CEID

Πολυδιάστατες Δομές Δεδομένων

Range and Similarity Queries σε Σύνολα Κειμένων

Project 2022 - 2023

Αβραμόπουλος Μιχαήλ
ΑΜ: 1067451
Έτος: 5ο

Αργυροκαστρίτης Ιωάννης
ΑΜ: 1067459
Έτος: 5ο

Καραγιάννης Αλέξανδρος
ΑΜ: 1067452
Έτος: 5ο

Πανταζάρας Ευστράτιος
ΑΜ: 1067490
Έτος: 5ο

Πίνακας περιεχομένων

Εισαγωγή

Επιλέξαμε το Project-1: Range and Similarity Queries σε Σύνολα Κειμένων. Στο zip περιέχεται το εκτελέσιμο αρχείο main.exe, ένας φάκελος src με τον πηγαίο κώδικα σε Python, και η παρούσα αναφορά.

Demo Εκτέλεση

Το εκτελέσιμο αρχείο ονομάζεται main.exe. Μπορεί να εκτελεστεί απευθείας σε command line ως *./main.exe.* Στην ουσία δημιουργεί τα ψεύτικα δεδομένα εισόδου, τα εισάγει στο δέντρο της επιλογής μας, εκτελεί range query σε αυτό το δέντρο, εκτελεί LSH στην έξοδο του range query και τέλος κάνει print την έξοδο του LSH μορφοποιημένη.

Ζητά από τον χρήστη να εισάγει τις απαραίτητες παραμέτρους για να εκτελέσει το επιλεγμένο data structure. Η δημιουργία των δεδομένων εισόδου (όπως περιγράφουμε παρακάτω) γίνεται στην αρχή κάθε εκτέλεσης, και για εισόδους μεγαλύτερες του 200 χρειάζεται μερικά δευτερόλεπτα αναμονής.

Στο τέλος της εκτέλεσης ο χρήστης ερωτάται αν θέλει να τρέξει το αρχείο των πειραμάτων, που παράγει χρόνους εκτέλεσης για κάθε δέντρο (εξηγείται αναλυτικά παρακάτω).

Δημιουργία Δεδομένων (data.py)

Scientist class

Η βασική μονάδα των δεδομένων μας, ένα αντικείμενο που αναπαριστά έναν επιστήμονα και περιέχει 3 πεδία:

- surname: string επώνυμο του επιστήμονα
- awards: int αριθμός βραβείων που έχει αποσπάσει ο επιστήμονας
- education: vector of strings που αντιστοιχούν σε ονόματα πανεπιστημίων στα οποία έχει σπουδάσει ο επιστήμονας

generate_data() function

Μετά από έλεγχο στα links που παρέχονταν από την εκφώνηση για άντληση δεδομένων, αποφασίσαμε πως τα δεδομένα για επιστήμονες στη Wikipedia είναι ανοργάνωτα και δύσκολα στην εξαγωγή. Επειδή αυτό θα απαιτούσε πολύ χρόνο (και δεν είναι στους στόχους του μαθήματος), υλοποιήσαμε μια δική μας συνάρτηση (τη παρούσα) για δημιουργία ψεύτικων δεδομένων.

Η συνάρτηση δέχεται σαν είσοδο τον αριθμό των ψεύτικων αντικειμένων-επιστημόνων που θέλουμε να δημιουργήσει, και επιστρέφει ένα list που τους περιέχει. Για κάθε επιστήμονα, δημιουργούμε ένα αληθοφανές ψεύτικο surname με τη βοήθεια της βιβλιοθήκης names. Προσέχουμε αυτά τα ονόματα να είναι μοναδικά (βλ. Παραδοχές παρακάτω). Έπειτα ως αριθμό βραβείων επιλέγουμε ένα τυχαίο int από 0 ως 30. Τέλος, για το education vector επιλέγουμε τυχαία 5 ονόματα πανεπιστημίων από μια προκαθορισμένη λίστα 10 τέτοιων ονομάτων. Αυτό το τελευταίο “5 στα 10” είναι αρκετά αυθαίρετο, αλλά η εκφώνηση ζητά ποσοστό ομοιότητας >60% στην εκπαίδευση, και η πιθανότητα να συμβεί κάτι τέτοιο μικραίνει όσο περισσότερα πανεπιστήμια προσθέτουμε.

string indices

Η εκφώνηση ζητά να χρησιμοποιήσουμε σαν indices τα πεδία surname και awards των δεδομένων. Αυτό σημαίνει πως το ένα index είναι string (και το άλλο int). Σε μερικές δομές (πχ k-d tree) αυτό δεν προκαλεί σημαντικά προβλήματα, αλλά σε μερικές άλλες χρειάστηκε να υλοποιήσουμε ολόκληρο σύστημα για την εκτέλεση πράξεων με strings (πχ ημιάθροισμα).

Παραδοχές

- Τα επώνυμα στα δεδομένα μας είναι μοναδικά. Αυτό εγγυάται πως 2 σημεία στον χώρο ποτέ δεν ταυτίζονται, γιατί η ταύτιση είναι προβληματική στο Quad Tree (όσο κι αν υποδιαιρέσουμε, η ταύτιση δεν αίρεται.
- Ό,που ζητείται είσοδος χρήστη, δεν έχουμε προγραμματίσει αμυντικά γιατί εστιάσαμε κυρίως στο ζητούμενο του μαθήματος
- Επειδή (όπως θα αναλύσουμε παρακάτω) οι υλοποιήσεις μας χρειάζονται περίπλοκες πράξεις με strings, δουλεύουμε μόνο με lowercase λατινικούς χαρακτήρες
- Επειδή η κάθε δομή υλοποιήθηκε διαφορετική χρονική στιγμή και από διαφορετικό μέλος της ομάδας, δυστυχώς δεν υπάρχει ένα αυστηρό κοινό API. Παρ'όλα αυτά, προσπαθήσαμε η κάθε δομή έχει δικό της αρχείο με δική της class, και όλες κάνουν implement μια instance method "range_query()"

K-d Tree (kd.py)

Η κλάση KDTree αναπαριστά έναν κόμβο k-d tree. Ο κάθε κόμβος είναι ένα 3-tuple αποτελούμενος από το αριστερό παιδί, το δεξί παιδί και τα δεδομένα.

make() function

Υπεύθυνη για την δημιουργία του δέντρου. Έχουμε x = surname και y = awards και με το i ελέγχουμε το dimension του κάθε επιπέδου. Η διάσταση X είναι string, αλλά η python χειρίζεται τους τελεστές (<,>) λεξικογραφικά, οπότε το θέμα δεν περιπλέκεται. Κάνουμε sort τους επόμενους κόμβους σύμφωνα με το κλειδί που αντιστοιχεί στη διάσταση i και ανανεώνουμε το i για το επόμενο επίπεδο εναλλάσοντας την τιμή του i από 1 σε 0 και αντίστροφα. Τέλος κάνοντας right shift στην δυαδική τιμή του αριθμού των υπόλοιπων κόμβων κάνουμε διαίρεση δια 2 και κρατάμε το ακέραιο κομμάτι, δηλαδή παίρνουμε τον μέσο και εκτελούμε αναδρομικά την make σύμφωνα με τον μέσο.

range_query() function

Εκτελεί αναζήτηση σύμφωνα με το κλειστό διάστημα επωνύμων και βραβείων που θα του δώσουμε ([surname_min, surname_max] και [awards_min, awards_max]). Αρχικά ελέγχει τον root κόμβο και αν είναι μέσα στο διάστημα το προσθέτει στο αποτέλεσμα. Μετά ελέγχοντας το i, που όπως είπαμε πριν είναι η μεταβλητή για την αλλαγή του dimension, ελέγχει εάν λ.χ. το επώνυμο αν είναι μεγαλύτερο από το ελάχιστο για να ψάξει στο αριστερό παιδί και μετά αν είναι μικρότερο από το μέγιστο για να ελέγξει αναδρομικά στο δεξί παιδί. Η αναζήτηση εκτελείται αναδρομικά μέχρι να φτάσουμε στα φύλλα του δέντρου. Το ίδιο γίνεται και στην περίπτωση που το i είναι στην 2η διάσταση όπου εκεί ελέγχει τα βραβεία.

Quad Tree (quad.py)

Η κλάση Quad αναπαριστά έναν κόμβο του Quadtree. Αποθηκεύει τα όρια του τετραγώνου (bot_right, top_left), τα τέσσερα παιδιά του (top_left_tree, top_right_tree, bot_left_tree, bot_right_tree) και ένα αντικείμενο τύπου Node υπεύθυνο για τα δεδομένα.

Point class

Αναπαριστά μια 2D θέση με x,y. Επειδή το ένα index του δέντρου μας είναι string, επιλέγουμε το x να είναι αυτή η διάσταση.

Node class

Αναπαριστά το θεμελιώδες node του δέντρου, που έχει ένα index και ένα δεδομένο. Το index είναι αντικείμενο Point που δηλώνει τη θέση του σημείου.

insert() function

Εισάγει ένα νέο στοιχείο σε έναν κόμβο. Εάν ο κόμβος δεν είναι φύλλο, επιλέγει τον κόμβο - παιδί στον οποίο ταιριάζει το δεδομένο, και καλείται αναδρομικά σε αυτόν. Σε περίπτωση που ο κόμβος έχει ήδη στοιχείο, τον διαιρεί περαιτέρω. Αυτό σημαίνει πως δημιουργεί τα 4 άδεια παιδιά που μαζί συμπληρώνουν τον κόμβο - γονέα. Είναι προφανές πως οι συντεταγμένες του “χωρίσματος” είναι (x1+x2)/2, όμως το ένα από τα 2 indices μας είναι string. Για αυτό το λόγο ορίσαμε τη συνάρτηση ημιαθροίσματος string **str_average** (περιγράφεται αναλυτικά παρακάτω στις Βοηθητικές Συναρτήσεις).

mass_insert() function

Συνάρτηση που δέχεται ως είσοδο λίστα από δεδομένα και εκτελεί την insert για να τα προσθέσει στο δέντρο. Σε περίπτωση που η ρίζα δεν υπάρχει, την δημιουργεί και ορίζει σαν άκρα (όρια του μεγάλου τετραγώνου) τις μέγιστες τιμές κάθε index.

print() function

Βοηθητική συνάρτηση που χρησιμοποιήσαμε για να κάνουμε print το δέντρο για debugging λόγους.

is_leaf() function

Συνάρτηση που επιστρέφει boolean εάν ο κόμβος είναι φύλλο.

intersect() function

Συνάρτηση που επιστρέφει boolean εάν το τετράγωνο του κόμβου τέμνει το ορθογώνιο που δίνεται σαν όρισμα.

in_boundary() function

Συνάρτηση που επιστρέφει boolean εάν το τετράγωνο του κόμβου περιέχει το σημείο που δίνεται σαν είσοδο.

range_query() function

Εκτελεί αναζήτηση σύμφωνα με το κλειστό διάστημα επωνύμων και βραβείων που θα του δώσουμε ([surname_min, surname_max] και [awards_min, awards_max]). Ελέγχει ποιά από τα παιδιά του κόμβου τέμνουν το ορθογώνιο που ορίζεται από το query, και καλείται αναδρομικά σε αυτά. Αν ο κόμβος είναι φύλλο, εκτελεί έναν τελευταίο έλεγχο στα κριτήρια και προσθέτει το στοιχείο στη λίστα εξόδου.

Range Tree (range.py)

Η κλάση Range εκπροσωπεί έναν κόμβο μιας δομής Range Tree. Είναι στην ουσία ένα binary tree που κάθε κόμβος έχει ένα nested binary tree. Κάθε κόμβος περιέχει τα δεδομένα του (data), αριστερό και δεξί παιδί (left, right), διάσταση (dimension = 0 ή 1) και το nested δέντρο στη δεύτερη διάσταση (twod_subtree).

build_tree()

Συνάρτηση που καλείται όταν δημιουργείται το δέντρο ταξινομεί τα δεδομένα, βάζει το κεντρικό στοιχείο στον κόμβο και καλείται αναδρομικά για το αριστερό και δεξιό παιδί του κόμβου. Αν ο κόμβος είναι στην πρώτη διάσταση τότε δημιουργεί και το αντίστοιχο δέντρο στη δεύτερη διάσταση.

query_nodes()

Συνάρτηση που καλείται από τη range_query() και επιστρέφει όλους τους κόμβους που ταιριάζουν στο query.Αυτή η συνάντηση λειτουργεί μόνο στη μία διάσταση και για αυτό δεν αρκεί από μόνη της.

range_query()

Καλεί την query_nodes() στην πρώτη διάσταση και για κάθε αποτέλεσμα την ξανακαλεί στη δεύτερη διάσταση. Προτού προσθέσει ένα αποτέλεσμα στην λίστα εξόδου ελέγχει πάλι τα κριτήρια.

R-Tree (rtree.py)

Η κλάση RTree εκπροσωπεί έναν κόμβο μιας δομής R-Tree. Κάθε κόμβος περιέχει τα δεδομένα του (data), μια λίστα με το πολύ 3 παιδιά (children) και ένα minimum bounding rectangle που υλοποιείται ως αντικείμενο της παρακάτω κλάσης:

MBR class

Βοηθητική class που εκφράζει το Minimum Bounding Rectangle για ένα R-Tree Node. Επειδή το ένα index μας είναι string, η μία διάσταση του παραλληλογράμμου είναι και αυτή string. Όπως θα δούμε αυτό περιπλέκει την έννοια του εμβαδού.

contains() function

Δέχεται σαν είσοδο ένα σημείο, και επιστρέφει boolean εάν το σημείο περιέχεται στο MBR.

intersects() function

Δέχεται σαν είσοδο ένα άλλο MBR και επιστρέφει boolean εάν τα 2 MBR τέμνονται.

extend() function

Δέχεται σαν είσοδο ένα σημείο, και επιστρέφει ένα νέο MBR, το οποίο είναι μια επέκταση του τρέχοντος MBR ώστε να χωράει το νέο σημείο. Εάν το νέο σημείο περιέχεται ήδη στο τρέχον MBR, η επέκταση στην ουσία επιστρέφει πάλι το τρέχον MBR (δε χρειάζεται επέκταση).

calc_area() function

Υπολογίζει το εμβαδόν του τρέχοντος MBR. Επειδή η μια διάσταση είναι string, χρειάζεται να υπολογίσουμε στην ουσία μια αφαίρεση της μορφής string1-string2. Αυτό γίνεται με την βοηθητική συνάρτηση **str_diff** (βλ. Βοηθητικές Συναρτήσεις παρακάτω).

area_increase() function

Δέχεται σαν είσοδο ένα σημείο και υπολογίζει την αύξηση σε εμβαδόν που απαιτείται ώστε να προστεθεί το νέο σημείο στο παρόν MBR. Σε περίπτωση που το σημείο περιέχεται ήδη στο MBR, επιστρέφει 0. Σε άλλη περίπτωση κάνει **extend** το παρόν MBR και αφαιρεί το παλιό εμβαδόν από το εμβαδόν του extended.

insert() function

Συνάρτηση του R-Tree που εισάγει ένα νέο δεδομένο στο δέντρο σαν κόμβο. Όταν καλείται να επιλέξει σε ποιό παιδί του κόμβου να κάνει το insert, συγκρίνει τις εν δυνάμει αυξήσεις σε εμβαδόν, και επιλέγει την ελάχιστη (με βάση την επόμενη συνάρτηση). Δεδομένα διατηρούνται μόνο στους κόμβους-φύλλα.

find_min_child() function

Συνάρτηση (στατική, εκτός της κλάσης) που δέχεται σαν είσοδο μια λίστα από κόμβους και ένα νέο δεδομένο, και αποφασίζει ποιός κόμβος έχει την ελάχιστη αύξηση εμβαδού ώστε να γίνει σε αυτόν την εισαγωγή.

mass_insert() function

Πολύ απλή συνάρτηση που εκτελεί insert για το τρέχον δέντρο σε βρόχο για λίστα δεδομένων.

is_leaf() function

Απλή συνάρτηση που ελέγχει αν ο τρέχον κόμβος είναι φύλο του δέντρου

range_query() function

Συνάρτηση που εκτελεί αναζήτηση σε εύρος αποδεκτών τιμών και για τα 2 indices, με βάση τις εισόδους surname_min/max, awards_min/max. Στην αρχή κατασκευάζει το MBR που αντιστοιχεί στο range query. Σε κάθε βήμα ελέγχει αναδρομικά ποιά από τα παιδιά του node τέμνουν το MBR του query, και εκτελείται αναδρομικά μόνο σε αυτά. Όταν φτάσει σε επίπεδο φύλλου, ελέγχει μια τελευταία φορά τη συνθήκη, και αν ισχύει επιστρέφει το δεδομένο.

LSH (lsh.py)

Το αρχείο lsh.py είναι ένα αρχείο με στατικές συναρτήσεις που στην ουσία υλοποιούν το LSH μας. Καλούμαστε να εφαρμόσουμε LSH στο τρίτο πεδίο των δεδομένων μας, την εκπαίδευση, που είναι string array. Θεωρήσαμε πως τα ονόματα των πανεπιστημίων εκπαίδευσης είναι σταθερά, και άρα η ομοιότητα ανάμεσα σε 2 vectors είναι στην ουσία ο αριθμός των κοινών στοιχείων προς τον συνολικό αριθμό στοιχείων.

Επίσης όταν τα αποτελέσματα ενός range query είναι πολλά, το “ομοιότητα 60%” δεν έχει ξεκάθαρη σημασία - τί σημαίνει 25 vectors να μοιάζουν κατά 60%? Η συνάρτηση lsh() παρακάτω εξηγεί την προσέγγισή μας.

lsh() function

Συνάρτηση που δέχεται σαν είσοδο μια λίστα και επιστρέφει σαν έξοδο μια υπολίστα στην οποία ανά 2 όλα τα στοιχεία έχουν ομοιότητα 60% (αλλά μπορεί να μην είναι η μέγιστη υπολίστα). Λειτουργεί ως εξής: βάζουμε τυφλά το πρώτο στοιχείο της εισόδου στην έξοδο. Από εκεί και έπειτα, προσθέτουμε από την είσοδο στοιχεία στην έξοδο μόνο αν η ομοιότητά τους με κάθε στοιχείο της υπάρχουσας εξόδου είναι πάνω από 60%. Επίσης το threshold δεν είναι όντως 60%, αλλά είναι input στη συνάρτηση.

lsh_ratio() function

Επιστρέφει το ποσοστό ομοιότητας 2 vectors, εκτελώντας τη διαίρεση (αριθμός ίδιων στοιχείων/ συνολικός αριθμός στοιχείων)

count_equal_elements() function

Δέχεται σαν είσοδο 2 vectors και μετρά τον αριθμό των κοινών στοιχείων που υπάρχουν.

Βοηθητικές Συναρτήσεις (str_utils.py)

Όπως έχουμε αναφέρει ήδη, οι υλοποιήσεις των δέντρων μας χρησιμοποιούν μερικά string indices, και έτσι χρειάστηκε να υλοποιήσουμε περίπλοκες πράξεις μεταξύ strings, κυρίως το ημιάθροισμα και την αφαίρεση. Έχουμε ορίσει ως αλφάβητο ένα string που έχει με τη σειρά όλα τα λατινικά γράμματα και το κενό (white space) στην αρχή.

str_average() function

Δέχεται ως είσοδο 2 strings. Αρχικά τα κάνει pad ώστε να έχουν ίδιο μήκος. Μετατρέπει στην ουσία τα 2 strings σε νούμερα του 27αδικου συστήματος (27 = 26

γραμματα της αλφαβητου + 1 κενο). Εκτελεί στα νούμερα το ημιάθροισμα, και κάνει την αντίστροφη μετατροπή για να επιστρέψει ένα string. Αυτό το string δεν έχει νόημα για κάποιον χρήστη, αλλά χρησιμοποιείται για λεξικογραφικές συγκρίσεις κατά την προσπέλαση του δέντρου.

str_diff() function

Δέχεται ως είσοδο 2 strings. Αρχικά τα κάνει pad ώστε να έχουν ίδιο μήκος. Μετατρέπει στην ουσία τα 2 strings σε νούμερα του 27αδικου συστήματος (27 = 26 γραμματα της αλφαβητου + 1 κενο). Εκτελεί στα νούμερα την αφαίρεση, και επιστρέφει τη ρίζα του αποτελέσματος (νούμερο). Η ρίζα είναι μονότονη συνάρτηση που “μικραίνει” την έξοδο μας, για να αποφύγουμε πιθανά number overflows.

pad() function

Συνάρτηση που μεγαλώνει το string εισόδου ώστε να φτάσει στο μήκος που ζητήθηκε.

debase() function

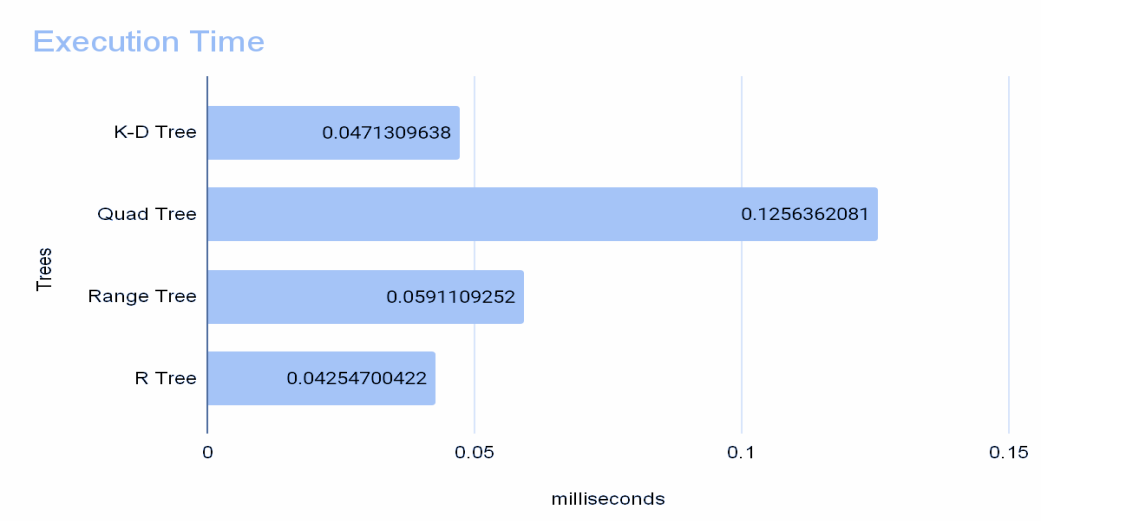
Μετατρέπει ένα string σε 27αδικό νούμερο.

enbase() function

Μετατρέπει ένα 27αδικό νούμερο σε string

Σύγκριση - Πειράματα (experiments.py)

Προκειμένου να συγκρίνουμε πειραματικά τις 4 μεθόδους, εκτελέσαμε ένα σταθερό range query ([‘bell’, ‘erdos’], [7, 12]) με μέγεθος εισόδου 1000 επί 10 φορές για κάθε δομή, και εξάγαμε την μέση καθυστέρηση. Η 10πλή εκτέλεση οφείλεται στην τυχαιότητα των δεδομένων, θέλουμε να αποκλείσουμε το ενδεχόμενο κάποια δομή να επωφεληθεί από την κατανομή των δεδομένων. Το κάθε query εκτελέστηκε 10.000 φορές στα ίδια δεδομένα ώστε να μετρηθεί αξιόλογο χρονικό διάστημα, και μετά πήραμε μέσο όρο. Τα χρονικά αποτελέσματα ήταν τα εξής:



Συμπεράσματα

Παρατηρούμε πως όλες οι δομές είχαν σχετικά παρόμοια απόδοση, με εξαίρεση το quad tree που είχε διπλάσιο χρόνο. Αυτό είναι λογικό, επειδή οι 2 διαστάσεις μας είναι ασύμμετρα κβαντισμένες: Τα string surnames έχουν μεγάλο εύρος τιμών, ενώ τα integer number of awards είναι μόνο 0-30. Αυτό σημαίνει πως όταν χωρίζουμε ένα τεταρτημόριο του quad tree στα 4, το κάνουμε κυρίως για τα surnames και όχι για τα awards (τα awards είναι 30~2^5). Με άλλα λόγια, σε βάθος>5, ο διαχωρισμός δεν έχει νόημα για την μια διάστασή μας.

Το r-tree είχε την καλύτερη απόδοση, ακόμη και όταν χρησιμοποιούσε μερικές από τις ακριβότερες (χρονικά) συναρτήσεις μας (εξηγήσαμε παραπάνω για πράξεις με strings). Αυτό είναι λογικό, αφού το r-tree είναι μια πολύ ευρέως χρησιμοποιούμενη δομή που προσαρμόζεται στα δεδομένα (τα bounding rectangles δημιουργούνται με βάση τα data points μας).

Το k-d tree στην ουσία επηρεάζεται από παρόμοιο πρόβλημα με το quad tree, αλλά η απλότητα της δομής και η binary φύση του κρατούν το χρόνο χαμηλό.

Τέλος το range tree παρότι επιτυγχάνει σχετικά καλό χρόνο, είναι εγγενώς πολυπλοκότερο από άλλες εναλλακτικές αφού εκτελεί στην ουσία 2 διαφορετικά range queries, ένα για κάθε διάσταση.