Quad Tree

Το Quadtree (Τετράδεντρο) 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 Quadtree 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 Quadtree έχει το πλεονέκτημα της σηματικής βελτίωσης στην αποτελεσματικότητα αυτών των λειτουργιών σε περιπτώσεις όπου τα δεδομένα είναι κατανεμημένα σε ένα μεγάλο δισδιάστατο χώρο.

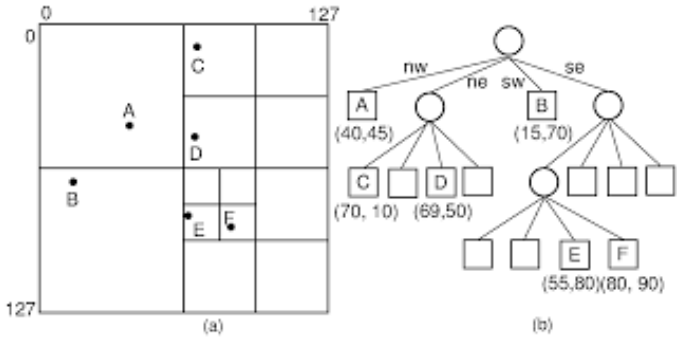

Figure 1

Στην ενότητα αυτή της εργασίας, το κέντρο ενδιαφέροντος στρέφεται στα 2D Quad-Tree. Η διαδικασία της υλοποίησης και αναζήτησης έχει ως εξής: Ορίζεται ένα δισδιάστατο Ευκλείδειο επίπεδο, στο οποίο εξετάζεται ένα κανονικοποιημένο τετράγωνο. Στη συνέχεια, δημιουργείται μια ρίζα για το δέντρο, η οποία αντιπροσωπεύει το συνολικό τετράγωνο στο οποίο γίνεται η αναζήτηση. Κάθε κόμβος (συμπεριλαμβανομένης και της ρίζας (*Figure 2*)), αντιπροσωπεύει μία περιοχή του τετραγώνου. Η διαδικασία που ακολουθεί ο αλγόριθμος είναι η επαναληπτική υποδιαίρεση του κάθε κόμβου σε 4 τεταρτημόρια έως ότου βρεθούν όλα τα στοιχεία αναζήτησης και τοποθετηθούν στο δέντρο, σύμφωνα με τους κανόνες που το διέπουν.

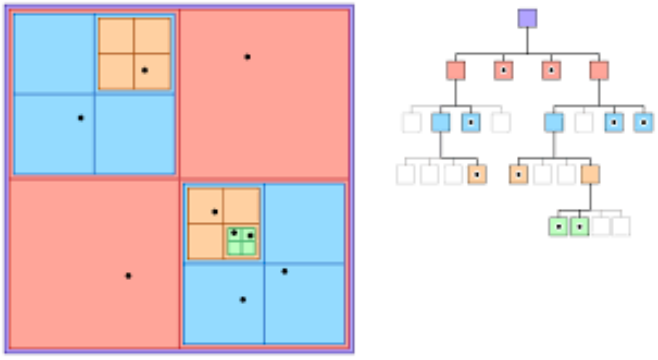

Figure 2

Η εκπόνηση αυτού του μέρους της εργασίας έγινε στο περιβάλλον του Visual Studio Code Editor, καθώς και οι βιβλιοθήκες που χρησιμοποιήθηκαν παρατίθενται ακολούθως:

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

QTree.py

Στο αρχείο αυτό περιέχονται όλες οι απαραίτητες μέθοδοι, των οποίων η κλήση αφορά στη δημιουργία ενός τετραγώνου αναζήτησης, στην τοποθέτηση των στοιχείων στο κανονικοποιημένο επίπεδο, στη διαδικασία των υποδιαιρέσεων των τετραγώνων καθώς και στην 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 χρησιμοποιείται το αντικείμενο (object) **Bounding Box**, το οποίο οριοθετεί τα σύνορα του τετραγώνου που ορίζει ο κάθε κόμβος περιέχοντάς τα ως **properties**.

def contains()

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

Class QuadNode()

Οι μέθοδοι της κλάσης αυτής καθορίζουν τους κόμβους του δέντρου. Ως properties τίθενται το **κέντρο** του Quadtree, το **ύψος**, το **βάθος** του αλλά και την **χωρητικότητα** αυτού. Με την χωρητικότητα γίνεται έλεγχος του βασικού κανόνα του Quadtree, όπου ο αριθμός των παιδιών του να μην ξεπερνά τα 4. Για το ψάξιμο του δέντρου χρησιμοποιείται η κατεύθυνση του κόμβου. Έτσι εξετάζεται αν βρίσκεται εκεί που επιθυμείται αναζήτηση.

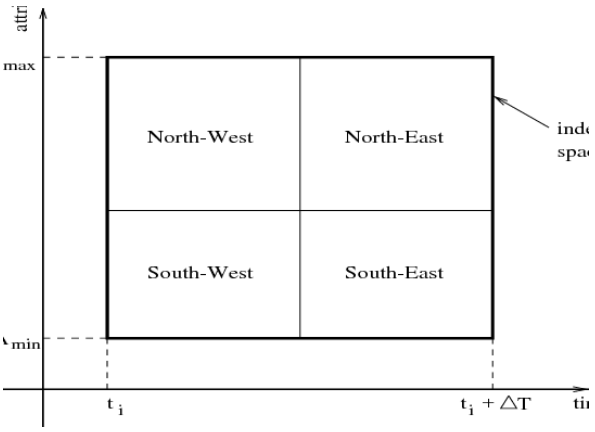

Figure 3

def contains(point)

Boolean συνάρτηση που επαληθεύεται στην περίπτωση που ένα σημείο (point) βρίσκεται μέσα στα δεδομένα του κόμβου.

def iter()

Μετατρέπει όλα τα σημεία (points) σε επαναληπτικά (iterable). Γι' αυτό χρησιμοποιείται η 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 η **return** κλήση.

def calc_bounding_box()

Υπολογισμός του τετραγώνου μέσα στο οποίο εμπεριέχεται ο κόμβος.

def contains_point(point)

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

def is_ul(), def issuer(), def is_ll(), def is_lr()

Σε ποιον προσανατολισμό βρίσκεται το σημείο (point) ενδιαφέροντος (NE, NW, SE, SW)(*Figure 3*).

def subdivide()

Αναδρομική σχέση που υποδιαιρεί έναν κόμβο και τα παιδιά του.

def insert()

Boolean μέθοδος, που αληθεύει στην εισαγωγή του σημείου (point). Πρώτα γίνεται έλεγχος της χωρητικότητας του κόμβου, στον οποίο εισάγεται το σημείο (point). Αν το έχει υπερβεί τότε υποδιαιρείται. Ακολούθως εξετάζεται ο προσανατολισμός του και αναλόγως τοποθετείται.

def find()

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

def find_node()

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

def all_points()

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

def within_bb()

Αναδρομική συνάρτηση για τον έλεγχο ενός τετραγώνου αν 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 αντικείμενο (object) **QuadTree**, όπου μέσα σε αυτό βρίσκονται κόμβοι και συντεταγμένες. Για την υλοποίηση του QuadTree απαιτείται η γνώση του κέντρου του, του ύψους του αλλά και του πλάτους του στο χώρο. Για τη χωρητικότητα η καθορισμένη (default) τιμή της ορίζεται σε **none**.

def convert_to_point()

Ελέγχεται η είσοδος (input) και επιστρέφεται με τον αποδεκτό τύπο (type).

def contains()

Boolean συνάρτηση που αληθεύει εφόσον ένα σημείο (point) περιέχει τη μέθοδο find().

def len()

Μήκος δέντρου.

def iter()

Μετατρέπει το δέντρο σε επαναληπτικό (iterable).

def insert()

Η είσοδος (input) μετατρέπεται σε *object point* και γίνεται μεταφόρτωση σε αυτήν τα χαρακτηριστικά. Στη συνέχεια, καλείται η **insert** (ορίστηκε παραπάνω).

def find()

Ψάχνει το σημείο (point) και το επιστρέφει ψς αποτέλεσμα κλήσης της μεθόδου **find()**.

Local_QTree.py

Class LocalQuadTree(object QUADTREE)

Αφορά σε μία κλάση προτύπου για το QuadTree έτσι ώστε να γίνει εφαρμογή των k κοντινότερων γειτόνων (K-Nearest-Neighbors).

def query()

Μέθοδος 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 επιτρέπει την προσθήκη ή αφαίρεση στοιχείων 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 (*Figure 4*).

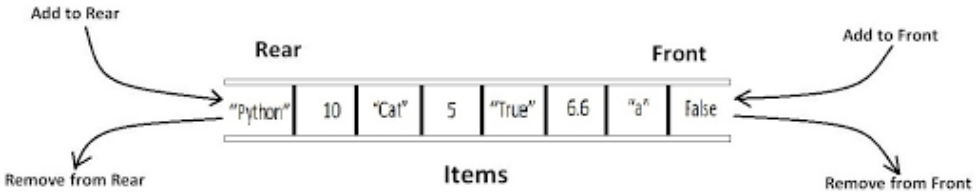

Figure 4

def get_knn(point, k)

Ως attributes δέχεται το σημείο (point) της αναζήτησης μας και τον αριθμό των k κοντινότερων γειτόνων που απαιτείται να ληφθούν από τη στοίβα, την οποία επιστρέφει από την κλήση της μεθόδους **query()**. Εφαρμόζεται η Ευκλείδεια απόσταση και γίνεται ταξινόμηση των αποτελεσμάτων.

@staticmethod def nodeIsLeaf()

Μέθοδος που ανήκει στην κλάση **LocalQuadTree**, και όπως προοικονομεί το όνομά της, επιστρέφει αν ο κόμβος που εξετάζεται είναι φύλλο.

run_QTree.py

Σε αυτό το αρχείο φορτώνεται το σύνολο των δεδομένων (dataset) και αποθηκεύονται κατάλληλα τα στοιχεία που απαιτούνται για την εφαρμογή των μεθόδων που υλοποιήθηκαν στα προηγούμενα εδάφια. 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()

Αφού γίνει το διάβασμα του αρχείου, σε μία for λούπα τοποθετούνται τα στοιχεία του csv που μας ενδιαφέρουν στις λίστες latitude, longitude, scientist_list.

def 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 συνόλου δεδομένων (dataset). Με σκοπό τη δημιουργία ενός **object point** για την εύρεση, λαμβάνει μέρος μία προ-επεξεργασία στη μέθοδο *main_scientists()*, στοχεύοντας στην εύρεση του **index** μέσα στη λίστα δεδομένων. Δίνεται η επιλογή για k-NN ή range_search. Στο σημείο αυτό, τονίζεται πως ο χρόνος ξεκινάει να μετράει με την έναρξη του k-NN και όχι με τη δημιουργία του δέντρου.

names.py

def name_filter(line, start, end)

Το συγκεκριμένο script δημιουργήθηκε για να φιλτράρονται τα ονόματα των επιστημόνων σύμφωνα με το πρώτο γράμμα που είναι καταχωρημένοι, σε ένα εύρος γραμμάτων (start, end).

Testing

K Nearest Neighbors

```
Reading scientists data...

Give the 1st letter of list: G
Give the last letter of list: Z
Data successfully read.


 MENU :

1: kNN
2: Range Search
1
Give the name of the scientist you want to search his neighbors: Victor Bahl
How many nearby scientists do you want to show: 5
********************
For Victor Bahl with coordinates 462, 4

********************
The algorithm run for :
0.002977sec
********************
********************
The 5 nearest neighbors are :


0  ====> Gerard Holzmannwith coordinates (463,4)
1  ====> Yanhong Annie Liuwith coordinates (461,6)
2  ====> Samir Daswith coordinates (464,6)
3  ====> Wendy Hallwith coordinates (459,3)
4  ====> Matthew Dillonwith coordinates (460,1)
```

Range Search

```
Reading scientists data...

Give the 1st letter of list: G
Give the last letter of list: Z
Data successfully read.


 MENU :

1: kNN
2: Range Search
2
Please give the minimum cord: 50
Please give the minimum award: 4
Please give the maximum cord: 150
Please give the maximum award: 6
Samson Abramsky[104, 5]
Sanjeev Arora[70, 4]
Rod Canion[87, 5]
John Carmack[51, 5]
Shimon Even[97, 5]
Raphael Finkel[86, 5]
Robert Floyd[134, 6]
Robert France[111, 4]
Kunihiko Fukushima[144, 5]
Robert M. Graham[123, 4]
Shelia Guberman[50, 4]
JoΩ½rg Gutknecht[93, 4]
Margaret Hamilton[72, 4]
Johan HoΩ½stad[117, 4]
Kenneth E. Iverson[95, 5]
William Kahan[56, 6]
Ken Kennedy[143, 5]
Leonard Kleinrock[101, 5]
Leslie Lamport[65, 6]
Nancy Lynch[115, 5]
Nadia Magnenat Thalmann[148, 6]
John McCarthy[75, 5]
Marshall Kirk McKusick[71, 5]
Robin Milner[133, 6]
J Strother Moore[84, 5]
Peter Naur[59, 5]
Paritosh Pandya[106, 6]
Juan PavoΩ½n[100, 4]
Jon Postel[142, 4]
Maciej Stachowiak[96, 5]
Richard Stallman[119, 6]
Richard E. Stearns[78, 4]
Robert Tarjan[124, 6]
Paul Vixie[76, 5]
Kevin Warwick[54, 4]
Joseph Weizenbaum[80, 4]
John Yen[127, 5]
Shlomo Zilberstein[125, 6]
```