ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΟΛΥΔΙΑΣΤΑΤΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

PROJECT 1

1 ΚΑΤΑΣΚΕΥΗ DATASET

Κατασκευάσαμε το δικό μας dataset με τα δεδομένα των επιστημόνων της επιστήμης υπολογιστών, τα οποία αντλήσαμε από τη βάση δεδομένων της Wikipedia. Το αρχείο `download_data.py` περιλαμβάνει το script και τη διαδικασία με την οποία αντλήσαμε και επεξεργαστήκαμε τις πληροφορίες αυτές από την Wikipedia. Ο κώδικας στοχεύει στη δημιουργία ενός `.csv` αρχείου, που περιέχει το επώνυμό κάθε επιστήμονα, όπως επίσης τον αριθμό των βραβείων που έχει λάβει, καθώς και πληροφορίες σχετικά με την εκπαίδευσή του.

Αρχικά, με τη συνάρτηση `get_urls()`, επιστρέφονται σε λίστα όλοι οι σύνδεσμοι (URLs) των επιστημόνων που περιέχονται στην κεντρική σελίδα με τη λίστα επιστημόνων της επιστήμης των υπολογιστών. Αυτό επιτυγχάνεται κάνοντας χρήση της βιβλιοθήκης `BeautifulSoup`, με την οποία μπορούμε να αναλύσουμε το HTML περιεχόμενο κάθε σελίδας, καθιστώντας την ανάκτηση και επεξεργασία των πληροφοριών εύκολη και αυτοματοποιημένη.[1]

Στη συνέχεια, καλείται η συνάρτηση `get_scientist_info()` για κάθε URL, όπου χρησιμοποιώντας regex expressions, HTML tag parsing και string manipulation εξάγονται οι ζητούμενες πληροφορίες για καθέναν από τους επιστήμονες της λίστας. Όταν έχουν ανακτηθεί όλα τα δεδομένα, δημιουργείται ένα `DataFrame`, το οποίο περιέχει τις πληροφορίες για κάθε επιστήμονα.

Λόγω της ιδιομορφίας της κάθε σελίδας, χρειάστηκε να δημιουργήσουμε διαφορετικές περιπτώσεις για να καλύψουμε όλα τα πιθανά σημεία όπου βρίσκονται τα awards και το education, αλλά και πάλι υπάρχουν περιπτώσεις που δεν μπορούν να καλυφθούν αυτοματοποιημένα. Για αυτές τις περιπτώσεις χρησιμοποιείται το αρχείο `corrections.txt`, το οποίο περιέχει πληροφορίες που εξάχθηκαν χειροκίνητα για να διορθωθούν αυτές οι αστοχίες στην εξαγωγή των δεδομένων.

Τέλος, τα δεδομένα εξάγονται στο αρχείο `scientists_data.csv`, που είναι και το τελικό dataset που χρησιμοποιήσαμε για την υλοποίηση της εργασίας. Περιέχει δεδομένα 254 επιστημόνων, που είναι αυτοί για τους οποίους υπήρχε κείμενο σχετικά με την εκπαίδευσή τους στην αντίστοιχη σελίδα τους.

	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...

Εικόνα 1: Παράδειγμα των πέντε πρώτων στοιχείων του dataset.

2 ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ ΚΑΙ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΠΟΛΥΔΙΑΣΤΑΤΗ ΜΟΡΦΗ

Προτού αναφερθούμε σε κάθε μία εκ των τεσσάρων πολυδιάστατων δομών που υλοποιήσαμε, αξίζει να σημειώσουμε πως για κάθε δομή υλοποιήθηκαν ξεχωριστά μία συνάρτηση κατασκευής της δομής και μία συνάρτηση αναζήτησης στη δομή.

Η συνάρτηση `build_tree()` είναι υπεύθυνη για την κατασκευή της εκάστοτε δομής και για την εισαγωγή σ' αυτήν όλων των διαθέσιμων σημείων. Αφού διαβάσει το αρχείο `.csv` με τα δεδομένα των επιστημόνων, δημιουργεί ένα αντιπροσωπευτικό σημείο (`x`, `y`) για τον κάθε επιστήμονα όπου `x` η αριθμητική

[1] Συγκεκριμένα για να εξαχθούν τα URLs των επιστημόνων, ψάχνονται, αν υπάρχουν, οι σύνδεσμοι που περιλαμβάνονται στα `<a href>` tags, μέσα στα `<li>` list items των unordered lists `<ul>`.

τιμή του πρώτου γράμματος του επωνύμου του επιστήμονα (['A', 'Z'] → [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 αναζήτησης δέχεται πάντα τέσσερις παραμέτρους:

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

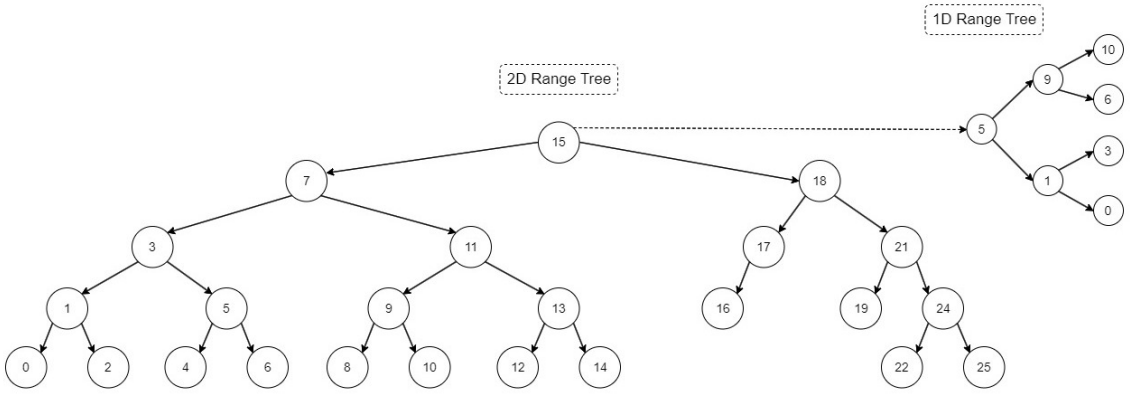
όπου tree, η πολυδιάστατη δομή που επιστράφηκε από τη συνάρτηση build_tree(), δύο γράμματα που αντιπροσωπεύουν το ελάχιστο και το μέγιστο όριο της συντεταγμένης x, και έναν αριθμό βραβείων που αντιπροσωπεύει το ελάχιστο όριο της συντεταγμένης y.

Η συνάρτηση αφού μετατρέπει τα δύο γράμματα στις αντίστοιχες αριθμητικές τους τιμές[2], αποστέλλει ερώτημα αναζήτησης στη δομή για τα δοθέντα διαστήματα τιμών, κάνοντας χρήση των μεθόδων της. Βάσει των αποτελεσμάτων της αναζήτησης, ανακτά τα δεδομένα των επιστημόνων από το .csv αρχείο (χρησιμοποιώντας το index) και τα επιστρέφει ως μία λίστα final_results με τη μορφή λεξικών.

2.1 RANGE TREE

Για την αποτελεσματική διαχείριση των δισδιάστατων σημείων με συντεταγμένες x, y, υλοποιήσαμε ένα 2D Range Tree. Η προσέγγιση που ακολουθήσαμε περιλαμβάνει την κατασκευή του από ισοσταθμισμένα 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 2: 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

[2] Η μετατροπή των γραμμάτων γίνεται χρησιμοποιώντας την συνάρτηση letter_normalization() η οποία μετατρέπει τον χαρακτήρα που εισάγουμε, ασχέτως αν είναι πεζός ή κεφαλαίος σε κάποιον αριθμό μεταξύ του [0-25]. Αυτό επιτυγχάνεται με την συνάρτηση ord() η οποία επιστρέφει τον Unicode κωδικό του γράμματος, τον οποίο κανονικοποιούμε στο διάστημα που θέλουμε.

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 ερωτήματος διαστήματος (range query).

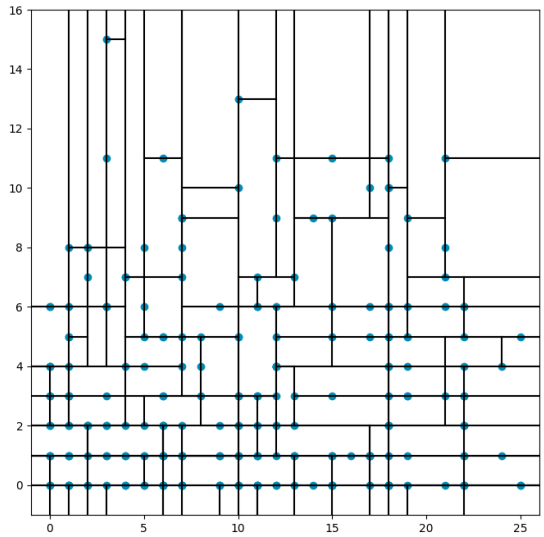
2.2 K-D TREE

Τα K-D Trees χρησιμοποιούνται για την αποτελεσματική αναζήτηση σημείων σε κοντινή απόσταση, κυρίως σε μικρές διαστάσεις. Επιλέγουμε τα δεδομένα να τοποθετούνται στο δισδιάστατο χώρο (συνεπώς `K=2`). Αυτός ο χώρος διαχωρίζεται σε ημιεπίπεδα, δημιουργώντας ένα ισοσταθμισμένο δυαδικό δέντρο. Κάθε κόμβος δημιουργεί και έναν διαχωρισμό των περιοχών. Ανάλογα με το βάθος του δέντρου οι διαχωρισμοί εναλλάσσονται διαδοχικά σε κάθετους και οριζόντιους, οι οποίοι αντιστοιχίζονται στον `x` και στον `y` άξονα.

Στην υλοποίηση του K-D Tree δημιουργούμε την κλάση `Node` που αναπαριστά έναν κόμβο του δέντρου. Αυτές είναι οι βασικές μέθοδοι της κλάσης:

- **`build()`**: δημιουργεί το K-D δέντρο τοποθετώντας τα σημεία τα οποία δημιουργούν εναλλάξ διαχωρισμούς στους άξονες x, y.
- **`insert()`**: εισαγάγει αναδρομικά ένα σημείο στο K-D δέντρο
- **`query()`**: πραγματοποιεί αναζήτηση διαστήματος στο K-D δέντρο. Το διάστημα αναπαρίσταται μέσω της μεταβλητής `rect = x1, y1, x2, y2.`

Το αποτέλεσμα της κατασκευής είναι κάθε κόμβος του δέντρου να αναπαρίσταται από ένα σημείο (`x`, `y`) στο επίπεδο και το δέντρο να έχει χωρίσει 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.



Εικόνα 3: Κατασκευή του K-D Tree χρησιμοποιώντας την Matplotlib. Παρατηρούμε τους κάθετους και οριζόντιους άξονες οι οποίοι δημιουργούνται από κάθε κόμβο και το χωρίζουν σε ημιεπίπεδα.

2.3 R-TREE

Το R-Tree είναι ισοσταθμισμένα δέντρα με κόμβους-δείκτες οι οποίοι συνδέονται με πολυδιάστατα data objects. Η πληροφορία του δέντρου δεν αποθηκεύεται απευθείας στο δέντρο, αλλά αποθηκεύουν ορθογώνιες περιοχές που αντιστοιχούν σε δεδομένα.

Για την υλοποίηση της δομής, με τη βοήθεια της βιβλιοθήκης `rtree`, έχουμε δημιουργήσει την κλάση `Rtree`. Περιλαμβάνει τις μεθόδους `insert()` και `search()`. Η `search()` περιλαμβάνει ως όρισμα ένα ορθογώνιο `query_bbox` το οποίο χωροθετεί την αναζήτηση. Χρησιμοποιεί την μέθοδο `intersection()` για να βρει σημεία που τέμνουν το `query_bbox`.

2.4 QUAD TREE

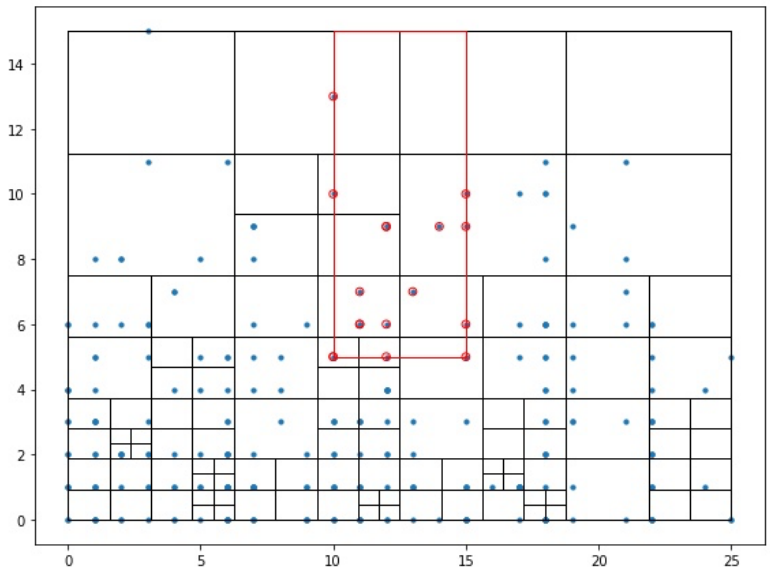
Ένα Quad Tree είναι μία δομή δεδομένων δέντρου που χρησιμοποιείται για την αποτελεσματική οργάνωση και αναζήτηση σημείων σε δισδιάστατους χώρους. Το δέντρο αυτό χωρίζει τον χώρο σε τέσσερα τμήματα (ή κόμβους) και κάθε τμήμα μπορεί να χωριστεί περαιτέρω ανάλογα με το πλήθος των σημείων που περιέχει.

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.



Εικόνα 4: Γραφική απεικόνιση του Quad Tree. Με κόκκινο χρώμα είναι σημειωμένο το ορθογώνιο αναζήτησης (search boundary) και τα σημεία που περιέχονται σε αυτό.