**Σύνδεση στη βάση δεδομένων**

1 def connect\_to\_mongodb(MONGO\_HOST, MONGO\_PORT, MONGO\_DB, MONGO\_USER, MONGO\_PASSWORD, colllection\_name):

2 client = MongoClient(MONGO\_HOST, MONGO\_PORT)

3 db = client[MONGO\_DB]

4 #db.authenticate(MONGO\_USER, MONGO\_PASSWORD)

5 print("Connection Succeed")

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

return db[colllection\_name]

Η παραπάνω συνάρτηση πραγματοποιεί την σύνδεση στη βάση δεδομένων. Τα ορίσματα της συνάρτησης είναι τα host, port, database name, username, password, collection name. H συνάρτηση επιστρέφει το collection της βάσης, έχοντας πραγματοποιήσει τη σύνδεση σε αυτό, έτοιμο για να χρησιμοποιηθεί από τις υπόλοιπες συναρτήσεις, οι οποίες θα παρουσιαστούν παρακάτω. To # μπροστά από το db.authenticate υποδηλώνει ότι η συγκεκριμένη γραμμή έχει μπει σε σχόλιο, καθώς στα παραδείγματα που θα παρουσιαστούν η σύνδεση στη βάση δεν απαιτεί username και password.

Ένα παράδειγμα χρήσης της παραπάνω συνάρτησης είναι το εξής:

mycol = connect\_to\_mongodb('localhost', 27017, 'trajectories', '', '','trajectory')

Στο παραπάνω παράδειγμα το host ισούται με localhost, το port είναι το 27017, το όνομα της βάσης είναι trajectories, η σύνδεση στην βάση δεν απαιτεί username και password και από τη βάση θα χρησιμοποιηθεί το collection objects.

Η παραπάνω γραμμή μπορεί να χρησιμοποιηθεί είτε μέσα σε κάποια άλλη συνάρτηση, είτε το mycol να χρησιμοποιηθεί σαν όρισμα σε κάποια άλλη συνάρτηση.

**Εύρεση k κοντινότερων γειτόνων**

Αρχικά η συνάρτηση λαμβάνει ως ορίσματα τα (mycol, x, y, k, t\_min, t\_max) που αντιστοιχούν στα εξής: το mycol αντιστοιχεί στη σύνδεση σε κάποιο collection της βάσης, όπως παρουσιάστηκε παραπάνω, x και y είναι οι συντεταγμένες για τις οποίες πρέπει να βρεθούν οι κοντινότεροι γείτονες. k είναι ο αριθμός των κοντινότερων γειτόνων και t\_min, t\_max το χρονικό διάστημα μέσα στο οποίο θα πραγματοποιηθεί η αναζήτηση των κοντινότερων γειτόνων.

Στη συνέχεια καλείται η παραπάνω συνάρτηση, η οποία πραγματοποιεί τη σύνδεση με τη βάση δεδομένων, όπως παρουσιάζεται παρακάτω:

def k\_neghbors(mycol,x,y,k,t\_min,t\_max):

Η λογική που ακολουθείται είναι η εξής: Πρώτα βρίσκονται όλες οι εγγραφές που ικανοποιούν τον χρονικό περιορισμό δηλαδή βρίσκονται μεταξύ t\_min και t\_max και στη συνέχεια υπολογίζονται οι k κοντινότερες εγγραφές στο x και y. Ο υπολογισμός των εγγραφών που βρίσκονται μεταξύ t\_min και t\_max γίνεται με τις εξής εντολές:

query='{"$and":[{"t":{"$gt":'+str(t\_min)+'}}, {"t":{"$lt":'+str(t\_max)+'}}]}'

k1 = json.loads(query)

cursor = mycol.find(k1)

Στη συνέχεια τα αποτελέσματα αποθηκεύονται στη λίστα documents

documents = []

counter = 0

for document in cursor:

documents.insert(counter, document)

counter += 1

Το επόμενο βήμα είναι για κάθε εγγραφή του documents να υπολογιστεί η Ευκλείδια απόσταση από τις παραμέτρους (x, y). Για να πραγματοποιηθεί η παραπάνω διαδικασία, οι παράμετροι (x, y) αποθηκεύονται σε έναν πίνακα b1 και τα στοιχεία (lon, lat) των εγγραφών του document σε έναν πίνακα b2. Με χρήση της συνάρτησης linalg.norm(b1-b2) του πακέτου numpy της Python υπολογίζεται η ευκλείδια απόσταση και αποθηκεύεται σε μια λίστα distances μαζί με τον αύξων αριθμό της εγγραφής του document.

b1 = np.array((x,y))

distances = []

counter = 0

for doc in documents:

b2 = np.array((doc['coordinates'][0],doc['coordinates'][1]))

dist = np.linalg.norm(b1-b2)

distances.append((counter,dist))

counter += 1

Τέλος, κάνοντας μια ταξινόμηση στη λίστα distances με τις αποστάσεις σε αύξουσα σειρά, μπορεί να γίνει η επιλογή των k πρώτων εγγραφών και με χρήση του αύξων αριθμού πραγματοποιείται εύρεση των εγγραφών στη λίστα documents.

Στην Python, αν το k είναι μεγαλύτερο από το πλήθος των εγγραφών στο distances, θα υπάρξει error. Επομένως, γίνεται και ο έλεγχος των δύο τιμών και αν το k είναι μεγαλύτερο, τότε επιστρέφονται όλες οι εγγραφές:

distances = sorted(distances, key=lambda distance: distance[1])

document = []

if len(distances) > k:

for j in range(k):

document.append(documents[distances[j][0]])

else:

for j in range(len(distances)):

document.append(documents[distances[j][0]])

return document

**Εύρεση k κοντινότερων γειτόνων με μοναδικό id**

Μια παραλλαγή της συνάρτησης που παρουσιάστηκε παραπάνω είναι να βρεθούν οι k κοντινότεροι γείτονες που θα έχουν διαφορετικό id. Πρακτικά αυτό σημαίνει ότι αν για ένα σημείο (x, y) βρεθούν τα 3 κοντινότερα σημεία, αυτά τα σημεία θα πρέπει να μην έχουν τα ίδια id ή διαφορετικά να μην είναι σημεία του ίδιου trajectory.

O αλγόριθμος είναι ίδιος με την παραπάνω συνάρτηση μέχρι το σημείο που γίνεται η ταξινόμηση στη λίστα distances με τη διαφορά ότι στη λίστα distances, αποθηκεύεται και το id της κάθε εγγραφής, εκτός από την απόσταση από το (x, y) και τον αύξων αριθμό.

Στη συνέχεια, έχοντας ταξινομημένη τη λίστα, η λογική που ακολουθείται είναι να βρεθούν τα πρώτα k στοιχεία με διαφορετικό id.

Επομένως για κάθε στοιχείο της λίστας distances ελέγχεται η τελική λίστα document. Αν η λίστα δεν είναι άδεια, τότε συγκρίνονται τα δύο id, αν είναι διαφορετικά τότε το στοιχείο που αντιστοιχεί στη συγκεκριμένη εγγραφή του distances εισάγεται στην τελική λίστα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να γίνει εισαγωγή των k στοιχείων.

for j in range(len(distances)):

if len(document) < k:

if document != []:

for doc in document:

traj\_object = documents[distances[j][0]]

if traj\_object['id'] != doc['id']:

document.append(documents[distances[j][0]])

else:

document.append(documents[distances[j][0]])

else:

break

return document

**Box Range Query**

Η Box Range συνάρτηση υπολογίζει πόσες εγγραφές υπήρξαν μέσα σε ένα ορθογώνιο ένα συγκεκριμένο χρονικό διάστημα.

Τα ορίσματα της συνάρτησης είναι (x\_min, y\_min) που αντιστοιχούν στην κάτω αριστερά γωνία του ορθογωνίου, (x\_max, y\_max) που αντιστοιχούν στην πάνω δεξιά γωνία του ορθογωνίου και (t\_min, t\_max) που είναι το χρονικό διάστημα από t\_min ως t\_max.

Επομένως, το query ζητάει όλες τις εγγραφές όπου το lon είναι ανάμεσα σε x\_min και x\_max, το lat ανάμεσα σε y\_min και y\_max και παράλληλα ικανοποιείται και ο χρονικός περιορισμός.

Η mongo περιλαμβάνει τον geospatial query operator $geowithin, ο οποίος μπορεί να ακολουθείται από κάποιους άλλους τελεστές όπως $center για κύκλο ή $box για ορθογώνιο και χρησιμοποιείται για να βρίσκονται συντεταγμένες που περιέχονται μέσα σε μια γεωγραφική περιοχή.

def range\_box(mycol, x\_min, x\_max, y\_min, y\_max, t\_min, t\_max):

query = '{"$and":[{"coordinates":{"$geoWithin": {"$box":[['+str(x\_min)+','+str(y\_min)+'] , ['+str(x\_max)+','+str(y\_max)+']]}}},{"t":{"$gt":'+str(t\_min)+'}},{"t":{"$lt":'+str(t\_max)+'}}]}'

k1 = json.loads(query)

cursor = mycol.find(k1)

documents = []

counter = 0

for document in cursor:

documents.insert(counter, document)

counter += 1

return documents

**Box Range query aggregate**

Ο παραπάνω αλγόριθμος μπορεί να εκτελεστεί και ως εξής: Εκτελούνται όλα τα βήματα μέχρι να βρεθούν όλες οι εγγραφές που ικανοποιούν τους περιορισμούς, με τη διαφορά ότι η συνάρτηση περιέχει μια ακόμα παράμετρο. Στο τέλος, η συνάρτηση δεν επιστρέφει όλες τις εγγραφές, αλλά ότι ορίζει η παράμετρος agg. Παρακάτω παρουσιάζεται η περίπτωση όπου το agg είναι ίσο με count και κατά συνέπεια η συνάρτηση επιστρέφει το πλήθος των εγγραφών.

def range\_box(mycol, x\_min, x\_max, y\_min, y\_max, t\_min, t\_max):

query = '{"$and":[{"coordinates":{"$geoWithin": {"$box":[['+str(x\_min)+','+str(y\_min)+'] , ['+str(x\_max)+',' +str(y\_max)+']]}}},{"t":{"$gt":'+str(t\_min)+'}},{"t":{"$lt":'+str(t\_max)+'}}]}'

k1 = json.loads(query)

cursor = mycol.find(k1)

documents = []

counter = 0

for document in cursor:

documents.insert(counter, document)

counter += 1

if agg == "count":

return len(documents)

**Circle Range Query**

Η Cirlcle Range συνάρτηση υπολογίζει πόσες εγγραφές υπήρξαν μέσα σε έναν κύκλο μια συγκεκριμένη χρονική στιγμή.

Ομοίως με το box range query, εκτελείται πρώτα το query που βρίσκει τις εγγραφές που βρίσκονται μέσα στον κύκλο και ικανοποιούν τον χρονικό περιορισμό.

def range\_circle(mycol, x, y, r, t\_min, t\_max):

query='{"$and":[{"coordinates":{"$geoWithin":{"$center": [['+str(x)+','+str(y)+'],'+str(r)+']}}} ,{"t":{"$gt":'+str(t\_min)+'}},{"t":{"$lt":'+str(t\_max)+'}}]}'

k1 = json.loads(query)

cursor = mycol.find(k1)

documents = []

counter = 0

for document in cursor:

documents.insert(counter, document)

counter += 1

return documents