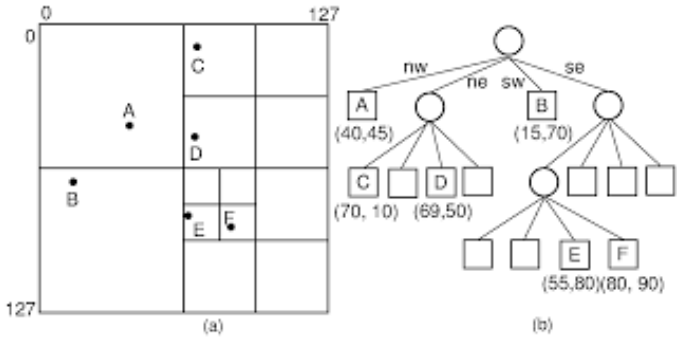
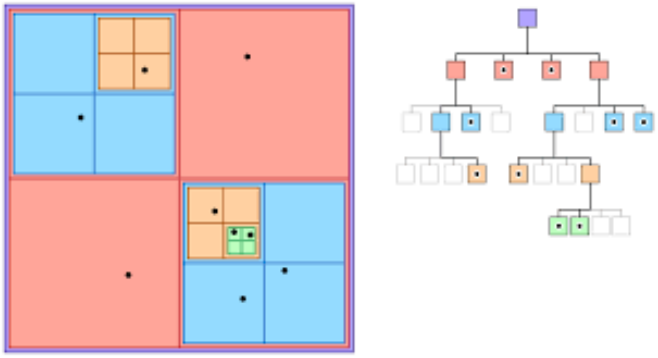
Quad Tree

Το Quadtree είναι μια δενδρική δομή δεδομένων στην οποία κάθε εσωτερικός κόμβος έχει ακριβώς τέσσερα παιδιά. Συνήθως χρησιμοποιείται για την κατάτμηση ενός δισδιάστατου χώρου σε ορθογώνιες περιοχές. Κάθε κόμβος φύλλου στο τετράδεντρο αντιπροσωπεύει μια περιοχή και η τιμή που αποθηκεύεται σε αυτόν τον κόμβο φύλλου είναι τα δεδομένα που σχετίζονται με την περιοχή αυτή. Το δέντρο μπορεί να χρησιμοποιηθεί για διάφορες λειτουργίες, όπως η αναζήτηση ενός συγκεκριμένου σημείου ή περιοχής, η εισαγωγή ενός νέου σημείου ή περιοχής και η διαγραφή ενός σημείου ή περιοχής. Η χρήση ενός τετραδένδρου μπορεί να βελτιώσει σημαντικά την αποτελεσματικότητα αυτών των λειτουργιών σε περιπτώσεις όπου τα δεδομένα είναι κατανεμημένα σε ένα μεγάλο δισδιάστατο χώρο.



Στην εργασία μας, θα επικεντρωθούμε στα 2D Quad-Tree. Η διαδικασία της υλοποίησης και αναζήτησης έχει ως εξής: Καλούμε ένα δισδιάστατο ευκλείδειο επίπεδο στο οποίο εξετάζεται ένα κανονικοποιημένο τετράγωνο. Δημιουργούμε μια ρίζα για το δέντρο, η οποία αντιπροσωπεύει το συνολικό τετράγωνο στο οποίο γίνεται η αναζήτηση. Κάθε κόμβος (όπως και η ρίζα), αντιπροσωπεύει μία περιοχή του τετραγώνου. Κάθε κόμβος υποδιαιρείται σε 4 τεταρτημόρια έως ότου βρεθούν όλα τα στοιχεία αναζήτησης και να μπορούν να τοποθετηθούν στο δέντρο, σύμφωνα με τους κανόνες που το διέπουν.



Η εκπόνηση αυτού του μέρους της εργασίας έγινε στο περιβάλλον του VS Code. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι:

```
import math
import sys
from collections import deque
import time
import re
import pandas as pd
```

QTree.py

Στο αρχείο αυτό περιέχονται όλες οι απαραίτητες συναρτήσεις έτσι ώστε να δημιουργήσουμε τόσο το τετράγωνο αναζήτησης, τοποθέτηση των στοιχείων στο κανονικοποιημένο επίπεδο, τη διαδικασία των υποδιαιρέσεων των τετραγώνων καθώς και την εισαγωγή των κόμβων στο δέντρο.

Class Point(object)

Οι συντεταγμένες που θα χαρακτηρίζει κάθε σημείο στο χώρο με την επιλογή να του δοθούν χαρακτηριστικά δεδομένα.

Class BoundingBox()

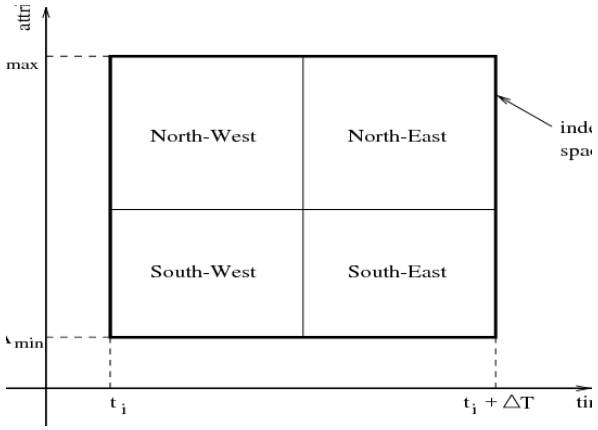
Η αναπαράσταση γίνεται σε ένα τετράγωνο το οποίο αποτελεί τη ρίζα. Για την περικοπή του χώρου αναδρομικά χρησιμοποιούμε το object Bounding Box, το οποίο τα σύνορα του τετραγώνου που περιέχει ο κάθε κόμβος τα περιέχει ως properties.

def contains()

Boolean συνάρτηση η οποία αληθεύει εφόσον το σημείο βρίσκεται μέσα στο bounding box που ελέγχουμε.

Class QuadNode()

Εδώ καθορίζεται ο κόμβος του δέντρου. Ως properties θέτουμε το κέντρο του quadtree, το ύψος, το βάθος του αλλά και την χωρητικότητα αυτού. Με την χωρητικότητα ελέγχουμε τον βασικό κανόνα του quadtree όπου ο αριθμός των παιδιών του να μην ξεπερνάει τα 4. Για το ψάξιμο του δέντρου χρησιμοποιείται η κατεύθυνση του κόμβου. Έτσι ξέρουμε αν βρίσκεται εκεί που θέλουμε να ψάξουμε.



def contains(point)

Boolean συνάρτηση που επιστρέφει true στην περίπτωση που ένα point βρίσκεται μέσα στα δεδομένα του κόμβου.

def iter()

Μετατρέπει όλα τα points σε iterable (επαναληπτικά). Γι' αυτό χρησιμοποιούμε τη συνάρτηση *yield* της python. Η δήλωση yield επιστρέφει ένα αντικείμενο γεννήτριας (generator) σε αυτόν που καλεί τη συνάρτηση που περιέχει yield, αντί να επιστρέφει απλώς μια τιμή όπως κάνει το return.

def calc_bounding_box()

Υπολογισμός του τετραγώνου μέσα στο οποίο εμπεριέχεται ο κόμβος.

def contains_point(point)
Συνάρτηση τύπου Boolean που ελέγχει αν ένα point υπάρχει μέσα σε έναν κόμβο.

def is_ul(), def issuer(), def is_ll(), def is_lr()
Σε ποιον προσανατολισμό βρίσκεται το point ενδιαφέροντος (NE, NW, SE, SW).

def subdivide()
Αναδρομική σχέση που υποδιαιρεί έναν κόμβο και τα παιδιά του.

def insert()
Boolean που αληθεύει στην εισαγωγή του point. Πρώτα ελέγχουμε τη χωρητικότητα του κόμβου που θα εισάγουμε το point. Αν έχει υπερβεί τότε υποδιαιρούμε. Έπειτα ελέγχουμε τον προσανατολισμό του και ανάλογα τοποθετείται.

def find()
Ψάχνει έναν κόμβο όπου το περιεχόμενο του point βρίσκεται στα παιδιά του. Επιστρέφει τον κόμβο και τα περιεχόμενά του.

def find_node()
Ψάχνει το κόμβο που μπορεί να περιέχει το point. Ως όρισμα δέχεται τη λίστα των σημείων που έχει ήδη επισκεφτεί ο αλγόριθμος και το σημείο ενδιαφέροντος. Επιστρέφει ένα QuadNode() object αν βρέθηκε και τη λίστα με όσα πέρασε για να το βρει.

def all_points()
Επιστρέφει μια λίστα με όλα τα points που περιέχει ένας κόμβος και τα παιδιά του.

def within_bb()
Αναδρομική συνάρτηση για να εξετάσουμε αν ένα τετράγωνο περιέχεται μέσα σε κάποιο μεγαλύτερο, συμπεριλαμβανομένου και των πιθανών κατευθύνσεων που μπορούν να ακολουθήσουν.

def Euclidean_distance()
Υπολογισμός της ευκλείδειας απόστασης 2 σημείων στο χώρο.

Class QuadTree()
Δημιουργείται το object QuadTree, όπου μέσα σε αυτό βρίσκονται κόμβοι και συντεταγμένες. Για την υλοποίηση του QuadTree χρειαζόμαστε το κέντρο του, το ύψος του αλλά και το πλάτος του στο χώρο. Η χωρητικότητα από default τη θέτουμε none.

def convert_to_point()
Ελέγχεται το input και επιστρέφει με το σωστό type.

def contains()
Boolean συνάρτηση που αληθεύει εφόσον ένα point περιέχει τη συνάρτηση find().

def len()
Μήκος δέντρου.

def iter()
Μετατρέπει το δέντρο σε επαναληπτικό (iterable).

def insert()

Το input μετατρέπεται σε object point και του μεταφορτώνονται τα χαρακτηριστικά. Μετά καλείται η insert που ορίστηκε παραπάνω.

def find()

Ψάχνει το point και το γυρνάει στη συνάρτηση find().

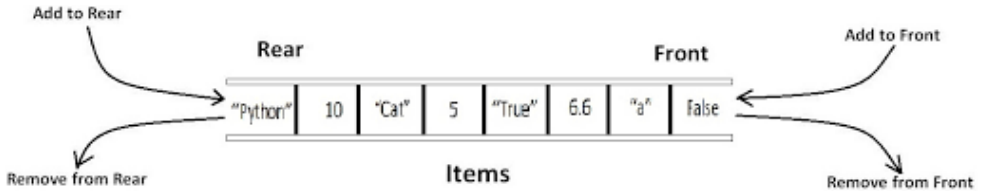
Local_QTree.py

Class LocalQuadTree(object QUADTREE)

Είναι μία κλάση προτύπου για το QuadTree έτσι ώστε να εφαρμόσουμε k Nearest Neighbors.

def query()

Συνάρτηση για την εύρεση γειτονικών σημείων συγκριτικά με το σημείο ενδιαφέροντος. Χρησιμοποιείται η συνάρτηση *deque()* ή αλλιώς «ουρά διπλού άκρου» της βιβλιοθήκης *collections*, η οποία είναι μια δομή δεδομένων που μας επιτρέπει να προσθέτετε ή να αφαιρείτε στοιχεία από κάθε άκρο σε σταθερό χρόνο, O(1). Εδώ, χρησιμοποιείται έτσι ώστε να υλοποιήσουμε μια στοίβα με τα σημεία που μπορεί να βρίσκονται «μέσα» στο σημείο ενδιαφέροντος. Εκεί, φτιάχνεται μία λίστα όπου από αριστερά μπαίνουν τα χαρακτηριστικά που απαρτίζουν τον κόμβο, ενώ από δεξιά κρατάει ένα ιστορικό με τους κόμβους που ελέγχθηκαν.



def get_knn(point, k)

Ως attributes δέχεται το point της αναζήτησης μας και τον αριθμό των k κοντινότερων γειτόνων που θέλουμε να πάρουμε από τη στοίβα που γυρνάει η συνάρτηση *query()*. Εφαρμόζουμε ευκλείδεια απόσταση και ταξινομούμε τα αποτελέσματα.

@staticmethod def nodeIsLeaf()

Μέθοδος που ανήκει στην κλάση *LocalQuadTree* και όπως λέει και το όνομά της μας επιστρέφει αν ο κόμβος που εξετάζουμε είναι φύλλο.

Run_QTree.py

Σε αυτό το αρχείο φορτώνεται το dataset και αποθηκεύονται κατάλληλα τα στοιχεία που χρειαζόμαστε για την εφαρμογή των μεθόδων που υλοποιήσαμε στα προηγούμενα αρχεία. Έπειτα ο χρήστης μέσω ενός στοιχειώδους menu αποφασίζει αν θα εξετάσει τους k κοντινότερους γείτονες ενός επιστήμονα, ή το εύρος των κόμβων για κάποια συγκεκριμένα όρια του ορθογωνίου.

def put_into_list(), get_points(), range_search()

Οι συγκεκριμένες συναρτήσεις παραμένουν ίδιες όσον αφορά την υλοποίησή τους, όπως και στις προηγούμενες πολυδιάστατες δομές που εξετάσαμε.

def run_scientists()

Δέχεται ως είσοδο το όνομα του επιστήμονα και τα στοιχεία του dataset. Για να γίνει η δημιουργία ενός object point για την εύρεση λαμβάνει μέρος μία προ επεξεργασία στη συνάρτηση *main_scientists()* για να βρεθεί το index μέσα στη λίστα δεδομένων. Δίνεται η επιλογή για kNN ή range_search. Να σημειωθεί πως ο χρόνος ξεκινάει να μετράει με την έναρξη του kNN και όχι με τη δημιουργία του δέντρου.

Testing

K Nearest Neighbors

```
Reading scientists data...

Data successfully read.


 MENU :

1: kNN
2: Range Search
1
Give the name of the scientist you want to search his neighbors: Andrew Appel
How many nearby scientists do you want to show: 7
********************
For Andrew Appel with coordinates 295, 1

********************
The algorithm run for :
0.002489sec
********************
********************
The 7 nearest neighbors are :


0  ====> Andrew Appel with coordinates (295,1)
1  ====> Peter J. Denning with coordinates (296,1)
2  ====> Manolis Kellis with coordinates (293,1)
3  ====> Herman Hollerithwith coordinates (294,3)
4  ====> Butler Lampson with coordinates (297,3)
5  ====> Ken Robinson with coordinates (292,2)
6  ====> Charles Bachman with coordinates (298,2)
```

Range Search

```
Please give the minimum cord: 10
Please give the minimum award: 4
Please give the maximum cord: 105
Please give the maximum award: 6
Serge Abiteboul [49, 6]
Samson Abramsky [104, 5]
Cecilia R. Aragon [16, 5]
Sanjeev Arora [70, 4]
John Vincent Atanasoff [30, 5]
David A. Bader[17, 5]
Cecilia Berdichevsky[21, 5]
Daniel J. Bernstein [102, 4]
Lenore Blum [31, 5]
Per Brinch Hansen[24, 5]
Rod Canion[87, 5]
John Carmack [51, 5]
Fernando J. CorbatoØ[67, 4]
Alexander Dewdney[77, 6]
Andrey Ershov [89, 5]
Don Estridge[103, 4]
Oren Etzioni [23, 6]
Shimon Even[97, 5]
Raphael Finkel[86, 5]
Xiaoming Fu[28, 5]
GastoØn Gonnet [47, 5]
Shelia Guberman [50, 4]
JoØrg Gutknecht [93, 4]
Margaret Hamilton [72, 4]
Eric Horvitz [58, 5]
Kenneth E. Iverson [95, 5]
Ivar Jacobson [29, 4]
David S. Johnson[73, 6]
Mathai Joseph[32, 5]
William Kahan [56, 6]
Leonard Kleinrock [101, 5]
John Krogstie [12, 4]
Leslie Lamport [65, 6]
Jochen Liedtke [43, 4]
John McCarthy [75, 5]
Andrew McCallum[26, 5]
Marshall Kirk McKusick [71, 5]
J Strother Moore [84, 5]
Peter Naur[59, 5]
Juan PavoØn[100, 4]
Roberto Pieraccini [45, 5]
Amir Pnueli [92, 5]
T. V. Raman [35, 5]
Dennis Ritchie [36, 4]
Rudy Rucker [18, 5]
Rudy Rucker [18, 5]
James Rumbaugh [22, 6]
Ben Shneiderman [63, 6]
Maciej Stachowiak [96, 5]
Richard E. Stearns [78, 4]
Paul Vixie [76, 5]
Kevin Warwick [54, 4]
Joseph Weizenbaum [80, 4]
David Wheeler [53, 5]
Niklaus Wirth [14, 5]
Stephen Wolfram [15, 4]
Beatrice Helen Worsley [40, 4]
```