονας

**for** triangle **in** bad\_triangles:

**del** self.triangles[triangle]

**del** self.circles[triangle]

new\_triangles=[]

**for** (e0, e1, triangle\_opposite) **in** boundary:

triangle = (idx, e0, e1) # νεο τρίγωνο με το σημείο p και τις κορυφες της ακμής (e0,e1) που συνορεύουν με το triangle\_opposite

self.circles[triangle] = self.circumcenter(triangle)

self.triangles[triangle] = [triangle\_opposite, **None**, **None**] # θέτω το απέναντι τρίγωνο γειτονα του τριγώνου

# προσπαθω να θέσω γειτονα του απέναντι τριγωνου το νεο τρίγωνο

**if** triangle\_opposite:

**for** i, neigh **in** enumerate(self.triangles[triangle\_opposite]):

**if** neigh:

**if** e1 **in** neigh **and** e0 **in** neigh:

self.triangles[triangle\_opposite][i] = triangle

new\_triangles.append(triangle)

# ενωνω τα τριγωνα μεταξυ τους (σχεση γειτνιασης)

N = len(new\_triangles)

**for** i, triangle **in** enumerate(new\_triangles):

self.triangles[triangle][**1**] = new\_triangles[(i+**1**) % N] # next

self.triangles[triangle][**2**] = new\_triangles[(i-**1**) % N] # prev

**def** **getInfo**(self):

**return** self.triangles, self.circles, self.coords

**def** **plotTriangles**(self,points,triangles,radius):

x, y = zip(\*points)

print(x)

print(y)

bounds = [min(x),max(x),min(y),max(y)]

fig, ax = plt.subplots()

ax.margins(**0.1**)

ax.set\_aspect('equal')

plt.axis([bounds[**0**]-**1**, bounds[**1**]+**1**, bounds[**2**]-**1**, bounds[**3**]+**1**])

ax.triplot(matplotlib.tri.Triangulation(x,y,triangles),'bo--')

plt.show()

Και ο κώδικας για την δημιουργία του Voronoi διαγράμματος βρίσκεται στο αρχείο ***Voronoi2D.py*** και παρουσιάζεται παρακάτω .

**import** **matplotlib.pyplot** **as** **plt**

**import** **matplotlib**

**from** **matplotlib.patches** **import** Polygon

**from** **matplotlib.collections** **import** PatchCollection

**import** **numpy** **as** **np**

**from** **core.triangulation** **import** Delaunay

**class** **Voronoi2D**:

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

x, y = zip(\*points)

self.bounds = [min(x),max(x),min(y),max(y)]

center = np.mean(points,axis=**0**)

self.d = Delaunay(center) # Αρχικοποιώ την τριγωνοποίηση (δημιουργία super triangles)

self.triangles = []

self.points = points

self.voronoiPoints = []

self.centers = []

self.voronoiRegions = {}

self.coords = []

**def** **start**(self):

# τριγωνοποιηση για κάθε νέο σημείο που εισάγεται (incremental εκτέλεση του αλγορίθμου τριγωνοποιησης)

**for** point **in** self.points:

self.d.addPoint(point)

self.triangles, self.centers, self.coords = self.d.getInfo() # παιρνω τα τριγωνα, τους κυκλους και στα σημεία απο την τριγωνοποίηση

triangleUseVertex = {i:[] **for** i **in** range(len(self.coords))}

triangleIndex = {}

**for** tidx, (a,b,c) **in** enumerate(sorted(self.triangles)):

self.voronoiPoints.append(self.centers[(a,b,c)][**0**])

# Περιστρέφω το υπαρχων τρίγωνο CCW ανάλογα με την κορυφή αναφοράς/ με την κορυφή που θέλουμε να είναι τελευταία .

# Αυτό γίνεται ωστε να έχω την σωστή σειρά αναφοράς των κορυφών του τριγώνου ανάλογα με το σημείο που χρησιμοποιείται εκείνη την στιγμή .

# Καποια κορυφή μπορεί να είναι και σε παραπάνω απο 1 τρίγωνα -> αρα το σημείο θα είναι κοινό για τουλάχιστον 2 τρίγωνα .

triangleUseVertex[a] += [(b,c,a)]

triangleUseVertex[b] += [(c,a,b)]

triangleUseVertex[c] += [(a,b,c)]

# Εδώ ορίζονται τα παραπάνω τρίγωνα με ένα id το οποίο είναι το ίδιο με το id της του τριγώνου που χρησιμοποιύμε .

# Οπότε με αυτο τον τρόπο δείχνω οτι τα παραπάνω τρίγωνα είναι ουσιαστικά τα ίδια αλλα περιστραμένα ανάλογα του σημείου με CCW φορά.

triangleIndex[(a,b,c)] = tidx

triangleIndex[(c,a,b)] = tidx

triangleIndex[(b,c,a)] = tidx

**for** point\_idx **in** range(**4**,len(self.coords)):

vertex = triangleUseVertex[point\_idx][**0**][**0**]

r = []

**for** \_ **in** range(len(triangleUseVertex[point\_idx])):

t = [t **for** t **in** triangleUseVertex[point\_idx] **if** t[**0**]==vertex][**0**] # Τρίγωνο που ξεκινάει με την κορυφή vertex και περιέχει και το σημειο αρα έχουν κοινη ακμή .

r.append(triangleIndex[t]) # προσθέτω το id του τριγωνου που ξεκινά με την κορυφη vertex, γειτονικα τριγωνα

# η λιστα r θα περιέχει τα indexes των τριγωνων που περιέχουν το σημείο αλλα και που έχουν κοινή ακμή με αυτο .

vertex = t[**1**] # επομενη κορυφη

self.voronoiRegions[point\_idx-**4**] = r

**return** self.voronoiPoints, self.voronoiRegions

**def** **plotVoronoi**(self,fill=**True**,city=**None**):

'''

fill : Αν True τοτε τα πολύγωνα θα είναι με χρώμα (default), αλλιως θα είναι μόνο οι γραμμές των πολυγώνων .

city : None τιμη αν θες να έχει μόνο τα σημεία στο plot(default), αλλιως μεταβλητη η οποία θα έχει σχέση με τα δεδομένα που έχουν εισαχθέι

(χρησιμοποιειται μονο στα γεωγραφικα δεδομενα)

'''

fig, ax = plt.subplots(num="Voronoi Diagram")

ax.margins(**0.1**)

ax.set\_aspect('equal')

plt.axis([self.bounds[**0**]-**1**, self.bounds[**1**]+**1**, self.bounds[**2**]-**1**, self.bounds[**3**]+**1**])

**if** city **is** **None**:

**if** **not** fill:

**for** point **in** self.points:

plt.plot(point[**0**],point[**1**],'rx')

**for** region **in** self.voronoiRegions:

poly = [self.voronoiPoints[i] **for** i **in** self.voronoiRegions[region]]

plt.plot(\*zip(\*poly),color='black')

**else**:

**for** point **in** self.points:

plt.plot(point[**0**],point[**1**],'kx')

**for** region **in** self.voronoiRegions:

poly = [self.voronoiPoints[i] **for** i **in** self.voronoiRegions[region]]

plt.fill(\*zip(\*poly),alpha=**0.5**)

**if** city **is** **not** **None**:

**if** **not** fill:

**for** point **in** self.points:

plt.plot(point[**0**],point[**1**],'rx')

**for** region **in** self.voronoiRegions:

poly = [self.voronoiPoints[i] **for** i **in** self.voronoiRegions[region]]

plt.plot(\*zip(\*poly),color='black')

city.plot(ax=ax,alpha=**0.3**, edgecolor="black", facecolor="white")

**else**:

**for** point **in** self.points:

plt.plot(point[**0**],point[**1**],'kx')

**for** region **in** self.voronoiRegions:

poly = [self.voronoiPoints[i] **for** i **in** self.voronoiRegions[region]]

plt.fill(\*zip(\*poly),alpha=**0.5**)

city.plot(ax=ax,alpha=**0.6**, edgecolor="black", facecolor="white")

plt.show()

Τα δεδομένα που μπορούν να χρησιμοποιηθούν για την δημιουργία του διαγράμματος Voronoi είναι 2 Datasets με πραγματικά γεωγραφικά δεδομένα καθώς και μια γεννήτρια τυχαίων σημείων , περισσότερα στο αρχείο ***main.py .*** Παρακάτω παρουσιάζεται ο κώδικας που υπάρχει στο αρχείο με πλήρη σχολιασμό σχετικά με τα δεδομένα που χρησιμοποιώ καθώς και τις συναρτήσεις που διαχειρίζονται αυτά τα δεδομένα . By-default το πρόγραμμα θα τρέξει το δεύτερο dataset αλλά μπορείτε να το αλλάξετε εφόσον βγάλετε από τα σχόλια κάποια άλλη από τις συναρτήσεις .

'''

Author : Κωνσταντίνος Αδαμόπουλος

ΑΜ: 236270 (1043750)

Ετος: 7ο

'''

**import** **random**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **geopandas** **as** **gpd**

**from** **core.Voronoi2D** **import** Voronoi2D

**def** **generatePoints**(numSeeds):

'''

Η συνάρτηση αυτή παράγει τυχαία 2D σημεία

το όρισμα numSeeds αφορά τόν αριθμό των σημείων .

'''

radius = **100**

seeds = radius \* np.random.random((numSeeds, **2**))

**return** seeds

**def** **loadGeoDataset1**(size):

'''

Η συναρτηση αυτη φορτώνει την πολιτεια της Νεας Υορκης

και καποια σημεια ενδιαφέροντος και τα χρησιμοποιώ ωστε να αναπαράγω το δίαγραμμα

Voronoi των σημείων αυτων .

Πηγες:

Το αρχείο της μεταβλητής gmap το πήρα απο :

https://tapiquen-sig.jimdofree.com/english-version/free-downloads/united-states/

Το αρχείο της μεταβλητής file το πήρα απο :

https://www1.nyc.gov/site/doitt/residents/gis-2d-data.page

'''

file = "data/dataset1/Points Of Interest/geo\_export\_d771d7a5-ef72-43f8-8b2c-67a3549235c5.shp"

gmap = "data/USA\_States/USA\_States.shp"

points = gpd.read\_file(file)

points = points.to\_crs({"init": "EPSG:4326"}) # μετατρέπω τις συντεταγμένες του αρχείου σε WGS 84

city = gpd.read\_file(gmap)

city = city.to\_crs(points.crs) # μετατρέπω τις συντεταγμένες του αρχείου στο ιδιο format με αυτο της μεταβλητης points

city = city[city["STATE\_NAME"] == "New York"] # παιρνω την γραμμή του dataframe οπου το STATE\_NAME ειναι ισο με New York

points = gpd.sjoin(points,city,how="left") # ενωνω τους πινακες και κραταω τα σημεία που αφορουν την τιμή New York

points = points.dropna(subset=["index\_right"])

numOfPoints = points.shape[**0**]

dataSize = points.shape[**0**]

**if** dataSize > size:

x = points.geometry.x[**0**:size]

y = points.geometry.y[**0**:size]

intrest\_points\_nparr = np.array([[i,j] **for** i,j **in** zip(x,y)])

**return** intrest\_points\_nparr[**0**:size,:], city, size

**else**:

**return** [],[],[],[]

**def** **loadGeoDataset2**():

'''

Η συναρτηση αυτη φορτώνει την πρωτέυουσα που έχει η κάθε πολιτεία

και τις πολιτείες της Αμερικής.

Πηγες:

Το αρχείο της μεταβλητής gmap το πήρα απο :

https://tapiquen-sig.jimdofree.com/english-version/free-downloads/united-states/

Το αρχείο της μεταβλητής file το πήρα απο :

https://tapiquen-sig.jimdofree.com/english-version/free-downloads/united-states/

'''

file = "data/dataset2/USA\_Capitals/USA\_Capitals.shp"

gmap = "data/USA\_States/USA\_States.shp"

points = gpd.read\_file(file)

points = points.to\_crs({"init": "EPSG:4326"}) # μετατρέπω τις συντεταγμένες του αρχείου σε WGS 84

x = points.geometry.x

y = points.geometry.y

capital\_points\_nparr = np.array([[i,j] **for** i,j **in** zip(x,y)])

states = gpd.read\_file(gmap,header=**None**)

states = states.to\_crs(points.crs) # μετατρέπω τις συντεταγμένες του αρχείου στο ιδιο format με αυτο της μεταβλητης points

**return** capital\_points\_nparr, states.geometry

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

# Παραγωγή διαγράμματος Voronoi τυχαίων σημείων

'''

points = generatePoints(100)

vor = Voronoi2D(points)

vor.start()

vor.plotVoronoi()

'''

# Παραγωγή διαγράμματος Voronoi ενος συνόλου των σημείων ενδιαφέροντος που υπάρχουν στην Νεα Υορκη .

'''

points, city, totalSize = loadGeoDataset1(100) # συναρτηση που διαβάζει και επιστρέφει καταλλήλως τα δεδομένα του 1ου dataset

#print('Number of Points:',totalSize)

if len(points)>0:

vor = Voronoi2D(points)

vor.start()

vor.plotVoronoi(city=city)

else:

print('Εχείς εισάγει παραπάνω σημεια απο ότι περιέχονται στο Dataset.')

'''

# Παραγωγή διαγράμματος Voronoi των πρωτευουσών των πολιτειών της Αμερικής.

points, states = loadGeoDataset2() # συναρτηση που διαβάζει και επιστρέφει καταλλήλως τα δεδομένα του 2ου dataset

vor = Voronoi2D(points)

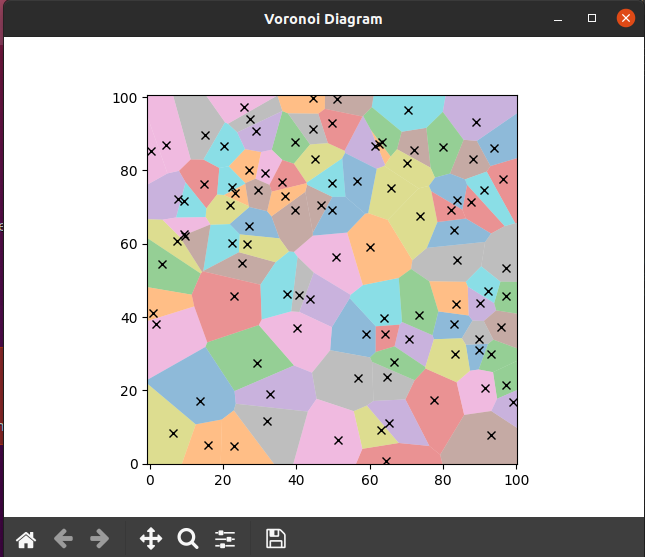
vor.start()

vor.plotVoronoi(city=states)

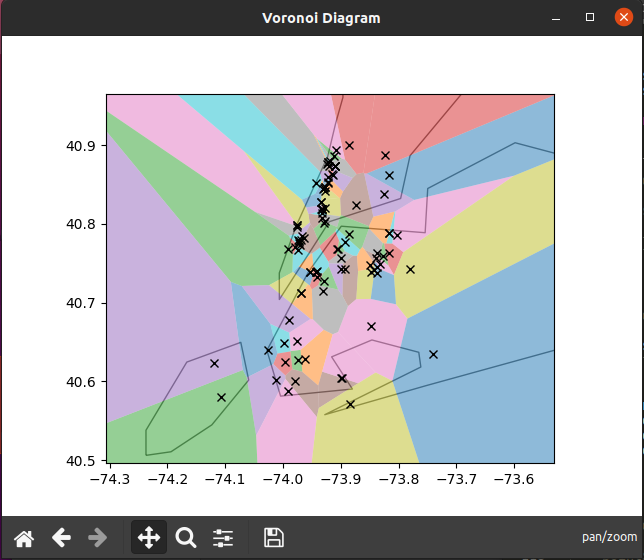
**Αποτελέσματα**

Παρακάτω παρουσιάζονται κάποια screenshots από την εκτέλεση του προγράμματος για τα datasets και τα τυχαία σημεία. Έχουμε:

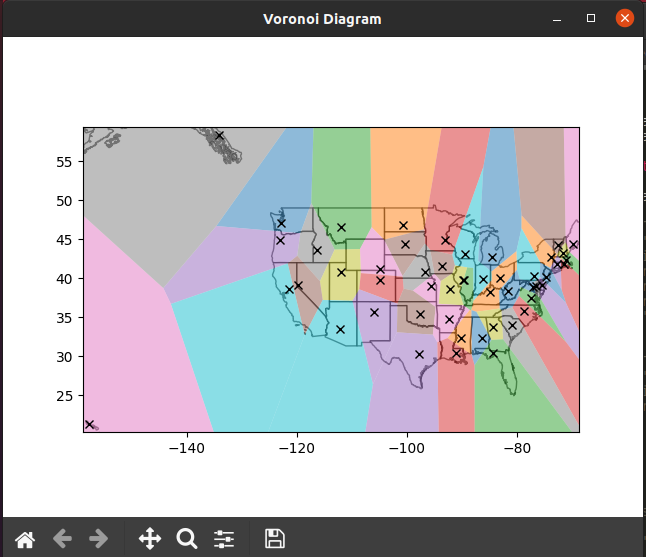
1. Screenshot για 100 τυχαία σημεία



1. Screenshot για το πρώτο dataset για 100 σημεία



1. Screenshot για το δεύτερο dataset(όλα τα σημεία)



**Οδηγίες**

***Σημείωση****: Προτείνεται η χρήση λειτουργικού* ***Linux*** *για την εκτέλεση της εφαρμογής* .

Για να εκτελεστεί σωστά η εφαρμογή θα πρέπει να έχουν εγκατασταθεί τα απαραίτητα modules τα οποία είναι:

* *numpy*
* *geopandas*
* *matplotlib*

Τα οποία μπορείτε να τα εγκαταστήσετε χρησιμοποιώντας την εντολή ***pip install numpy geopandas matplotlib*** ή ***python -m pip install numpy geopandas matplotlib*** και να έχετε μια έκδοση της ***python*** στον υπολογιστή σας η οποία να είναι από έκδοση ***3.6*** και άνω . Για να τρέξετε το πρόγραμμα εκτελέστε το αρχείο **main.py** με την εντολή ***python3 main.py*** .