3η Εργασία στις Δομές Δεδομένων *(2D Trees)*

Μωραϊτίδης Δημήτρης

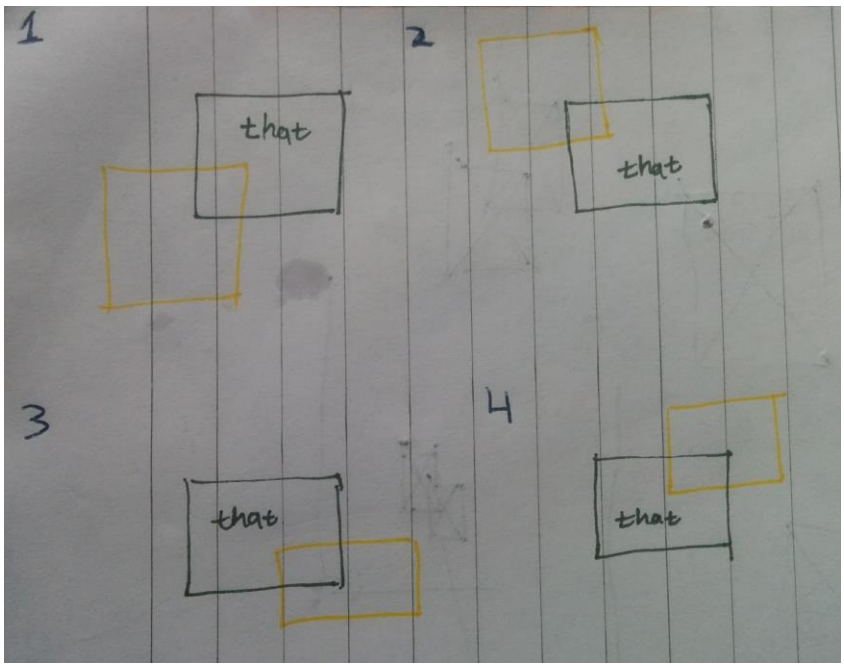
ΑΜ: 3100240

Μέρος Α:

1) Point.java
   Ο υπολογισμός της απόστασης μεταξύ δύο σημείων (distanceTo method) προκύπτει απ' τον τύπο:
   Math.sqrt((z.x-x)^2 + (z.y-y)^2)
   Η μέθοδος squareDistanceTo κάνει το ίδιο μόνο που εδώ χρειαζόμαστε το τετράγωνο της παραπάνω πράξης, και αλλαγή του τύπου που θα επιστρέφει η μέθοδος από double σε int, οπότε:
   (int)Math.sqrt((z.x-x)^2 + (z.y-y)^2)^2

2) Rectangle.java
   Για την μέθοδο contains, ελέγχουμε αν τα σημεία x,y βρίσκονται ταυτόχρονα μεταξύ των xmax, xmin και ymax, ymin.

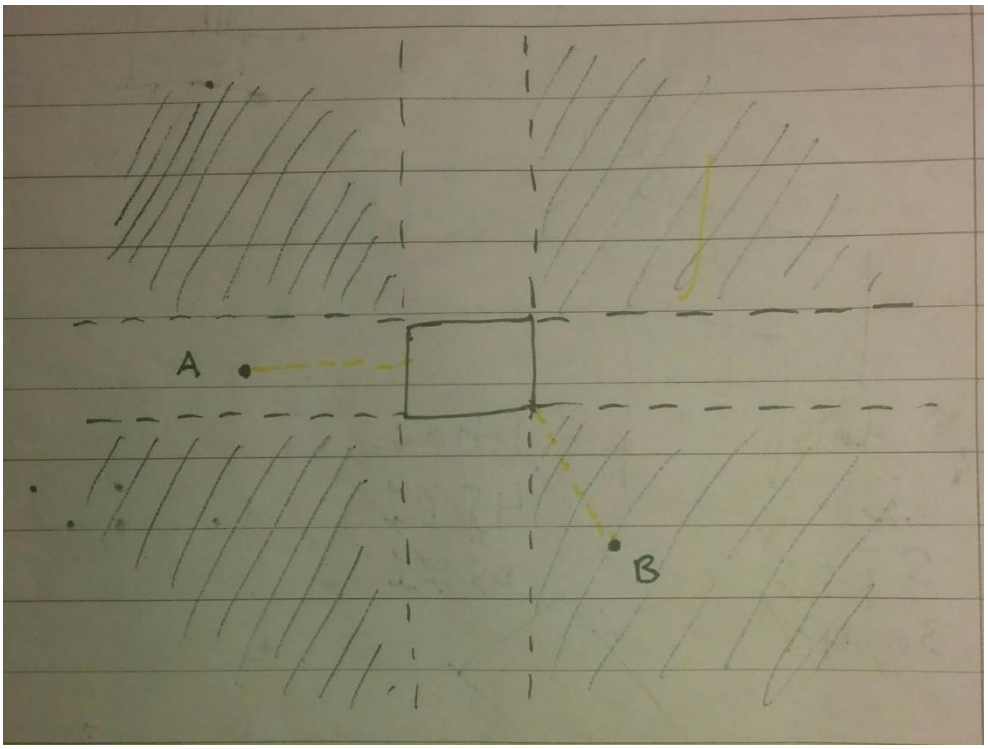
   Η μέθοδος intersects υλοποιείται με την χρήση τεσσάρων γενικευμένων περιπτώσεων όπως φαίνεται στο παρακάτω σχήμα:



   Η κάθε μία από τις 4 περιπτώσεις αντιστοιχίζεται σε κάθε ένα από τα τέσσερα return true; statements που βρίσκονται στην μέθοδο. Οι περίπτωση 1 περιλαμβάνει το

ενδεχόμενο το κίτρινο παραλληλόγραμμο να βρίσκεται 100% εντός του that, ενώ η περίπτωση 3 το that να βρίσκεται εντός του κίτρινου.

Στην μέθοδο distanceTo ο υπολογισμός της ελάχιστης απόστασης γίνεται παίρνοντας δύο υποπεριπτώσεις, ανάλογα με το που βρίσκεται το σημείο.
Αρχικά βρίσκουμε την απόσταση από την κοντινότερη κορυφή του παραλληλογράμμου. Στην περίπτωση που η ελάχιστη απόσταση δεν σχηματίζει κάθετη γωνία με μια από τις πλευρές του παραλληλόγραμμου, όπως συμβαίνει δηλαδή με το σημείο B της εικόνας, τότε η ελάχιστη απόσταση που επιστρέφει η μέθοδος είναι αυτή που έχουμε ήδη βρει από τον υπολογισμό της κοντινότερης κορυφής νωρίτερα.
Σε αντίθετη περίπτωση (A), βρίσκουμε την ελάχιστη απόσταση κάνοντας μια από τις πράξεις: xmin-p.x, p.x-xmax, ymin-p.y, p.y-ymax. Στην περίπτωση της εικόνας επιλέγουμε την πράξη xmin-A.x



Τέλος, η squareDistanceTo κάνει το ίδιο μόνο που για κάθε αποτέλεσμα γίνεται και πολλαπλασιασμός με τον εαυτό του.
Η διαχείριση ων Exceptions γίνεται στο πρόγραμμα του Μέρους Β.

Μέρος Β:

Για την κλάση TwoDTree υλοποίησα τις μεθόδους insert και search σε αντιστοιχία με τον κώδικα του εργαστηρίου 7. Για την ικανοποίηση της εναλλάξ εισαγωγής και εύρεσης κόμβων στο δέντρο, χρησιμοποίησα την μεταβλητή xORy για να μεταφέρω τον έλεγχο σε διαφορετικό άξονα κάθε φορά.

Η υλοποίηση της nearestneighbor έγινε ως εξής, λαμβάνοντας υπόψην το hint της εκφώνησης:
Δημιουργία ενός αρχικού παραλληλόγραμμου με μέγεθος [0,100]χ[0,100].
Σε κάθε διαδοχικό έλεγχο στον επόμενο κόμβο, υπολόγισε την απόσταση του δωσμένου p σημείου από το παραλληλόγραμμο που εμπεριέχει τον κόμβο. Αν είναι μικρότερη του ήδη υπάρχοντος mindist, τότε δώσε αυτήν τη νέα τιμή στο mindist και συνέχισε στα υποδέντρα του κόμβου αυτού.
Για τον σωστό ορισμό του παραλληλογράμμου, σε κάθε νέα βύθιση στο δέντρο προς τα κάτω γίνεται axis splitting, ανάλογα με το αν έχουμε x ή y κλειδί εκείνη τη στιγμή.

Με την ίδια λογική υλοποιείται και η RangeSearch.

Έπειτα γίνεται η εισαγωγή στο δέντρο διαβάζοντας και εισάγοντας κατάλληλα τα σημεία που αναγράφονται στο αρχείο που έχει δοθεί ως argument κατά την εκτέλεση του προγράμματος.
Αφού διαβαστεί το αρχείο, το πρόγραμμα τυπώνει: το όνομα του αρχείου, τον αριθμό των περιεχόμενων σημείων, καθώς και τα ίδια τα σημεία αριθμημένα με την σειρά.
Αμέσως μετά εμφανίζεται το μενού το οποίο παροτρύνει τον χρήστη να επιλέξει μια από τις περιγραφόμενες λειτουργίες.
Στην περίπτωση που διαλέξει μια απ’ τις άλλες δύο, υπάρχει επιπλέον μήνυμα παρότρυνσης για περαιτέρω εισαγωγή δεδομένων, με τρόπο ανάλογο με την περίπτωση.
Χρησιμοποιώντας τα εισαχθείσα δεδομένα καλείται η ζητούμενη μέθοδος της κλάσης TwoDTree η οποία επιστρέφει με την σειρά της τα απαραίτητα αποτελέσματα.
Στην συνέχεια το μενού παρότρυνσης ξαναεμφανίζεται με νέα παρότρυνση επιλογής λειτουργίας.
Το πρόγραμμα τερματίζει όταν ο χρήστης επιλέξει ‘0’ (exit).

*Σε όλα τα στάδια εισαγωγής δεδομένων, όπως και στο διάβασμα του αρχείου, γίνεται έλεγχος για το αν πληρούν τους απαραίτητους περιορισμούς. Σε αντίθετη περίπτωση το πρόγραμμα εμφανίζει κατάλληλο σφάλμα και το πρόγραμμα τερματίζει.