ΑΜ : 2017030021 ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΙΩΑΝΝΗΣ-ΙΑΣΩΝ ΓΕΩΡΓΑΚΑΣ

ΑΝΑΦΟΡΑ 2ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

ΥΛΟΠΟΙΗΣΗ ΑΣΚΗΣΗΣ

Για την υλοπιήση της 2ης εργαστηριακης άσκησης δημιούργησα τρία πακετα, ένα για την υλοποιήση του Kd-Tree, ένα για το pr-QuadTree και ένα για το εκτελέσιμο αρχείο τη main. Στη κλάση main επιδεικνύω τη λειτουργία τόσο του kd όσο και του pr, εισάγωντας και στις δύο δομές 1.000,10.000,30.000,50.000, 70.000,100.000 με τη σειρά και στη συνέχεια αναζητώ 100 στοιχεία που υπάρχουν(επιτυχής αναζήτηση) και 100 που δεν υπαρχούν(ανεπιτυχής αναζήτηση). Στο project η έκδοση του eclipse που χρησιμοποιήσα ειναι jdk 1.8.0_161.

1ο Μέρος : Δημιουργία Kd-Tree

Για την κατασκευή του Kd-Tree δημιούργησα μια κλάση NodeKdTree και μία KdTree. Στη κλάση Kd-Tree δημιούργησα δύο βασικές μεθόδους την insertElement() και την searchElement. Ειδικότερα η μέθοδος εισαγωγής εισάγει το κάθε στοιχείο συγκρίνοντας με τη τιμή του αντίστοιχου κόμβου και αν είναι μικρότερη τοποθετείται στο αριστερό παιδί του κομβου αλλίως στο δεξί παιδί του δένδρου. Η σύγκριση γίνεται είτε με την χ μεταβλήτη του κόμβου είτε με τη y ανάλογα σε ποιοι επίπεδο είμαστε δηλαδή αρχίζουμε απο την ρίζα που συγκρίνουμε με βαση το x και στη συνέχεια συγκρίνουμε εναλλάξ τις τιμές. Η μέθοδος αναζήτησης έχει ίδια λογική δηλαδη για να βρούμε αν ένα στοιχείο υπάρχει η όχι στο δένδρο συγκρίνουμε με βάση την αντίστοιχη τιμή ανάλογα στο επίπεδο που είμαστε οπως και προηγουμένως. Αν το βρεί επιστρέφει επιτυχία αν όχι αποτυχία.

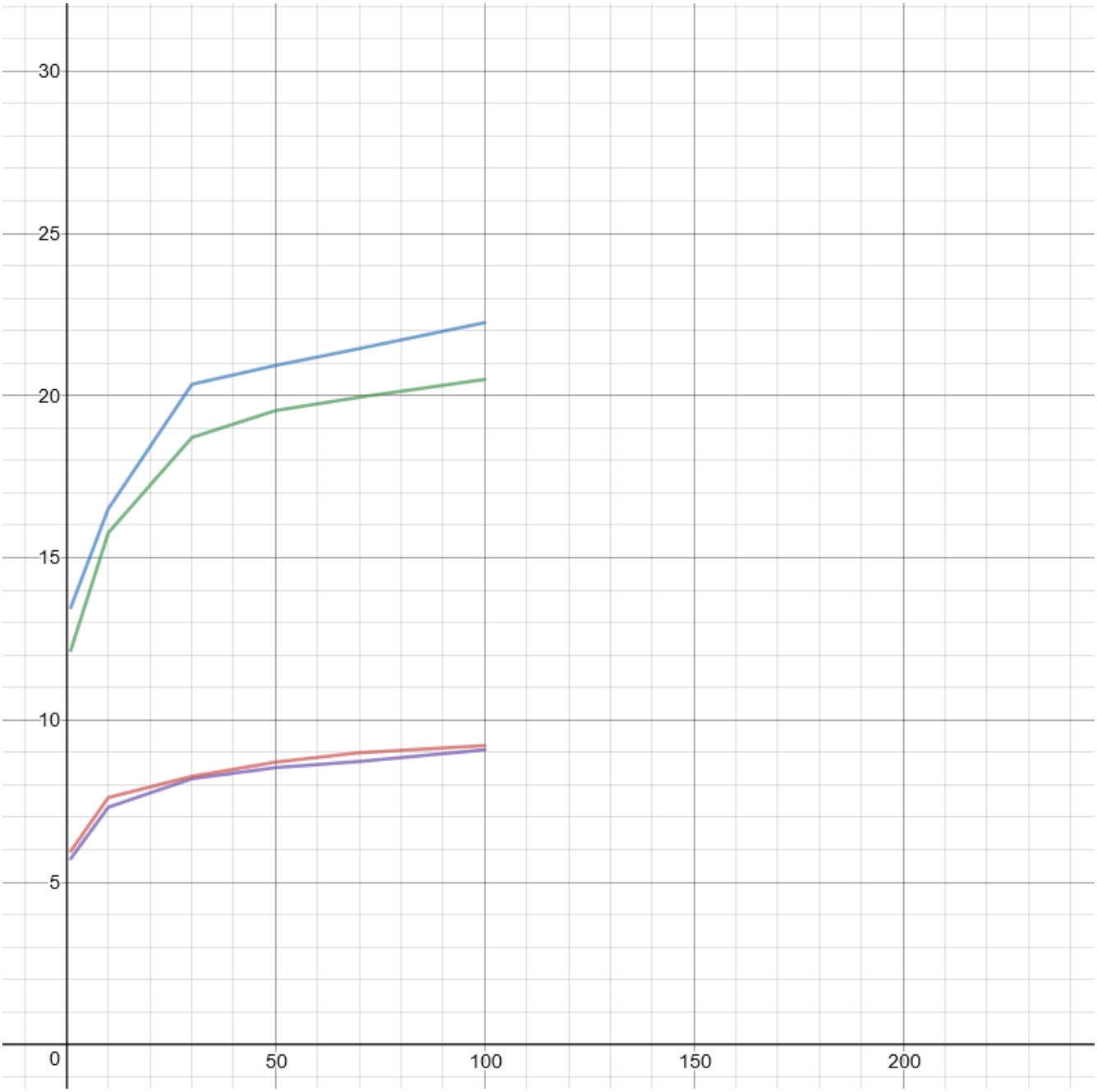
2ο Μέρος : Δημιουργία Pr-QuadTree

Για την κατασκευή του Pr-QuadTree δημιούργησα μια κλάση NodePrQuadTree και μία PrQuadTree. Στη κλάση PrQuadTree δημιούργησα δύο βασικές μεθόδους την insertElement και την searchElement.
Πιο συγκεκριμένα η μέθοδος εισαγωγής εισάγει κάθε στοιχείο στο κόμβο βλέποντας αν είναι φύλλο η όχι αν δεν είναι εισάγει κανονικά το στοιχείο στο σωστό φύλλο αν έιναι τότε υποδιαιρει το φύλλο σε τέσσερις ίσες περιοχές και το τοποθετεί κατάλληλα και ταυτόχρονα είσαγει και το στοιχείο του φύλλου σε ένα απο τα καινούργια φύλλα έτσι ώστε να μην υπάρχει στοιχείο σε κανέναν εσωτερικό κόμβο . Στην μέθοδο αναζήτησης ακολουθώ την ίδια λογική με τη διαφορά ότι δεν δημιουργώ φύλλο ή κόμβο αλλα απλά υποδιαιρώ σε τέσσερις ίσες περιοχές τον εκάστοτε κόμβο μέχρι να το βρώ, αν το βρώ επιστρέφω επιτυχία αλλιώς αποτυχία.

3ο Μέρος: Συγκρίσεις-Εκτιμίσεις απόδοσης

Για να συγκρίνω και να εκτιμήσω την απόδοση των δύο δομών δεδομένων δημιούργησα στο πακέτο main ένα εκτελέσιμο αρχείο(κλάση main).Αρχικά δημιουργώ ένα instance του Kd-Tree και του Pr-QuadTree,παράγω με την συνάρτηση generateNumbersForInsertion Μ τυχαίους αριθμούς .Ταυτόχρονα τα αποθηκεύω σε δύο πίνακες μεγέθους Μ έτσι ώστε να τα χρησιμοποιήσω στην επιτυχή αναζήτηση αλλα και στη παραγωγή τιμών που δεν υπάρχουν για να έχουμε αποτυχή αναζήτηση.Ακόμη παράγω αριθμούς που δεν υπάρχουν με την μέθοδο generateNumbersForUnsuccessSearch και τους αποθηκεύω σε δύο πίνακες μεγέθους Μ .Στη συνέχεια εισάγω στα δύο δέντρα τους αριθμούς ,που βρίσκονται στους αντιστοιχους πίνακες που τα έχω αποθηκεύσει, με την αντίστοιχη συνάρτηση insertElementsInKdTree και insertElementsInQuadTree.Μετά κάνω 100 επιτυχημένες αναζητήσεις και 100 αποτυχημένες αναζητήσεις για κάθε δομή αντλώντας τα στοιχεία απο τον αντίστοιχο πίνακα και χρησιμοποιώντας τις αντίστοιχες συναρτήσεις searchElementInKdTree και searchElementInQuadTree.
Τέλος όλο ο κώδικας που περιγράφτηκε περιβάλεται από ένα for loop που εκτελείται 6 φορές κάθε φορά για διαφορετικό αριθμό δεδομένων Μ = 1.000,10.000,30.000,50.000,70.000,100.000 .

Data	KdTree Success Search	KdTree Unsuccess Search	Pr-QuadTree Success Search	Pr-QuadTree Unsuccess Search
1000	12.14	13.46	5.95	5.72
10000	15.77	16.51	7.61	7.31
30000	18.71	20.35	8.26	8.19
50000	19.54	20.93	8.7	8.53
70000	19.95	21.45	8.99	8.72
100000	20.5	22.25	9.21	9.08



4ο Μέρος : Τεκμηρίωση αποτελεσμάτων

Για το Kd-Tree η απόδοση της επιτυχημένης αναζήτησης κυμαίνεται απο 12-20 συγκρίσεις για την εύρεση του στοιχείου και απο 13-22 για την αποτυχή αναζήτηση του στοιχείου που είναι λογικό καθώς θέλει λιγότερες συγκρίσεις για να βρεί ένα στοιχείο που υπάρχει απο ότι για να δηλώσει ότι το στοιχείο που ψάχνουμε δεν υπάρχει αφού είναι ένα δυαδικο δένδρο δηλαδή κάθε κόμβος έχει δύο παιδιά.Για το Pr-QuadTree η απόδοση της επιτυχημένης αναζήτησης κυμαίνεται απο 5-9 συγκρίσεις για την εύρεση του στοιχείου που ψάχνουμε ενώ για την αποτυχημένη αναζήτηση κυμαίνεται και αυτη απο 5-9 συγκρίσεις δηλαδή θέλουν τον ίδιο χρόνο με μικρές διαφορές της τάξης του 0.03. Αυτό ειναι λογικό διότι το quadTree έχει τέσσερα παιδιά και δεν περιέχει τιμη σε κανένα ενδιάμεσο κόμβο παρά μόνο στα φύλλα του.

Η έκφραση πολυπλοκότητας που εκφράζει τις παραπάνω καμπύλες είναι για το Kd-Tree logN με βάση το 2 και N είναι τα δεδομένα που περιέχει το δένδρο και για το Pr-QuadTree είναι logN με βάση 4 και το N είναι τα δεδομένα που περιέχει το δένδρο.Η διαφορετική βάση στους λογαρίθμους των δύο δένδρων δικαιολογείται απο το γεγονός ότι το Kd-Tree έχει δύο παιδία σε κάθε κόμβο ενώ το Pr-QuadTree έχει τέσσερα παιδιά σε κάθε κόμβο.

Συγκρίνοντας τις δύο δομές καταλαβαίνουμε ότι η πιο γρήγορη δομή δεδομένων από τις δύο προαναφερθείσες είναι το Pr-QuadTree καθώς για τον ίδιο αριθμό στοιχείων στο δένδρο έχουμε λιγότερες συγκρίσεις τόσο στην επιτυχημένη όσο και στην αποτυχημένη αναζήτηση ενός στοιχείου.Αυτο είναι απόλυτα λογικό λόγω της πολυπλοκότητας τους και οτι το pr-QuadTree έχει περισσότερα παιδία από το KdTree.

Τέλος η απόδοση της επιτυχημένης και της αποτυχημένης αναζήτησης στο Pr-QuadTree η οποία κυμαίνεται από 5-9 είναι μέσα στα πλαίσια που θα έπρεπε καθώς το log με βάση του δύο είναι περίπου 4.9 για 1000 στοιχεία και 8.3 για 100000 στοιχεία. Ενώ για την επιτυχημένη και την αποτυχημένη στο Kd-Tree η οποία κυμαίνεται απο 12-20 και 13-22 αντίστοιχα είναι και αυτή στα όρια που θα έπρεπε καθώς το log με βάση το 4 είναι περίπου 9.9 για 1000 στοιχεία και 16.6 για 100000 στοιχεία.

Σημείωση :

Τη κλίμακα στο διάγραμμα την έχω βάλει απο 1-100 δηλαδή διαίρεσα με το 1000 τη κλίμακα των δεδομένων για να έχω καλύτερη γραφική απεικόνιση.Ακόμη στο διάγραμμα με τις γραφικές παραστάσεις, η κόκκινη αντιστοιχεί στην επιτυχημενη αναζήτηση στο Pr-QuadTree , η μπλε στην αποτυχημένη αναζήτηση του KdTree, η μωβ στην αποτυχημένη του Pr-QuadTree και η πράσινη στην επιτυχημένη αναζήτηση του KdTree.