ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ- Η.Μ.Μ.Υ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΡΧΕΙΩΝ – ΠΛΗ 201

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

ΑΤΟΜΙΚΗ ΑΝΑΦΟΡΑ ΥΛΟΠΟΙΗΣΗΣ

Αντώνης Χρυσοφάκης 2015030116

Σκοπός Εργασίας:

Εξοικείωση με πρακτικές συγγραφής κώδικα σε Java .Η άσκηση έχει σκοπό της εξοικείωση με δομές που χειρίζονται δεδομένα δύο διαστάσεων της μορφής (x,y). Και οι δύο δομές που θα φτιάξαμε μπορεί να θεωρηθούν επεκτάσεις του δυαδικού δένδρου έρευνας σε δύο διαστάσεις (όπως είναι τα σημεία στο επίπεδο). Δηλαδή η αναζήτηση (όπως και η εισαγωγή) γίνονται ταυτόχρονα πάνω σε δύο τιμές κλειδιών αντί για μία (όπως είναι στο Δυαδικό Δένδρο Έρευνας).

Περι Υλοποίησης:

Ο κώδικας για την παρούσα άσκηση γράφτηκε στην γλώσσα Java στο IDE Eclipse, ενώ η ανάλυση και υλοποίηση αυτής κατατμείται σε 3 διακριτά υποτμήματα:

• Δημιουργία των δύο διαφορετικών δέντρων.
• Αναζήτηση ενος σημείου στα δέντρα μας.
• Σύγκριση μεθόδων και τεκμηρίωση.

Οργάνωση Κώδικα και Documentation:

Ο πηγαίος κώδικας βρίσκεται οργανωμένος σε 2 πακέτα (packages) τα περιεχόμενα των οποίων περιγράφονται αναλυτικά στον παρακάτω πίνακα.

Packages:	**Use:**
trees	Εδώ βρίσκονται όλοι οι μέθοδοι δημιουργίας του εκάστοτε δέντρου και της

	αναζήτησης ενός σημείου(K_d_Tree,PR_Quadtree).
pck	Κυρίως κλάση (App.java),δημιουργία τυχαίων σημείων(randomInts),τέλος εδώ υπολογίζονται όλοι οι ΜΟ και εκτυπώνονται τα αποτελέσματα.

Υλοποίηση:

Αρχικά δημιουργήσαμε M*2 τυχαία μη μοναδικά κλειδιά στη μνήμη με τιμές κλειδιών από 1 έως 2^{16}.Στη συνέχεια ομαδοποιούμε ανά δύο τα κλειδιά μας και τα αποθηκεύουμε σε μια ArrayList που δημιουργήσαμε με όνομα points και μέγεθος Μ.Περνάμε την λίστα μας στο αντίστοιχο δέντρο που θέλουμε να υλοποιήσουμε και αφου τα δημιουργήσουμε με την βοήθεια συγκεκριμένων μεθόδων κάνουμε αναζήτηση μέσα στα δέντρα μας για 100 κλειδιά που υπάρχουν και για 100 που δεν υπάρχουν.Τέλος υπολογίζουμε τους μέσους όρους που χρειαζόμαστε για κάθε δομή και εκτυπώνουμε τα αντίστοιχα αποτελέσματα.Επίσης να σημειωθεί ότι οι τιμές που παίρνει το Μ είναι οι εξής:

- *200, 500, 1.000, 10.000, 30.000,50.000,70.000, 100.000*

<u>Δημιουργία Δέντρων:</u>

Καλούμε το αντίστοιχο δέντρο που θέλουμε να δημιουργήσουμε περνόταν σαν όρισμα μία λίστα από τυχαία double points που έχουμε δημιουργήσει.Στη συνέχεια με τη βοήθεια της μεθόδου **<u>buildTree</u>** που υπάρχει μέσα και στα δύο δέντρα μας αρχίζει η εισαγωγή στοιχείων σε αυτά.

Η μέθοδος buildTree στο *K-d_Tree* μας χρησιμοποιεί το **<u>median point</u>** της λίστας ως κόμβο στον οποίο στη συνέχεια διαιρούμε τη λίστα σε δύο μέρη, ανάλογα με τη θέση του κάθε σημείου σε σχέση με το median point.*

Ο κώδικας του *K-d_Tree* περιλαμβάνει μια κλάση **<u>Node</u>** που περιέχει ένα σημείο (point), και τις αναφορές σε αριστερό (left) και δεξί (right) παιδί του κόμβου,ενώ ο κώδικας του *PR_QuadTree* αντίστοιχα περιλαμβάνει ένα σημείο (point),και τις αναφορές σε NW,NE,SE,SW παιδιά του κόμβου.

Τέλος η μέθοδος **<u>contains</u>** που βρίσκεται στο *PR_QuadTree* μας χρησιμοποιείται για να ελέγξει αν ένα σημείο υπάρχει μέσα στο δέντρο.

**Να σημειωθεί ότι δεν είμαι σίγουρος αν γίνεται σωστά η χρήση του median point και στο PR_QuadTree(απλα δεν είχα άλλη λύση).*

<u>Μέθοδοι Αναζήτησης:</u>
Αφου φτιαξαμε τα δύο παραπάνω δέντρα μας σκοπός μας είναι να δημιουργήσουμε 100 τυχαία σημεία(Points) που υπάρχουν στα δέντρα μας και 100 τυχαία σημεία(points) που δεν υπάρχουν στα δέντρα.Για το κάθε δέντρο κάνουμε την αναζήτηση αυτών των σημείων και επιστρέφουμε το βάθος που φτάσαμε κάθε φορά αν βρήκαμε ή ακομα και αν δεν βρήκαμε το αντίστοιχο σημείο.Η διαδικασία αυτή επαναλαμβάνεται για κάθε Μ αριθμό δεδομένων που έχουμε εισάγει στα δέντρα.

- ***SearchKDTree***:Η μέθοδος αυτή παίρνει σαν όρισμα μία λίστα double σημείων που θέλουμε να αναζητήσουμε μέσα στο K-d_Tree μας και αμέσως καλεί την searchNode.

  Η μέθοδος ***searchNode*** αναζητά ένα σημείο στο δυαδικό δέντρο K-d. Δέχεται ως ορίσματα έναν κόμβο του δέντρου, ένα σημείο που αναζητούμε και έναν ακέραιο αριθμό που αντιπροσωπεύει το βάθος του κόμβου στο δέντρο.
  Στην αρχή της κλήσης, ο κόμβος που περνιέται είναι η ρίζα του δέντρου και το βάθος είναι 0. Η μέθοδος ελέγχει αν ο κόμβος είναι null, περιπτώση στην οποία το σημείο δεν βρέθηκε και επιστρέφει το βάθος του κόμβου. Αν ο κόμβος δεν είναι null, ελέγχει αν το σημείο που αναζητούμε είναι ίδιο με το σημείο του κόμβου. Αν είναι, επιστρέφει το βάθος του κόμβου. Διαφορετικά, εξετάζει την τιμή του κατάλληλου στοιχείου του σημείου και συγκρίνει την τιμή αυτή με αυτή του κατάλληλου στοιχείου του σημείου του κόμβου. Αν είναι μικρότερο, καλείται αναδρομικά η searchNode για το αριστερό υποδέντρο του κόμβου, με αύξηση του βάθους κατά 1. Διαφορετικά, καλείται αναδρομικά η searchNode για το δεξί υποδέντρο του κόμβου, επίσης με αύξηση του βάθους κατά 1.

- ***SearchPRTree***:Η μέθοδος αυτή παίρνει σαν όρισμα μία λίστα double σημείων που θέλουμε να αναζητήσουμε μέσα στο PR_QuadTree μας και αμέσως καλεί την search μέθοδο.

  Η μέθοδος ***search*** αναζητά ένα σημείο στο δέντρο PR Quadtree και επιστρέφει το βάθος του κόμβου που περιέχει το σημείο. Ξεκινώντας από τη ρίζα του δέντρου, η μέθοδος πραγματοποιεί αναδρομικές κλήσεις στον υποδέντρο που περιέχει το σημείο, μέχρι να βρει το φύλλο του δέντρου που περιέχει το σημείο. Κάθε φορά που η μέθοδος καλείται για ένα υποδέντρο, αυξάνει το βάθος κατά 1. Όταν βρεθεί το φύλλο που περιέχει το σημείο, η μέθοδος επιστρέφει το βάθος αυτού του κόμβου στο δέντρο. Εάν το σημείο δεν βρεθεί, η μέθοδος επιστρέφει το βάθος του κόμβου που φτάσαμε στην αναζήτηση μας.

ΠΕΙΡΑΜΑ:
Αφου δημιουργήσουμε τα 200 τυχαία σημεία μας(100 υπάρχουν μέσα στα δέντρα και 100 δεν υπάρχουν) κάνουμε 100 αναζητήσεις περνώντας τα σημεία που υπάρχουν στα δέντρα μας και 100 περνώντας αυτα που δεν υπάρχουν στις μεθόδους searchKDTree και searchPDTree.Τα αποτελέσματα που παίρνουμε φαίνονται στους παρακάτω πίνακες και εικόνες

ΠΙΝΑΚΑΣ Α

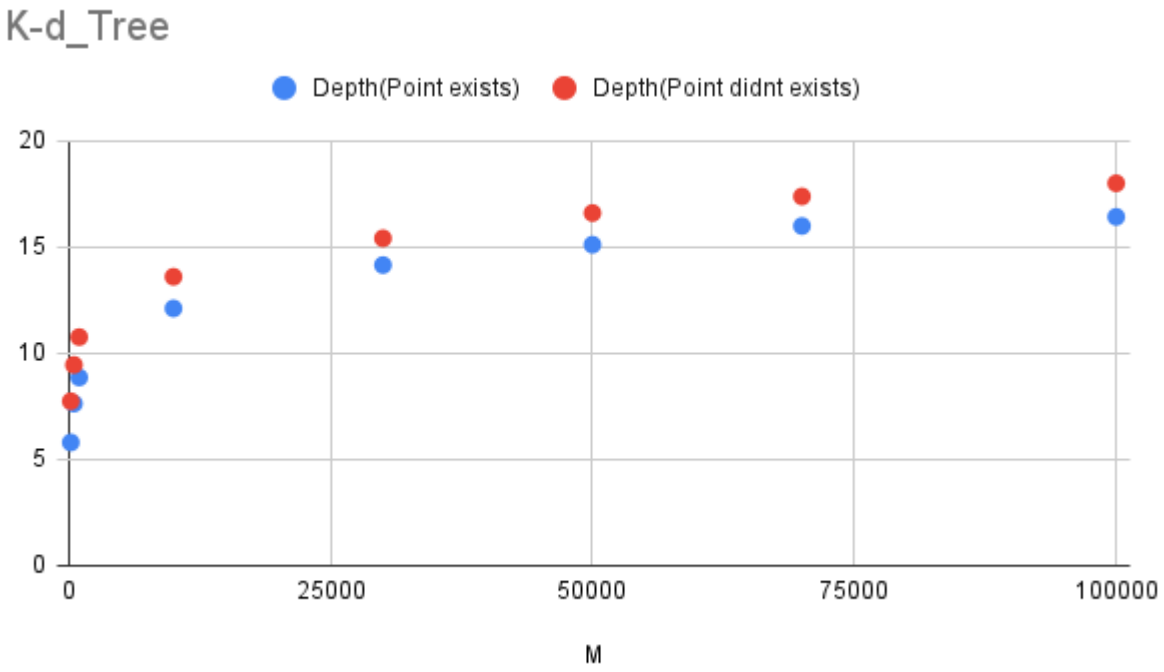
	K-d_Tree		**PR_QuadTree**	
M	*ΜΟ βάθους στο K-d_Tree σε 100 αναζητήσεις σημείων που υπαρχουν στο δέντρο.*	*ΜΟ βάθους στο K-d_Tree σε 100 αναζητήσεις σημείων που δεν υπαρχουν στο δέντρο.*	*ΜΟ βάθους στο PR_QuadTree σε 100 αναζητήσεις σημείων που υπαρχουν στο δέντρο.*	*ΜΟ βάθους στο PR_QuadTree σε 100 αναζητήσεις σημείων που δεν υπαρχουν στο δέντρο.*
200	5.80	7.74	4.20	4.37
500	7.63	9.45	5.34	5.12
1000	8.86	10.76	5.49	5.48
10000	12.11	13.60	7.66	7.17
30000	14.15	15.41	8.12	7.33
50000	15.11	16.60	9.01	8.80
70000	15.99	17.38	8.53	8.69
100000	16.42	18.00	9.09	8.55

```
MO of depth if the point we are looking for exists in our KD_Tree is  5.80  if it doesnt exists:  7.74  for  M = 200
MO of depth if the point we are looking for exists in our PR_QuadTree is  4.20  if it doesnt exists:  4.37  for  M = 200

MO of depth if the point we are looking for exists in our KD_Tree is  7.63  if it doesnt exists:  9.45  for  M = 500
MO of depth if the point we are looking for exists in our PR_QuadTree is  5.34  if it doesnt exists:  5.12  for  M = 500

MO of depth if the point we are looking for exists in our KD_Tree is  8.86  if it doesnt exists:  10.76  for  M = 1000
MO of depth if the point we are looking for exists in our PR_QuadTree is  5.49  if it doesnt exists:  5.48  for  M = 1000

MO of depth if the point we are looking for exists in our KD_Tree is  12.11  if it doesnt exists:  13.60  for  M = 10000
MO of depth if the point we are looking for exists in our PR_QuadTree is  7.66  if it doesnt exists:  7.17  for  M = 10000

MO of depth if the point we are looking for exists in our KD_Tree is  14.15  if it doesnt exists:  15.41  for  M = 30000
MO of depth if the point we are looking for exists in our PR_QuadTree is  8.12  if it doesnt exists:  7.33  for  M = 30000

MO of depth if the point we are looking for exists in our KD_Tree is  15.11  if it doesnt exists:  16.60  for  M = 50000
MO of depth if the point we are looking for exists in our PR_QuadTree is  9.01  if it doesnt exists:  8.80  for  M = 50000

MO of depth if the point we are looking for exists in our KD_Tree is  15.99  if it doesnt exists:  17.38  for  M = 70000
MO of depth if the point we are looking for exists in our PR_QuadTree is  8.53  if it doesnt exists:  8.69  for  M = 70000

MO of depth if the point we are looking for exists in our KD_Tree is  16.42  if it doesnt exists:  18.00  for  M = 100000
MO of depth if the point we are looking for exists in our PR_QuadTree is  9.09  if it doesnt exists:  8.55  for  M = 100000
```

Διαγράμματα-Σχολιασμός Αποτελεσμάτων:

Σε όλα τα διαγράμματα το μπλε χρώμα αντιπροσωπεύει τον ΜΟ του βάθους που φτάσαμε όταν το σημείο που αναζητάμε υπάρχει στο δέντρο,ενώ το κόκκινο τον ΜΟ του βάθου που φτάσαμε όταν το σημείο που αναζητάμε δεν υπάρχει στο δέντρο

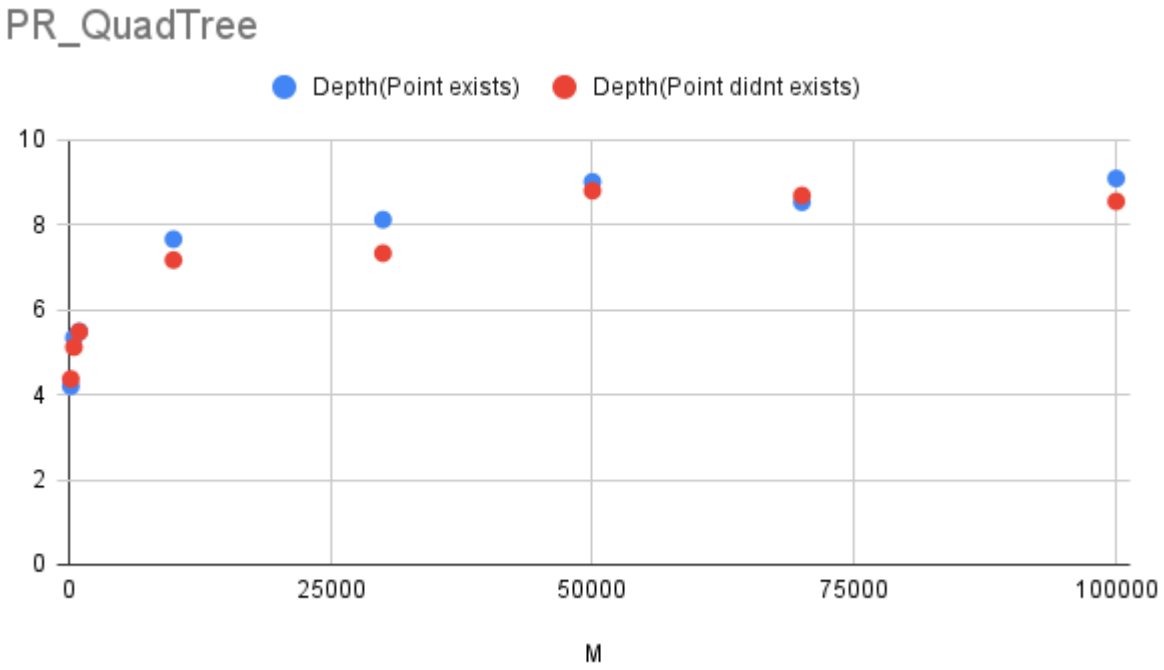
Στο ***διάγραμμα 1.1*** φαίνεται Μέσος όρος βάθους για το K-d_Tree μας.



Διαγραμμα 1.1

Είναι φανερό ότι οι μορφές των καμπυλών μας είναι γραμμικές και αυξάνονται όσο μεγαλώνει το Μ. Όπως θα περιμέναμε όταν ένα σημείο υπάρχει στο δέντρο μας ο Μέσος όρος βάθους που φτάνουμε στην αναζήτηση μας είναι μικρότερος από όταν δεν υπάρχει το σημείο αυτό στο δέντρο.

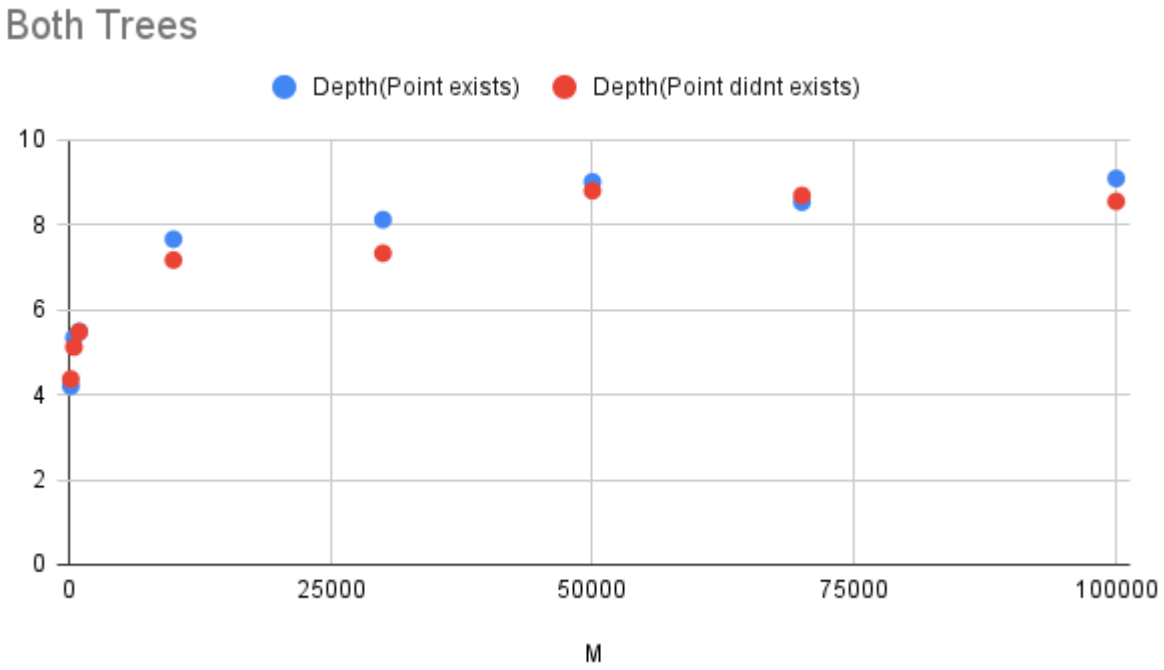
Στο ***διάγραμμα 1.2*** φαίνεται Μέσος όρος βάθους για το PR_QuadTree μας.



Διαγραμμα 1.2

Σε αυτό το δέντρο βλέπουμε ότι οι μορφές των καμπυλών μας είναι και πάλι γραμμικές.Επιπλέον παρατηρούμε ένα αρκετά μικρότερο Μέσο όρο βάθους που φτάνουμε.Θέλω να επισημάνω ότι σε κάποιες περιπτώσεις όταν ένα σημείο υπάρχει στο δέντρο μας ο Μέσος όρος βάθους στην αναζήτηση μας είναι μικρότερος από όταν δεν υπάρχει το σημείο αυτό στο δέντρο,αλλα σε κάποιες άλλες γίνεται το αντίθετο. Αυτό το γεγονός μπορεί να σημαίνει ότι υπάρχει κάποιο λάθος στον κώδικα μας.

Τέλος στο ***διάγραμμα 2.1*** φαίνονται οι Μέσοι όροι και των δύο δέντρων μαζί.



Διαγραμμα 2.1

Τέλος όταν αναπαριστούμε και τα δύο δέντρα σε ένα διάγραμμα,βλέπουμε ότι οι μορφές των καμπυλών μας είναι όπως ήταν αναμενόμενο γραμμικες .Παρατηρούμε ότι σε γενικά πλαίσια όταν ένα σημείο υπάρχει στα δέντρα μας ο Μέσος όρος του βάθους που φτάνουμε είναι μικρότερος από το όταν δεν υπάρχει.

Συμπεράσματα:

Σκοπός του παρόντος εδαφίου αποτελεί η σύγκριση των δύο δυαδικών δέντρων που υλοποιήσαμε.Είναι φανερό ότι ο Μέσος όρους βάθους στο K-d Tree για μικρό M βρίσκεται αρκετά κοντά σε αυτόν του PR QuadTree,παρολα αυτα όσο μεγαλώνει το M η διαφορά αυτών των δύο αυξάνεται με το PR QuadTree να είναι πάντα πιο αποδοτικό.Η πολυπλοκότητα και των δύο δέντρων για την αναζήτηση σημείων είναι O(log n).

Παραπομπές – Πηγές:

1. Φροντιστήριο Μαθήματος.

2. Σελίδα Μαθήματος eclass.

3.www.geeksforgeeks.org