Project Πολυδιάστατες Δομές Δεδομένων και Υπολογιστική Γεωμετρία

PROJECT2 20232024

Χριστοδουλόπουλος Ευστάθιος Παναγιώτης
up1093513
3ο έτος

Χριστοδούλου Νικόλαος
up1093514
3ο έτος

Σωτήρης Χατζηγιάννης
up1078730
5ο έτος

Πίσσουρος Χάρης
up1069673
5ο έτος

Contents

Installation and Requirements

Τα requirements για την εκτέλεση του κώδικα είναι τα εξής:

- Εγκατάσταση της Rust
  Για την εγκατάσταση της Rust, ακολουθήστε τις οδηγίες που βρίσκονται στον παρακάτω σύνδεσμο:
  https://www.rust-lang.org/tools/install

- Εγκατάσταση της Python
  Για την εγκατάσταση της Python, ακολουθήστε τις οδηγίες που βρίσκονται στον παρακάτω σύνδεσμο
  https://www.python.org/downloads/

- Εγκατάσταση του pip
  Μετά την εγκατάσταση της Python, εγκαταστήστε το pip ακολουθώντας τις οδηγίες που βρίσκονται στον παρακάτω σύνδεσμο: https://pip.pypa.io/en/stable/installation/

Απλά χρησιμοποιήστε την εντολή pip install -r requirements.txt στο τερματικό σας στο home dir (Project_MDDS) . Αυτό θα εγκαταστήσει όλα τα πακέτα που αναφέρονται στο αρχείο requirements.txt.

Data Gathering And Processing

Code Documentation

Modules

change_data.py :
Περιέχει τον κώδικα για την μετατροπή δεδομένων όπου αυτό κριθεί απαραίτητο.
crawler :
Περιέχει τον κώδικα για την μεταφορά των δεδομένων που μας ενδιαφέρουν για computer scientists από την ιστοσελίδα https://dblp.org.pers/ και την δημιουργία ενός json αρχείου pol.json όπου αποθηκεύονται τα δεδομένα για κάθε επιστήμονα ξεχωριστά.
get_hash_size.py :
Περιέχει τον κώδικα για την δημιουργία ενός envfile στο οποίο περιέχονται τα μήκη των DBLP_Records και Surnames για να μπορούμε όπου χρειαστεί να μετατρέψουμε τα strings DBLP_Records και Surnames σε integers (hash).

Crawler

Εδώ θα αναλύσουμε λίγο μόνο τον κώδικα του Crawler εφόσον ο κώδικας του change_data.py και get_hash_size.py είναι αρκετά απλός και δεν υπάρχει ανάγκη για περεταίρω ανάλυση.

Βιβλιοθήκη requests: χρησιμοποιείτε για αποστολή αιτήματος σε μια ιστοσελίδα για άδεια χρήσης των δεδομένων της.

Με την βοήθεια της βιβλιοθήκης beautifulsoup4 αναλύουμε HTML και XML εγγραφές.

Δημιουργούμε αίτηση προς την ιστοσελίδα [DBLP_Records for computer scientists](https://dblp.org.pers/) και αποθηκεύουμε τα δεδομένα της σε μια μεταβλητή response = requests.get(url) και στην συνέχεια χρησιμοποιώντας την βιβλιοθήκη BeautifulSoup βρίσκουμε τα tags και έπειτα τα URL που αντιστοιχούν στις ιστοσελίδες για τον κάθε επιστήμονα ξεχωριστά. Στην συνέχεια βρίσκουμε τα tags για τα δεδομένα που μας ενδιαφέρουν και αποθηκεύουμε τα δεδομένα που αντιστοιχούν στα tags σε ένα json αρχείο pol.json όπου γίνεται έλεγχος εάν οι εγγραφές έχουν μεταφερθεί σωστά και τα μεταφέρουμε στο αρχείο poll.son όπου υπάρχουν πλέον τα δεδομένα των computer scientists για να τα χρησιμοποιήσουμε στα επόμενα ερωτήματα.

Αρχείο pol.json: Περιέχει τα δεδομένα των computer scientists.

Τα δεδομένα αυτά είναι:

1.author's name
2.title
3.gap of year
4.year of release
5.DBLP_Record
6.Awards
7.kind
8.co-author
9.surname


```
    {
        "author's name": "Aadhitya A",
        "title": "HexE - Securing Audio Contents in Voice Chat using
Puzzle and Timestamp.",
        "gap of years": [
            2020,
            2024
        ],
        "year of release": 2024,
        "DBLP_Record": "journals/corr/abs-2401-00765",
        "Awards": 2,
        "kind": "Journal articles",
        "co-author": [],
        "surname": "A"
    },
```


A) Segment And Interval Trees

A1) Interval Tree Implementation for Interval and Point Queries

Γενική Επισκόπηση

Αυτό το πρόγραμμα υλοποιεί ένα interval tree, έναν ειδικό τύπο δυαδικού δέντρου αναζήτησης που έχει σχεδιαστεί για την αποτελεσματική εύρεση όλων των intervals που επικαλύπτονται με οποιοδήποτε δεδομένο διάστημα ή σημείο. Είναι ιδιαίτερα χρήσιμο στην υπολογιστική γεωμετρία και σε εφαρμογές όπως ο χρονοπρογραμματισμός, όπου συχνά πρέπει να αναζητήσει κανείς αλληλεπικαλυπτόμενα διαστήματα. Η υλοποίηση περιλαμβάνει λειτουργίες για τη δημιουργία του δέντρου από ένα σύνολο intervals, την εισαγωγή νέων interval στο tree και την εκτέλεση ερωτημάτων τόσο ανά σημείο (εύρεση όλων των intervals που περιέχουν ένα δεδομένο σημείο) όσο και ανά διάστημα (εύρεση όλων των interval που τέμνονται με ένα δεδομένο διάστημα).

Χρήση

Βασική Χρήση

Για να χρησιμοποιήσετε αυτή την υλοποίηση του interval tree, θα πρέπει να έχετε το JSON αρχείο pol.json. Το script διαβάζει αυτό το αρχείο, δημιουργεί το interval tree και σας επιτρέπει να εκτελείτε ερωτήματα σημείων και διαστημάτων.

```
# Example of performing a point query
point_query_result = interval_tree.query(point=1995)
print("Point Query Result:", point_query_result)

# Example of performing an interval query
interval_query_result = interval_tree.interval_query(start=1995, end=1997)
print("Interval Query Result:", interval_query_result)
```

Προηγμένα Χαρακτηριστικά

Η Advanced χρήση περιλαμβάνει την τροποποίηση του δέντρου διαστημάτων με την εισαγωγή νέων διαστημάτων και τη χρήση του δέντρου για σύνθετα ερωτήματα που περιλαμβάνουν πολλαπλά διαστήματα ή σημεία.

Code Documentation

Modules

- json: Για να φορτωσει interval data απο ενα JSON file.

Functions

- **Μέθοδος main():**
  Το main entry point του script, που δείχνει πώς να δημιουργείται το δέντρο και να εκτελούνται queries.

- **Μέθοδος insert(self, record, node=None):**
  Η μέθοδος insert προσθέτει ένα νέο διάστημα στο δέντρο διαστημάτων. Εξασφαλίζει ότι το δέντρο διατηρεί τις ιδιότητές του, όπου κάθε κόμβος αντιπροσωπεύει ένα διάστημα και το δέντρο είναι δομημένο με βάση τα σημεία εκκίνησης αυτών των διαστημάτων.

  Ξεκινώντας από τη ρίζα, η μέθοδος συγκρίνει το σημείο εκκίνησης του νέου διαστήματος με το σημείο εκκίνησης του τρέχοντος κόμβου για να αποφασίσει αν θα διασχίσει αριστερά (για μικρότερα σημεία εκκίνησης) ή δεξιά (για ίσα ή μεγαλύτερα σημεία εκκίνησης). Η διαδικασία αυτή συνεχίζεται αναδρομικά μέχρι να βρεθεί μια κατάλληλη θέση φύλλου για το νέο διάστημα. Καθώς προστίθενται διαστήματα, η μέθοδος ενημερώνει την ιδιότητα max_end κάθε κόμβου που διασχίζεται ώστε να αντικατοπτρίζει το μέγιστο τελικό σημείο όλων των διαστημάτων στο υποδέντρο του, εξασφαλίζοντας την αποτελεσματική αναζήτηση.

- **Μέθοδος interval_query (self, start, end, node=None, results=None):**
  Αυτή η μέθοδος ανακτά όλα τα διαστήματα στο δέντρο που τέμνονται με ένα δεδομένο διάστημα ερωτήματος. Είναι χρήσιμη για την εύρεση όλων των εγγραφών που επικαλύπτουν μια συγκεκριμένη χρονική περίοδο ή εύρος.

  Η μέθοδος διατρέχει το δέντρο, ξεκινώντας από τη ρίζα, και ελέγχει το διάστημα κάθε κόμβου με το διάστημα ερωτήματος. Εάν το διάστημα του τρέχοντος κόμβου τέμνει το διάστημα του ερωτήματος, προστίθεται στα αποτελέσματα. Η μέθοδος χρησιμοποιεί επίσης τις τιμές max_end για το κλάδεμα της αναζήτησης, αποφεύγοντας την περιττή διάσχιση υποδέντρων που δεν είναι δυνατόν να τέμνονται με το διάστημα του ερωτήματος, βελτιώνοντας έτσι την αποδοτικότητα του ερωτήματος.

- **Μέθοδος query (self, point, node=None, results=None):**
  (Stabbing Query)

Η μέθοδος ερωτήματος έχει σχεδιαστεί για να βρίσκει όλα τα διαστήματα που περιέχουν ένα συγκεκριμένο σημείο, γνωστό ως stabbing query. Αυτό είναι ιδιαίτερα χρήσιμο για τον εντοπισμό των διαστημάτων που περιλαμβάνουν μια συγκεκριμένη χρονική στιγμή.

Παρόμοια με την interval_query, αυτή η μέθοδος διατρέχει το δέντρο, συγκρίνοντας το σημείο του ερωτήματος με τα διαστήματα που αντιπροσωπεύονται από κάθε κόμβο. Εάν το σημείο του ερωτήματος εμπίπτει στο διάστημα ενός κόμβου, το διάστημα αυτό προστίθεται στα αποτελέσματα. Η διάσχιση εκμεταλλεύεται την ιδιότητα max_end για να παραλείπει τα κλαδιά του δέντρου όπου το σημείο του ερωτήματος δεν θα μπορούσε ενδεχομένως να εμπίπτει σε κανένα διάστημα, βελτιστοποιώντας έτσι τη διαδικασία αναζήτησης.

Classes

- **Node:**
  Αντιπροσωπεύει ένα node στο interval tree.
- **IntervalTree:**
  Υλοποιεί τη δομή του interval tree και τις σχετικές μεθόδους.

A2) Segment Tree Implementation for Interval and Point Queries

Γενική Επισκόπηση

Η εφαρμογή Segment Tree Implementation for Interval and Point Queries είναι μια προσαρμοσμένη λύση Python που έχει σχεδιαστεί για την αποτελεσματική διαχείριση και αναζήτηση δεδομένων διαστημάτων. Αναπτυγμένη μέσω προηγμένων τεχνικών προγραμματισμού, αυτή η βάση κώδικα διευκολύνει την ταχεία υποβολή ερωτημάτων σημείων και διαστημάτων σε εκτεταμένα σύνολα δεδομένων. Είναι ιδιαίτερα επωφελής για εφαρμογές στην επιστήμη των δεδομένων, την ιστορική έρευνα και κάθε τομέα που απαιτεί ακριβή ανάλυση διαστημάτων. Αυτή η εφαρμογή ξεχωρίζει για την αποτελεσματική χρήση των Segment Trees, καθιστώντας την ένα βασικό πλεονέκτημα για την επεξεργασία και ανάλυση σύνθετων διαστημικών δεδομένων με ταχύτητα και ακρίβεια.

Χρήση

Βασική Χρήση

Το σύστημα λειτουργεί με interval data που παρέχονται σε μορφή JSON. Κάθε interval θα πρέπει να περιλαμβάνει ένα σημείο έναρξης και λήξης (υπό το κλειδί "gap of years") και ένα σχετικό "authors name". Αφού φορτώσετε τα δεδομένα σας, το σύστημα επιτρέπει διάφορους τύπους ερωτημάτων:

- **Point Query:** Προσδιορίζει όλα τα διαστήματα που περιέχουν ένα δεδομένο σημείο.
- **Range Query:** Εντοπίζει όλα τα διαστήματα που επικαλύπτονται με ένα καθορισμένο εύρος.

Code Documentation

Modules

- json: Για να φορτωσει interval data απο ενα JSON file.

Functions
Οι σημαντικές λειτουργίες του script περιλαμβάνουν:

- **build_tree():** Κατασκευάζει το segment tree από ένα αρχικό σύνολο διαστημάτων.

  Κατασκευάζει τη βασική δομή του segment tree δημιουργώντας κόμβους για κάθε τμήμα εντός του συνολικού εύρους δεδομένων.

  Προσδιορίζει τις ελάχιστες και μέγιστες τιμές σε όλα τα διαστήματα για τον καθορισμό του εύρους δεδομένων. Στη συνέχεια, διαιρεί αναδρομικά αυτό το εύρος για να δημιουργήσει ένα δυαδικό δέντρο όπου κάθε κόμβος αντιπροσωπεύει ένα τμήμα των δεδομένων.

  Ένα πλήρως κατασκευασμένο segment tree όπου κάθε κόμβος αντιπροσωπεύει ένα συγκεκριμένο εύρος, έτοιμο για εισαγωγή διαστημάτων και αναζήτηση.

- **insert():** Προσθέτει ένα νέο διάστημα στο segment tree.

  Προσθέτει ένα νέο διάστημα στο segment tree, διασφαλίζοντας ότι το δέντρο αντιπροσωπεύει με ακρίβεια όλα τα διαστήματα στο σύνολο δεδομένων.

  Η συνάρτηση διατρέχει το δέντρο για να εντοπίσει κόμβους των οποίων τα τμήματα καλύπτονται από το νέο διάστημα. Στη συνέχεια αποθηκεύει το διάστημα σε κάθε κόμβο που καλύπτεται πλήρως από αυτό. Για τις μερικές επικαλύψεις, η εισαγωγή συνεχίζεται αναδρομικά για να εξασφαλιστεί η πλήρης κάλυψη.

  Το διάστημα αποθηκεύεται σε όλους τους σχετικούς κόμβους του δέντρου, επιτρέποντας την αποδοτική υποβολή ερωτημάτων με βάση αυτό το ενημερωμένο σύνολο δεδομένων.

- **query():** Εκτελεί ερωτήματα σημείου ή εύρους στο δέντρο.

  Εκτελεί ερωτήματα στο segment tree για την εύρεση διαστημάτων με βάση ένα συγκεκριμένο σημείο ή εύρος.

  Η Point Query Process διατρέχει το δέντρο, συλλέγοντας διαστήματα από κόμβους των οποίων τα τμήματα περιλαμβάνουν το σημείο του ερωτήματος. Αυτό περιλαμβάνει τον έλεγχο των διαστημάτων σε κάθε σχετικό κόμβο και συνεχίζει προς τα κάτω σε κόμβουςπαιδιά, όπως είναι απαραίτητο.

  Η Interval Query Process ψάχνει για κόμβους που επικαλύπτονται με το εύρος του ερωτήματος, συλλέγοντας διαστήματα από αυτούς τους κόμβους και ελέγχοντας αναδρομικά τους κόμβουςπαιδιά για πρόσθετες επικαλύψεις.

  Το Outcome επιστρέφει έναν κατάλογο διαστημάτων που ικανοποιούν τα κριτήρια του ερωτήματος, επιτρέποντας στους χρήστες να έχουν γρήγορη πρόσβαση σε δεδομένα που σχετίζονται με το ερώτημά τους.

Classes
Η αρχιτεκτονική του συστήματος περιλαμβάνει διάφορες κλάσεις:

- **SegmentTreeNode:**
  Μια κλάση που αναπαριστά έναν κόμβο στο segment tree, κρατώντας το εύρος του και όλα τα διαστήματα πλήρους κάλυψης.

- **SegmentTree:**
  Διαχειρίζεται τη συνολική δομή του δέντρου, συμπεριλαμβανομένης της κατασκευής, της εισαγωγής διαστημάτων και της εκτέλεσης ερωτημάτων.

B) 3D Convex Hull Implementing Quickhull

Γενική Επισκόπηση

Το παρακάτω υποερώτημα του project αποτελεί την υλοποίηση του αλγορίθμου του 3D Convex Hull σε Rust.

Πιο συγκεκριμένα, ο αλγόριθμος που υλοποιήθηκε είναι ο αλγόριθμος του QuickHull. Ο αλγόριθμος αυτός είναι ένας αλγόριθμος που χρησιμοποιείται για την εύρεση του Convex Hull ενός συνόλου σημείων στον τρισδιάστατο χώρο (στην περίπτωση μας).

Ο αλγόριθμος αυτός έχει πολυπλοκότητα O(nlogn) και στη χειρότερη περίπτωση έχει πολυπλοκότητα O(n^2) όταν όλα τα σημεία στον τρισδιάστατο χώρο αποτελούν το convex hull .

Χρήση

Βασική Χρήση

Για την εκτέλεση του κώδικα, ακολουθήστε τα παρακάτω βήματα:

1. Εκτελέστε τα προγράμματα που βρίσκονται στον φάκελο **Crawler** με όνομα change_data.py και get_hash_size.py (επεξήγηση τους στον κατάλληλο φάκελο).
2. Τώρα στον φάκελο **QueryB/convex_hull** εκτελέστε την εντολή **cargo run** για να τρέξετε τον κώδικα.

Προηγμένα Χαρακτηριστικά

Εκτελέστε τον κώδικα του **visualise.py** για να δείτε το αποτέλεσμα του Convex Hull σε γραφική μορφή.

```
┌─mauragkas@archlinuxlp ~/git/Project_MDDS/QueryB/convex_hull ‹main●›
└─λ ./visualise.py
Usage: ./visualiser.py [points|edges|planes]
```

Code Documentation

Modules

- convex_hull.rs: Περιέχει τον κώδικα του αλγορίθμου του QuickHull καθώς και την υλοποίηση του Convex Hull structure.
- point.rs: Περιέχει τον κώδικα για την υλοποίηση των σημείων στον τρισδιάστατο χώρο.
- edge.rs: Περιέχει τον κώδικα για την υλοποίηση των ακμών στον τρισδιάστατο χώρο.
- plane.rs: Περιέχει τον κώδικα για την υλοποίηση των επιπέδων στον τρισδιάστατο χώρο. Επιπλέον περιέχει μια μέθοδο για τον υπολογισμό της normal ενός επιπέδου.
- functions.rs: Περιέχει τον κώδικα για την υλοποίηση των βασικών συναρτήσεων που χρησιμοποιούνται στον αλγόριθμο του QuickHull.
- hush_stuff.rs: Περιέχει τον κώδικα για την υλοποίηση των βασικών συναρτήσεων που χρησιμοποιούνται για την δημιουργία των points μέσω του hashing.

Functions

fn init_simplex(&mut self) :

Συνάρτηση που δημιουργεί το αρχικό simplex που θα χρησιμοποιηθεί για την κατασκευή του Convex Hull.

1. Εύρεση τεσσάρων μη γραμμικών σημείων από ένα δεδομένο σύνολο, διασφαλίζοντας ότι μπορούν να σχηματίσουν ένα έγκυρο τετράεδρο.

2. Δημιουργία τεσσάρων επιπέδων, καθένα από τα οποία αντιπροσωπεύει μια όψη του τετραέδρου.

3. Push τα επίπεδα σε ένα Vec που είναι μέρος της δομής του simplex, κατασκευάζοντας αποτελεσματικά τη γεωμετρία του simplex.

pub fn quick_hull(&mut self) :

Συνάρτηση που υλοποιεί τον αλγόριθμο του QuickHull.
1. Αρχικοποίηση του Convex Hull με την κατασκευή του αρχικού simplex.

2. Επαναληπτική εφαρμογή του αλγορίθμου μέχρι να εξαντληθούν τα επίπεδα που αντιπροσωπεύουν το Convex Hull.

3. Εύρεση των σημείων που βρίσκονται πάνω από το επίπεδο.

4. Εύρεση του σημείου που βρίσκεται πιο μακριά από το επίπεδο.

5. Εύρεση των επιπέδων που το σημείο βρίσκεται πάνω από αυτά.

6. Εύρεση των ακμών που αντιπροσωπεύουν τα επίπεδα.

7. Κατασκευή νέων επιπέδων με βάση τις ακμές.

8. Αφαίρεση των παλιών επιπέδων και προσθήκη των νέων.

9. Επανάληψη των παραπάνω βημάτων μέχρι να εξαντληθούν τα επίπεδα.

functions.rs:
Περιέχει τον κώδικα για την υλοποίηση των βασικών συναρτήσεων που χρησιμοποιούνται στον αλγόριθμο του QuickHull.

create_rng_ponts(it: u32) :
Συνάρτηση που δημιουργεί τυχαία σημεία στον τρισδιάστατο χώρο. (Χρησιμοποιείται για το testing του αλγορίθμου)

populate_point_vec() :
Συνάρτηση που περνει τα data απο το json και τα μετατρεπει σε σημεια.

cross_product(a: &Point, b: &Point) :
Συνάρτηση που υπολογίζει τον cross product.

dot_product(a: &Point, b: &Point) :
Συνάρτηση που υπολογίζει τον dot product.

magnitude(vector: &Point) :

Συνάρτηση που υπολογίζει το μέτρο ενός διανύσματος.

point_to_plane_distance(plane: &Plane, point: &Point) :
Βοηθητική συνάρτηση που υπολογίζει την απόσταση ενός σημείου από ένα επίπεδο.

farthest_point_from_plane(plane: &Plane, points: &[Point]) :
Συνάρτηση που υπολογίζει το σημείο που είναι το πιο μακριά από ένα επίπεδο.

point_above_plane(plane: &Plane, point: &Point) :
Συνάρτηση που ελέγχει αν ένα σημείο είναι πάνω από ένα επίπεδο return true αν είναι αλλιώς false βαση του dot product.

find_non_collinear_points(points: &Vec<Point>) :
Συνάρτηση που βρίσκει τα σημεία που δεν είναι συγγραμικά για την κατασκευή του initial simplex. (περιέχει βοηθητικές συναρτήσεις)

save_to_json<T>(filename: &str, data: &T):
Αποθηκεύει γενικά δεδομένα σε json σε ένα δοσμένο filename.

hash_stuff.rs :
Χρησιμοποιεί ένα συνδυασμό μιας προσαρμοσμένης δομής Env και της μακροεντολής lazy_static! για να φορτώσει και να αναλύσει τις μεταβλητές περιβάλλοντος από ένα αρχείο μία φορά κατά την εκτέλεση, παρέχοντας έναν αποτελεσματικό και επαναχρησιμοποιήσιμο μηχανισμό για πρόσβαση σε αυτές τις ρυθμίσεις σε όλη την εφαρμογή.

Υλοποιεί μια δομή Data για την αναπαράσταση δομημένων δεδομένων, αξιοποιώντας τη βιβλιοθήκη serde για εύκολη σειριοποίηση και αποσειριοποίηση. Αυτό επιτρέπει τον ευέλικτο χειρισμό δεδομένων με προσαρμοσμένα ονόματα πεδίων για να ταιριάζουν με εξωτερικές μορφές δεδομένων (π.χ. JSON).

Προσφέρει μια απλή συνάρτηση κατακερματισμού συμβολοσειρών.

Visualization

Αυτό το Python script είναι σχεδιασμένο για την οπτικοποίηση γεωμετρικών δεδομένων 3D, ειδικότερα σημείων, ακμών και επιπέδων. Είναι ιδιαίτερα χρήσιμο για εφαρμογές όπως η απεικόνιση των κορυφών, των ακμών και των όψεων ενός κυρτού περιβλήματος. Το script υποστηρίζει τρεις λειτουργίες: σημεία, ακμές και επίπεδα, κάθε μία προσφέρει μια μοναδική προοπτική στα δεδομένα.

Data Structure

Το script αναμένει ένα αρχείο JSON με όνομα convex_hull.json που περιέχει τα γεωμετρικά δεδομένα δομημένα ως εξής:

Σημεία: Μια λίστα από λεξικά, καθένα αντιπροσωπεύοντας ένα σημείο με συντεταγμένες x, y, και z.

Ακμές: (Για τη λειτουργία ακμών) Μια λίστα από λεξικά που αντιπροσωπεύουν τις ακμές, με κάθε ακμή ορισμένη από δύο σημεία (start και end).

Επίπεδα: (Για τη λειτουργία επιπέδων) Μια λίστα από λεξικά, κάθε ένα αντιπροσωπεύοντας ένα επίπεδο ορισμένο από τρία σημεία και ένα κανονικό διάνυσμα.

Παράδειγμα δομής JSON: (μας αφορούν τα σημεία, τα επίπεδα και οι ακμές)

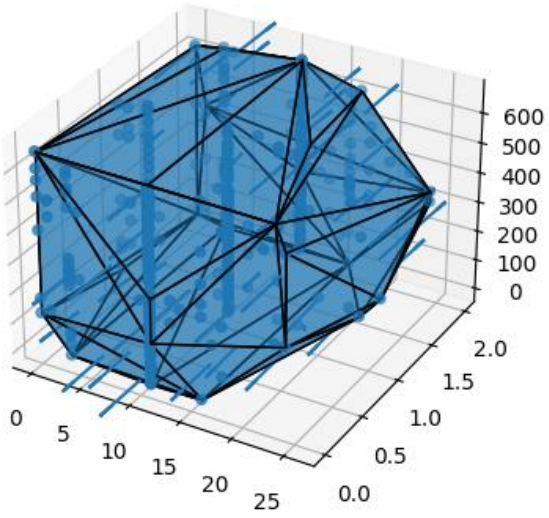
```
{
```

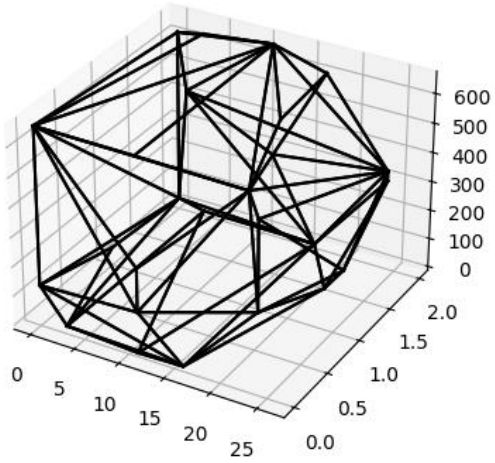
```
  "points": [{"x": 1, "y": 2, "z": 3}, ...],
  "edges": [{"start": {"x": 1, "y": 2, "z": 3}, "end": {"x": 4, "y": 5, "z": 6}}, ...],
  "planes": [{"point_a": {"x": 1, "y": 2, "z": 3}, "point_b": {"x": 4, "y": 5, "z": 6},
"point_c": {"x": 7, "y": 8, "z": 9}, "normal": {"x": 0, "y": 0, "z": 1}}, ...]
}
```

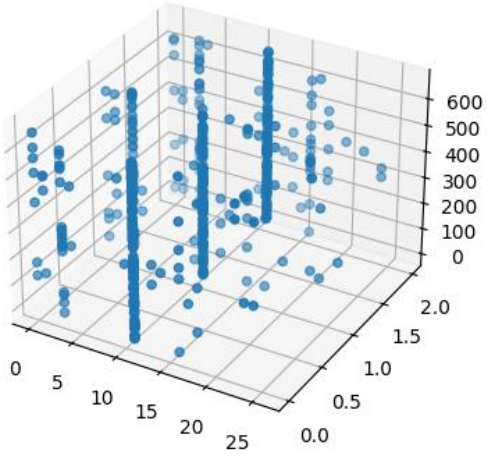
Function Visualiser

- **get_the_data_from_file(filename):**
  Διαβάζει και αναλύει τα δεδομένα JSON από το καθορισμένο αρχείο.
- **plot_points(data):**
  Οπτικοποιεί τα σημεία στον 3D χώρο.
- **plot_edges(data):**
  Οπτικοποιεί τις ακμές που συνδέουν σημεία.
- **plot_planes(data):**
  Οπτικοποιεί τα επίπεδα, συμπεριλαμβανομένων των κανονικών διανυσμάτων.

Output:


600
500
400
300
200
100
0
0
5
10
15
20
25
0.0
0.5
1.0
1.5
2.0


600
500
400
300
200
100
0
0
5
10
15
20
25
0.0
0.5
1.0
1.5
2.0


600
500
400
300
200
100
0
0
5
10
15
20
25
0.0
0.5
1.0
1.5
2.0

Γ) Skyline Operator & 2D Convex Hull

Γενική Επισκόπηση

Το παρακάτω υποερώτημα του project αποτελεί την υλοποίηση του αλγορίθμου του 2D Convex Hull και του 2D Skyline Operator σε Python.

Πιο συγκεκριμένα, ο αλγόριθμος που υλοποιήθηκε είναι ο αλγόριθμος του Graham Scan. Ο αλγόριθμος αυτός είναι ένας αλγόριθμος που χρησιμοποιείται για την εύρεση του Convex Hull ενός συνόλου σημείων στον δυσδιάστατο χώρο (στην περίπτωση μας).

Ο αλγόριθμος Skyline Operator 2D επιστρέφει το σύνολο των σημείων που αποτελούν το Skyline, δηλαδή τα σημεία που δεν μπορούν να κυριαρχηθούν από άλλα σημεία σε κάποιο άλλο κριτήριο. Στην περίπτωση μας υπολογίζουμε τα 4 υποσύνολα: 1ο Subset μικρότερες τιμές για d1 και d2, 2ο Subset μια μικρότερη τιμή d1 και μια μεγαλύτερη τιμή d2, 3ο Subset μια μεγαλύτερη τιμή d1 και μια μικρότερη τιμή d2, 4ο Subset μεγαλύτερες τιμές σε όλες τις διαστάσεις.

Χρήση

Βασική Χρήση

Για το αρχείο test.py εκτελέστε τον κώδικα με την εντολή ./test.py

Προηγμένα Χαρακτηριστικά

Η Advanced χρήση περιλαμβάνει την τροποποίηση του δέντρου διαστημάτων με την εισαγωγή νέων διαστημάτων και τη χρήση του δέντρου για σύνθετα ερωτήματα που περιλαμβάνουν πολλαπλά διαστήματα ή σημεία.

Code Documentation

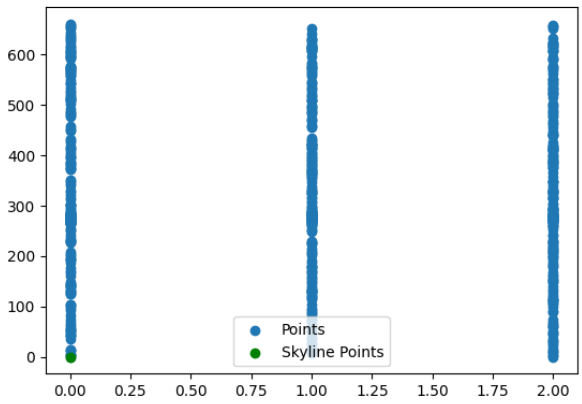
Functions

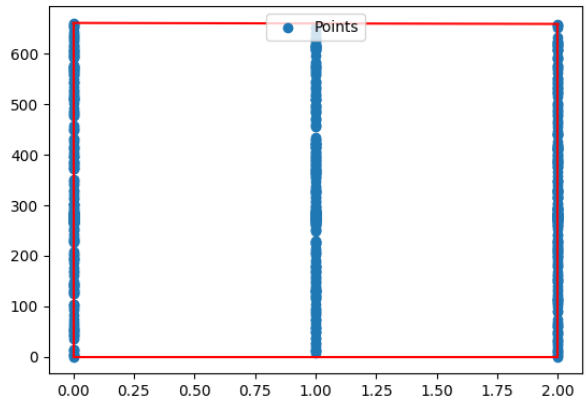
- **def convex_hull(points):**
  Περιέχει τον κώδικα του αλγορίθμου του Graham Scan καθως και την υλοποίηση του Convex Hull structure ενος συνόλου 2D σημείων.
- **def orientation(p, q, r):** Βοηθητική συνάρτηση για την εύρεση του Convex Hull.
  Συγκρίνει 3 σημεία και βρίσκει τον προσανατολισμό τους. Επιστρέφει 0 αν είναι συγγραμικά, 1 αν είναι δεξιόστροφος και 2 αν είναι αριστερόστροφος.
- **def skyline:**
  Περιέχει τον κώδικα για την υλοποίηση των skyline σημείων στον δυσδιάστατο χώρο.
- **def is_dominated:**
  Είναι βοηθητική συνάρτηση για την υλοποίηση του Skyline Operator. Περιέχει τον κώδικα για την εύρεση των

σημείων που δεν κυριαρχούνται από όλα τα υπόλοιπα. Ανάλογα με το case που επιλέγουμε βρίσκει τα σημεία για:
1ο subset: MIN d1, MIN d2
2ο subset: MIN d1, MAX d2
3ο subset: MAX d1, MIN d2
4ο subset: MAX d1, MAX d2

- **hash_function(string):**
  Περιέχει τον κώδικα για την μετατροπή του DBLP_Records το οποίο είναι σε μορφή string σε μορφή integer μέσω του hashing.

Output:





Δ) Skyline Layers & Convex Hull

Γενική Επισκόπηση

Το παρακάτω υποερώτημα του project αποτελεί την υλοποίηση του αλγορίθμου του 2D Convex Hull και του 2D Skyline Layers σε Python.

Πιο συγκεκριμένα, ο αλγόριθμος που υλοποιήθηκε είναι ο αλγόριθμος του 1Graham Scan1. Ο αλγόριθμος αυτός είναι ένας αλγόριθμος που χρησιμοποιείται για την εύρεση του Convex Hull ενός συνόλου σημείων στον δυσδιάστατο χώρο (στην περίπτωση μας).

Ο αλγόριθμος Skyline Layers 2D όπως τον αλγόριθμο Skyline Operator, επιστρέφει το σύνολο των σημείων που αποτελούν το Skyline για όσα όμως layers καθορίσει ο χρήστης. Δηλαδή τα σημεία που δεν μπορούν να κυριαρχηθούν από άλλα σημεία σε κάποιον άλλο κριτήριο για το 1ο layer και επαναλαμβάνουμε την διαδικασία χωρίς τα σημεία που έχουμε βρεί στα προηγούμενα layers μέχρι να φτάσουμε στο επιθυμητό layer. Στην περίπτωση μας υπολογίζουμε τα 4 υποσύνολα: 1ο Subset μικρότερες τιμές για d1 και d2, 2ο Subset μια μικρότερη τιμή d1 και μια μεγαλύτερη τιμή d2, 3ο Subset μια μεγαλύτερη τιμή d1 και μια μικρότερη τιμή d2, 4ο Subset μεγαλύτερες τιμές σε όλες τις διαστάσεις.

Χρήση

Βασική Χρήση

Για το αρχείο test.py εκτελέστε τον κώδικα με την εντολή ./test.py

Προηγμένα Χαρακτηριστικά

Η Advanced χρήση περιλαμβάνει την τροποποίηση του δέντρου διαστημάτων με την εισαγωγή νέων διαστημάτων και τη χρήση του δέντρου για σύνθετα ερωτήματα που περιλαμβάνουν πολλαπλά διαστήματα ή σημεία.

Code Documentation

Functions

- **def convex_hull(points):**
  Περιέχει τον κώδικα του αλγορίθμου του Graham Scan καθως και την υλοποίηση του Convex Hull structure ενος συνόλου 2D σημείων.

- **def orientation(p, q, r):** Βοηθητική συνάρτηση για την εύρεση του Convex Hull.
  Συγκρίνει 3 σημεία και βρίσκει τον προσανατολισμό τους. Επιστρέφει 0 αν είναι κολινέα, 1 αν είναι δεξιόστροφος και 2 αν είναι αριστερόστροφος.

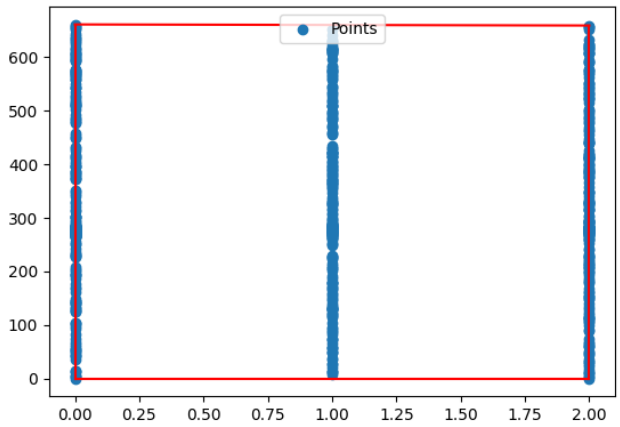
- **def find_skyline_layers:**
  Περιέχει τον κώδικα για την υλοποίηση των skyline layers στον δυσδιάστατο χώρο μέχρι και το τελευταίο layer που έχει καθοριστεί απο τον χρήστη.

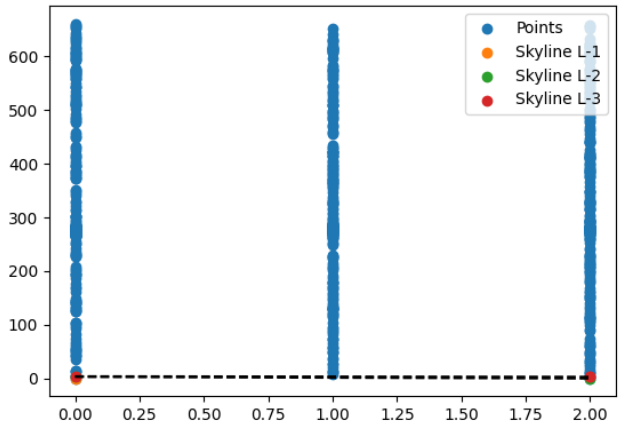
- **def is_dominated:** Είναι βοηθητική συνάρτηση της συνάρτησης def modified_skyline(points, dominance_case).
  Περιέχει τον κώδικα για την εύρεση των σημείων που δεν κυριαρχούνται από όλα τα υπόλοιοπα. Ανάλογα με το case που επιλέγουμε βρίσκει τα σημεία για:
  1ο subset: MIN d1, MIN d2
  2ο subset: MIN d1, MAX d2
  3ο subset: MAX d1, MIN d2
  4ο subset: MAX d1, MAX d2

- **def modified_skyline(points, dominance_case):** Είναι βοηθητική συνάρτηση που καλείται μέσω της συνάρτησης find_skyline_layers για να βρει τα skyline σημεία ενός layer.
  Περιέχει τον κώδικα όπου ξεχωρίζουμε σε διαφορετικές λίστες τα σημεία που αποτελούν το skyline μέχρι το τελευταίο layer που έχει εκτελεστεί και υπόλοιπα σημεία έτσι ώστε εάν πρέπει να συνεχίσουμε την αναζήτηση των skylines σε μεγαλύτερο layer να επαναλάβουμε την διαδικασία χωρίς τα σημεία που έχουμε ήδη βρει και αποθηκεύσαμε στην λίστα skyline_points = [].

- **hash_function(string):**
  Περιέχει τον κώδικα για την μετατροπή του DBLP_Records το οποίο είναι σε μορφή string σε μορφή integer μέσω του hashing.

Output:





E) Priority Search Tree για την Αναγνώριση Σημείων Skyline

Γενική Επισκόπηση

Αυτός ο συνοπτικός οδηγός επικεντρώνεται σε ένα Python script που χρησιμοποιεί ένα Priority Search Tree (PST) για να αναγνωρίσει σημεία skyline από ένα σύνολο δεδομένων περιγραφόμενο σε ένα αρχείο JSON. Το script τονίζει τη χρήση του PST για αποδοτική διαχείριση χωρικών δεδομένων και υπολογισμό σημείων skyline.

Ένα Priority Search Tree (PST) είναι μια εξειδικευμένη δομή δεδομένων που επιτρέπει την αποδοτική ερώτηση χωρικών δεδομένων. Αυτό το script χρησιμοποιεί ένα PST για να επεξεργαστεί εγγραφές που περιέχουν τα πεδία 'Awards' και 'DBLP_Record', αναγνωρίζοντας σημεία που αποτελούν το "skyline", όπου κανένα σημείο δεν κυριαρχείται από άλλο σε και τις δύο διαστάσεις.

Key Components

Priority Search Tree (PST)

Το PST οργανώνει σημεία σε δύο διαστάσεις, επιτρέποντας την αποδοτική ερώτηση εύρους και την αναγνώριση σημείων skyline και τα σημεία ταξινομούνται και τοποθετούνται αναδρομικά στο δέντρο βάσει των χωρικών τους χαρακτηριστικών. Το PST χρησιμοποιείται για να φιλτράρει και να αναγνωρίσει σημεία που δεν κυριαρχούνται από άλλα, καθορίζοντας το skyline.

Data Processing

- **Είσοδος:** Ένα αρχείο JSON με εγγραφές, κάθε μία έχοντας τα πεδία 'Awards' και 'DBLP_Record'.
- **Hash Function:** Μετατρέπει το 'DBLP_Record' σε μια αριθμητική τιμή hash, υπηρετώντας ως μία από τις χωρικές διαστάσεις.
- **Προετοιμασία:** Φιλτράρει τα κυριαρχημένα σημεία πριν από την κατασκευή του PST, εξασφαλίζοντας ότι το δέντρο περιέχει μόνο πιθανούς υποψηφίους για skyline.

Skyline Identification

Ο αλγόριθμος διασχίζει το PST για να βρει σημεία που δεν κυριαρχούνται από άλλα σε και τις δύο διαστάσεις, τον αριθμό των βραβείων και την τιμή hash του 'DBLP_Record'.
Κριτήρια Κυριαρχίας: Ένα σημείο θεωρείται ότι κυριαρχεί άλλο αν είναι ανώτερο σε και τις δύο διαστάσεις.

Visualization

Εργαλείο: Χρησιμοποιεί τη βιβλιοθήκη Matplotlib για να διαγραμματίσει το σύνολο δεδομένων και να επισημάνει τα σημεία skyline, παρέχοντας μια οπτική ανάλυση των αποτελεσμάτων.

Implementation Steps

1. Ανάγνωση και Επεξεργασία Δεδομένων JSON: Εξαγωγή και μετατροπή των 'Awards' και 'DBLP_Record' σε κατάλληλες αριθμητικές μορφές.
2. Κατασκευή PST: Δημιουργία του Priority Search Tree από τα επεξεργασμένα δεδομένα σημεία.
3. Αναγνώριση Σημείων Skyline: Χρήση του PST για την αποδοτική ανεύρεση και συγκέντρωση σημείων skyline.
4. Οπτικοποίηση Αποτελεσμάτων: Διαγράμμιση όλων των σημείων και διάκριση των σημείων skyline για εύκολη αναγνώριση.

Χρήση:

Για να χρησιμοποιήσετε το script, βεβαιωθείτε ότι το αρχείο δεδομένων JSON είναι σωστά διαμορφωμένο με τα απαραίτητα πεδία. Ενημερώστε το script με τις σωστές διαδρομές αρχείων τόσο για τα δεδομένα όσο και για οποιεσδήποτε απαιτούμενες διαμορφώσεις περιβάλλοντος.

Output: