Εικόνα που περιέχει στιγμιότυπο οθόνης, κύκλος, κείμενο, ασπόνδυλο

Περιγραφή που δημιουργήθηκε αυτόματα

**ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΣΠΥΡΟ ΣΟΥΛΙ**

**ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ : 1070263**

**ΕΤΟΣ : 6ο**

**Project 2ο**

[**Πολυδιάστατες Δομές Δεδομένων και Υπολογιστική Γεωμετρία**](https://eclass.upatras.gr/courses/CEID1162/)

**Ανάπτυξη Γεωμετρικών Πολυδιάστατων Δομών**

**1ο Θέμα**

3D R-trees for Spatio-Temporal Queries σε ΒΔ τροχιών στο επίπεδο: Υλοποίηση ερωτημάτων σε τροχιές κινούμενων αντικειμένων (trajectory queries) στο επίπεδο, κάνοντας χρήση τρισδιάστατων R-trees που δεικτοδοτούν σημεία της μορφής (x,y,t).

Για το συγκεκριμένο θέμα θα υλοποιήσω τα ακόλουθα ερωτήματα :

Ερωτήματα πλησιέστερου γείτονα: Βρείτε την τροχιά που είναι πιο κοντά σε ένα δεδομένο σημείο στον τρισδιάστατο χώρο.

Ερωτήματα χρονικού εύρους: Ανάκτηση όλων των τροχιών που ήταν ενεργές εντός ενός καθορισμένου χρονικού διαστήματος

Ερωτήματα διασταύρωσης: Προσδιορισμός τροχιών που τέμνονται ή συγκρούονται μεταξύ τους.

Για να υλοποιήσουμε τα ερωτήματα θα πρέπει πρώτα να δημιουργήσουμε μια κλάση που θα να κρατά πληροφορίες σχετικά με τη θέση ενός κινούμενου αντικειμένου σε ένα χώρο 2D για ένα συγκεκριμένο χρονικό διάστημα(class MovingObject). Ύστερα δημιουργώ τυχαία αντικείμενα μέσα στην κλάση ώστε να μπορέσουμε να κάνουμε τα πειράματα που θέλουμε. Το κάθε ερώτημα υλοποιείται με βάση μιας συνάρτησης, η πρώτη συνάρτηση(def nearest\_neighbor\_query(*query\_x*, *query\_y*, *query\_time*)) χρησιμοποιείται για την εύρεση του πλησιέστερου αντικειμένου. Υπολογίζει την μικρότερη απόσταση από το σημείο του ερωτήματος και εκτυπώνει τις λεπτομέρειές του, συμπεριλαμβανομένου του αναγνωριστικού του αντικειμένου. Το δεύτερο ερώτημα υλοποιείται με βάση την συνάρτηση (def range\_query()) όπου βρίσκει τις τροχιές που ήταν ενεργές εντός ενός καθορισμένου χρονικού διαστήματος, αυτό γίνεται ελέγχοντας αν η ώρα έναρξης ή λήξης του αντικειμένου εμπίπτει στο καθορισμένο χρονικό εύρος του ερωτήματος και εκτυπώνει εάν υπάρχουν οι συγκεκριμένες τροχιές και τις λεπτομέρειες τους. Το τρίτο ερώτημα υλοποιείται με βάση την σηνάρτηση (def intersection\_query()) όπου προσδιορίζει τροχιές που τέμνονται ή συγκρούονται μεταξύ τους, η συνάρτηση ελέγχει διάφορες συνθήκες όπως ο χρόνος εκκίνησης και λήξης των αντικειμένων και εάν υπάρχουν τέμνουσες τροχιές, εκτυπώνονται οι λεπτομέρειές τους, συμπεριλαμβανομένων των αναγνωριστικών αντικειμένων, των χρόνων έναρξης και λήξης. Τέλος μέσω διαγραμμάτων θα δείξω το αποτέλεσμα των πειραμάτων.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

**2ο ΘΕΜΑ**

Interval trees και Segment trees για interval και stabbing Queries αντίστοιχα. Μελέτη απόδοσης των βασικών πράξεων στις δύο δομές.

Αρχικά χρησιμοποίησα κλάση μόνο για το Segment tree διότι για το Interval μπορώ απλά να το πάρω έτοιμο από την βιβλιοθήκη intervaltree και να το χρησιμοποιήσω. Τα ερωτήματα που θα γίνουν και για τις 2 δομές θα είναι build,update,query,insert,delete και το εύρος των ερωτημάτων θα είναι 100, 500, 1000, 2000, 5000, 10000. Για το Segment tree πρέπει πρώτα να το δημιουργήσουμε βάση θεωρίας, έτσι δημιούργησα ένα Segment tree με μήκος 4 φορες τον πίνακα εισόδου. Ύστερα χρησιμοποίησα για κάθε ερώτημα μια συνάρτηση ώστε να υπολογίσω τον μέσο χρόνο υλοποίησης στο Segment tree. Η def build\_tree φτιάχνει τα φύλλα του δέντρου δηλαδή το υλοποιεί και η def measure\_build\_time μετρά τον μέσο χρόνο που απαιτείται για τις λειτουργίες εισαγωγής. Η def update χρησιμοποιείται για την ενημέρωση του Segment tree και η measure\_update\_time μετράει τον μέσο χρόνο για την εκτέλεση πράξεων ενημέρωσης. Η def query χρησιμοποιείται για την εκτέλεση ενός ερωτήματος εύρους στο δέντρο και η measure\_query\_time μέθοδος μετράει το μέσο χρόνο που απαιτείται. Η measure\_insertion\_time μετρά τον μέσο χρόνο που απαιτείται για εισαγωγή δεδομένων σε ένα δέντρο τμημάτων. Τέλος, measure\_deletion\_time μετρά τον μέσο χρόνο που απαιτείται για την εκτέλεση πράξεων διαγραφής (προσομοίωση της αφαίρεσης) σε ένα δέντρο τμημάτων.

Για τα Interval trees απλά μπορούμε να χρησιμοποιήσουμε μια εντολή εκχώρησης όπως η tree = IntervalTree(interval\_list) για να ορίσουμε την λίστα με τα δεδομένα που θα του δώσουμε, να ορίσουμε με την num\_iterations πόσες φορές θα επαναληφθεί το ερώτημα στο δέντρο και τέλος να ορίσουμε με την num\_intervals το εύρος του δέντρου. Οι αντίστοιχες λειτουργίες θα υλοποιηθούν με τις συναρτήσεις measure\_interval\_tree\_build\_performance, measure\_insertion\_performance, measure\_deletion\_performance, measure\_query\_overlap\_performance και measure\_query\_point\_performance. Σημειώνω ότι στα Interval tree θα κάνουμε 2 είδη ερωτημάτων, ερωτήματα για τα σημεία του δέντρου και ερωτήματα για αλληλεπικαλυπτόμενα διαστήματα στο δέντρο. Τέλος δημιουργώ γραφήματα για να συγκρίνουμε την θεωρητική πολυπλοκότητα των δέντρων με την πραγματική με βάση τα πειράματα μου και για τα 2 δέντρα με γραφήματα γραμμών και συγκρίνω τα 2 δέντρα μεταξύ τους μέσω ραβδογραφήματος.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

**3ο ΘΕΜΑ**

Convex Hull: Υλοποίηση κυρτών περιβλημάτων σε 2 διαστάσεις.

Υπάρχουν διάφοροι αλγόριθμοι που έχουν σχεδιαστεί για τον υπολογισμό του κυρτού περιβλήματος ενός συνόλου σημείων. Εμείς θα ασχοληθούμε με 3 από αυτόυς, τον Graham’s Scan, τον Quickhull και τον Jarvis March (gift wrapping). Αρχικά πρέπει να ορίσουμε μια συνάρτηση που θα προσδιορίζει τον προσανατολισμό των σημείων, αυτό γίνεται με την συνάρτηση orientation(p, q, r). Ύστερα υλοποιώ τον αλγόριθμο Graham’s Scan όπου ελέγχει αν ο αριθμός των σημείων είναι μικρότερος από 3, αν ναι επιστρέφει τα σημεία εισόδου, καθώς δεν μπορεί να σχηματιστεί κυρτό περίβλημα με λιγότερα από 3 σημεία.Το χαμηλότερο (κάτω αριστερά) σημείο επιλέγεται ως σημείο εκκίνησης για τη δημιουργία του κυρτού περιβλήματος.Tα σημεία ταξινομούνται με βάση τις πολικές τους γωνίες ως προς το χαμηλότερο σημείο, αλγόριθμος επαναλαμβάνει τα ταξινομημένα σημεία και ελέγχει τον προσανατολισμό τους με τα δύο τελευταία σημεία της στοίβας convex\_hull.Εάν ο προσανατολισμός δεν είναι αριστερόστροφος, το τελευταίο σημείο αφαιρείται από τη στοίβα μέχρι να επιτευχθεί αριστερόστροφος προσανατολισμός. Τέλος, η στοίβα convex\_hull περιέχει όλες τις κορυφές του κυρτού περιβλήματος.

Ο αλγόριθμος jarvis\_march(points) ελέγχει αν ο αριθμός των σημείων είναι μικρότερος από 3. Αν ναι, επιστρέφει τα σημεία εισόδου, καθώς δεν μπορεί να σχηματιστεί κυρτό περίβλημα με λιγότερα από 3 σημεία.Αρχικοποιεί μια κενή λίστα με το όνομα hull για να αποθηκεύσει τις κορυφές του κυρτού περιβλήματος.Το αριστερότερο σημείο (με τη μικρότερη συντεταγμένη x) επιλέγεται ως σημείο εκκίνησης.Ο αλγόριθμος επαναλαμβάνει μέχρι να επισκεφθεί εκ νέου το σημείο εκκίνησης.Επαναλαμβάνει όλα τα άλλα σημεία και υπολογίζει τον προσανατολισμό του τρέχοντος σημείου, του σημείου που προστέθηκε προηγουμένως και του υποψήφιου σημείου.Εάν ο προσανατολισμός είναι αριστερόστροφος ή τα σημεία είναι κολλητά και το υποψήφιο σημείο απέχει περισσότερο από το τρέχον σημείο από ό,τι το προηγουμένως εξεταζόμενο σημείο, το υποψήφιο σημείο γίνεται το επόμενο σημείο του περιβλήματος.Ο αλγόριθμος συνεχίζει μέχρι να επιστρέψει στο πιο αριστερό σημείο, ολοκληρώνοντας το κυρτό περίβλημα.

Ο αλγόριθμος quickhull(points): Εάν υπάρχουν 3 ή λιγότερα σημεία, η συνάρτηση επιστρέφει τα σημεία εισόδου ως κυρτό περιβλήματος, δεδομένου ότι ένα κυρτό περίβλημα δεν μπορεί να σχηματιστεί με λιγότερα από 3 σημεία.Βρίσκει το αριστερότερο και το δεξιότερο σημείο και αρχικοποιεί το κυρτό περίβλημα με αυτά τα δύο σημεία.Διαχωρίζει τα σημεία σε δύο σύνολα: points\_above (σημεία πάνω από την ευθεία που σχηματίζεται από το αριστερότερο και το δεξιότερο σημείο) και points\_below (σημεία κάτω από την ευθεία).Καλεί τη συνάρτηση quickhull\_recursive δύο φορές: μία για τα σημεία πάνω από τη γραμμή και μία για τα σημεία κάτω από τη γραμμή. Ο αλγόριθμος χρησιμοποιεί την συνάρτηση quickhull\_recursive(convex\_hull, points, p1, p2): όπου καλείται για την εύρεση των σημείων του κυρτού περιβλήματος μέσα σε ένα δεδομένο εύρος. Τέλος δημιουργώ ένα μεγάλο dataset για τα πειράματα και τα εμφανίζω με την βοήθεια γραφημάτων.

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, πολυχρωμία

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

**4ο ΘΕΜΑ**

Line Segment Intersection: Υλοποίηση αλγορίθμων εύρεσης τομών μεταξύ ευθυγράμμων τμημάτων στο επίπεδο που προκύπτουν από τα «σύνορα» πολυγωνικών περιοχών π.χ. σε εφαρμογές υπέρθεσης χαρτών στα GIS κ.τ.λ.

Eφόσον δόθηκε το ελεύθερο από τους καθηγητές μας αποφάσισα να χρησιμοποιήσω μια δικιά μου υλοποίηση και να την συγκρίνω με την υλοποίηση του Sweep Line αλγορίθμου όπου είναι από τους ευρέως χρησιμοποιούμενους αλγορίθμους για την αποτελεσματική επίλυση αυτού του προβλήματος. Τα dataset που θα χρησιμοποιήσω είναι custom φτιαγμένα από εμένα.

Ξεκινάμε εισάγωντας διάφορες βιβλιοθήκες της Python, όπως τα sortedcontainers (για τη διατήρηση ταξινομημένων dictionaries), matplotlib.pyplot (για τη σχεδίαση), shapely.geometry (για την εργασία με γεωμετρικά σχήματα) και time (για τη μέτρηση του χρόνου εκτέλεσης).Ορίζουμε τις συντεταγμένες πολυγώνου: μια λίστα που ονομάζεται polygon\_coords, όπου κάθε στοιχείο είναι μια λίστα με 2D συντεταγμένες που αντιπροσωπεύουν τις κορυφές διαφορετικών πολυγώνων.

Δημιουργούμε αντικείμενα Shapely Polygon χρησιμοποιώντας τις συντεταγμένες που ορίστηκαν προηγουμένως.Μετά, μετράμε το χρόνο εκτέλεσης χρησιμοποιώντας τη συνάρτηση time.time() για να ξέρουμε το χρόνο εκτέλεσης των ακόλουθων βημάτων. Ύστερα δημιουργούμε τμήματα γραμμής για κάθε πολύγωνο συνδέοντας διαδοχικές κορυφές. Αυτά τα τμήματα γραμμής δημιουργούνται ως αντικείμενα LineString. Συγκρίνουμε κάθε ζεύγος τμημάτων γραμμής για να ελέγξουμε αν τέμνονται. Εάν ναι, υπολογίζουμε την τομή και αποθηκεύουμε τα επικαλυπτόμενα τμήματα σε ένα λεξικό που ονομάζεται overlapping\_groups. Τα κλειδιά του λεξικού είναι τα σημεία αρχής και τέλους των επικαλυπτόμενων τμημάτων και οι τιμές είναι λίστες ζευγών τμημάτων γραμμής που τέμνονται. Μετράμε και εκτυπώνουμε τον χρόνο που έχει παρέλθει για την υλοποίηση του αλγορίθμου. Για κάθε επικαλυπτόμενο τμήμα γραμμής, εκτυπώνουμε τα σημεία αρχής και τέλους του τμήματος και τα πολύγωνα που έχουν γειτονικά όρια κατά μήκος αυτού του τμήματος μέσω του overlapping\_groups. Τέλος, κάνουμε οπτικοποίηση του κώδικα.

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Με άλλα dataset

Εικόνα που περιέχει κείμενο, διάγραμμα, χάρτης, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Ο αλγόριθμος Sweep Line όπως θα δείτε δεν μου βγήκε ακριβώς όπως έπρεπε γιαυτο και δημιούργησα δικό μου αλγόριθμο

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εδώ βλέπουμε ότι ο δικός μου αλγόριθμος για το μικρό dataset (αυτό που έδειξα παραπάνω), έχει καλύτερο χρόνο εκτέλεσης από τον SweepLine. Βέβαια επειδή η πολυπλοκότητα του sweepline είναι λογαριθμική όσο μεγαλύτερα γίνουν τα δεδομένα τόσο καλύτερος θα είναι ο χρόνος που θα έχει σε σχέση με τον δικο μου αλγόριθμο. Ο αλγόριθμος ονομάστηκε PipiSweep 😊

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Για περισσότερο υλικό για το Project ανατρέξτε στο Powepoint όπου αναλύω περισσότερο τους αλγορίθμους με πιο διαδραστικό τρόπο.

**Το GITHUB μου :** [**https://github.com/ssoulis**](https://github.com/ssoulis)

**ΕΥΧΑΡΙΣΤΩ ΓΙΑ ΤΟΝ ΧΡΟΝΟ ΣΑΣ !**

**Εικόνα που περιέχει κείμενο, ρουχισμός, στιγμιότυπο οθόνης, άτομο

Περιγραφή που δημιουργήθηκε αυτόματα**