Διαχείριση Σύνθετων Δεδομένων

Εργασία 2: Χωρικά Δεδομένα

Κωνσταντίνος Παπαντωνίου - Χατζηγιώσης, ΑΜ: 4769

Μέρος 1ο:

Στο πρώτο μέρος της εργασίας, μας ζητήθηκε η υλοποίηση ενός προγράμματος το οποίο κατασκευάζει R-δέντρα χρησιμοποιώντας τον sort-tile-recursive αλγόριθμο. Τα R-δέντρα είναι μια δομή δεδομένων η οποία χρησιμοποιείται για την αποθήκευση πολύ διάστατων δεδομένων τα οποία μπορούν να ανακτηθούν μέσω των δεικτών των δεδομένων. Τα δεδομένα οργανώνονται με βάση τις χωρικές τους σχέσεις.

Ο αλγόριθμος sort-tile-recursive, είναι ένας αλγόριθμος που κατασκευάζει το δέντρο αναδρομικά, χωρίζοντας τα δεδομένα σε μικρότερες ομάδες βάσει μιας επιλεγμένης διάστασης και στην συνέχεια τα ταξινομεί.

Αρχικά διαβάζουμε το αρχείο που περιέχει χωρικά δεδομένα από την γραμμή εντολών το οποίο πρέπει να είναι με συγκεκριμένο φορμάτ για να δουλέψει ορθά το πρόγραμμα. Θα πρέπει το αρχείο στην πρώτη γραμμή να έχει τον αριθμό των συντεταγμένων και οι υπόλοιπες γραμμές να περιέχουν, μία η κάθε γραμμή, δισδιάστατη συντεταγμένη στον χώρο.

Για την υλοποίηση του δέντρου θα χρησιμοποιήσουμε την δυνατότητα αντικειμενοστραφούς προγραμματισμού που μας προσφέρει η Python ώστε να φτιάξουμε ξεχωριστά αντικείμενα για τα βασικά στοιχεία του δέντρου. Αυτό επιτυγχάνεται με τον ορισμό κλάσεων, συγκεκριμένα ορίζουμε τις κλάσεις:

1. Point, η οποία κράτα τις συντεταγμένες των σημείων μαζί με έναν ξεχωριστό δείκτη για κάθε σημείο και επιστρέφει τις συντεταγμένες σαν αλφαριθμητικά.
2. Rect, η κλάση αυτή δέχεται δύο σημεία σαν όρισμα τα οποία είναι το ελάχιστο και το μέγιστο και επιστρέφει ένα παραλληλόγραμμο με όρια τα σημεία αυτά.
3. EntryBlock, η συγκεκριμένη κλάση είναι χρησιμοποιείται για να αναπαραστήσουμε τις εγγραφές του δέντρου και περιέχει τον

αναγνωριστικό αριθμό της κάθε εγγραφής καθώς και τα παραλληλόγραμμα που θα περιέχει η εγγραφή. Επιπλέον καταγράφει και τον αριθμό των σημείων που υπάρχουν στην εγγραφή στα οποία μπορεί να προσθέσει σημεία η να τα επιστρέψει.

4. LeafNode, η κλάση αυτή αναπαριστά τους κόμβους-φύλλα του δέντρου τα οποία αποθηκεύουν τις εγγραφές των δεδομένων μαζί με το αναγνωριστικό του κόμβου. Επιπλέον υπολογίζουν το MBR(Minimum Bounding Rectangle) που περιβάλλει τις εγγραφές του και το εμβαδό αυτού.
5. LeafLessNode, καθώς το δέντρο διαφοροποιεί τους κόμβους που περιέχουν άλλους κόμβους και τους κόμβους που περιέχουν εγγραφές, η κλάση αυτή αναπαριστά επίσης κόμβο αλλά τους ενδιάμεσους κόμβους που έχουν παιδιά (κόμβους, κόμβους-φύλλα), η λειτουργία τους είναι παρόμοια με τους άλλους κόμβους καθώς και αυτοί υπολογίζουν το MBR των παιδιών τους και προσθέτουν παιδιά τα οποία τα επιστρέφουμε με συγκεκριμένο φορμάτ χάρη στην __str__ .
6. RTree, τέλος η κλάση αυτή είναι η τελική αναπαράσταση του δέντρου στην μνήμη. Περιέχει τον κόμβο πηγή και τα παιδιά του καθώς και την βασική λειτουργία κατασκευής των δέντρων.

Στην κλάση RTree κατασκευάζουμε τα βασικά στοιχεία του δέντρου, την πηγή και τους ενδιάμεσους κόμβους καθώς για τους κόμβους φύλλα χρειαζόμαστε να τροφοδοτούμε δεδομένα. Έχουμε μία συνάρτηση η οποία δημιουργεί τον κόμβο ρίζα σαν τον πρώτο μη φύλλο-κόμβο και τον βάζει στην αρχή της λίστας που περιέχει όλους τους κόμβους καθώς και προσθέτουμε σε αυτόν όλα τα παιδιά. Επιπλέον έχουμε μια αναδρομική συνάρτηση η οποία για για τον μέγιστο αριθμό των κόμβων που μπορούμε να χρησιμοποιήσουμε προσθέτει κόμβους μη φύλλα στο δέντρο αφού πρώτα ελέγχει αν ο αριθμός αυτός είναι μόνο ένα κόμβος ώστε να καλέσει την συνάρτηση για την δημιουργία της ρίζας. Επιπλέον υπάρχουν οι συναρτήσεις που προσθέτουν κόμβους στο δέντρο είτε είναι φύλλα είτε ενδιάμεσοι κόμβοι. Η πρόσθεση τον φύλλων-κόμβων σε αυτό το σημείο είναι απλή, καθώς απλά τους προσθέτουμε στο δέντρο μαζί με τα δεδομένα τους, αλλά η πρόσθεση των μη-φύλλων είναι λίγο πιο πολύπλοκη. Αρχικά χωρίζουμε σε κομμάτια τα παιδιά που θα έχει ο κάθε κόμβος δεδομένο τον μέγιστο αριθμό των παιδιών που επιτρέπουμε, αφού βρούμε το μέγεθος περνάμε σε κάθε κομμάτι των αριθμό των παιδιών που επιτρέπεται και μετά αποθηκεύουμε σε έναν προσωρινό πίνακα τα κομμάτια σε κάθε θέση ξεχωριστά. Σε επόμενο βήμα δημιουργείται για κάθε κομμάτι έναν ενδιάμεσο κόμβο και του περνάμε τα παιδιά που αντιστοιχούν στο κομμάτι καθώς και τον υπολογισμό του mbr. Τέλος ενημερώνουμε πίνακα που κρατάει τα παιδιά του δέντρου.

Οι δύο τελικές λειτουργίες που υλοποιούνται την κλάση του δέντρου είναι η αποθήκευση του δέντρου σε ένα αρχείο το οποίο αναπαριστά το δέντρο όπως είναι αποθηκευμένο στην μνήμη σαν block όπου περιέχουν δείκτες που δείχνουν σε άλλα

block και κρατάνε τα δεδομένα του δέντρου. Επιπλέον εκτυπώνουμε τα στατιστικά του δέντρου, συγκεκριμένα το ύψος του δέντρου τον αριθμό των κόμβων του κάθε επιπέδου και το μέσο εμβαδόν κάθε επιπέδου από το MBR. Η συνάρτηση για την εκτύπωση των στατιστικών χρησιμοποιεί την δομή των ουρών για να υλοποιήσουμε το επιθυμητό αποτέλεσμα. Ορίζουμε μια ουρά που την αρχικοποιούμε με δεδομένα τον ρίζα του δέντρου και το αρχικό επίπεδο. Στο επόμενο βήμα μέσα σε έναν επαναληπτικό βρόχο παίρνουμε τα αντικείμενα στο τελευταίο επίπεδο της ουράς, όπου και το διαγράφουμε για να το ελέγχουμε μόνο μια φορά, και ελέγχουμε αρχικά εάν σε ένα λεξικό υπάρχει τιμή με το κλειδί του επιπέδου, που μας δίνει η ουρά μαζί με τον κόμβο που θα ψάξουμε. Εάν υπάρχει το επίπεδο στο λεξικό αυξάνουμε τον αριθμό τον αντικειμένων του επιπέδου, εδώ δείχνουμε πόσα παιδιά υπάρχουν στο επίπεδο, αλλιώς δημιουργούμε μια νέα θέση στο λεξικό με κλειδί το επίπεδο. Εάν ο κόμβος που παίρνουμε από την ουρά είναι ενδιάμεσος κόμβος τότε υπολογίζουμε το μέσο εμβαδό των MBR τον παιδιών του κόμβου και το προσθέτουμε σε ένα λεξικό που κρατάει τα μέσα εμβαδά και των άλλον κόμβων στο επίπεδο που βρισκόμαστε. Σε αυτό το σημείο παρακάμπτουμε την περίπτωση του κόμβου να είναι φύλλο καθώς περιέχει σημεία αυτός ο τύπος κόμβων οπότε το εμβαδό είναι πάντα μηδέν και η λογική τις συναρτήσεις ελέγχει τα παιδία κάθε φορά των κόμβων οπότε οι κόμβοι φύλλα δεν έχουν παιδία αλλά εγγραφές. Τέλος τυπώνουμε τον αριθμό των επιπέδων, τον αριθμό των κόμβων σε κάθε επίπεδο καθώς και τον μέσο όρο των μέσων εμβαδών σε κάθε επίπεδο.

Έξω από τις κλάσεις του προγράμματος υλοποιούμε τον STR αλγόριθμο και παράλληλα τροφοδοτούμε τα δεδομένα στο δέντρο με την συνάρτηση back_load. Στην συνάρτηση αυτή αρχικά ταξινομούμε τις εγγραφές με βάση το ελάχιστη ‘x’ συντεταγμένη, αφού υπολογίσουμε τον μέγιστο αριθμό των παιδιών για κάθε κόμβο αλλά και τον μέγιστο αριθμό των εγγραφών((8+8+4=20) για τους κόμβους φύλλα αφού έχουν 2 συντεταγμένες και (8+8+8+8+4=36) για τους ενδιάμεσους κόμβους εφόσον έχουν 4 συντεταγμένες). Στην συνέχεια θα χωρίσουμε τις εγγραφές σε κομμάτια με βάση των μέγιστο επιτρεπτό αριθμό και θα ταξινομήσουμε το κάθε κομμάτι με βάση την ελάχιστη συντεταγμένη στον άξονα ‘y’. Τέλος δημιουργούμε την οντότητα του δέντρου και τροφοδοτούμε τους κόμβους φύλλα στο δέντρο έχοντας πρώτα αποθήκευση κομμάτια των εγγραφών σε κάθε κόμβο. Αυτό το κομμάτι γίνεται πρώτο πριν καλέσουμε την add_sarting_nodes διότι κατασκευάζουμε το δέντρο αναδρομικά και οι ενδιάμεσοι κόμβοι κατασκευάζονται με βάση των αριθμό τον παιδιών που χρειάζεται να έχουν. Οπότε πρώτα αποθηκεύουμε στο δέντρο τους κόμβους φύλλα που θα πρέπει να έχει σε έναν πίνακα και μετά στο δέντρο που επιστρέφει η back_load καλούμε την add_starting_node για να χωρίσει να παιδία(κόμβους φύλλα) στους ενδιάμεσους κόμβους, όπως αναφέραμε παραπάνω.

Τέλος στο ‘main’ κομμάτι του προγράμματος διαβάζουμε το αρχείο με τα δεδομένα από την γραμμή εντολών και για κάθε γραμμή δημιουργούμε τα σημεία και τα αρχικοποιούμε σε ένα παραλληλόγραμμο για να τα αποθηκεύσουμε σαν εγγραφές

στο σύστημα και ύστερα καλούμε τις συναρτήσεις για την κατασκευή του δέντρου με τις συγκεκριμένες εγγραφές.

Μέρος 2ο:

Στο δεύτερο μέρος της άσκησης μας ζητείται να διαβάσουμε το δέντρο που έχουμε κατασκευάσει και να υπολογίσουμε τον incremental nearest neighbor search αλγόριθμο που βασίζεται σε best-first search ώστε να ψάξουμε τους κοντινότερους k, k+1 και k+2 γείτονες από ένα τυχαίο σημείο που επιλέγουμε. Αρχικά διαβάζουμε από την γραμμή εντολών τα δεδομένα με το format (: πρώτο όρισμα το αρχείο που έχει αποθηκευμένο το δέντρο, δεύτερο όρισμα την 'χ' συντεταγμένη του σημείου που θέλουμε να βρούμε, τρίτο όρισμα την 'y' συντεταγμένη και τέλος τον αριθμό 'k' που μας δείχνει τον αριθμό τον πλησιέστερων γειτόνων που θέλουμε να βρούμε).

Αφού πάρουμε τα ορίσματα από την γραμμή εντολών ανοίγουμε και διαβάζουμε το αρχείο και κάθε αποθηκεύουμε την κάθε γραμμή του αρχείου σε έναν πίνακα data χωρίζοντας τα δεδομένα με την εμφάνιση του ',' και διαγράφοντας τους κενούς χαρακτήρες. Ξέροντας την δομή του δέντρου που διαβάζουμε μπορούμε να το αναπαραστήσουμε στην μνήμη όπως στο μέρος ένας αποθηκεύοντας κάθε στοιχείο του δέντρου σε ξεχωριστούς πίνακες όπως node_id όπου κρατάει τα αναγνωριστικά των κόμβων , nums_children όπου κρατάει των αριθμό των παιδιών κάθε κόμβου, flags για το αν ο κόμβος είναι φύλλο η μη και τέλος children_data όπου τέλος κρατάει τα δεδομένα του κάθε κόμβου, κόμβους παιδία και mbr αν πρόκειται για ενδιάμεσους αλλιώς αναγνωριστικά σημείων και τις συντεταγμένες τους).

Στο επόμενο βήμα θέλουμε να προσπελάσουμε τα δεδομένα ταυτόχρονα ώστε να έχουμε την αντιστοίχιση για κάθε κόμβο με τα σωστά δεδομένα, αυτό το καταφέρνουμε με την χρήση του zip() όπου μας επιτρέπει να ανατρέξουμε ταυτόχρονα τους 4 πίνακες δεδομένων δημιουργώντας στο τέλος ένα tuple που κρατάει τα αντιστοιχισμένα δεδομένα για κάθε επανάληψη. Θα χρειαστεί να 'καθαρίσουμε κάποια από τα δεδομένα από χαρακτήρες όπως '(',')' και να δημιουργήσουμε έναν εσωτερικό πίνακα για τα δεδομένα του children_data καθώς περιέχει παιδιά και δεδομένα τα οποία θέλουμε να είναι το καθένα διαχωρισμένο και ο κάθε πίνακας να αντιστοιχεί σε ένα παιδί. Αυτό το επιτυγχάνουμε χωρίζονται τα δεδομένα σε ένα tuple που το πρώτο στοιχείο περιέχει το αναγνωριστικό και το δεύτερο περιέχει αριθμούς κινητής υποδιαστολής(χάρη στην map,float), είτε 4 αν flag είναι 1 άρα ενδιάμεσος κόμβος είτε 2 εάν flag είναι 0 άρα φύλλο. Τέλος χρησιμοποιούμε ένα λεξικό ώστε να αποθηκεύουμε τα δεδομένα που φτιάξαμε στο κόμβο με το συγκεκριμένο αναγνωριστικό κάθε φορά. Με αυτό τον τρόπο έχουμε αποθηκεύσει και αναπαραστήσει το δέντρο στην μνήμη μέσα στο λεξικό και μπορούμε να έχουμε πρόσβαση σε κάθε κόμβο συγκεκριμένα και στα δεδομένα του.

Επιπλέον για την υλοποίηση της αναζήτησης πλησιέστερου γείτονα θα χρειαστούμε μια συνάρτηση που υπολογίζει την ευκλείδεια απόσταση μεταξύ ενός σημείου και ενός mbr. Για την λειτουργία αυτή φτιάχνουμε την mindist όπου δέχεται σαν παράμετρο το σημείο και το mbr που θέλουμε. Αρχικά ελέγχουμε εάν το σημείο εμπεριέχεται στο παραλληλόγραμμο, όπου επιστρέφει 0 εφόσον έχει μηδενική απόσταση. Ύστερα συγκρίνουμε από ποια πλευρά το σημείο είναι πιο κοντά σε σχέση με τους δύο άξονες και τα αποτελέσματα τα βάζουμε στον τύπο sqrt(dx^2+dy^2).

Τέλος ορίζουμε την συνάρτηση INNS που υλοποιεί την βασική λειτουργία της αναζήτησης. Δέχεται σαν όρισμα το σημείο που ψάχνουμε, το δέντρο που έχουμε στην μνήμη και τον αριθμό κ γειτόνων. Αρχικοποιούμε μια ουρά που παίρνει σαν όρισμα ένα tuple που περιέχει την ελάχιστη απόσταση μεταξύ του σημείου και των mbr των παιδιών της ρίζας, το αναγνωριστικό της ρίζας καθώς και τα δεδομένα της ρίζας. Σε επόμενο βήμα για όλους τους υπόλοιπους κόμβους φτιάχνουμε μια while loop και μέσα κάθε φορά παίρνουμε τα περιεχόμενα της ουράς με βάση την ελάχιστη τιμή του πρώτου στοιχείου της ουράς (στην περίπτωση μας την ελάχιστη απόσταση από τις αποστάσεις των mbr από κάθε παιδί με το σημείο). Το οποίο μετά διαγράφουμε ώστε στο επόμενο πέρασμα να είμαστε ένα επίπεδο κάτω. Με βάση τα δεδομένα του κόμβου που επιλέγεται με την ελάχιστη απόσταση ελέγχουμε εάν ο κόμβος αυτός είναι φύλλο ή ενδιάμεσος κόμβος. Στην περίπτωση που ο κόμβος είναι φύλλο αποθηκεύουμε σε έναν πίνακα, που κρατά τους γείτονες, την ευκλείδεια απόσταση κάθε σημείου από το σημείο μας, το αναγνωριστικό του σημείου και τις συντεταγμένες του. Εάν ο κόμβος είναι ενδιάμεσος (flag ==1) τότε ελέγχουμε αν έχουμε περάσει από τον συγκεκριμένο κόμβο, μέσο ενός set που προσθέτουμε κάθε φορά τους κόμβους που ελέγχουμε, όπου επαναληπτικά συγκρίνουμε την απόσταση του σημείου μας μεταξύ των mbr των παιδιών του και στο τέλος προσθέτουμε στην ουρά τον κόμβο και τα δεδομένα που βρίσκεται πιο κοντά στο σημείο μας. Τέλος μετά την επαναληπτική while επιστρέφουμε τα k+3 αντικείμενα της ταξινομημένης λίστας με τους πλησιέστερους γείτονες ως προς την μικρότερη τιμή όπου και τα εκτυπώνουμε στην οθόνη. Επιπλέον ομοίως με πριν έχουμε βάλει την λειτουργία να καταγράφει τον χρόνο υλοποίησης του προγράμματος.