Διαχείριση Σύνθετων Δεδομένων

2ο Σετ ασκήσεων

Μπακάλης Δημήτριος Α.Μ. 3033

Το πρόγραμμα αποτελείται από 7 συναρτήσεις και μια τελική συνάρτηση main. Όταν εκτελείται το πρόγραμμα, καλείται η main() και έτσι ξεκινάει η διαδικασία παραγωγής αποτελεσμάτων.
Το πρόγραμμα για να τρέξει χρειάζεται να δώσουμε την εξής εντολή στο τερματικό: **« python3 main.py data_rectangles.txt »**
και στο φάκελο που θα υπάρχει το πρόγραμμα, να βρίσκονται και τα αρχεία *data_rectangles.txt* και *query_rectangles.txt.*

Βασική λειτουργία προγράμματος:

1ο Μέρος:

Το πρόγραμμα ξεκινάει από την **main()** και η πρώτη «δουλειά» που κάνω είναι να διαβάσω τα MBRs μέσα από το δοθέν αρχείο και να τα βάλω μέσα σε μια λίστα (mbrs). Μετά κάνω sort στην λίστα με βάση το x-low στοιχείο και υπολογίζω τα R P και S.
Προχωράω με το να δημιουργήσω τους κόμβους φύλλα του R-TREE δέντρου μου, σπάζοντας τα MBRs σε «φέτες» των S*N και κάνοντας sort σε αυτά τα MBRs με βάση την y-low τιμή τους. Μόλις φτάσω στο τελευταίο MBR τότε ολοκλήρωσα την διαδικασία παραγωγής φύλλων κόμβων.
Τους κόμβους του δέντρου τους αποθηκεύω σε μια λίστα (nodes_array) καθώς επίσης κρατάω σε μια διαφορετική λίστα (level_nodes) τον αριθμό των κόμβων σε κάθε επίπεδο, θεωρώντας το επίπεδο των κόμβων-φύλλων το επίπεδο 0.
Στην συνέχεια υπολογίζω τους κόμβους όλων των υπόλοιπων επιπέδων μέχρι και την ρίζα, καλώντας την συνάρτηση **create_nodes()**.
Όταν τελειώσει η διαδικασία έχω την λίστα μου (nodes_array) γεμάτη με όλους τους κόμβους του δέντρου μου καθώς και την λίστα (level_nodes) γεμάτη με τον αριθμό των κόμβων κάθε επιπέδου.
Επιπλέον, γράφω στο ζητούμενο αρχείο (rtree.txt) το node-id της ρίζας, τον αριθμό των επιπέδων στο δέντρο και όλους του κόμβους.
Τέλος, τυπώνω στο τερματικό τα ζητούμενα της άσκησης (ύψος δέντρου, αριθμός κόμβων σε κάθε επίπεδο, και μέσο εμβαδό των MBRs σε κάθε επίπεδο).

2ο Μέρος:

Για το δεύτερο μέρος της εργασίας διαβάζω από το query_rectangles τα MBRs και τα κρατάω σε μια λίστα (query).
Διατρέχω την λίστα και για κάθε MBR καλώ 3 συναρτήσεις, τις: **range_query()** , **range_query_inside()** και **range_query_contain()**

Η **range_query()** συνεργάζεται με την συνάρτηση **intersect()** και μου ενημερώνει τις global μεταβλητές *used_nodes* και *inter_mbrs.*
Η μεταβλητή *used_nodes* μας δείχνει πόσοι κόμβοι προσπελάθηκαν και η μεταβλητή *inter_mbrs* πόσα mbrs του R-tree μας, τέμνουν το εκάστοτε mbr του ερωτήματος .

Η **range_query_inside()** συνεργάζεται με την συνάρτηση **inside()** και μου ενημερώνει τις global μεταβλητές *used_nodes* και *ins_mbrs.*
Η μεταβλητή *used_nodes* μας δείχνει πόσοι κόμβοι προσπελάθηκαν και η μεταβλητή *ins_mbrs* πόσα mbrs του R-tree μας, περιέχονται στο εκάστοτε mbr του ερωτήματος .

Η **range_query_ contain ()** συνεργάζεται με την συνάρτηση **contain()** και μου ενημερώνει τις global μεταβλητές *used_nodes* και *cont_mbrs.*
Η μεταβλητή *used_nodes* μας δείχνει πόσοι κόμβοι προσπελάθηκαν και η μεταβλητή *cont_mbrs* πόσα mbrs του R-tree μας, περιέχουν το εκάστοτε mbr του ερωτήματος .

Οι 3 συναρτήσεις **range_query()** , **range_query_inside()** και **range_query_contain()** έχουν παρόμοια λειτουργία μιας και χρησιμοποιούν την intersect() μέχρι να φτάσουμε σε κόμβο-φύλλο. Μετά η κάθε συνάρτηση, όταν φτάσει σε κόμβο-φύλλο, χρησιμοποιεί την κατάλληλη συνάρτηση από τις *intersect() inside()* και *contain ()* αντίστοιχα.

Τέλος για κάθε mbr από το query_rectangles, τυπώνω τα ζητούμενα της άσκησης.