Μάθημα: Διαχείριση μη Παραδοσιακών Δεδομένων

Εργασία 2 – Χωρικά Δεδομένα

Ονοματεπώνυμο: Κωνσταντίνος Βαρδάκας

ΑΜ: 522

Email: pcs0522@uoi.gr

# Μέρος 1: Κατασκευή R-Tree μέσω Bulk Loading

Στο πρώτο μέρος της εργασίας υλοποιείται η κατασκευή ενός R-Tree με τη μέθοδο bulk loading. Ο σχετικός κώδικας βρίσκεται στα αρχεία rtree_init_functions.py (βοηθητικές συναρτήσεις) και rtree_init_main.py (κύριο πρόγραμμα). Η διαδικασία ξεκινά με την ανάγνωση των αρχείων που περιλαμβάνουν τα δεδομένα coords.txt και offsets.txt, μέσω της συνάρτησης read_data(coords_path, offsets_path), η οποία επιστρέφει δύο λίστες. Η πρώτη λίστα που επιστρέφεται, η λίστα coords, περιέχει τις συντενταγμένες του σημείου (coords = [[$x_1$, $y_1$], [$x_2$, $y_2$], …]) και η δεύτερη περιλαμβάνει πληροφορίες για την ομαδοποίηση των σημείων προς δημιουργία των χωρικών δεδομένων (offsets = [[$id_1$, $start\_ind_1$, $end\_ind_1$], [$id_2$, $start\_ind_2$, $end\_ind_2$], …]).

Στη συνέχεια, απαιτείται ο υπολογισμός του παραθύρου MBR που περιλαμβάνει κάθε πολύγωνο, ο οποίος γίνεται με την εύρεση της ελάχιστης και μέγιστης τιμής κάθε άξονα (x και y) για κάθε πολύγωνο (συνάρτηση compute_mbr(coords)). Αφού καθοριστεί το MBR, βρίσκεται το κέντρο του, υπολογίζοντας για κάθε άξονα το μέσο όρο των ελάχιστων και μεγιστων τιμών (συνάρτηση mbr_center(mbr)). Το κέντρο αυτό χρησιμοποιείται στη συνέχεια για την εξαγωγή της z-order τιμής κάθε MBR, η οποία βοηθά στην ομαδοποίηση των πολυγώνων κατά την κατασκευή του R-Tree. Αυτός ο υπολογισμός της τιμής Z-order για δεδομένο κέντρο MBR γίνεται χρησιμοποιώντας τη συνάρτηση interleave_latlng(y, x) της βιβλιοθήκης pymorton. Η συνάρτηση build_mbrs(coords, offsets), χρησιμοποιεί τις προηγούμενες συναρτήσεις, και έτσι δημιουργούνται τα δεδομένα, τα πολύγωνα και οι τιμές z values και η λίστα που επιστρέφεται περιλαμβάνει τα δεδομένα ταξινομημένα ως προς το z- order value. Επειδή στα φύλλα που δημιουργούνται στη συνέχεια, οι

εγγραφές πρέπει να είναι του τύπου [id, MBR], στην επιστρεφόμενη λίστα, διαγράφεται το z-order value.

Με σκοπό να είναι έτοιμα τα δεδομένα σε φύλλα, δημιουργήθηκε η συνάρτηση group_entries(entries), η οποία ομαδοποιεί τα δεδομένα σε εικοσάδες, που είναι ο μέγιστος αριθμός που μπορεί να διαθέτει κάθε φύλλο. Αν υπάρχουν περισσευούμενα δεδομένα που είναι παραπάνω από το ελάχιστο επιτρεπτό όριο (min_entries = 8) δημιουργείται μία ακόμη ομάδα. Αν όμως τα περισσευούμενα είναι λιγότερα από αυτό το όριο, η τελευταία πλήρης ομάδα που εισάχθηκε απομακρύνεται, και δημιουργείται πάλι, αφού μετακινηθούν πρώτα αρκετά δεδομένα από αυτή, ώστε να συμπληρωθεί μία ακόμη ομάδα με 8 (ή min_entries) δεδομένα.

Σε αυτό το σημείο τα δεδομένα είναι έτοιμα σε φύλλα, και μπορούν να δημιουργηθούν οι κόμβοι. Αυτή η διαδικασία πραγματοποιείται με τη συνάρτηση build_nodes(entries, node_type). Η ομαδοποίηση σε φύλλα πραγματοποιείται μέσα στη συνάρτηση και στη συνέχεια για κάθε φύλλο υπολογίζεται το MBR το οποίο περιλαμβάνει όλα τα MBR των δεδομένων ενός φύλλου (με τη συνάρτηση compute_union_mbr(mbr_list)). Έτσι, για δεδομένο φύλλο, δημιουργείται ένας κόμβος με MBR, με children το ίδιο το φύλλο, και με type = ‘leaf’ που στη συνέχεια θα ξεχωρίζει το leaf nodes από τα υπόλοιπα. Στη συνέχεια, ανατίθεται και ένα node id, αρχίζοντας από την αρίθμηση του πρώτου φύλλου (φύλλο με τα δεδομένα με το μικρότερο z-order curve). Η ίδια συνάρτηση χρησιμοποιείται για τη δημιουργία των parent nodes, καθώς κάθε κόμβος που δημιουργείται έχει και αυτός ένα MBR.

Πλέον μπορεί να πραγματοποιηθεί η δημιουργία του δέντρου με τη συνάρτηση build_rtree, κατά την οποία αρχικά δημιουργούνται οι κόμβοι των φύλλων όπως αναλύθηκε προηγουμένως και επακόλουθα ανατίθενται ids για κάθε κόμβο. Στη συνέχεια, χτίζονται τα επόμενα επίπεδα του δέντρου (node_type=’node’) έως ότου το επίπεδο που χτιστεί να έχει από 2 έως 20 nodes, τα οποία ομαδοποιούνται κάτω από το root node.

Για την εκτύπωση των κόμβων ανά επίπεδο, χρησιμοποιήθηκε μια ουρά που περιείχε tuples της μορφής (κόμβος, επίπεδο), η οποία αρχικοποιήθηκε με το root node στο επίπεδο 0. Η αρίθμηση των επιπέδων χρησίμευσε αποκλειστικά για την παρακολούθηση της δομής κατά την επεξεργασία, καθώς τα αποτελέσματα εκτυπώνονται τελικά με αντιστροφή της σειράς, ξεκινώντας από τα φύλλα και

καταλήγοντας στη ρίζα. Αναλυτικότερα, η διαδικασία περιλάμβανε την αφαίρεση του πρώτου στοιχείου από την ουρά (node, level), την καταγραφή του επιπέδου στο οποίο βρίσκεται ο κόμβος και, στη συνέχεια, την εισαγωγή όλων των παιδιών του στην ουρά ως (child_node, level + 1) για περαιτέρω επεξεργασία. Όταν η διαδικασία φτάσει στα φύλλα του δέντρου, τα οποία δεν διαθέτουν περαιτέρω παιδιά, δεν προστίθενται νέα στοιχεία στην ουρά. Έτσι, η ουρά αδειάζει σταδιακά και η καταμέτρηση ολοκληρώνεται. Αφού έχει συμπληρωθεί το λεξικό level_counts που διαθέτει τις πληροφορίες που συλλέχθηκαν, γίνεται αναστροφή της σειράς των επιπέδων ώστε η εκτύπωση να ξεκινά από τα φύλλα και να καταλήγει στη ρίζα, εμφανίζοντας τον αριθμό κόμβων σε κάθε επίπεδο.

Για την αποθήκευση της δομής του R-Tree σε αρχείο μορφής .txt, υλοποιήθηκε η συνάρτηση write_rtree_to_file_dfs_sorted. Η διαδικασία ξεκινά με μία αναδρομική αναζήτηση κατά βάθος (DFS) του δέντρου, αρχίζοντας από τη ρίζα. Κατά τη διάρκεια αυτής της αναζήτησης, κάθε κόμβος του δέντρου καταγράφεται σε μια λίστα entries, η οποία περιέχει λίστες της μορφής [isnonleaf, id, children_data]. Το πεδίο isnonleaf υποδεικνύει αν ο κόμβος είναι εσωτερικός ή φύλλο, ενώ το children_data περιλαμβάνει τα ids και τα MBRs των παιδιών του κόμβου. Μετά την ολοκλήρωση της αναζήτησης, η λίστα entries ταξινομείται με βάση το id κάθε κόμβου. Τέλος, τα ταξινομημένα δεδομένα αποθηκεύονται στο αρχείο Rtree.txt, με κάθε γραμμή να αντιστοιχεί σε έναν κόμβο του δέντρου.

## Μέρος 2: Ερωτήσεις Εύρους (Range Queries)

Ως πρώτο κομμάτι του δεύτερου μέρος είναι η φόρτωση του rtree από το αρχείο που αποθηκεύτηκε προηγουμένως. Αυτό πραγματοποιείται με τη συνάρτηση load_tree, η οποία αναδημιουργεί του R-Tree από το αρχείο κειμένου που έχει παραχθεί κατά την αποθήκευση του δέντρου στο πρώτο μέρος της εργασίας. Αρχικά, η συνάρτηση διαβάζει το περιεχόμενο του αρχείου και με τη χρήση της ast.literal_eval, αποδομεί κάθε εγγραφή στα τρία στοιχεία που είχαν αποθηκευτεί προηγουμένως κατά τη διαδικασία εγγραφής του δέντρου: έναν ακέραιο δείκτη (isnonleaf) που υποδηλώνει αν ο κόμβος είναι εσωτερικός ή φύλλο, το μοναδικό αναγνωριστικό του κόμβου (node_id), και τη λίστα των παιδιών του με τη μορφή ζευγών [child_id, mbr]. Με αυτά τα

δεδομένα δημιουργείται ένα λεξικό nodes, όπου κάθε καταχώρηση περιέχει βασικές πληροφορίες του κόμβου, όπως το είδος του (φύλλο ή εσωτερικός κόμβος), τη λίστα των παιδιών του (αρχικά με μορφή [id, mbr]) και το ίδιο το αναγνωριστικό.

Στη συνέχεια, αντικαθίστανται τα child_id στις λίστες παιδιών με τα αντίστοιχα πλήρη λεξικά των child κόμβων. Η αντικατάσταση αυτή δεν γίνεται με deepcopy, γεγονός που διατηρεί κοινές αναφορές μεταξύ parent και child nodes, εξασφαλίζοντας συνεκτικότητα των δεδομένων. Τα φύλλα του δέντρου διατηρούν απλούστερη δομή, περιλαμβάνοντας μόνο id και mbr. Στο τελευταίο στάδιο, εντοπίζεται ο κόμβος με το μέγιστο id ο οποίος θεωρείται root, υπολογίζεται το mbr του με συνένωση των mbrs των παιδιών του (αφού δεν αποθηκεύτηκε κατά την εγγραφή), και τροποποιείται ώστε να έχει την αρχική δομή δέντρου. Στην παρακάτω εικόνα φαίνεται ότι το ανακατασκευασμένο δέντρο ταυτίζεται με το αρχικό, καθώς η αποθήκευση και η ανάγνωση των κόμβων διατηρούν την ίδια δομή και περιεχόμενο. Παρά τη μικρή διαφορά στη σειρά των πεδίων του root κόμβου (όπου στο αρχικό δέντρο το mbr εμφανίζεται τελευταίο, ενώ στο ανακατασκευασμένο προηγείται του id), οι κόμβοι είναι ισοδύναμοι και παράγουν ίδιες hash τιμές, επιβεβαιώνοντας ότι η αναπαράσταση του δέντρου δεν έχει επηρεαστεί.

```
coords, offsets = read_data('coords.txt', 'offsets.txt')
objects = build_mbrs(coords, offsets)
objects = delete_z(objects)
rtree_root = build_rtree(objects)
write_rtree_to_file_dfs_sorted(rtree_root)
rtree = load_tree()

with open('Rtree.txt', 'r') as f:
    initial = f.read()

with open('Rtree_temp.txt', 'r') as f:
    second = f.read()

hash(initial) == hash(second)
```

```
True
```

*Figure 1: Σύγκριση hash value των δύο αρχείων που διαθέτουν τους κόμβους του δέντρο με κοινή hash function*

Στη συνέχεια, για την υλοποίηση του range query χρησιμοποιήθηκε Depth First Search του R-tree, ξεκινώντας από τη ρίζα και εξετάζοντας σε κάθε βήμα αν το ελάχιστο MBR ενός κόμβου τέμνεται με το παράθυρο ερωτήματος w. Αν δεν υπάρχει τομή, το υποδέντρο κάτω από αυτόν τον κόμβο αγνοείται. Η υλοποίηση βασίζεται στη

συνάρτηση mbrs_intersect(w, mbr), η οποία ελέγχει γεωμετρικά αν δύο MBRs τέμνονται σε δύο διαστάσεις. Όταν η αναδρομή φτάσει σε φύλλα, ελέγχεται το κάθε αντικείμενο ξεχωριστά, και όσα τέμνονται με το παράθυρο ερωτήματος προστίθενται στη λίστα αποτελεσμάτων.

## Μέρος 3: Ερωτήσεις Πλησιέστερου Γείτονα (kNN queries)

Για την υλοποίηση του kNN query, χρησιμοποιήθηκε η τεχνική Best-First Search με προτεραιότητα την εγγύτητα του κάθε κόμβου ή δεδομένου προς το σημείο ερωτήματος q. Η αναζήτηση ξεκινάει από τη ρίζα του δέντρου και διατηρεί ένα min-heap (ουρά προτεραιότητας) όπου αποθηκεύονται υποψήφιοι κόμβοι ή φύλλα με βάση την ελάχιστη Ευκλείδεια απόστασή τους από το σημείο q. Η συνάρτηση point_to_mbr_distance υπολογίζει την απόσταση ενός σημείου από ένα MBR, επιστρέφοντας 0 αν το σημείο βρίσκεται εντός του MBR, ή διαφορετικά την μικρότερη απόσταση από το περίγραμμα.

Κάθε φορά που αφαιρείται το πλησιέστερο στοιχείο από την ουρά, αν πρόκειται για φύλλο, προστίθεται στη λίστα αποτελεσμάτων. Αν είναι εσωτερικός κόμβος, προστίθενται τα παιδιά του στην ουρά με υπολογισμένη την απόστασή τους από το σημείο. Η διαδικασία επαναλαμβάνεται μέχρι να βρεθούν τα k πλησιέστερα αντικείμενα. Χρησιμοποιείται επιπλέον ένας μετρητής counter ως tie-breaker για να αποφεύγονται συγκρίσεις μεταξύ αντικειμένων που δεν μπορούν να ταξινομηθούν άμεσα (π.χ. dictionaries). Η προσέγγιση αυτή εξασφαλίζει ότι η αναζήτηση κατευθύνεται πάντα προς τις περιοχές του χώρου που έχουν μεγαλύτερη πιθανότητα να περιέχουν κοντινά αντικείμενα, βελτιώνοντας σημαντικά την αποδοτικότητα.

## Οδηγίες εκτέλεσης

Για την εκτέλεση του εκάστοτε αρχείου, ακολουθείται η παρακάτω συλλογιστική:

- Ανοίγουμε το τερματικό στο φάκελο των αρχείων .py
- Προσθέτουμε τα αρχεία των δεδομένων στον ίδιο φάκελο
- Η γενική εντολή είναι: python {scriptname.py} args

Για κάθε αρχείο ξεχωριστά:

- python rtree_init_main.py [coords_path] [offsets_path] [output_path] (default values 'coords.txt', 'offsets.txt', 'Rtree.txt' αντίστοιχα)
- python range_queries_main.py [rtree_file] [queries_file] (default values 'Rtree.txt', 'Rqueries.txt')
- python knn_queries_main.py [rtree_file] [queries_file] [k] (default values 'Rtree.txt', 'NNqueries.txt', 10)

Το filename μπορεί να δίνεται είτε με κατάληξη .txt είτε χωρίς αυτή.