1059633 Νικηφόρος - Γιώργος Παπαγεωργίου
1041815 Εμμανουήλ Μηναδάκης
Αλέξανδρος Ξιάρχος 1059619

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΟΛΥΔΙΑΣΤΑΤΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ & ΥΠΟΛΟΓΙΣΤΙΚΗ ΓΕΩΜΕΤΡΙΑ

PROJECT 1 · 2022-2023

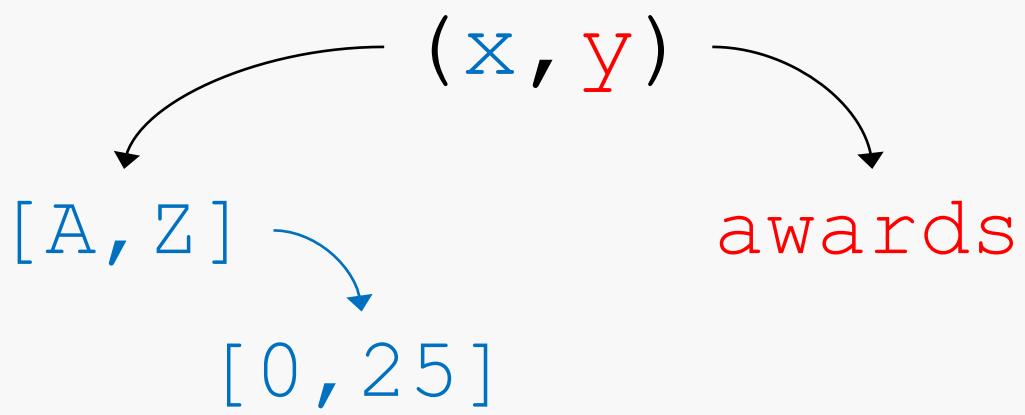
1 ΚΑΤΑΣΚΕΥΗ DATASET

- Κατασκευή δικού μας dataset χρησιμοποιώντας την βιβλιοθήκη `BeautifulSoup` για ανάλυση του HTML περιεχομένου της σελίδας με τη λίστα επιστημόνων της επιστήμης των υπολογιστών και εξαγωγή όλων των επιστημόνων.
- Από τα URLs των επιστημόνων συλλέγονται οι ζητούμενες πληροφορίες (επώνυμο, βραβεία και εκπαίδευση) χρησιμοποιώντας regex expressions, HTML tag parsing και τεχνικές string manipulation. Τα δεδομένα εισάγονται σε ένα `DataFrame`.
- Λόγω της ιδιομορφίας κάθε σελίδας χρησιμοποιήθηκε το διορθωτικό αρχείο `corrections.txt` για τις περιπτώσεις που δεν μπορούσαν να εξαχθούν αυτοματοποιημένα.
- Τα τελικά δεδομένα των 254 επιστημόνων εξάγονται στο αρχείο `scientists_data.csv`.

	surname	awards	education
0	Khan	10	Khan was a Bright Sparks scholar and received ...
1	Aaronson	4	Aaronson grew up in the United States, though ...
2	Abebe	3	Abebe was born and raised in Addis Ababa, Ethi...
3	Abelson	1	Abelson graduated with a Bachelor of Arts degr...
4	Abiteboul	4	The son of two hardware store owners, Abitebou...

2 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ ΚΑΙ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΠΟΛΥΔΙΑΣΤΑΤΗ ΜΟΡΦΗ

- Για κάθε δομή υλοποιήθηκαν ξεχωριστά μια συνάρτηση κατασκευής της δομής (`build_tree()`) και μια συνάρτηση αναζήτησης στη δομή (`query_tree()`):
- Η `build_tree()` αφού διαβάσει το `.csv`, δημιουργεί ένα αντιπροσωπευτικό σημείο (`x`, `y`) για κάθε επιστήμονα. Ως x ορίζεται η αριθμητική τιμή του αρχικού του επωνύμου του επιστήμονα, και ως y ο αριθμός των βραβείων που έχει λάβει.
- Σε κάποιες δομές χρησιμοποιήθηκε και ο αριθμός γραμμής (`index`) του επιστήμονα του `DataFrame` για να ξεχωρίσουμε τους επιστήμονες με ίδια (`x`, `y`).
- Με την δημιουργία των σημείων κατασκευάζουμε την κάθε δομή βάσει αυτών.



2 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ ΚΑΙ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΠΟΛΥΔΙΑΣΤΑΤΗ ΜΟΡΦΗ

- Η συνάρτηση αναζήτησης δέχεται τέσσερις παραμέτρους:

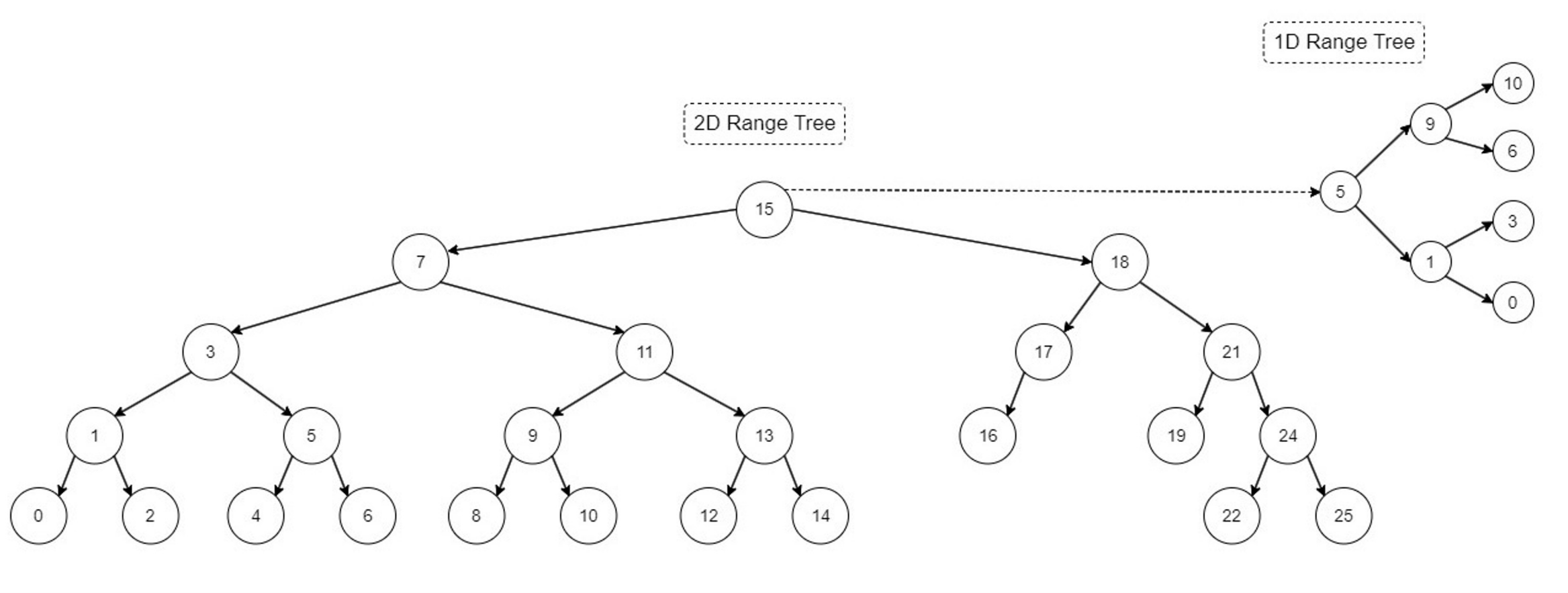
```
query_tree(tree, min_letter, max_letter, num_awards)
```

- `tree`: η πολυδιάστατη δομή που επιστράφηκε από τη συνάρτηση `build_tree()`, δύο γράμματα που αντιπροσωπεύουν το ελάχιστο και το μέγιστο όριο της συντεταγμένης `x` και έναν αριθμό βραβείων που αντιπροσωπεύει το ελάχιστο όριο της `y`.
- Η συνάρτηση αποστέλλει ερώτημα αναζήτησης στη δομή για τα δοθέντα διαστήματα τιμών. Βάσει των αποτελεσμάτων της αναζήτησης ανακτά τα δεδομένα των επιστημόνων από το `.csv` αρχείο και το επιστρέφει στη λίστα `final_results`.

2.1 RANGE TREE

- Η υλοποίηση του 2D **Range Tree** πραγματοποιήθηκε με την κατασκευή ισοσταθμισμένων δυαδικών δέντρων αναζήτησης (BBSTs).
- Αρχικά κατασκευάζεται ένα κύριο BBST βάσει των συντεταγμένων x των σημείων. Κάθε κόμβος του αποθηκεύει ένα 1D Range Tree (y-tree) με όλα τα σημεία με ίδιο x με τον κόμβο. Κάθε y-tree είναι και αυτό BBST, κατασκευασμένο βάσει των συντεταγμένων y των σημείων που περιέχει.
- Έτσι επιτρέπεται η αναζήτηση σημείων χρησιμοποιώντας το y για σημεία που έχουν ίδιο x.
- Κατά την αναζήτηση ενός εύρους, το κύριο δέντρο προσπελαύνεται πρώτα για να βρεθούν οι κόμβοι με τα x που ανήκουν στο επιθυμητό διάστημα x. Για αυτούς τους κόμβους προσπελαύνεται το αντίστοιχο y-δέντρο τους για τα σημεία y.
- Η συνδυασμένη προσπέλαση των δύο δέντρων επιτρέπει την αποτελεσματική εύρεση όλων των σημείων που βρίσκονται εντός ενός ερωτήματος αναζήτησης (range query).

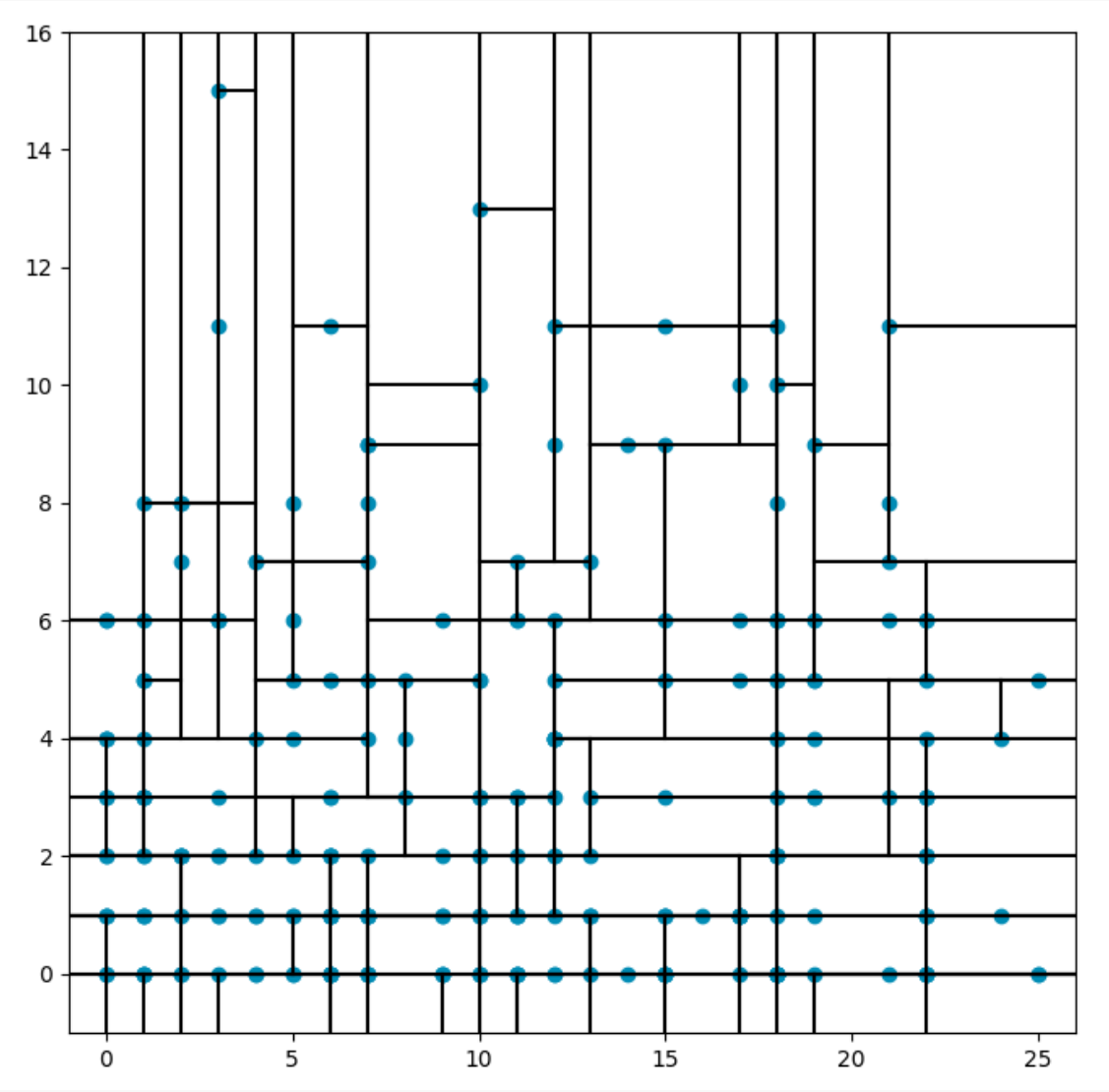
2.1 RANGE TREE



2.2 K-D TREE

- Βρισκόμαστε στον δισδιάστατο χώρο, άρα $K=2$. Τα **k-d Trees** διαχωρίζουν αυτόν τον χώρο σε ημιεπίπεδα. Η κατασκευή του K-D Tree συνεπάγει την διχοτόμηση αυτού του χώρου σε ημιεπίπεδα από δύο άξονες, τον x και τον y.
- Επιλέγεται ένας αρχικός άξονας και ένα σημείο (x, y) που θα τον τέμνει. Τα υπόλοιπα σημεία θα ανήκουν πλέον σε δύο υποσύνολα, αριστερά και δεξιά του, ανάλογα με τις συντεταγμένες τους. Όσο προστίθενται σημεία, ανάλογα με το βάθος του δέντρου, οι διαχωρισμοί θα εναλλάσσονται διαδοχικά σε κάθετους και οριζόντιους, οι οποίοι αντιστοιχίζονται στον x και στον y άξονα.
- Το αποτέλεσμα της κατασκευής είναι ένα ισοσταθμισμένο δυαδικό δέντρο με κάθε κόμβο του να έχει έναν "προσωπικό" άξονα ο οποίος χωρίζει το χώρο σε όλο και μικρότερα ημιεπίπεδα. Για την αναζήτηση των σημείων οδηγούμαστε όλο και βαθύτερα στο δέντρο, αναζητώντας μόνο τα σημεία που βρίσκονται μέσα στο ορθογώνιο διάστημα που επιθυμούμε.

2.2 K-D TREE



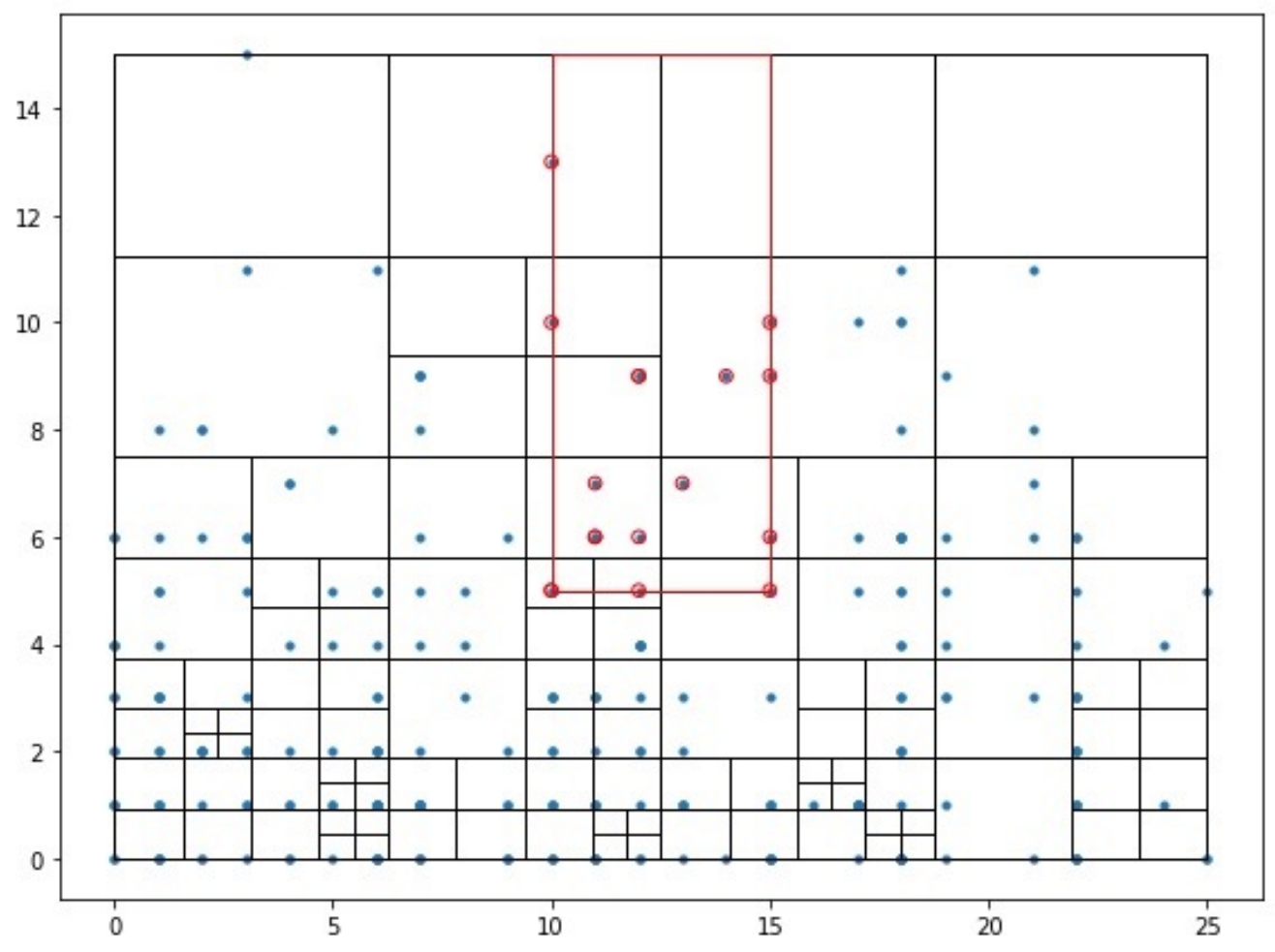
2.3 R-TREE

- Η βασική ιδέα του **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.
- Ο κώδικας που υλοποιήσαμε περιλαμβάνει την κλάση `Rtree`, χρησιμοποιώντας τη βιβλιοθήκη `rtree` της `libspatialindex`. Η μέθοδος `insert()` προσθέτει ένα στοιχείο στο δέντρο ψάχνοντας τον πιο κατάλληλο κόμβο για την εισαγωγή του. Αν αυτός ο κόμβος υπερβαίνει το μέγιστο πλήθος στοιχείων, διαιρείται σε δύο νέους κόμβους. Χρησιμοποιούνται αλγόριθμοι που βελτιστοποιούν την διαίρεση ελαχιστοποιώντας την υπερκάλυψη τους ώστε το δέντρο να παραμένει ισορροπημένο.
- Η μέθοδος `search()` χρησιμοποιεί ένα δοθέν bounding box ως όρισμα και καλεί την `intersection()` η οποία βρίσκει όλα τα στοιχεία που τέμνονται με αυτό το bounding box.

2.4 QUAD TREE

- Το **Quad Tree** χωρίζει το χώρο σε τέσσερα τμήματα (ή κόμβους) και κάθε τμήμα μπορεί να χωριστεί περαιτέρω ανάλογα με το πλήθος των σημείων που περιέχει.
- 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 ορίζουμε ως 4), ο κόμβος διασπάται και το σημείο προστίθεται στο 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. Αν ζητηθούν να βρεθούν όλα τα σημεία εντός ενός ορθογωνίου, το δέντρο ελέγχει μόνο τους κόμβους που τέμνουν το ορθογώνιο, αγνοώντας όλους τους υπόλοιπους.

2.4 QUAD TREE



Με κόκκινο χρώμα είναι σημειωμένο το ορθογώνιο αναζήτησης
(search boundary) και τα σημεία που περιέχονται σε αυτό.

3 ΥΛΟΠΟΙΗΣΗ LOCALITY SENSITIVITY HASHING (LSH)

- Ο αλγόριθμος **Locality-Sensitive Hashing** (LSH) είναι ένας αλγόριθμος που χρησιμοποιείται στον τομέα της αναζήτησης και της αναγνώρισης πλησιέστερων γειτόνων σε δεδομένα σε 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.
- **Συνάρτηση Hashing**: Ο LSH χρησιμοποιεί μια συνάρτηση κατακερματισμού (hash function) για να μετατρέψει τα δεδομένα από τον αρχικό χώρο σε έναν χώρο μικρότερων διαστάσεων.
- **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.
- **Αναζήτηση Γειτόνων**: Αφού τα δεδομένα αντιστοιχήθηκαν σε κάδους, μπορούμε να αναζητήσουμε γρήγορα πλησιέστερους γείτονες, εξετάζοντας μόνο τους κάδους που περιέχουν τα δεδομένα που μας ενδιαφέρουν.
- Ο αλγόριθμος υλοποιείται από τις κλάσεις `MinHash()` και `LSH()` στο αρχείο `lsh/lsh.py`.

3.1 ΜΕΤΡΙΚΕΣ ΟΜΟΙΟΤΗΤΑΣ

- Υπάρχουν αρκετές μέθοδοι για την σύγκριση των ομοιοτήτων, όπως η μετρική cosine που χρησιμοποιεί vectors για την εύρεση των ομοιοτήτων. Η φυσιολογία των κειμένων που έχουμε μας οδηγεί στη χρήση της μετρικής Jaccard:
- Η μετρική ομοιότητας **Jaccard** είναι ένα μέτρο που χρησιμοποιείται για να αξιολογήσει τον βαθμό της ομοιότητας ανάμεσα σε δύο σύνολα δεδομένων. Το πλεονέκτημα της στην σύγκριση που θέλουμε να πραγματοποιήσουμε 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()`.
- Η **Κωδικοποίηση One-Hot** (One-Hot Encoding) είναι μια τεχνική που χρησιμοποιείται στη μηχανική μάθηση και την επεξεργασία φυσικής γλώσσας για τη μετατροπή κατηγορικών δεδομένων, όπως κατηγορικές μεταβλητές ή ετικέτες, σε δυαδική (0 ή 1) αριθμητική μορφή.
- Στην παρούσα εργασία αναπαριστούμε το σύνολο των δεδομένων, δηλαδή το πεδίο "education", σε ένα One-Hot μητρώο το οποίο και εισάγουμε στον αλγόριθμο LSH. Αυτό υλοποιείται μέσω της συνάρτησης `one_hot_encoding()` στο αρχείο `lsh/tools.py`.

3.2 ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΚΕΙΜΕΝΟΥ

- Ο στόχος της προεπεξεργασίας κειμένου είναι να καθαρίσει τα δεδομένα κειμένου, μετατρέποντάς τα σε μορφή κατάλληλη για ανάλυση και εύκολη επεξεργασία.
- Έχουμε υλοποιήσει τη συνάρτηση `stemming_and_stopwords()` για να επεξεργαζόμαστε τα κείμενα εκπαίδευσης των επιστημόνων που θα εισάγουμε στο LSH. Εκεί υλοποιοιούνται οι παρακάτω λειτουργίες:
  - **Tokenization**: Σπάσιμο του κειμένου σε μικρότερες λέξεις ή υπολέξεις.
  - **Stopwords Removal**: Κατάργηση των κοινών λέξεων (stopwords) όπως "and", "the", "in" κ.λπ., καθώς συχνά έχουν μικρό νόημα και μπορεί να είναι υπολογιστικά δαπανηρή η επεξεργασία τους.
  - **Noise Removal**: Κατάργηση τυχόν χαρακτήρων και συμβόλων που δεν συνεισφέρουν νόημα στο κείμενο, όπως παρενθέσεις, αριθμούς και HTML tags.
  - **Stemming**: Μείωση των λέξεων στη βάση ή τη ρίζα τους και αφαίρεση επιθημάτων και προθεμάτων για να βρεθεί τη ρίζα της λέξης (π.χ. "running" σε "run").

4 ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΠΑΡΑΔΕΙΓΜΑΤΑ

```
Εισάγετε ελάχιστο ποσοστό ομοιότητας (0 - 1): 0.4
Εισάγετε διάστημα ονομάτων στη μορφή X,X: e,k
Εισάγετε ελάχιστο αριθμό βραβείων: 2

1. k-d tree
2. Quad tree
3. Range tree
4. R-tree
Επιλέξτε δομή: 3
165 candidates with at least 40 % similarity using jaccard_binary:

1. Similarity: 40.0%
+----------+--------+-----------------------------------------------------------------------------------------------------------------------------------+
| Surname  | Awards |                                                                 Education                                                         |
+----------+--------+-----------------------------------------------------------------------------------------------------------------------------------+
| Geschke  |   3    | Charles Matthew Geschke[4] was born in Cleveland, Ohio, on September 11, 1939.[5]  He attended Saint Ignatius High School.[6]  |
|          |        | Geschke earned an AB in classics in 1962 and an MS in mathematics in 1963, both from Xavier University.[5]  He taught mathemat |
|          |        | ics at John Carroll University from 1963 to 1968.[7]  In 1972, he completed his PhD studies in computer science at Carnegie Me |
|          |        | llon University under the advice of William Wulf.[4]  He was a co-author of Wulf's 1975 book The Design of an Optimizing Compi |
|          |        |                                                                 ler.[8]                                                           |
+----------+--------+-----------------------------------------------------------------------------------------------------------------------------------+
| Goldberg |   2    | Goldberg was born in Cleveland, Ohio, on July 22, 1945.  Her parents moved to Chicago, Illinois when she was 11, where she spe |
|          |        | nt the rest of her childhood.[1] She enjoyed problem solving and mathematics from a young age and was encouraged by her teache |
|          |        | rs to pursue mathematics.[1] In 1967, she earned a bachelor's degree in mathematics at the University of Michigan.[2]  Interes |
|          |        | ted in the subject of computing, Goldberg worked as an intern with IBM during the summer of her junior year of college, where  |
```

4 ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΠΑΡΑΔΕΙΓΜΑΤΑ

Input: 0.4 threshold, [K, L], 5 awards
Output: Similarity 57.1%

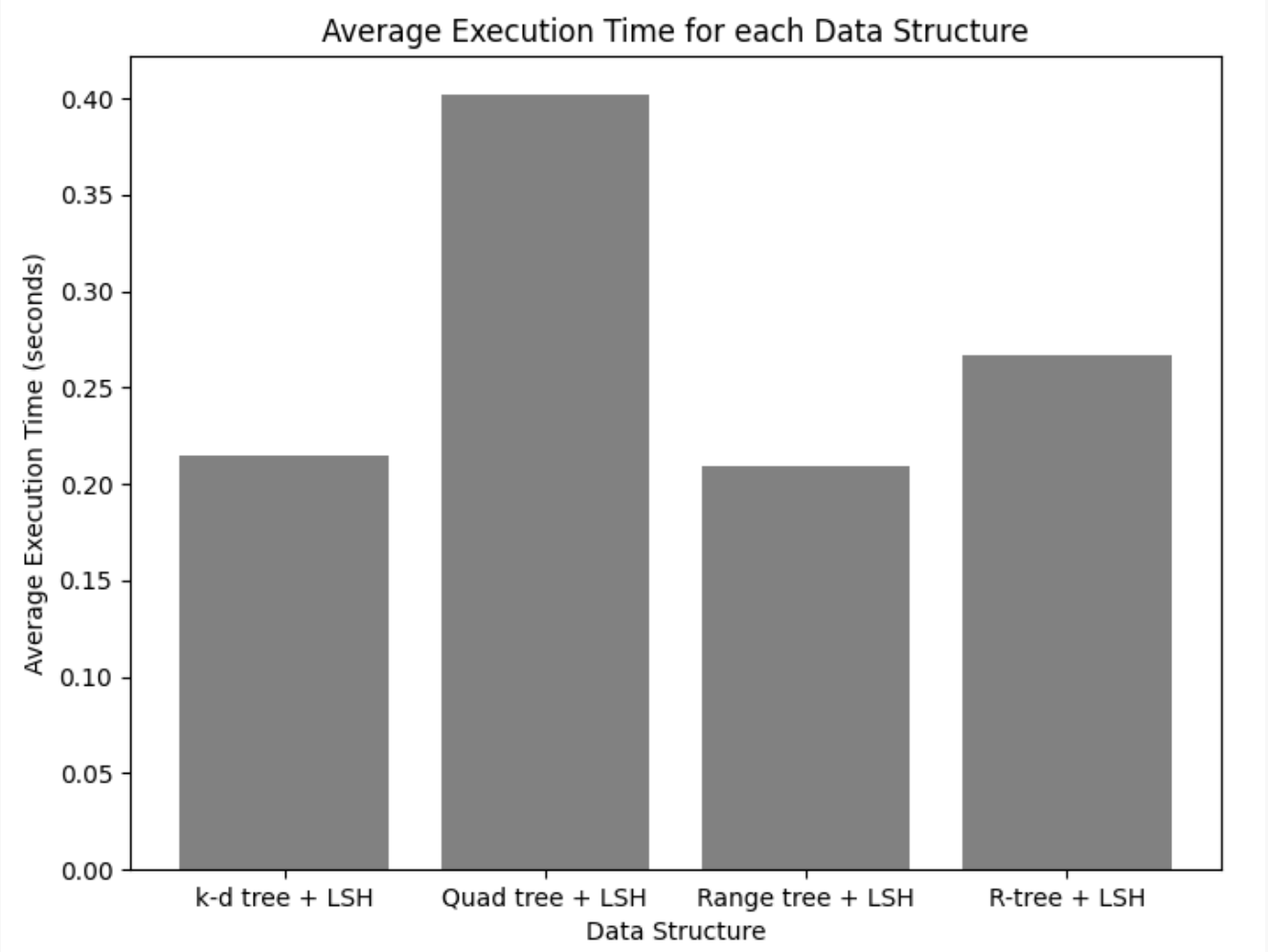
Surname	Awards	Education
Kleinrock	5	Leonard Kleinrock **was born** in **New York** City on June 13, 1934, to a **Jewish family**,[3] and graduated from the noted Bronx **High School of Science** in 1951. He **received** a Bachelor of Electrical Engineering degree in 1957 from the City College of **New York**, and a master's degree and a doctorate (**Ph.D.**) in electrical engineering and computer science from the **Massachusetts Institute of Technology** in 1959 and 1963 respectively. He then joined the faculty at the **University** of California at Los Angeles (UCLA), where he remains to the present day; during 1991–1995 he served as the chairman of the Computer Science Department there.[4]
Lamport	6	Lamport **was born** into a **Jewish family** in Brooklyn, **New York**, the son of Benjamin and Hannah Lamport (née Lasser).[citation needed] His father was an immigrant from Volkovisk in the Russian Empire (now Vawkavysk, Belarus)[10] and his mother was an immigrant from the Austro-Hungarian Empire, now southeastern Poland. A graduate of Bronx **High School of Science**, Lamport **received** a B.S. in mathematics from the **Massachusetts Institute of Technology** in 1960, followed by M.A. (1963) and **Ph.D.** (1972) degrees in mathematics from Brandeis **University**.[11] His dissertation, The analytic Cauchy problem with singular data, is about singularities in analytic partial differential equations.[12]

4 ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΠΑΡΑΔΕΙΓΜΑΤΑ

Input: 0.6 threshold, [M, X], 2 awards
Output: Similarity: 60.0%

Surname	Awards	Education
Muggleton	4	Muggleton **received** his **Bachelor of Science degree** in **computer science** (1982) and **Doctor of Philosophy** in artificial intelligence (1986) **supervised** by Donald Michie at the **University** of Edinburgh.[12]
Wadler	4	Wadler **received** a **Bachelor of Science degree** in mathematics from Stanford University in 1977, and a Master of Science degree in **computer science** from Carnegie Mellon **University** in 1979.[6] He completed his **Doctor of Philosophy** in **computer science** at Carnegie Mellon **University** in 1984. His thesis was entitled "Listlessness is better than laziness" and was **supervised** by Nico Habermann.[7][8]

4.1 ΣΥΓΚΡΙΣΗ ΥΛΟΠΟΙΗΣΕΩΝ



4.1 ΣΥΓΚΡΙΣΗ ΥΛΟΠΟΙΗΣΕΩΝ

Παρατηρούμε τα εξής:

- **Quad tree**: Η δομή αυτή κατανάλωσε τον περισσότερο χρόνο, φτάνοντας τα 0,4 δευτερόλεπτα. Αυτό μπορεί να οφείλεται σε διάφορους παράγοντες, όπως η διανομή των σημείων ή ο τρόπος με τον οποίο έχει υλοποιηθεί το Quad tree.
- **R-tree**: Ακολούθησε με χρόνο λίγο πάνω από 0,25 δευτερόλεπτα. Τα R-trees είναι γενικά πιο πολύπλοκα στην κατασκευή τους, και η απόδοσή τους μπορεί να επηρεαστεί από τη διανομή των σημείων και το μέγεθος των bounding boxes.
- **k-d tree** και **Range tree**: Και οι δύο δομές είχαν παρόμοιο χρόνο, με το k-d tree να καταγράφει περίπου 0,21 δευτερόλεπτα και το Range tree 0,2 δευτερόλεπτα. Αυτό δείχνει ότι οι δύο δομές είναι σχετικά αποτελεσματικές για το συγκεκριμένο σύνολο δεδομένων και για τα ερωτήματα που τέθηκαν.

Ενώ το Quad tree μπορεί να είναι ιδανικό για ορισμένες εφαρμογές, στη συγκεκριμένη περίπτωση φαίνεται να είναι το πιο αργό. Αντίθετα, τα k-d και Range trees παρουσίασαν την καλύτερη απόδοση για το συγκεκριμένο σύνολο δεδομένων.