ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ

2η Άσκηση

Λαμπράκης Μιχάλης 2020030077

1)Μετρήσεις

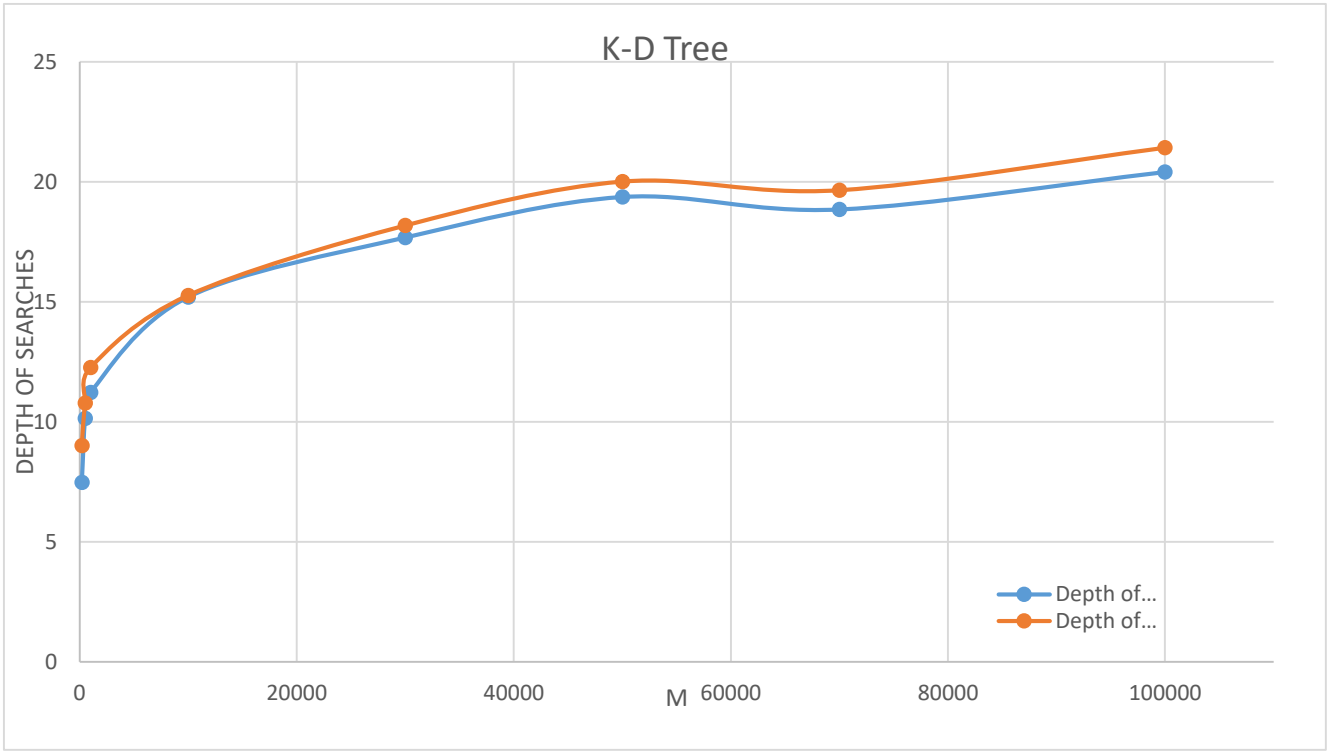
Αρχικά, έχοντας υλοποιήσει σε Java τα δύο ζητούμενα δέντρα και κάνοντας τις ζητούμενες μετρήσεις καταγράφτηκαν τα εξής αποτελέσματα:

K-D Tree		
M	**Depth of True Searches**	**Depth of False Searches**
200	7.48	9.01
500	10.15	10.78
1000	11.23	12.26
10000	15.2	15.27
30000	17.68	18.18
50000	19.37	20.01
70000	18.85	19.65
100000	20.41	21.42

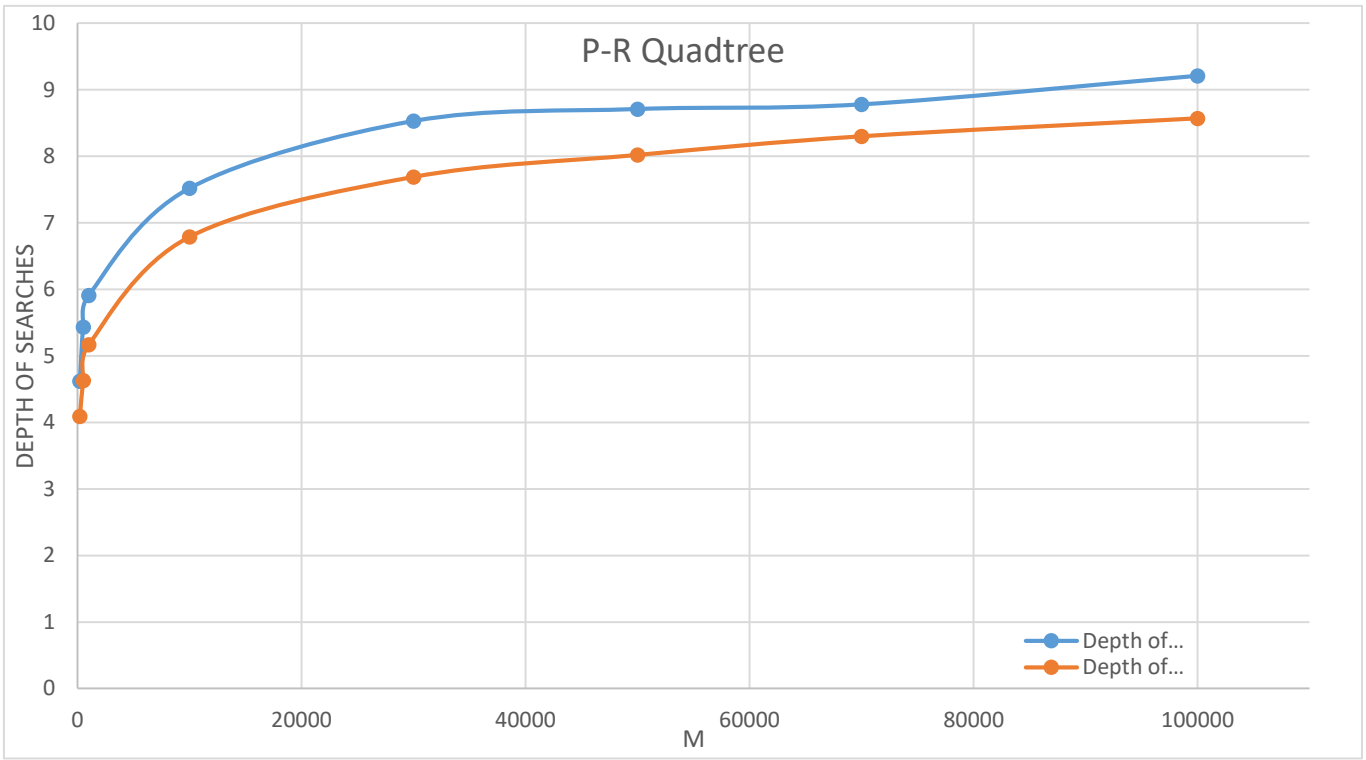
P-R Quadtree		
M	**Depth of True Searches**	**Depth of False Searches**
200	4.62	4.09
500	5.43	4.63
1000	5.91	5.17
10000	7.52	6.79
30000	8.53	7.69
50000	8.71	8.02
70000	8.78	8.3
100000	9.21	8.57

2)Διαγράμματα

Από τα παραπάνω δεδομένα σχεδιάστηκαν τα διαγράμματα στο Excel χρησιμοποιώντας “Scatter Chart” διαγράμματα για να επιτευχθεί μια καθαρή απεικόνιση των αποτελεσμάτων.



Εικόνα 2.1: Μέσος όρος βάθους δέντρου για αναζητήσεις σε **K-D Tree**.



Εικόνα 2.3:Μέσος όρος βάθους δέντρου για αναζητήσεις σε **P-R Quadtree**.

3)Σχόλια & Παρατηρήσεις

Ξεκινώντας με το K-D Tree παρατηρούμε ότι οι καμπύλες επιτυχημένης και αποτυχημένης αναζήτησης έχουν την ίδια μορφή. Το βάθος του δέντρου σε μια επιτυχής αναζήτηση είναι λίγο μικρότερο από μια αποτυχημένη και αυτό συμβαίνει διότι ο αλγόριθμος αναζήτησης εκμεταλλεύεται τη δομή του δέντρου για να φτάσει στο επιθυμητό σημείο. Σε μια αποτυχημένη αναζήτηση μπορεί να απαιτείται εξερεύνηση μεγαλύτερου τμήματος του δέντρου. Στο P-R Quadtree οι επιτυχημένες αναζητήσεις έχουν μεγαλύτερο βάθος με πολύ μικρή απόκλιση και αυτό οφείλεται στο ότι τα δεδομένα βρίσκονται στα φύλλα του δέντρου, οπότε η πρόσβαση σε αυτά απαιτεί να φτάσει ο αλγόριθμος επιτυχημένης σε μεγαλύτερο βάθος σε σχέση με αυτόν της αποτυχημένης .

Η έκφραση πολυπλοκότητας που εκφράζει τις καμπύλες είναι λογαριθμική (**O(logn)**) γιατί και στις δυο περιπτώσεις ο αλγόριθμος χωρίζει αναδρομικά τον χώρο, οπότε κατά την αναζήτηση ή την εισαγωγή στοιχείου απορρίπτουμε ένα κομμάτι του δέντρο.

Το P-R Quadtree είναι γρηγορότερο διότι σε κάθε αναδρομική κλίση της συνάρτησης μειώνεται το δέντρο κατά τα 3 / 4 (αφού ο κάθε κόσμος έχει 4 ΄παιδιά΄). Αυτό αυξάνει σημαντικά και την ταχύτητα αναζήτησης στο δέντρο.

Συγκρίνοντας τις γραφικές παραστάσεις συμπεραίνουμε ότι το P-R Quadtree έχει πολύ μικρότερο μέσο βάθος από το K-D Tree και άρα πολύ καλύτερη απόδοση. Αυτό συμβαίνει διότι με κάθε αναδρομή ο αλγόριθμος χωρίζει κάθε ενδιάμεσο κόμβο σε 4 υπό-δέντρα, σε αντίθεση με το K-D Tree που χωρίζεται σε 2.

4)Επεξήγηση κώδικα / κλάσεις και μέθοδοι

Ο Κώδικας περιέχει 6 κλάσεις.

KDnode: Κλάση αναπαράστασης ενός κόμβου στο K-D Tree. Περιέχει τις συντεταγμένες x και y καθώς και 2 μεταβλητές KDnode που αναπαριστούν το δεξί και αριστερό υπό-δέντρο του K-D Tree.

KDTree: Κλάση υλοποίησης της δομής K-D Tree. Περιέχει συνάρτηση εισαγωγής στοιχείου στο δέντρο καθώς και αναζήτησης σε αυτό.

- insertNode: Συνάρτηση εισαγωγής στοιχείου στο δέντρο. Αυτό που συμβαίνει είναι ότι η συνάρτηση συγκρίνει τον καινούργιο κόμβο με τον τρέχοντα κόμβο βασισμένη στην boolean μεταβλητή 'compVar'. Εάν η μεταβλητή είναι αληθής, συγκρίνει την 'x' συντεταγμένη του καινούργιου κόμβου με την 'x' του τρέχοντα και αν είναι μεγαλύτερη, εισάγει τον καινούργιο κόμβο στο δεξί υπό-δέντρο του τρέχοντα κόμβου αλλιώς στο αριστερό (στο επόμενο επίπεδο αναδρομής η σύγκριση γίνεται με βάση την 'y' συντεταγμένη). Εάν η μεταβλητή 'compVar' είναι ψευδής η σύγκριση των κόμβων γίνεται με βάση την 'y' συντεταγμένη και ακολουθείται η ίδια διαδικασία εισαγωγής. Η συνάρτηση επιστρέφει την ρίζα του τροποποιημένου K-D Tree.

- searchWithDepth: Συνάρτηση αναζήτησης κόμβου στο δέντρο. Πραγματοποιείται η σύγκριση του κόμβου αναζήτησης με τον τρέχοντα κόμβο με βάση την boolean μεταβλητή 'compVar'. Εάν είναι αληθής η σύγκριση γίνεται με βάση την 'x' συντεταγμένη των δύο κόμβων και ανάλογα με το εάν η ζητούμενη τιμή είναι μεγαλύτερη ή μικρότερη, η αναζήτηση συνεχίζεται αναδρομικά αντίστοιχα στο δεξί ή αριστερό υπό-δέντρο του τρέχοντα κόμβου (στο επόμενο επίπεδο αναδρομής η σύγκριση γίνεται με βάση την 'y' συντεταγμένη). Σε κάθε επίπεδο αναδρομής η μεταβλητή 'depth' αυξάνεται κατά 1. Εάν βρεθεί ο ζητούμενος κόμβος η συνάρτηση επιστρέφει το βάθος και το προσθέτει στην global μεταβλητή 'totalDepthOfTrueSearches' . Αντίστοιχα εάν ο τρέχοντας κόμβος είναι 'null' σημαίνει ότι ο αλγόριθμος διάσχισε όλο το δέντρο και δεν βρήκε το ζητούμενο σημείο, οπότε επιστρέφει το βάθος της αναδρομής και το προσθέτει στην global μεταβλητή 'totalDepthOfFalseSearches'. Οι global αυτές μεταβλητές αποθηκεύουν το συνολικό βάθος αναδρομής του αλγόριθμου για έναν αριθμό αναζητήσεων ώστε να υπολογιστεί ο μέσος όρος.

Qnode: Κλάση αναπαράστασης ενός κόμβου στο P-R Quadtree. Περιέχει τις συντεταγμένες ‘x’ και ‘y’ , τα όρια της περιοχής που καλύπτει ο κόμβος καθώς και 4 μεταβλητές Qnode που αναπαριστούν τα 4 υπό-δέντρα του P-R Quadtree. Περιέχει και 2 boolean μεταβλητές οι οποίες μας δείχνουν εάν ο κόμβος είναι φύλλο του δέντρο ή άμα έχει διαιρεθεί σε υπό-δέντρα.

Qtree: Κλάση υλοποίησης του P-R Quadtree. Περιέχει τις εξής μεθόδους.

- Qtree: Κατασκευαστής της κλάσης, δημιουργεί την ρίζα του δέντρο και την αρχικοποιεί. Χαρακτηριστικό του Quadtree είναι ότι οι τιμές βρίσκονται μόνο στα φύλλα του δέντρου και όχι στους εσωτερικούς κόμβους, οπότε σε κάθε εσωτερικό κόμβο ορίζουμε τις τιμές των ‘x’ και ‘y’ να είναι ίσες με -1.

- devideRect: Συνάρτηση που διαιρεί το παραλληλόγραμμο (στον δισδιάστατο χώρο) που αντιπροσωπεύει ο κόμβος, που δέχεται η συνάρτηση ως είσοδο, σε 4 τεταρτημόρια και τα αναθέτει στα σωστά ‘παιδία’ του κόμβου αυτού.

- inBounds: Ελέγχει εάν το σημείο που δίνεται είναι μέσα στα όρια του κόμβου. Εάν είναι επιστρέφει true.

- insert: Κλάση εισαγωγής κόμβου στο Quadtree. Αρχικά ελέγχουμε εάν το σημείο βρίσκεται ήδη μέσα στο δέντρο και εάν αυτό υπάρχει η σηνάρτηση επιστρέφει χωρίς να κάνει τίποτα. Εάν δεν βρίσκεται στο δέντρο, ελέγχουμε αν ο τρέχοντα κόμβος είναι φύλλο του δέντρου και εάν το σημείο είναι μέσα στα όρια του κόμβου. Εάν οι συνθήκες είναι αληθείς εισάγουμε το σημείο στον κόμβο και θέτουμε την τιμή false στην μεταβλητή ‘leaf’. Εάν ο τρέχοντας κόμβος δεν είναι φύλλο του δέντρο και το σημείο είναι μέσα στα όρια του κόμβου, η συνάρτηση ελέγχει εάν αυτός ο κόμβος έχει διαιρεθεί (μέσο της μεταβλητής ‘devided’). Εάν δεν έχει διαιρεθεί τον χωρίζει σε 4 υπό-κόμβους καλώντας την συνάρτηση ‘devideRec’ , διαφορετικά η συνάρτηση ελέγχει σε ποιο από τα όρια των ‘παιδιών’ βρίσκεται το σημείο και αναδρομικά καλεί την συνάρτηση insert στο νέο υπό-δέντρο.

- searchWithDepth: Συνάρτηση αναζήτησης κόμβου στο δέντρο. Αρχικά η συνάρτηση ελέγχει άμα το σημείο βρίσκεται μέσα στα όρια του τρέχοντα κόμβου και αν τα σημεία ‘x’ και ‘y’ είναι ίσα επιστρέφει το βάθος και το προσθέτει στην global μεταβλητή ‘totalDepthOfTrueSearches’. Αλλιώς η

συνάρτηση αναδρομικά καλώντας τον εαυτό της ελέγχει κάθε επόμενο επίπεδο μέχρι να φτάσουμε στα φύλλα του δέντρου. Εάν δεν βρεθεί το σημείο επιστρέφει το βάθος της αναδρομής και το προσθέτει στην global μεταβλητή ‘totalDepthOfFalseSearches’. Οι global αυτές μεταβλητές αποθηκεύουν το συνολικό βάθος αναδρομής του αλγόριθμου για έναν αριθμό αναζητήσεων ώστε να υπολογιστεί ο μέσος όρος.

RandomGenerator: Παράγει τυχαίους ακεραίους και τυχαίες συμβολοσειρές.

- getAlphaNumericString: Παραγωγή τυχαίου αλφαριθμητικού.
- getUniqueRandomKey: Παραγωγή ενός array από τυχαίους ακεραίους χωρίς να επαναλαμβάνεται κάποιος από αυτούς.
- getRandomKey: Παραγωγή ενός array από τυχαίους (επιτρέπεται η επανάληψη).

Main: Κλάση που δημιουργεί τα δέντρα, εισάγει τα τυχαία δεδομένα και πραγματοποιεί τις ζητούμενες αναζητήσεις (100 με στοιχεία που περιέχονται στα δέντρα και 100 με στοιχεία που δεν περιέχονται).

5)Σχετικά με το πρόγραμμα

Το πρόγραμμα τρέχει κανονικά εκτελώντας την Main και τυπώνει στην κονσόλα ,για κάθε αριθμό δεδομένων, τους ζητούμενους μέσους όρους για κάθε δομή δέντρου ξεχωριστά.

6)Πηγές και εξωτερική βοήθεια.

Χρησιμοποιήθηκαν αναφορές σε δομές δέντρων από το GeekForGeeks και από το GitHub.