Κατασκευή real dataset

Αποφασίσαμε να κατασκευάσουμε το δικό μας dataset με πραγματικά δεδομένα επιστημόνων, αντλούμενα από τη βάση δεδομένων της Wikipedia. Στο αρχείο **“download\_data.py”**, πραγματοποιείται ανάκτηση και επεξεργασία πληροφοριών σχετικά με επιστήμονες της επιστήμης των υπολογιστών, από τη Wikipedia. Ο κώδικας στοχεύει στη δημιουργία ενός dataset, που για κάθε επιστήμονα περιέχει το επώνυμό του, τον αριθμό βραβείων που έχει λάβει, καθώς και πληροφορίες σχετικά με την εκπαίδευσή του.

Αρχικά, με τη συνάρτηση **“get\_urls”**, ανακτώνται όλοι οι σύνδεσμοι (URLs) των επιστημόνων που περιέχονται στην κεντρική σελίδα με τη λίστα επιστημόνων της επιστήμης των υπολογιστών: <https://en.wikipedia.org/wiki/List_of_computer_scientists>. Καθένας από αυτούς τους συνδέσμους αποθηκεύεται σε μία λίστα, που είναι και αυτό που επιστρέφει η συνάρτηση.

Στη συνέχεια, καλούμε τη συνάρτηση **“get\_scientist\_info”** για κάθε σελίδα επιστήμονα, προκειμένου να εξάγουμε τις ζητούμενες πληροφορίες για καθέναν από τους επιστήμονες της λίστας. Για το σκοπό αυτό, κάνουμε χρήση της βιβλιοθήκης **BeautifulSoup**, με την οποία μπορούμε να επεξεργαστούμε το HTML περιεχόμενο κάθε σελίδας προκειμένου να βρούμε και να εξάγουμε συγκεκριμένα στοιχεία, όπως συνδέσμους, κείμενο και άλλα χαρακτηριστικά. Είναι ένα απαραίτητο εργαλείο γι’ αυτόν τον κώδικα, καθώς παρέχει τη δυνατότητα ευέλικτης και αποτελεσματικής ανάλυσης του HTML περιεχομένου, καθιστώντας την ανάκτηση και επεξεργασία των πληροφοριών ευκολότερη και πιο αυτοματοποιημένη.

Όταν έχουν ανακτηθεί όλα τα δεδομένα, δημιουργείται ένα DataFrame, το οποίο περιέχει τις πληροφορίες για κάθε επιστήμονα. Ακολούθως, ο κώδικας πραγματοποιεί διορθώσεις στο DataFrame, βάσει των πληροφοριών που περιέχονται στο αρχείο **“corrections.txt”**. Το αρχείο αυτό περιέχει εγγραφές με το επώνυμο του επιστήμονα, τον αριθμό των βραβείων του, καθώς και ενδεχομένως τον αριθμό γραμμής (index) του συγκεκριμένου επιστήμονα στο DataFrame. Σκοπός αυτής της διαδικασίας είναι να διορθώσουμε ορισμένες λανθασμένες καταχωρήσεις βραβείων που έγιναν κατά την εξαγωγή των δεδομένων, έτσι ώστε να ανταποκρίνονται στην πραγματικότητα. Οι λανθασμένες καταχωρήσεις οφείλονται στο γεγονός ότι δεν είναι όλες οι σελίδες της Wikipedia δομημένες κατά τον ίδιο τρόπο όσον αφορά το HTML περιεχόμενό τους.

Αφού εφαρμοστούν όλες οι απαραίτητες διορθώσεις στο DataFrame, τα δεδομένα εξάγονται σε ένα αρχείο CSV με την ονομασία **“scientists\_data.csv”**, που είναι και το τελικό dataset που χρησιμοποιήσαμε για την υλοποίηση της εργασίας. Το τελικό dataset περιέχει δεδομένα 254 επιστημόνων, που είναι αυτοί για τους οποίους υπήρχε κείμενο σχετικά με την εκπαίδευσή τους στην αντίστοιχη σελίδα τους.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Συναρτήσεις κατασκευής και αναζήτησης σε πολυδιάστατη δομή

Προτού ξεκινήσουμε την αναφορά μας σε κάθε μία εκ των τεσσάρων πολυδιάστατων δομών που υλοποιήσαμε, αξίζει να σημειώσουμε πως για κάθε δομή υλοποιήθηκαν ξεχωριστά μία συνάρτηση κατασκευής της δομής (συναρτήσεις build) και μία συνάρτηση αναζήτησης στη δομή (συναρτήσεις query).

Η συνάρτηση build είναι υπεύθυνη για την κατασκευή της εκάστοτε δομής και για την εισαγωγή σ’ αυτήν όλων των διαθέσιμων σημείων. Αρχικά, διαβάζουμε το αρχείο CSV με τα δεδομένα των επιστημόνων και δημιουργούμε ένα αντιπροσωπευτικό σημείο (x, y) για τον κάθε επιστήμονα, όπου x η αριθμητική τιμή του πρώτου γράμματος του επωνύμου του επιστήμονα ([‘Α’, ‘Ζ’] → [0, 25]) και y ο αριθμός των βραβείων που έχει λάβει. Σε συγκεκριμένες περιπτώσεις, χρησιμοποιήθηκε και ο αριθμός γραμμής (index) του επιστήμονα στο DataFrame ως χρήσιμη πληροφορία για να ξεχωρίζουμε τους επιστήμονες που τυγχάνει να έχουν ίδιες συντεταγμένες x, y βάσει των στοιχείων τους. Αφού δημιουργηθούν τα σημεία, κατασκευάζουμε την πολυδιάστατη δομή βάσει αυτών.

Η συνάρτηση query δέχεται πάντα τέσσερις παραμέτρους, που είναι: η πολυδιάστατη δομή που επιστράφηκε από τη συνάρτηση build, δύο γράμματα που αντιπροσωπεύουν το ελάχιστο και το μέγιστο όριο της συντεταγμένης x, και έναν αριθμό βραβείων που αντιπροσωπεύει το ελάχιστο όριο της συντεταγμένης y. Η συνάρτηση αφού μετατρέπει τα δύο γράμματα στις αντίστοιχες αριθμητικές τους τιμές, αποστέλλει ερώτημα αναζήτησης στη δομή για τα δοθέντα διαστήματα τιμών, κάνοντας χρήση της αντίστοιχης μεθόδου αναζήτησης. Βάσει των αποτελεσμάτων της αναζήτησης, ανακτά τα δεδομένα των επιστημόνων από το CSV αρχείο (χρησιμοποιώντας το index) και τα αποθηκεύει σε μία λίστα final\_results με τη μορφή λεξικών.

Range Tree

Για την αποτελεσματική διαχείριση σημείων δύο διαστάσεων με συντεταγμένες x, y, υλοποιήσαμε ένα δισδιάστατο (2D) Range Tree. Η προσέγγιση που ακολουθήσαμε περιλαμβάνει την κατασκευή του από ισορροπημένα δυαδικά δέντρα αναζήτησης (BBSTs). Αρχικά, κατασκευάζεται ένα κύριο BBST με βάση τις συντεταγμένες x των σημείων. Κάθε κόμβος του κύριου δέντρου αποθηκεύει ένα 1D Range Tree (y-tree) που περιλαμβάνει όλα τα σημεία που έχουν την ίδια συντεταγμένη x με τον κόμβο. Κάθε y-tree είναι με τη σειρά του κι αυτό ένα BBST, αλλά κατασκευασμένο με βάση τις συντεταγμένες y των σημείων που περιέχει. Επιτρέπει την αναζήτηση σημείων με βάση τη συντεταγμένη y εντός ενός διαστήματος, για σημεία που έχουν την ίδια συντεταγμένη x.

Κατά την εισαγωγή ενός νέου σημείου, ελέγχεται αν υπάρχει ήδη κόμβος με την ίδια συντεταγμένη x. Αν ναι, το σημείο προστίθεται στο αντίστοιχο y-tree του κόμβου. Διαφορετικά, δημιουργείται ένας νέος κόμβος στο 2D δέντρο. Τόσο το 2D δέντρο όσο και τα 1D δέντρα διατηρούνται ισορροπημένα μέσω περιστροφών κόμβων, με βάση τον παράγοντα ισορροπίας του κάθε κόμβου, ο οποίος υπολογίζεται ως η διαφορά των υψών των υπο-δέντρων του. Ακολουθεί μία γραφική απεικόνιση του 2D δέντρου.

Εικόνα που περιέχει σκίτσο/σχέδιο, διάγραμμα, λευκό, ζωγραφιά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 1: Κατασκευή ενός 2D Range Tree, χρησιμοποιώντας BBSTs. Κάθε κόμβος του 2D δέντρου έχει ένα associated 1D Range Tree για την αναζήτηση εύρους εντός ενός y-διαστήματος.

Κατά την αναζήτηση ενός εύρους στο 2D Range Tree, το κύριο δέντρο προσπελαύνεται πρώτα για να βρεθούν οι κόμβοι που «πέφτουν» εντός του διαστήματος x. Για κάθε κόμβο που βρίσκεται εντός του διαστήματος x, προσπελαύνεται το αντίστοιχο y-tree για να βρεθούν τα σημεία που «πέφτουν» εντός του διαστήματος y. Η συνδυασμένη προσπέλαση των δύο δέντρων επιτρέπει την αποτελεσματική εύρεση όλων των σημείων που βρίσκονται εντός ενός ερωτήματος διαστήματος (range query).

Quad Tree

Ένα Quad Tree είναι μία δομή δεδομένων δέντρου που χρησιμοποιείται για την αποτελεσματική οργάνωση και αναζήτηση σημείων σε δισδιάστατους χώρους. Το δέντρο αυτό χωρίζει τον χώρο σε τέσσερα τμήματα (ή κόμβους) και κάθε τμήμα μπορεί να χωριστεί περαιτέρω ανάλογα με το πλήθος των σημείων που περιέχει.

Στον κώδικα που υλοποιήσαμε συναντάμε τρεις κλάσεις:

* **Point**: αναπαριστά ένα σημείο στον δισδιάστατο (2D) χώρο, με συντεταγμένες x, y, το οποίο μπορεί να φέρει κάποια ωφέλιμη πληροφορία data.
* **Rect**: αναπαριστά ένα ορθογώνιο, με συντεταγμένες κέντρου (cx, cy), πλάτος w και ύψος h. Διαθέτει μία μέθοδο για τον έλεγχο αν ένα σημείο βρίσκεται εντός του ορθογωνίου (μέθοδος contains) και άλλη μία για τον έλεγχο αν τέμνει με κάποιο άλλο ορθογώνιο (μέθοδος intersects).
* **QuadTree**: αναπαριστά τον κόμβο ενός Quad Tree. Κάθε κόμβος έχει ένα ορθογώνιο boundary που είναι ο χώρος που καταλαμβάνει, μία λίστα points με τα σημεία που φιλοξενεί και τέσσερις υπο-κόμβους (sw, se, ne, nw). Αν ένας κόμβος φτάσει σε έναν καθορισμένο αριθμό σημείων, τότε χωρίζεται στους τέσσερις υπο-κόμβους, καθένας από τους οποίους καταλαμβάνει ένα τεταρτημόριο του αρχικού χώρου.

Κατά την εισαγωγή ενός νέου σημείου, το δέντρο ελέγχει σε ποιον κόμβο ανήκει και το προσθέτει σ’ αυτόν. Αν ο κόμβος έχει ήδη το μέγιστο επιτρεπόμενο πλήθος σημείων, το οποίο ορίζουμε τυπικά ως 4, τότε ακολουθείται η διαδικασία διάσπασης του κόμβου και το σημείο προστίθεται στον κατάλληλο υπο-κόμβο απ’ αυτούς που προκύπτουν.

Η αναζήτηση σημείων σε ένα Quad Tree είναι αποτελεσματική, καθώς το δέντρο επιτρέπει την ταχεία πρόσβαση σε συγκεκριμένες περιοχές του χώρου. Πιο συγκεκριμένα, αν ζητηθούν να βρεθούν όλα τα σημεία εντός ενός ορθογωνίου, το δέντρο ελέγχει μόνο τους κόμβους που τέμνουν το ορθογώνιο, αγνοώντας όλους τους υπόλοιπους.

A diagram of a graph

Description automatically generated

Εικόνα 2: Γραφική απεικόνιση του Quad Tree. Με κόκκινο χρώμα είναι σημειωμένο το ορθογώνιο αναζήτησης (search boundary) και τα σημεία που περιέχονται σ’ αυτό.

R-Tree

Η βασική ιδέα ενός R-tree είναι να ομαδοποιεί τα δεδομένα σε ορθογώνιες περιοχές, οι οποίες αποτελούν τους κόμβους του δέντρου. Καθώς το δέντρο αναπτύσσεται, τα ορθογώνια μπορεί να υπερκαλύπτονται, αλλά προσπαθούν να ελαχιστοποιούν την υπερκάλυψη και το μέγεθός τους.

Για την υλοποίηση της δομής, χρησιμοποιήσαμε τη βιβλιοθήκη της Python rtree. Η βιβλιοθήκη rtree βασίζεται στη βιβλιοθήκη libspatialindex, η οποία παρέχει υλοποιήσεις για διάφορες δομές δεδομένων χωρικής ευρετηρίασης, συμπεριλαμβανομένων των R-trees. Ένα από τα κύρια πλεονεκτήματα της χρήσης της βιβλιοθήκης rtree, είναι η αποτελεσματικότητα και η ταχύτητα που προσφέρει στις βασικές λειτουργίες εισαγωγής και αναζήτησης ενός R-tree, ακόμα και σε μεγάλα σύνολα δεδομένων.

Ο κώδικας που υλοποιήσαμε περιλαμβάνει την κλάση RTree, η οποία χρησιμοποιεί το ευρετήριο (Index) της βιβλιοθήκης rtree για να διαχειριστεί την εσωτερική δομή του δέντρου και να παρέχει αποτελεσματικές λειτουργίες εισαγωγής και αναζήτησης. Η μέθοδος insert της κλάσης επιτρέπει την εισαγωγή στοιχείων στο δέντρο με βάση τις χωρικές τους συντεταγμένες, ενώ η μέθοδος search παρέχει τη δυνατότητα αναζήτησης στοιχείων με βάση ένα δοθέν bounding box (query\_bbox).

Όταν καλείται η μέθοδος insert, το στοιχείο προστίθεται στο δέντρο με βάση τις συντεταγμένες του. Αυτό γίνεται με την εύρεση του πιο κατάλληλου κόμβου (ή φύλλου) για την εισαγωγή του στοιχείου, ξεκινώντας από τη ρίζα του δέντρου. Μετά την εισαγωγή, ελέγχεται αν ο κόμβος υπερβαίνει το μέγιστο επιτρεπόμενο πλήθος στοιχείων. Εάν συμβεί αυτό, τότε ο κόμβος πρέπει να διαιρεθεί σε δύο νέους κόμβους. Η βιβλιοθήκη rtree χρησιμοποιεί εσωτερικά αλγορίθμους που βελτιστοποιούν τη διαδικασία της διαίρεσης (split), ελαχιστοποιώντας την υπερκάλυψη και το μέγεθος των bounding boxes. Αν ένας κόμβος διαιρεθεί, μπορεί να χρειαστεί να διαιρεθεί και ο γονικός του κόμβος, και ούτω καθεξής, μέχρι να φτάσουμε στη ρίζα του δέντρου. Συνολικά η μέθοδος insert διασφαλίζει ότι το δέντρο παραμένει ισορροπημένο και ότι τα bounding boxes είναι όσο το δυνατόν πιο αποτελεσματικά, προκειμένου να βελτιώσει την απόδοση των αναζητήσεων.

Όταν καλείται η μέθοδος search με ένα δοθέν bounding box ως όρισμα, τότε αυτή καλεί με τη σειρά της τη μέθοδο intersection της βιβλιοθήκης rtree, με όρισμα το δοθέν bounding box. Η μέθοδος αυτή ελέγχει τα bounding boxes των κόμβων του R-tree για να βρει όλα τα στοιχεία που τέμνονται με το δοθέν bounding box. Αυτό γίνεται με αποτελεσματικό τρόπο, χρησιμοποιώντας τη δομή του R-tree για να περιορίσει τον αριθμό των στοιχείων που πρέπει να ελεγχθούν. Τέλος, τα στοιχεία που επιστρέφονται από τη μέθοδο intersection, συλλέγονται και επιστρέφονται από τη search.

Υλοποίηση Locality Sensitivity Hashing (LSH)

Ο αλγόριθμος Locality-Sensitive Hashing (LSH) είναι ένας αλγόριθμος που χρησιμοποιείται κυρίως στον τομέα της αναζήτησης και της αναγνώρισης πλησιέστερων γειτόνων σε δεδομένα πολυδιάστατους χώρους. Ο στόχος του αλγορίθμου LSH είναι να μετατρέψει τα δεδομένα ώστε να είναι ευαίσθητα στη γεωμετρική ομοιότητα, επιτρέποντας την αποδοτική αναζήτηση πλησιέστερων γειτόνων χωρίς την ανάγκη εξέτασης όλων των δυνατών συνδυασμών. Στην παρούσα εργασία ο αλγόριθμος υλοποιείται από τις κλάσεις MinHash και LSH στο αρχείο lsh.py.

Τα βασικά «συστατικά» στοιχεία του LSH είναι τα παρακάτω

**Συνάρτηση Hashing:** Ο LSH χρησιμοποιεί μια συνάρτηση κατακερματισμού (hash function) για να μετατρέψει τα δεδομένα από τον αρχικό χώρο σε έναν χώρο μικρότερων διαστάσεων. Αυτή η συνάρτηση hash προσπαθεί να είναι ευαίσθητη στη γεωμετρική ομοιότητα, δηλαδή να "κατακερματίζει" παρόμοια δεδομένα κοντά στην ίδια θέση. Αυτή η λειτουργία υλοποιείται στην κλάση MinHash από τις μεθόδους \_hash, build\_functions και \_signature\_matrix.

**Σύγκριση Buckets:** Οι μετατροπές των δεδομένων δημιουργούν "κάδους" ή "buckets" στον νέο χώρο. Δύο δεδομένα που καταλήγουν στον ίδιο κάδο θεωρούνται ότι είναι πιθανά πολύ κοντά γεωμετρικά στον αρχικό χώρο. Αυτή η λειτουργία υλοποιείται στη μέθοδο \_get\_candidates της κλάσης LSH.

**Αναζήτηση Γειτόνων:** Αφού τα δεδομένα αντιστοιχίστηκαν σε κάδους, μπορούμε να αναζητήσουμε γρήγορα πλησιέστερους γείτονες εξετάζοντας μόνο τους κάδους που περιέχουν τα δεδομένα που μας ενδιαφέρουν. Αυτή η λειτουργία υλοποιείται στη μέθοδο neighbors της κλάσης LSH.

Μετρικές ομοιότητας

Στο αρχείο tools.py έχουμε υλοποιήσει τις συναρτήσεις που χρησιμοποιούνται για την εύρεση της ομοιότητας μεταξύ των διανυσμάτων στον πολυδιάστατο χώρο.

Η μετρική ομοιότητας cosine, γνωστή και ως "cosine similarity," είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει τον βαθμό της ομοιότητας ανάμεσα σε δύο διανύσματα σε έναν πολυδιάστατο χώρο. Συνήθως, αυτός ο πολυδιάστατος χώρος είναι χώρος χαρακτηριστικών όπου κάθε διάσταση αντιστοιχεί σε ένα χαρακτηριστικό ή μια μετρική. Η cosine similarity είναι ιδιαίτερα χρήσιμη στον τομέα της ανάκτησης πληροφοριών, της αναζήτησης κειμένου και της συστατικής φιλτράρισης. Συνήθως, χρησιμοποιείται για τη σύγκριση της ομοιότητας μεταξύ κειμένων ή σετ δεδομένων όπου κάθε στοιχείο αναπαρίσταται ως ένα διάνυσμα χαρακτηριστικών. Η cosine similarity επιτρέπει να αξιολογηθεί πώς δύο διανύσματα σχετίζονται μεταξύ τους, ανεξάρτητα από το μέγεθος των διανυσμάτων, και μπορεί να βοηθήσει στην ανίχνευση παρόμοιων στοιχείων σε μεγάλα σύνολα δεδομένων. Αυτή η μετρική ομοιότητας υλοποιείται στη μέθοδο cosine\_similarity.

Η μετρική ομοιότητας Jaccard είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει τον βαθμό της ομοιότητας ανάμεσα σε δύο σύνολα δεδομένων. Συνήθως, χρησιμοποιείται όταν τα δεδομένα προτιμούνται να αναπαριστούν ως δυαδικές (binary) μεταβλητές, που μπορούν να είναι είτε παρόντα (1) είτε απόντα (0). Η Jaccard similarity υπολογίζει την ομοιότητα ανάμεσα σε δύο σύνολα A και B ως τον λόγο του αριθμού των στοιχείων που υπάρχουν και στα δύο σύνολα προς τον αριθμό των στοιχείων που υπάρχουν σε οποιοδήποτε από τα δύο σύνολα. Ουσιαστικά, αυτή η μετρική συγκρίνει τον αριθμό των κοινών δυαδικών στοιχείων στα δύο σύνολα με τον συνολικό αριθμό των διαφορετικών δυαδικών στοιχείων που υπάρχουν σε αυτά τα σύνολα. Ο κύριος περιορισμός της Jaccard similarity είναι ότι δεν λαμβάνει υπόψη τη συχνότητα των διαφορετικών στοιχείων, αλλά μόνο το γεγονός ότι αυτά τα στοιχεία υπάρχουν ή όχι στα σύνολα. Αυτή η μετρική ομοιότητας υλοποιείται στη μέθοδο jaccard\_binary.

Εφαρμογή του αλγόριθμου στα αποτελέσματα αναζήτησης

Στο αρχείο main.py έχουμε υλοποιήσει τη μέθοδο lsh\_test η οποία δέχεται ως όρισμα μια λίστα αποτελεσμάτων από την αναζήτηση, ένα κατώτατο όριο ομοιότητας και των αριθμό των buckets (ανάλογα με το πλήθος των αποτελεσμάτων που θα έχουμε). Ακόμη, έχουμε θέσει ως default μετρική ομοιότητας την Jaccard. Η επιλογή αυτή έχει προτιμηθεί εξαιτίας του ότι το μητρώο των αποτελεσμάτων γίνεται encode σε one hot αντί για tf-idf (όπου και θα ήταν προτιμότερο να χρησιμοποιούμε τη μετρική συνημιτόνου). Έχουμε επιλέξει να μεταβάλλουμε τον αριθμό των buckets ανάλογα με το πλήθος των αποτελεσμάτων ώστε να έχουμε όσο γίνεται πιο σταθερό πλήθος candidate pairs (π.χ. όταν έχουμε λίγα results και χρησιμοποιούμε πάρα πολλά buckets είναι απίθανο να παρατηρήσουμε όμοιες εξόδους από το lsh\_test). Παρακάτω βλέπουμε πως εφαρμόζεται ο αλγόριθμος LSH στα αποτελέσματα της αναζήτησης του R-Tree όπου έχουμε θέσει threshold = 0.6, letters = Α-Ζ, awards = 3.

A black background with many small colored lines

Description automatically generated with medium confidence

Εικόνα 1: Εφαρμογή του αλγόριθμου LSH σε R-Tree search results με similarity threshold.

Από τα 115 αποτελέσματα, έχουμε συνολικά 4896 πιθανά ζεύγη και 130 από αυτά παρουσιάζουν υψηλή μετρική ομοιότητας ( > 0.6).

Αν κάνουμε μια απόπειρα να επαληθεύσουμε την ομοιότητα πάνω στα ζεύγη που βρέθηκαν παρατηρούμε την ύπαρξη των παρακάτω ζευγών

A screenshot of a computer

Description automatically generated

Εικόνα 2: Ένα ζεύγος που παρουσιάζει μετρική ομοιότητας jaccard 0.662

A screenshot of a computer

Description automatically generated

Εικόνα 3: Ένα ζεύγος που παρουσιάζει μετρική ομοιότητας jaccard 0.727

Πειραματική σύγκριση των πολυδιάστατων δομών μέσα από τον αλγόριθμο

Στο αρχείο experiments.py έχουμε υλοποιήσει ένα script που τρέχει τον αλγόριθμο για n φορές σε κάθε μια από τις 4 πολυδιάστατες δομές και επιστρέφει τους μέσους χρόνους εκτέλεσης του όπως παρακάτωA graph of blue rectangular bars

Description automatically generated with medium confidence

Εικόνα 4: Διάγραμμα μέσου χρόνου εκτέλεσης αλγόριθμου LSH για κάθε μια από τις πολυδιάστατες δομές δεδομένων.