Υλοποίηση Locality Sensitivity Hashing

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 : 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

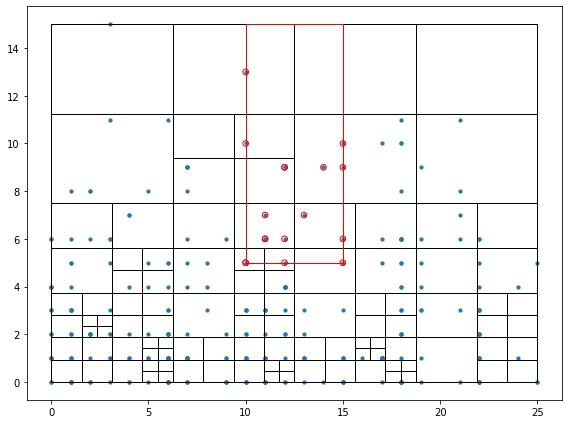
Ένα Quad Tree είναι μία δομή δεδομένων δέντρου που χρησιμοποιείται για την αποτελεσματική οργάνωση και αναζήτηση σημείων σε δισδιάστατους χώρους. Το δέντρο αυτό χωρίζει τον χώρο σε τέσσερα τμήματα (ή κόμβους) και κάθε τμήμα μπορεί να χωριστεί περαιτέρω ανάλογα με το πλήθος των σημείων που περιέχει.

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.



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