ΠΟΛΥΔΙΑΣΤΑΤΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΓΕΩΜΕΤΡΙΑ

Project 1

ΜΕΛΗ ΟΜΑΔΑΣ:

Χρονόπουλος Θεοδόσιος, 1072595

Καρβούνη Άντρια, 1080481

Πουρής Βασίλειος, 1072485

Ταμβάκης Θωμάς Χρυσοβαλάντης, 1072631

Περιεχόμενα

CRAWLER……..3

KD TREE……..4

QUAD TREE……5

R TREE……6

RANGE TREE……...7

LSH……8

MAIN……10

RESULTS……...11

CRAWLER

Συνάρτηση *get_dblp_records(dblp_url):*
Η συνάρτηση παίρνει ως όρισμα το url του επιστήμονα στο site dblp.org και επιστρέφει τον αριθμό των records του επιστήμονα που υπάρχουν στο site. Χρησιμοποιεί την βιβλιοθήκη BeautifulSoup της python και πιο συγκεκριμένα κάνει αναζήτηση τα “li” στο class = ‘section’ με id ‘publ-section’.

Συνάρτηση *get_scientist_data(url):*
Η συνάρτηση παίρνει ως όρισμα το url του επιστήμονα στη Wikipedia και επιστρέφει το επώνυμο , τα βραβεία, την εκπαίδευση και τα dblp_records από το site dblp. Κάνει χρήση της βιβλιοθήκης BeautifulSoup. Το επώνυμο εξάγεται από ‘span’ της κλάσης ‘mw-page-title-main’. Τα βραβεία εξάγονται και από το κείμενο της wikipedia και από το infobox του επιστήμονα στη wikipedia και επιλέγεται ο μεγαλύτερος αριθμός για να πιο ακριβή τα αποτελέσματα. Η εκπαίδευση εξάγεται από το κείμενο της wikipedia που έχει τίτλο που περιέχει τη λέξη education και τα dblp_records εξάγονται με την χρήση της παραπάνω συνάρτησης αφού έχουμε πρώτα εξάγει το link για το site dblp του επιστήμονα από authority_control_box του site της Wikipedia.

Συνάρτηση *get_scientist_urls():*
Η συνάρτηση χρησιμοποιώντας την βιβλιοθήκη BeautifulSoup επιστρέφει μια λίστα με τα url του κάθε επιστήμονα στη wikipedia από το site που μας δίνεται στην εκφώνηση.

Το πρόγραμμα αρχικά με την συνάρτηση ***get_scientist_urls()*** κάνει extract τα url του κάθε επιστήμονα στη wikipedia. Στη συνέχεια για κάθε url, με την χρήση της συνάρτησης ***get_scientist_data(url)*** εξάγει το επώνυμο, τα βραβεία, την εκπαίδευση και dblp_records του κάθε επιστήμονα. Αν το education δεν είναι κενό και τα dblp_records δεν είναι 0 προσθέτονται στη λίστα, καθώς σε αρκετούς επιστήμονες δεν υπήρχε πεδίο education ή link για dblp τα δεδομένα προσθέτονται στον πίνακα data[]. Ο πίνακας αυτός χρησιμοποιείται για την δημιουργία DataFrame με τα στοιχεία αυτά και με το DataFrame δημιουργείται ένα αρχείο csv με τα δεδομένα μας και αποθηκεύεται στον φάκελο μας.

KD TREE

Τα βασικά σημεία λειτουργίας ενός KD δένδρου είναι τα εξής:

1. Κατασκευή (Construction): Κατά τη διαδικασία κατασκευής, τα δεδομένα διαιρούνται ανά διάσταση, έτσι ώστε κάθε υποδένδρο να περιέχει σημεία που βρίσκονται κοντά μεταξύ τους σε αυτή τη διάσταση. Η επιλογή της διάστασης για κάθε διαίρεση εναλλάσσεται καθώς προχωράμε κατά βάθος στο δένδρο.
2. Εισαγωγή (Insertion): Κατά την εισαγωγή ενός νέου σημείου, το δένδρο ψάχνει την κατάλληλη θέση για το νέο σημείο, λαμβάνοντας υπόψη τον τρέχοντα άξονα διαίρεσης. Στη συνέχεια, το δένδρο διαιρείται ανά διάσταση, προχωρώντας προς το αριστερό ή το δεξί υποδένδρο ανάλογα με την τιμή του νέου σημείου στον τρέχοντα άξονα.
3. Αναζήτηση (Search): Κατά την αναζήτηση, το δένδρο διατρέχεται αναδρομικά, αρχίζοντας από τη ρίζα. Κάθε φορά που διασχίζεται ένας κόμβος, ελέγχεται αν το σημείο που αναζητούμε βρίσκεται στο αριστερό ή δεξί υποδένδρο του κόμβου. Η αναζήτηση συνεχίζεται ανάλογα με το πώς το σημείο συγκρίνεται με το σημείο του κόμβου.

Σύμφωνα με τα παραπάνω έχουν υλοποιηθεί οι αντίστοιχες συναρτήσεις και κλάσεις στον κώδικα μας:

- Κλάση KDNode: Κλάση που αντιπροσωπεύει έναν κόμβο στο KD δένδρο. Κάθε κόμβος έχει ένα σημείο (point), δύο παιδιά (αριστερό και δεξί), καθώς και μια διάσταση (dimension), που χρησιμοποιείται για τον διαχωρισμό των δεδομένων.
- Κλάση KDDimensionalTree: Αυτή είναι η κύρια κλάση για το KD δένδρο. Περιλαμβάνει μεθόδους για την κατασκευή του δένδρου (construct), την εισαγωγή ενός σημείου στο δένδρο (insert_point) και την αναζήτηση σημείων εντός μιας περιοχής (search).
- Συνάρτηση create_and_query_kdtree: Αυτή η συνάρτηση είναι υπεύθυνη για τη δημιουργία ενός KD δένδρου από ένα αρχείο CSV και την αναζήτηση σε αυτό, βάσει καθορισμένων κριτηρίων περιοχής. Χρησιμοποιεί τη βιβλιοθήκη pandas για τη φόρτωση δεδομένων από ένα CSV αρχείο.

QUAD TREE

Το Quadtree είναι μια δομή δεδομένων για τον 2D χώρο όπου κάθε κόμβος έχει ακριβώς τέσσερα παιδιά. Για τον τρισδιάστατο χώρο χρησιμοποιείται η αναλογία του Quadtree στον τρισδιάστατο χώρο που ονομάζεται Octree, όπου κάθε κόμβος έχει ακριβώς 8 παιδιά. Και στις δύο παραπάνω μορφές του Quadtree, κάθε χώρος αντιστοιχεί σε ένα node. Αρχικά, ο χώρος διασπάται σε 4 (2D) ή 8 (3D) υποχώρους, ανάλογα με το αν βρισκόμαστε σε δισδιάστατο ή τρισδιάστατο χώρο, οι οποίοι αποτελούν παιδιά του αρχικού χώρου. Οι υπόχωροι με τη σειρά τους διασπώνται σε 4 ή 8 υποχώρους αν περιέχουν παραπάνω από έναν αριθμό point και η διαδικασία συνεχίζεται μέχρι να μην χρειάζονται άλλες διασπάσεις. Κάθε υπόχωρος έχει μέγεθος ίσο με το ¼ ή το ⅛ του πατέρα του. Στον κώδικα μας βρισκόμαστε στον τρισδιάστατο χώρο, επομένως υλοποιούμε ένα Octree, δηλαδή την μορφή του Quadtree στον 3D χώρο. Πιο συγκεκριμένα:

Κλάση DataPoint*:** Περιέχει τη συνάρτηση ***_init_(self,x,y,z,data) και αναπαριστά ένα σημείο στον τρισδιάστατο χώρο.

Κλάση DataSpace: Η κλάση αναπαριστά έναν τρισδιάστατο χώρο και περιέχει τις εξής συναρτήσεις. Η ***_init_(self, cx, cy, cz, w, h, d)*** αρχικοποιεί τις μεταβλητές του χώρου και υπολογίζει τα edges του χώρου. Η ***contains(self, point)*** ελέγχει αν ένα σημείου βρίσκεται μέσα στον χώρο και επιστρέφει True αν περιέχεται και False αν όχι. Η ***intersects(self,other)*** ελέγχει αν δύο χώροι διασταυρώνονται και επιστρέφει True αν διασταυρώνονται και False αν όχι.

Κλάση DataQuadTree: Είναι η κύρια κλάση του δέντρου και περιέχει τις εξής συναρτήσεις:

Η ***_init_(self, space, max_points=8, depth=0)*** αρχικοποιεί το Quadtree. Η ***divide(self)*** διασπάει τον χώρο (node) στον οποίο καλείται σε 8 υποχώρους με ίσο μέγεθος ο καθένας. Η ***insert(self,point)*** εισάγει ένα point στο δέντρο κάνοντας τους απαραίτητους ελέγχους και διασπάσεις αν χρειαστούν. Η ***query(self, space, found_points)*** αναζητάει και επιστρέφει τα points του Quadtree που βρίσκονται στον χώρο που δίνεται ως όρισμα.

Επίσης έχουμε και τις εξής συναρτήσεις για την λειτουργία του δέντρου:
read_data(): Η συνάρτηση διαβάζει τα δεδομένα από το αρχείο, τα μετατρέπει σε points στο χώρο και τα επιστρέφει.

build_data_quad_tree(): Χρησιμοποιώντας την παραπάνω συνάρτηση διαβάζει τα δεδομένα από το αρχείο csv , δημιουργεί ένα Quadtree με τον κατάλληλο χώρο και εισάγει τα δεδομένα σε αυτό. Τέλος επιστρέφει το δέντρο που δημιουργήθηκε.

test_data_quad_tree(quad_tree,min_l,max_l,min_aw,_min_dblp,_max_dblp):
Παίρνει ως όρισμα ένα Quadtree, το μικρότερο γράμμα, το μεγαλύτερο γράμμα, τον ελάχιστο αριθμό βραβείων, τον ελάχιστο αριθμό dblp_records και τον μέγιστο αριθμό dblp_records και τυπώνει τους επιστήμονες που τα στοιχεία τους πληρούν τις προϋποθέσεις από τα ορίσματα της συνάρτησης.

R TREE

Τα βασικά χαρακτηριστικά του R-Tree είναι τα εξής:

- Κόμβοι (Nodes): Οι κόμβοι του R-tree μπορεί να είναι είτε φύλλα (leaf nodes) που περιέχουν τα πραγματικά δεδομένα, είτε εσωτερικοί κόμβοι που αντιπροσωπεύουν περιοχές που καλύπτουν τα φύλλα τους.
- Ενδιάμεσοι Κόμβοι (Intermediate Nodes): Οι ενδιάμεσοι κόμβοι περιέχουν πληροφορίες σχετικά με το πεδίο κάλυψης (bounding box) των υποκόμβων τους.
- Bounding Box: Κάθε κόμβος (είτε φύλλο είτε ενδιάμεσος) έχει ένα πεδίο κάλυψης (bounding box), το οποίο είναι ένα περιοριστικό ορθογώνιο που περιλαμβάνει όλα τα δεδομένα που βρίσκονται κάτω από αυτόν τον κόμβο.
- Εισαγωγή Δεδομένων: Κατά την εισαγωγή νέων δεδομένων, το R-tree διασπά τον χώρο σε περιοχές με βάση τον χωρικό χαρακτήρα των δεδομένων και προσπαθεί να τα ομαδοποιήσει σε τρόπο που να ελαχιστοποιεί το συνολικό εμβαδόν των bounding boxes.
- Αναζήτηση (Query): Κατά τη διάρκεια μιας αναζήτησης, το R-tree μειώνει τον χώρο αναζήτησης αποκλείοντας τους κόμβους που δεν τέμνουν την περιοχή αναζήτησης. Αυτό γίνεται με τη χρήση των πεδίων κάλυψης.

Αντίστοιχα έχουμε χρησιμοποιήσει την έτοιμη βιβλιοθήκη για υλοποίηση R-Trees και στην συνέχεια έχουμε δημιουργήσει κάποιες συναρτήσεις για την χρήση της:

- **init(self, csv_file):** Αρχικοποιεί ένα αντικείμενο της κλάσης RTreeIndexer. Διαβάζει ένα CSV αρχείο με επιστήμονες, δημιουργεί τα properties ενός R-tree για τρισδιάστατη αναζήτηση και καλεί τη συνάρτηση _build_index() για να δημιουργήσει το R-tree.
- **_build_index(self):** Κατασκευάζει το R-tree, εισάγοντας κάθε επιστήμονα στο δέντρο. Επίσης, αποθηκεύει τα δεδομένα του κάθε επιστήμονα σε μια λίστα για μελλοντική χρήση.
- **query(self, left_letter, right_letter, awards_min, dblp_min, dblp_max):** Κάνει μια αναζήτηση στο R-tree χρησιμοποιώντας το διάστημα κατά την αναζήτηση (left_letter, right_letter) και ελάχιστες/μέγιστες τιμές για τα βραβεία (awards) και τα αριθμός εγγραφών στη βάση δεδομένων DBLP (dblp_records). Επιστρέφει μια δομημένη λίστα αποτελεσμάτων που περιέχει τα δεδομένα των επιστημόνων που πληρούν τα κριτήρια.

RANGE TREE

Ένα Range Tree διαχωρίζει τις διαστάσεις ενός συνόλου σημείων στον χώρο, δημιουργώντας ιεραρχικές δομές δεδομένων. Συγκεκριμένα, ένα 2D Range Tree διαχωρίζει τον χώρο στον άξονα των x και για κάθε σημείο στον άξονα των x, δημιουργεί ένα δεύτερο Range Tree στον άξονα των y.

Για παράδειγμα, ένα 2D Range Tree θα διαχωρίσει αρχικά τα σημεία στον άξονα των x και, για κάθε επίπεδο, θα δημιουργήσει ένα 1D Range Tree στον άξονα των y. Το ίδιο συμβαίνει με το 3D Range Tree, όπου δημιουργούνται πρώτα Range Trees για τον άξονα των x, στη συνέχεια για τον άξονα των y, και τέλος για τον άξονα των z.

Κωδικας:

Ο κώδικας μας υλοποιεί Range Trees για 1D, 2D και 3D διαστάσεις (Τα 1D και 2D απαιτούνται για την δημιουργία 3D). Ο κώδικας έχει τις εξής κλάσεις και συναρτήσεις:

1. Κλάση Node: Ορίζει έναν κόμβο του δέντρου με συγκεκριμένες διαστάσεις (dimension) και ανάλογα πεδία (y, y_tree, z, xy_tree) για τις συντεταγμένες.
2. Κλάσεις RangeTree1D, RangeTree2D, RangeTree3D: Υλοποιούν τα Range Trees για 1D, 2D και 3D διαστάσεις αντίστοιχα και περιλαμβάνουν συναρτήσεις για την εισαγωγή κόμβων, τον υπολογισμό του ύψους και την εκτέλεση ερωτημάτων στο Range Tree.
3. Συναρτήσεις εισαγωγής (insert) και περιστροφής (rotateRight, rotateLeft): Χρησιμοποιούνται για την ενημέρωση του δέντρου ώστε να διατηρείται η ισορροπία.
4. Συναρτήσεις query: Εκτελούν ερωτήματα στο Range Tree για την εύρεση σημείων εντός κάποιου εύρους.
5. Συναρτήσεις constructTree: Δημιουργούν το Range Tree από μια λίστα σημείων.

LSH

Λίγα λόγια για το LSH:

Το Locality sensitive hashing (LSH) είναι μια ευρέως δημοφιλής τεχνική που χρησιμοποιείται στην αναζήτηση κατά προσέγγιση πλησιέστερου γείτονα. Το LSH είναι μία από τις πρωτότυπες τεχνικές για την παραγωγή υψηλής ποιότητας αναζήτησης, διατηρώντας ταυτόχρονα αστραπιαίες ταχύτητες αναζήτησης. Παράγει στην καλύτερη περίπτωση πολυπλοκότητα O(n²). Πρέπει επίσης να ληφθεί υπόψη η πολυπλοκότητα πίσω από έναν μόνο υπολογισμό ομοιότητας - κάθε δείγμα αποθηκεύεται ως διάνυσμα, συχνά διανύσματα πολύ υψηλών διαστάσεων - αυτό αυξάνει ακόμη περισσότερο την πολυπλοκότητα. Η λύση είναι η κατά προσέγγιση αναζήτηση. Αντί να συγκρίνουμε κάθε διάνυσμα (εξαντλητική αναζήτηση) — μπορούμε να προσεγγίσουμε και να περιορίσουμε το εύρος αναζήτησής μόνο στα πιο σχετικά διανύσματα. Ο LSH είναι ένας αλγόριθμος που μας παρέχει αυτούς τους υπογραμμικούς (sub-linear) χρόνους αναζήτησης.

Συνάρτηση shingle():

Η συνάρτηση παίρνει σαν όρισμα ένα *string* (*text*) και μία παράμετρο *k* και επιστρέφει ένα set από shingle (*shingle_set*). Επιλέγονται k συνεχόμενες λέξεις που ξεκινάνε από το index i και καθώς συνενώνονται με " ", γίνονται append στο shingle_set.

Συνάρτηση one_hot_encoding():

Η συνάρτηση παίρνει σαν όρισμα μια λίστα από μοναδικές (unique) λέξεις (*vocab*) και μια λίστα με λέξεις (*data*) και επιστρέφει μια λίστα με 0 και 1. Ειδικότερα, για κάθε x που ανήκει στη vocab list, αν ανήκει και στο data list, τότε το x παίρνει την τιμή 1, αλλιώς παίρνει την τιμή 0.

Συνάρτηση create_hash_func():

Αυτή η συνάρτηση, δημιουργεί μια λίστα με τυχαία hash values ενός συγκεκριμένου μεγέθους (size). Αυτό το size, είναι το μοναδικό όρισμα της συνάρτησης. Δημιουργείται μια λίστα που περιέχει αριθμούς από το 1 έως το size+1. Στη συνέχεια, μέσω της shuffle, η λίστα με τις προαναφερθείσες τιμές ανακατεύεται με απώτερο σκοπό να μειωθούν όσο περισσότερο γίνεται οι συγκρούσεις.

Συνάρτηση build_minhash_func():

Χάρη σε αυτή τη συνάρτηση, κατασκευάζονται πολλαπλά MinHash Vectors, όπου το καθένα αποτελείται από μια λίστα με τυχαίες hash τιμές. Παίρνει σαν όρισμα το *μέγεθος του vocabulary*, *nbits* και τον αριθμό των hashes (*num_hashes*) που προσδιορίζει πόσα MinHash Vectors θα δημιουργηθούν. Τέλος, η συνάρτηση επιστρέφει τη λίστα με τα MinHash Vectors, όπου κάθε Vector εκπροσωπεί μια λίστα από τυχαίες hash τιμές.

Συνάρτηση create_hash():

Αυτή η συνάρτηση παίρνει σαν ορίσματα ένα *binary vector* και μια *MinHash συνάρτηση* και επιστρέφει μια υπογραφή (signature) για το δοθέν Vector με βάση την MinHash συνάρτηση. Ειδικότερα, αυτή η συνάρτηση, δημιουργεί μια υπογραφή MinHash για ένα binary vector, βρίσκοντας σε ποιο σημείο το διάνυσμα ταιριάζει με

τη συνάρτηση κατακερματισμού. Η υπογραφεί που προκύπτει, είναι μια λίστα με δείκτες που υποδεικνύουν τις θέσεις όπου υπάρχει ταίριασμα (match)

Συνάρτηση jaccard_similarity():

Η συνάρτηση παίρνει σαν ορίσματα *δυο set* και συγκρίνει τον αριθμό των κοινών στοιχείων που έχουν.

Συνάρτηση split_vector():

Η συνάρτηση παίρνει σαν ορίσματα την υπογραφή (*signature*) και μια *μεταβλητή b*. Η συνάρτηση έχει σκοπό να διαιρεί την υπογραφή (signature) σε b ίσου μεγέθους subvectors. Αυτό το βήμα είναι κοινό στους αλγορίθμους LSH για κατά προσέγγιση αναζήτηση του πλησιέστερου γείτονα.

MAIN

Η main κάνει import όλες τις συναρτήσεις των δέντρων και του lsh και παρέχει στον χρήστη ένα interface για την εμφάνιση των αντίστοιχων αποτελεσμάτων. Επίσης για την σύγκριση των μεθόδων έχουμε χρησιμοποιήσει την βιβλιοθήκη timeit για την χρονομέτρηση των δέντρων σε συνδυασμό με την lsh καθώς και την βιβλιοθήκη matplotlib.pyplot για την εμφάνιση ενός γραφήματος με τον μέσο όρο των αποτελεσμάτων.

```
Welcome to our program!
=========================
->To insert your values and view results for a tree of your choice combined with the lsh method press 1
->To compare the methods with random inputs press 2
Your choice:
```

Ο χρήστης έχει δύο επιλογές όταν τρέχει την main:

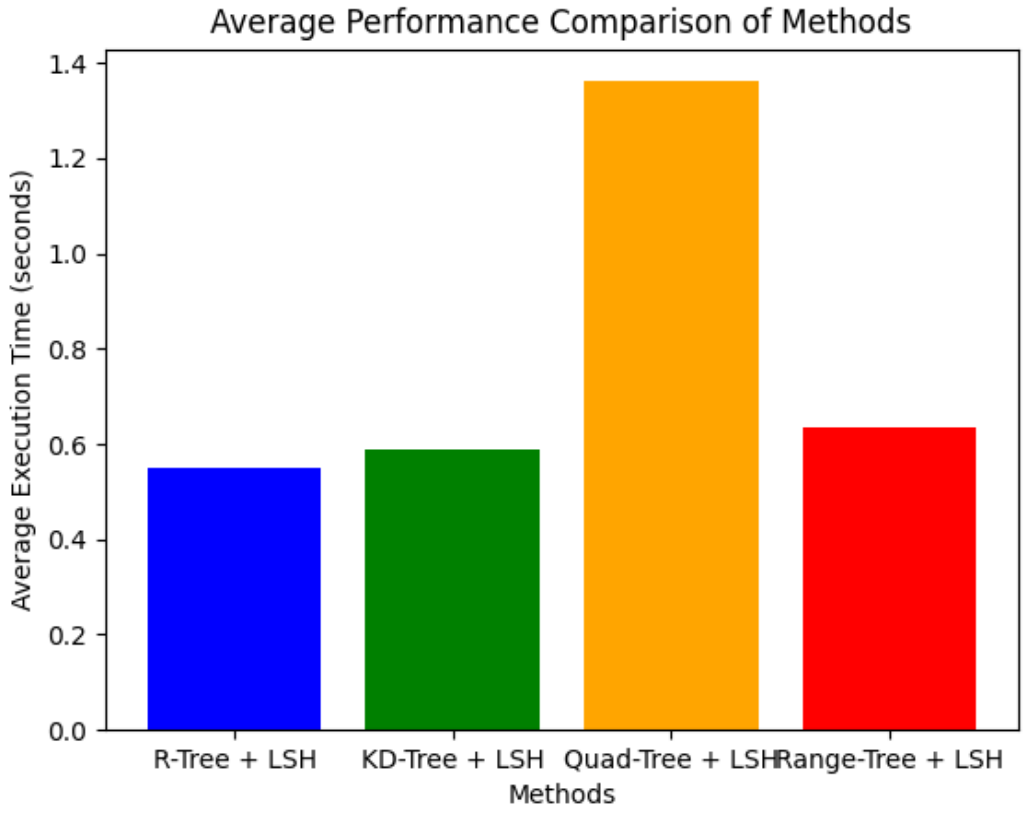
- Χειροκίνητη επιλογή εύρους τιμών και επιλογή δέντρου που θέλει να χρησιμοποιήσει και τέλος εμφάνιση αντίστοιχων αποτελεσμάτων:

```
Welcome to our program!
=========================
->To insert your values and view results for a tree of your choice combined with the lsh method press 1
->To compare the methods with random inputs press 2
Your choice: 1
Εισάγετε ελάχιστο ποσοστό ομοιότητας των πεδίων education(0 - 1): 0.12
Εισάγετε το εύρος τον ονομάτων στη μορφή X,X: a,a
Εισάγετε τον ελάχιστο αριθμό βραβείων: 2
Εισάγετε τον ελάχιστο αριθμό των dblp_records: 0
Εισάγεται τον μέγιστο αριθμό dblp_records: 200
==============================
->For r_tree + LSH press 1
->For kd_tree + LSH press 2
->For quad_tree + LSH press 3
->For range_tree + LSH press 4
Your choice: 2
KD-Tree Data Results =  5

Jaccard Similarity between 1 and 2 with similarity 12.02%

Education 1:

b'Luis von Ahn was born in and grew up in Guatemala City. Von Ahn grew up in a wealthy household with bot
eat privilege.[10][11] When von Ahn was eight years old, his mother bought him a Commodore 64 computer, b
hn began studying at Duke University, where he received a Bachelor of Science (BS) in Mathematics, summa

Education 2:

b'Allen grew up on a farm in Peru, New York, near Lake Champlain, as the oldest of six children. Her fath
e degree in mathematics in 1954 and began teaching school in Peru, New York.[10] After two years, she enr


Additional information:

Info 1: Name: Ahn, Awards: 3, DBLP Records: 39

Info 2: Name: Allen, Awards: 6, DBLP Records: 21

================================================================================
```

- Τυχαία επιλογή εύρους τιμών από το πρόγραμμα 10 φορές και έπειτα για κάθε επιλογή τιμών τρέχει με τις ίδιες εισόδους 10 φορές στο κάθε δέντρο ώστε να πάρουμε όσο το δυνατόν πιο ακριβή αποτελέσματα. Τέλος παρουσιάζεται το γράφημα με τους μέσους χρόνους εκτέλεσης για το κάθε δέντρο.

RESULTS:

Τα αποτελέσματα που προέκυψαν από την εισαγωγή τυχαίων τιμών(10 διαφορετικά είσοδοι και 10 φορές εκτέλεση της κάθε εισόδου από το κάθε δέντρο) από το πρόγραμμα είναι τα εξής:



Παρατηρούμε ότι:

- Η πιο αργή μέθοδος είναι η Quadtree+LSH κάτι το οποίο μπορεί να οφείλεται σε πολλούς παράγοντες όπως η κατανομή των δεδομένων, το πλήθος των αποτελεσμάτων και η υλοποίηση του ίδιου του Quadtree
- Οι υπόλοιπες έχουν παρόμοιο χρόνο εκτέλεσης με την πιο γρήγορη από τις μεθόδους να είναι η R-Tree+LSH κάτι το οποίο μπορεί να οφείλεται και στην χρήση έτοιμης βιβλιοθήκης για την υλοποίηση του R-Tree.
- Γενικότερα τα αποτελέσματα μας εξαρτώνται από πολλούς παράγοντες και ίσως σε διαφορετικές εισόδους ή τύπους δεδομένων να βλέπαμε εντελώς διαφορετικούς χρόνους εκτέλεσης.

Ένας από τους λόγους που το Quadtree είναι αργό είναι ότι έχει πολλά κενά nodes και η κύρια χρήση του είναι image processing και image representation.