3η Εργασία στις Δομές Δεδομένων
2019-2020
Υπ. Καθηγητής: Ε. Μαρκάκης
Φοιτητές: Πιπιλίκας Ευάγγελος 3180157 / Αλεξόπουλος Νικόλαος 3180004

Μέρος Α:

Class Rectangle:

Μέθοδος contains():

Εφόσον ένα αντικείμενο της Rectangle ορίζεται από τους 4 ακέραιους αριθμούς x_{min}, x_{max}, y_{min}, y_{max}, τότε για να αποφανθούμε εάν ένα αντικείμενο της Point περιέχεται εντός κάποιου ορθογωνίου πρέπει να ελέγξουμε εάν ισχύουν **ταυτόχρονα** (&&) οι παρακάτω συνθήκες:

Η τετμημένη του x:

- Να ανήκει στο διάστημα [x_{min}, x_{max}] **και**

Η τεταγμένη του y:

- Να ανήκει στο διάστημα [y_{min}, y_{max}].

Εάν ικανοποιούνται όλες οι συνθήκες τότε η contains() επιστρέφει true, διαφορετικά επιστρέφει false.

Μέθοδος intersects():

Για την υλοποίηση της intersects() εκμεταλλευόμαστε την μέθοδο contains() που περιγράψαμε παραπάνω. Η ιδέα μας είναι πως για να τέμνονται 2 παραλληλόγραμμα πρέπει τουλάχιστον μία από τις κορυφές του ενός να περιέχεται εντός του άλλου. Για να ελέγξουμε κάτι τέτοιο, εντός της intersects() ορίζουμε 4 αντικείμενα της Point με συντεταγμένες τις 4 κορυφές του ορθογωνίου που δέχεται ως παραμέτρο η μέθοδος. Έπειτα για το ορθογώνιο που καλεί την intersects() ελέγχουμε εάν περιέχει τουλάχιστον μία από τις κορυφές του ορθογωνίου-παραμέτρου. Χρησιμοποιούμε λογική διάζευξη (||) αφού θέλουμε η μέθοδος να επιστρέφει true εάν έστω και μία κορυφή του ενός ορθογωνίου περιέχεται εντός του άλλου. Επειδή υπάρχει περίπτωση το ορθογώνιο-παράμετρος να περικλείει εντός του ολόκληρο το ορθογώνιο που καλεί την intersects() ελέγχουμε και αν το τελευταίο περιέχει έστω και μία κορυφή του πρώτου με χρήση της διάζευξης.

Μέθοδος distanceTo():

Η μέθοδος δέχεται ως παράμετρο ένα αντικείμενο της Point. Για τον υπολογισμό του αποτελέσματος της distanceTo() ελέγχουμε 3 περιπτώσεις:

1) Εάν η παράμετρος βρίσκεται εντός ή επί του ορθογωνίου που καλεί την distanceTo() και σε περίπτωση που αυτό ισχύει, επιστρέφουμε 0.

2) Εάν το Point βρίσκεται εκτός του ορθογωνίου τότε πρέπει να δούμε εάν μπορούμε να φέρουμε κάθετα την απόσταση από το Point προς το ορθογώνιο. Αυτό είναι εφικτό εάν η τετμημένη του Point ανήκει στο διάστημα [x_{min}, x_{max}] **ή** η τεταγμένη του Point ανήκει στο διάστημα [y_{min}, y_{max}].

3) Εάν το Point βρίσκεται εκτός του ορθογωνίου και ΔΕΝ μπορούμε να φέρουμε κάθετα την απόσταση από το Point προς το ορθογώνιο τότε υπολογίζουμε την απόσταση του Point από όλες τις κορυφές του ορθογωνίου και βρίσκουμε και επιστρέφουμε την μικρότερη από αυτές.

Μέρος Β:

Class TwoDTree:

Για την εύρεση του Point που απέχει την λιγότερη απόσταση από ένα Point που δίνει ο χρήστης, δημιοργήσαμε 2 μεθόδους, την nearestNeighbor(), όπως ορίζεται στην εκφώνηση του Project και την nearestNeighborR() η οποία εκτελεί μία αναδρομική διάσχιση του TwoDTree. Θα περιγράψουμε πρώτα την αναδρομική μέθοδο.

Μέθοδος nearestNeighborR():

Η μέθοδος nearestNeighborR() δέχεται ως παραμέτρους ένα TreeNode, το Point για το οποίο αναζητούμε την ελάχιστη απόσταση από κάποιο Point που περιέχεται στο δέντρο, το εκάστοτε «κοντινότερο» Point όπως και την εκάστοτε ελάχιστη απόσταση που έχουμε υπολογίσει μέχρι κάποιο σημείο της αναδρομικής αναζήτησης εντός του δέντρου, μια ακέραια μεταβλητή που συμβολίζει το επίπεδο στο οποίο βρίσκεται κάθε φορά η αναδρομή (βοηθά στο να συγκρίνουμε τα Point's από 2 TreeNode's με βάση το x ή το y.) και τέλος το ορθογώνιο του οποίου τα x_{min}, x_{max}, y_{min}, y_{max} εκμεταλλευόμαστε για να το χωρίζουμε κάθε φορά σε 2 υπο-ορθογώνια προχωρώντας την διάσχιση του δέντρου.
Αναλυτικά:

Ελέγχουμε εάν το τρέχον/εκάστοτε TreeNode βρίσκεται σε άρτιο ή περιττό επίπεδο εντός του δέντρου.
Αν το επίπεδο είναι άρτιο δημιουργούμε 2 «κατακόρυφα» υπο-ορθογώνια ως εξής: το «αριστερό» υπο-ορθογώνιο ορίζεται ως: [$ParentRectangle_{xMin}$, $CurrentNode_{x}$]x[$ParentRectangle_{yMin,}$ $ParentRectangle_{yMax}$] ενώ το «δεξιό» ως: [$CurrentNode_{x}$, $ParentRectnagle_{xMax}$]x[$ParentRectangle_{yMin,}$ $ParentRectangle_{yMax}$]. Εάν το επίπεδο είναι περιττό τότε δημιουργούμε 2 «οριζόντια» υπο-ορθογώνια ως εξής: το «κάτω» υπο-ορθογώνιο ορίζεται ως: [$ParentRectangle_{xMin}$, $ParentRectangle_{xMax}$]x[$ParentRectangle_{yMin,}$ $CurrentNode_{y}$] ενώ το «άνω» ως: [$ParentRectangle_{xMin}$, $ParentRectangle_{xMax}$]x[$CurrentNode_{y}$, $ParentRectangle_{yMax}$].

Αφού δημιουργήσουμε τα 2 υπο-ορθογώνια ελέγχουμε εάν κάποιο (πρώτα το «αριστερό» και μετά το «δεξί» // αντίστοιχα το «πάνω» ή το «κάτω») από αυτά έχει απόσταση από το Point που περνά ο χρήστης στη μέθοδο ως παράμετρο, αυστηρά μικρότερη από την απόσταση που έχουμε ήδη υπολογίσει έως το προηγούμενο/γονικό επίπεδο στο δέντρο **Ή** η απόσταση του κάθε παιδιού (όσα υπάρχουν) από το Point του χρήστη είναι μικρότερη από την απόσταση του Point του χρήστη και του τελευταία υπολογισμένου κοντινότερου σημείου. Ενημερώνουμε την μεταβλητή που περιέχει την μικρότερη απόσταση και έπειτα, αν το TreeNode στο οποίο βρισκόμαστε δεν είναι φύλλο, καλούμε αναδρομικά την rangeSearchR() για τα παιδιά του TreeNode. Εάν ο αλγόριθμος φτάσει σε κάποιο φύλλο το οποίο περιέχει το Point με την μικρότερη απόσταση τότε δεν καλείται η rangeSearchR() ξανά αλλά απλώς επιστρέφεται το Point αυτό. Ακόμη, εάν η απόσταση κάποιου από τα υπο-ορθογώνια δεν έχει απόσταση αυστηρά μικρότερη από την ήδη υπολογισμένη τότε και πάλι δεν κάνουμε αναδρομή (όπως περιγράφεται στα hints της εκφώνησης.).

Μέθοδος nearestNeighbor ():

Η μέθοδος αυτή πρακτικά ξεκινά την αναδρομή, περιέχοντας την πρώτη κλήση της nearestNeighborR(), με αρχικές παραμέτρους το παραλληλόγραμμο το οποίο αναπαριστά όλον τον χώρο στον οποίο εργαζόμαστε ([0, 100]x[0, 100]), την ρίζα του δέντρου, την απόσταση του αρχικού ορθογωνίου από το Point που περνά ο χρήστης και τέλος τον ακέραιο 0, που συμβολίζει το ανώτερο επίπεδο του δέντρου.

Υλοποίηση της rangeSearch()

Όπως και παραπάνω, η δημιουργία της Λίστας που θα περιέχει όλα τα Points τα οποία βρίσκονται εντός ενός ορθογωνίου που περνά ο χρήσης ως παράμετρο της μεθόδου γίνεται αναδρομικά. Γι' αυτό έχουμε γράψει 2 μεθόδους, την rangeSearch(), όπως ορίζεται στην εκφώνηση και την rangeSearchR() η οποία είναι αναδρομική. Περιγράφουμε αρχικά την rangeSearchR().

Μέθοδος rangeSearchR():

Η rangeSearchR() δέχεται ως παραμέτρους το εκάστοτε TreeNode στο οποίο βρίσκεται η αναδρομή, το ορθογώνιο το οποίο κάθε TreeNode του δέντρου χωρίζει σε δύο υπο-ορθογώνια, το ορθογώνιο που δίνει ο χρήστης, την λίστα με τα σημεία που πρέπει να επιστραφούν και έναν ακέραιο ο οποίος όπως και προηγουμένως αναπαριστά το επίπεδο στο οποίο βρίσκεται η αναδρομή.

Αναλυτικά:

Ελέγχουμε το επίπεδο που βρίσκεται η αναδρομή και με βάση το αν ο ακέραιος που υποδεικνύει το επίπεδο είναι άρτιος ή περιττός ορίζουμε τα 2 υπο-ορθογώνια στα οποία χωρίζει το κάθε TreeNode το γονικό/ανώτερο ορθογώνιο (Αρχικό ορθογώνιο είναι το [0,100]x[0, 100] που αναπαριστά τον χώρο στον οποίο εργαζόμαστε). Η διαδικασία είναι η ίδια με αυτήν που ακολουθούμε στην nearestNeighborR().

Αφού οριστούν τα υπο-ορθογώνια ελέγχουμε για καθένα από τα υπο-ορθογώνια εάν:

- Το ορθογώνιο που περνά ο χρήστης τέμνεται με κάποιο από αυτά **και**
- Το ορθογώνιο που περνά ο χρήστης περιέχει το Point που έχει το εκάστοτε TreeNode, στο οποίο βρίσκεται η αναδρομή.

Εάν κάποιο ή και τα δύο από τα υπο-ορθογώνια ικανοποιούν τις 2 συνθήκες τότε προσθέτουμε στην λίστα προς επιστροφή το Point που περιέχει το TreeNode που βρίσκεται η αναδρομή.

Στο σημείο αυτό προετοιμάζουμε την αναδρομική κλήση της rangeSearchR() αυξάνοντας κατά 1 τον ακέραιο που αναπαριστά το επίπεδο του δέντρου στο οποίο πρέπει να τρέξει η μέθοδος. Εφόσον το παραλληλόγραμμο που δίνει ο χρήστης τέμνεται με το παραλληλόγραμμο που αναπαριστά κάποιο TreeNode και το TreeNode αυτό δεν είναι φύλλο, τότε καλούμε αναδρομικά την rangeSearch() για κάθε ένα από τα παιδιά του εκάστοτε TreeNode.