ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων

(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2019-20)

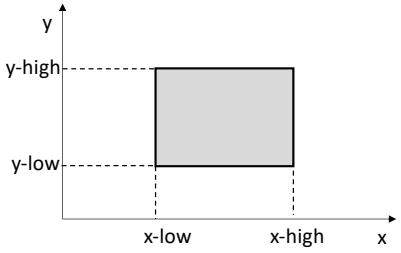
ΕΡΓΑΣΙΑ 2 – Χωρικά Δεδομένα

Προθεσμία: 30 Απριλίου 2020, 9μ.μ.

Στόχος της εργασίας είναι η ανάπτυξη τεχνικών δεικτοδότησης (δηλ. ευρετηρίασης) και αναζήτησης χωρικών δεδομένων.

Μέρος 1 (κατασκευή R-δέντρου)

Κατεβάστε το αρχείο data_rectangles.txt από το ecourse. Το αρχείο περιέχει τις συντεταγμένες 126437 ορθογωνίων τα οποία είναι MBRs χωρικών αντικειμένων. Η κάθε γραμμή του αρχείου έχει την εξής μορφή:



object-id x-low x-high y-low y-high

όπου object-id είναι το προσδιοριστικό του αντικειμένου (ακέραιος), x-low είναι το κάτω όριο του ορθογωνίου στη διάσταση x, x-high είναι το πάνω όριο του ορθογωνίου στη διάσταση x, y-low είναι το κάτω όριο του ορθογωνίου στη διάσταση y, και y-high είναι το πάνω όριο του ορθογωνίου στη διάσταση y.

Γράψτε ένα πρόγραμμα το οποίο υλοποιεί την τεχνική sort-tile-recursive (STR) για να διαβάσει τα ορθογώνια από το αρχείο και να φτιάξει ένα R-tree στη μνήμη για αυτά. Η τεχνική αυτή ταξινομεί αρχικά όλα τα ορθογώνια με βάση τη x-low τιμή τους και μετά διαβάζει τμήματα από την ταξινομημένη λίστα (ή αρχείο) που αντιστοιχούν στην τετραγωνική ρίζα του συνολικού αριθμού των φύλλων του δέντρου, τα αναταξινομεί με βάση την y-low τιμή τους και φτιάχνει τα φύλλα σταδιακά (όπως και το υπόλοιπο δέντρο). Λεπτομέρειες μπορείτε να δείτε στις σημειώσεις του μαθήματος και στην εργασία:

Scott T. Leutenegger, J. M. Edgington, and Mario A. López. 1997. STR: A Simple and Efficient Algorithm for R-Tree Packing. In ICDE. 497–506.

https://apps.dtic.mil/sti/pdfs/ADA324493.pdf

Θεωρήστε ότι (α) ο κάθε κόμβος έχει χωρητικότητα 1024 bytes και ότι χρησιμοποιούμε 4 bytes για κάθε object-id ή node-id και ένα double (8 bytes) για κάθε συντεταγμένη. Άρα κάθε εγγραφή χρειάζεται 36 bytes για να αποθηκευτεί.

Στην υλοποίησή σας θα πρέπει να διαβάσετε τα δεδομένα από το αρχείο, να κάνετε την ταξινόμηση και να φτιάξετε το δέντρο στη μνήμη. Θα χρησιμοποιήσετε μια δομή array ή vector για να αποθηκεύσετε τους κόμβους. Καθώς φτιάχνετε το δέντρο, θα προσθέτετε τους κόμβους στο array ή vector και **το node-id ενός κόμβου θα είναι η θέση του** στο array ή vector. Με αυτό τον τρόπο προσομοιώνουμε μια ακολουθία από blocks στο δίσκο που αποθηκεύουν το δέντρο.

Το πρόγραμμά σας **θα πρέπει να τυπώνει στατιστικά για το δέντρο**: ύψος (δηλ. αριθμός επιπέδων), αριθμός κόμβων σε κάθε επίπεδο, και μέσο εμβαδό των MBRs σε κάθε επίπεδο.

Επίσης **θα πρέπει να γράφει σε ένα text αρχείο εξόδου rtree.txt** την αναπαράσταση του δέντρου. Η πρώτη γραμμή θα πρέπει να έχει μόνο το node-id της ρίζας του δέντρου. Η δεύτερη γραμμή θα πρέπει να έχει μόνο τον αριθμό των επιπέδων του δέντρου. Καθεμία από τις επόμενες γραμμές του αρχείου θα περιέχει τα δεδομένα ενός κόμβου στην εξής μορφή:

node-id, n, (ptr1, MBR1), (ptr2, MBR2), …, (ptrn, MBRn)

όπου node-id είναι το id του κόμβου, n ο αριθμός των εγγραφών στον κόμβο, και ακολουθούν οι n εγγραφές μέσα στον κόμβο. Σε κάθε εγγραφή, το ptr είναι είτε ένα node-id (αν η εγγραφή δείχνει σε ενδιάμεσο κόμβο) είτε ένα object-id αν η εγγραφή δείχνει σε αντικείμενο. Το MBR είναι είτε μια ακολουθία 4 doubles <x-low> <x-high> <y-low> <y-high>.

Το πρόγραμμά σας θα πρέπει να τρέχει στη γραμμή διαταγών και να παίρνει σαν όρισμα το αρχείο με τα δεδομένα.

Για τις λεπτομέρειες του STR bulk loading μπορείτε να ανατρέξετε στις σημειώσεις του μαθήματος ή στο παρακάτω άρθρο:

https://pdfs.semanticscholar.org/e680/839472e7f5cab0f12fcc7c4aba6834a2e096.pdf

Χρησιμοποιήστε έτοιμες συναρτήσεις ταξινόμησης στη μνήμη (όχι external sorting).

Μέρος 2 (ερωτήσεις στο R-δέντρο)

Υλοποιήστε συναρτήσεις αποτίμησης ερωτήσεων επιλογής (range queries) στο R-tree που φτιάξατε. Καλείστε να υλοποιήσετε 3 τύπους ερωτήσεων:

1) Range intersection query: δίνεται ένα ορθογώνιο q στο χώρο και ο στόχος είναι να βρούμε τα ορθογώνια που **τέμνουν** (δηλαδή, έχουν ένα κοινό σημείο με) το q.

2) Range inside query: δίνεται ένα ορθογώνιο q στο χώρο και ο στόχος είναι να βρούμε τα ορθογώνια που **περιέχονται** στο q.

3) Containment query: δίνεται ένα ορθογώνιο q στο χώρο και ο στόχος είναι να βρούμε τα ορθογώνια που **περιέχουν** το q

Για καθεμιά από τις παραπάνω ερωτήσεις υλοποιήστε μια συνάρτηση, η οποία χρησιμοποιεί το R-tree που φτιάξατε στο 1ο μέρος για να **μετρήσει** τον αριθμό των αποτελεσμάτων της

ερώτησης και **τον αριθμό των κόμβων του R-δέντρου** που προσπελαύνονται για να αποτιμηθεί η ερώτηση.

Διαβάστε από το αρχείο query_rectangles.txt τις ερωτήσεις που περιέχονται σε αυτό με τη μορφή:

query-id x-low x-high y-low y-high

και υπολογίστε για κάθε ερώτηση τον αριθμό των απαντήσεων και τον αριθμό των node accesses για κάθε ερώτηση χρησιμοποιώντας τις συναρτήσεις σας. Συγκεκριμένα, διατρέξτε το αρχείο των ερωτήσεων και τυπώστε στην έξοδο τον αριθμό των αποτελεσμάτων και τον αριθμό των κόμβων για κάθε ορθογώνιο ερώτησης και για καθένα από τους τύπους ερωτήσεων.

Υπόδειξη: Για να ελέγξετε την ορθότητα των αποτελεσμάτων των ερωτήσεων με χρήση του δέντρου, μπορείτε να φτιάξετε και εναλλακτικές εκδόσεις των συναρτήσεών σας οι οποίες δεν θα υπολογίζουν τα αποτελέσματα με χρήση του δέντρου, αλλά με γραμμική αναζήτηση. Αυτές οι εκδόσεις θα είναι φυσικά πιο αργές, αλλά θα δίνουν τα σωστά αποτελέσματα. Αν τα αποτελέσματα του δέντρου είναι ίδια με αυτά της γραμμικής αναζήτησης τότε οι μέθοδοι αναζήτησης που χρησιμοποιούν το δέντρο θα είναι σωστές.

Παράδειγμα

΄Εστω ότι θέλουμε να κατασκευάσουμε ένα STR R-tree που να έχει τα 10 πρώτα ορθογώνια του αρχείου και έστω ότι το block size = 120bytes, που σημαίνει ότι μόνο 3 ζεύγη (object-id, MBR) μπορούν να χωρέσουν σε έναν κόμβο. Σε αυτή την περίπτωση, σχηματίζεται δέντρο με 4 φύλλα, 2 κόμβους στο επίπεδο 1 και 1 κόμβο στο επίπεδο 2 (ρίζα). Η STR τεχνική πρώτα ταξινομεί όλα τα ορθογώνια ως προς x-low, και μετά ανά δύο φύλλα με βάση το y-low. ΄Ετσι τα δύο πρώτα φύλλα είναι τα:

node-id=0 [(2, (0.254807, 0.254809, 0.131919, 0.131922)), (3, (0.254978, 0.25498, 0.132005, 0.132008)), (1, (0.255048, 0.255052, 0.132625, 0.132627))]

node-id=2 [(8, (0.254411, 0.254413, 0.132954, 0.132956)), (6, (0.254975, 0.254979, 0.132965, 0.132968)), (9, (0.254734, 0.254735, 0.133035, 0.133037))]

και τα δύο επόμενα τα:

node-id=3 [(0, (0.255158, 0.25517, 0.132399, 0.132426)), (7, (0.255256, 0.255258, 0.133137, 0.13314)), (5, (0.255117, 0.25512, 0.133244, 0.133246))]

node-id=4 [(4, (0.25522, 0.255222, 0.133289, 0.133292))]

Προσέξτε ότι το τελευταίο φύλλο έχει μόνο 1 ορθογώνιο, αλλά αυτό είναι οκ (γενικά στο δεξιότερο μονοπάτι του δέντρου μπορούν οι κόμβοι να έχουν λίγα στοιχεία. Τα node-ids δίνονται με τη σειρά που φτιάχνονται οι κόμβοι.

Όμοια, ο πρώτος κόμβος στο επίπεδο 1, ο οποίος δεικτοδοτεί τα 3 πρώτα φύλλα είναι ο:

node-id=1 [(0, [0.254807, 0.255052, 0.131919, 0.132627]), (2, [0.254411, 0.254979, 0.132954, 0.133037]), (3, [0.255117, 0.255258, 0.132399, 0.133246])]

και ο δεύτερος είναι ο

node-id=5 [(4, [0.25522, 0.255222, 0.133289, 0.133292])]

Τέλος η ρίζα είναι η

node-id=6 [(1, [0.254411, 0.255258, 0.131919, 0.133246]), (5, [0.25522, 0.255222, 0.133289, 0.133292])])

Παραδοτέα: Κάντε turnin στο assignment2@mye041 τα προγράμματά σας και ένα PDF αρχείο το οποίο τεκμηριώνει τα προγράμματα και περιέχει οδηγίες εκτέλεσης/χρήσης.

Προσοχή: τα προγράμματά σας πρέπει να τρέχουν, χωρίς εξαρτήσεις από μη στάνταρ βιβλιοθήκες. Ο κώδικάς σας θα πρέπει να είναι αυτόνομος και να τρέχει με τη minimal εγκατάσταση της γλώσσας προγραμματισμού. Αν χρησιμοποιείτε Python, μην υποβάλλετε κώδικα σε jupyter αλλά αρχεία .py και μη χρησιμοποιείτε μη στάνταρ βιβλιοθήκες (π.χ. pandas, numpy). Αν χρησιμοποιείτε java μην υποβάλετε κώδικα με πακέτα.