Διαχείριση Σύνθετων Δεδομένων

Αναφορά 2ης Εργαστηριακής Άσκησης

ΙΩΑΝΝΗΣ ΜΠΟΥΖΑΣ

ΑΜ:5025

## Κατασκευή R-Tree μέσω Bulk Loading

Μας ζητήθηκε να κατασκευάσουμε ένα **χωρικό R-tree** από ένα σύνολο 2D αντικειμένων, χρησιμοποιώντας τη **Z-order (Morton) ταξινόμηση**.

Το δέντρο R είναι μια δεντρική δομή δεδομένων που χρησιμοποιείται για την εύρεση πολυδιάστατων πληροφοριών, όπως γεωγραφικές συντεταγμένες, ορθογώνια ή πολύγωνα. Εν ολίγης το πρόγραμμα μας κάνει τα παρακάτω:

* Διαβάζει αντικείμενα 2D από τα αρχεία μας.
* Υπολογίζει το MBR κάθε αντικειμένου.
* Υπολογίζει τον κώδικα Z-order (Morton) για ταξινόμηση.
* Κατασκευάζει το R-Tree από κάτω προς τα πάνω.
* Εκτυπώνει τον αριθμό των κόμβων ανά επίπεδο.

Πιο συγκεκριμένα έχουμε τις εξής μεθόδους:

**find\_mbr(coordinates)**

Βρίσκει το **Minimum Bounding Rectangle (MBR)** για (x, y) συντεταγμένες. Επιστρέφει σε συγκεκριμένο φορματ τις διαστάσεις του MBR [min\_x, max\_x, min\_y, max\_y].

**center\_of\_mbr(mbr)**

Υπολογίζει το κέντρο του MBR το οποίο χρησιμοποιούμε μετά για να υπολογίσουμε το Morton code

**combine\_mbrs(mbrs)**

Παίρνει πολλαπλά MBRs και επιστρέφει ένα το οποίο τα περικλείει όλα τους. Την χρησιμοποιούμε αυτή την μέθοδο για να δημιουργήσουμε parent nodes από MBRs

**distribute\_entries(entries, node\_capacity, min\_entries)**

Διανέμει τις καταχωρήσεις σε κόμβους με βάση τους περιορισμούς χωρητικότητας. Εξασφαλίζει ότι η τελευταία ομάδα δεν πέφτει κάτω από το ελάχιστο όριο εισόδου.

**build\_rtree(entries, start\_node\_id, node\_capacity, min\_entries)**

Η κύρια αναδρομική συνάρτηση που κατασκευάζει το δέντρο R:

* + Εάν οι καταχωρήσεις χωράνε σε έναν κόμβο, αυτός είναι φύλλο.
  + Διαφορετικά, είναι:
    - Κατανέμει τις καταχωρήσεις σε ομάδες (κόμβους φύλλων).
    - Υπολογίζει τα MBR για αυτές τις ομάδες.
    - Δημιουργεί έναν γονικό κόμβο για κάθε ομάδα.
    - Επαναλαμβάνει αναδρομικά τη διαδικασία για τα ανώτερα επίπεδα.
  + Επιστρέφει όλους τους κόμβους και το επόμενο διαθέσιμο ID κόμβου.

Κάθε κόμβος αναπαρίσταται ως λίστα: [is\_nonleaf\_flag, node\_id, entries], όπου:

* is\_nonleaf\_flag = 0 για φύλλα, 1 για κόμβους χωρίς φύλλα.
* Οι καταχωρήσεις σε ένα φύλλο είναι [object\_id, mbr].
* Οι καταχωρήσεις σε ένα μη φύλλο είναι [child\_node\_id, mbr]

**count\_nodes\_by\_level(nodes)**

Βρίσκει πόσοι κόμβοι υπάρχουν σε κάθε επίπεδο δέντρου:

* Καθορίζει τον κόμβο-ρίζα (που δεν αναφέρεται από κανέναν άλλο κόμβο).
* Πραγματοποιεί BFS.
* Παρακολουθεί και μετράει τους κόμβους ανά επίπεδο.

Το main workflow είναι το εξής:

**Διαβάζουμε τις συντεταγμένες και τα offsets**

* Βάζουμε τις συντεταγμένες σε μια λίστα.
* Κάνουμε map κάθε αντικείμενο ID στις συγκεκριμένες συντεταγμένες του χρησιμοποιώντας τα offsets

**2. Υπολογίζουμε MBRs και Z-Order Codes**

* Για κάθε αντικείμενο:
  + Υπολογίζουμε MBR.
  + Υπολογίζουμε το Z-order (Morton) από το κέντρο των MBRs.

**3. Σορταρουμε τα αντικείμενα χρησιμοποιώντας Z-Order**

* Διατηρούμε το spatial locality όταν σορταρουμε τα αντικείμενα.

**4. Φτιάχνουμε τα Φύλλα**

* Δημιουργία λίστας [object\_id, mbr] για κάθε αντικείμενο..

**5. Κατασκευάζουμε το R-Tree**

* Καλεί αναδρομικά την build\_rtree.
* Χωρητικότητα κόμβων 20 και ελάχιστες καταχωρήσεις 8.

**6. Καταμέτρηση κόμβων ανά επίπεδο**

* Χρησιμοποιεί το count\_nodes\_by\_level για την έξοδο της δομής του δέντρου

**7. Σορταρουμε το δέντρο**

* Η ρίζα του δέντρου είναι στο τέλος του αρχείου.

**8. Αποθηκεύουμε το αποτέλεσμα στο αρχείο Rtree.txt**

## Ερωτήσεις Εύρους (Range queries)

Πρέπει να υλοποιήσουμε μια συνάρτηση αποτίμησης ερωτήσεων εύρους στο R-Tree που φτιάξατε. Το εύρος της ερώτησης καθορίζεται από ένα ορθογώνιο W και το ζητούμενο είναι να βρεθούν τα MBRs που τέμνουν το W.

Πιο συγκεκριμένα έχουμε τις εξής μεθόδους:

**1. Φόρτωση R-tree**

Η συνάρτηση load\_rtree() διαβάζει όλους τους κόμβους από το αρχείο R-tree και τους αποθηκεύει σε ένα **λεξικό (nodes\_dict)** με κλειδί το node\_id, για πιο γρήγορη πρόσβαση. Ο τελευταίος κόμβος στο αρχείο θεωρείται η **ρίζα** του δέντρου.

**2. Διαχείριση Ερωτημάτων**

Η συνάρτηση read\_queries() διαβάζει τα ερωτήματα από αρχείο και τα μετατρέπει σε μορφή [x\_min, x\_max, y\_min, y\_max], ώστε να είναι συμβατή με τις συναρτήσεις του προγράμματος.

**3. Range Queries**

Η βασική λειτουργία range\_query(root\_id, query\_rect):

* Ξεκινά από τη ρίζα του δέντρου και ελέγχει αν τα MBRs των κόμβων τέμνονται με το ορθογώνιο ερώτημα.
* Χρησιμοποιεί **ουρά BFS (Breadth-First Search)** για να εξετάσει όλους τους σχετικούς κόμβους.
* Αν ένας κόμβος είναι φύλλο και το αντικείμενο του τέμνεται με την περιοχή, τότε προστίθεται στο αποτέλεσμα.

Η συνάρτηση mbr\_intersects() ελέγχει αν δύο MBRs τέμνονται.

**Έξοδος**

Για κάθε ερώτημα, το πρόγραμμα επιστρέφει στη γραμμή εντολών:

<αριθμός\_ερωτήματος> (<πλήθος\_αποτελεσμάτων>): <λίστα\_id\_αντικειμένων>

## Ερωτήσεις Πλησιέστερου Γείτονα (kNN queries)

Μας ζητήθηκε να υλοποιήσουμε τον αλγόριθμο best-first search για ανάκτηση του πλησιέστερου object MBR σε ένα σημείο αναφοράς q.

Πιο συγκεκριμένα κάνουμε τα εξής στο πρόγραμμα μας:

**1. Υπολογισμός Απόστασης – mindist()**

Η συνάρτηση mindist(point, mbr) υπολογίζει την ελάχιστη ευκλείδεια απόσταση ενός σημείου από ένα MBR (Minimum Bounding Rectangle).

* Αν το σημείο βρίσκεται *εντός* του MBR → απόσταση = 0.
* Αν βρίσκεται *εκτός*, τότε βρίσκουμε τις πλησιέστερες ακμές.

**2. Αναζήτηση k-NN – knn\_search()**

Η βασική συνάρτηση του προγράμματος:

* Ξεκινά από τη ρίζα του R-tree.
* Εισάγει όλα τα παιδιά της ρίζας σε μια **προτεραιότητα (min-heap)** βάσει της mindist από το σημείο-ερώτημα.
* Επαναληπτικά:
  + Εξάγει τον πλησιέστερο κόμβο ή αντικείμενο από το heap.
  + Αν είναι αντικείμενο, το προσθέτει στο αποτέλεσμα.
  + Αν είναι εσωτερικός κόμβος, επεκτείνει τα παιδιά του στο heap.
* Η διαδικασία σταματά όταν έχουν βρεθεί k αντικείμενα.

**3. Φόρτωση R-tree – process\_knn\_queries()**

* Διαβάζει και αποθηκεύει όλους τους κόμβους του δέντρου σε λεξικό.
* Διαβάζει τα ερωτήματα σημείων και για κάθε σημείο:
  + Εκτελεί το knn\_search()
  + Εκτυπώνει τα αποτελέσματα.

**Έξοδος Προγράμματος**

Για κάθε σημείο-ερώτημα, εμφανίζεται:

<αριθμός\_ερωτήματος> (<πλήθος\_αποτελεσμάτων>): <λίστα\_id\_αντικειμένων>