Υλοποίηση RTree

εργασία στα πλαίσια του μαθήματος ανάκτησης δεδομένων

2/15/2008

Κρητικός Απόστολος (914) & Σκαλιστής Στέφανος (1024)

Το R-δένδρο

Αν και γνωστό από ήδη από την θεωρία θα γίνει συνοπτική αναφορά. Το R-δένδρο είναι μία δενδρική δομή για πολυδιάστατα δεδομένα. Στη συγκεκριμένη υλοποίηση το δένδρο δεν αποθηκεύει ορθογώνια, όπως ορίζεται στο άρθρο Guttman, αλλά σημεία. Το δένδρο υποστηρίζει τρείς βασικές λειτουργίες:

- Εισαγωγή σημείου στο δένδρο.
- Ερώτημα διαστήματος με βάση ένα σημείο αναφοράς p στο n-διάστατο χώρο και μία απόσταση d, όπου πρέπει να βρεθούν όλα τα σημεία, τα οποία περιέχονται στη δομή, που απέχουν το πολύ d από το σημείο d.
- Ερώτημα k-Κοντινότερων Γειτόνων με βάση ένα σημείο αναφοράς p και έναν ακέραιο αριθμό k, όπου πρέπει να ανακτηθούν τα k-Κοντινότερα σημεία στο p.

  (Σημείωση: Συμφώνα με τη γενικότερη θεωρία των k-Κοντινότερων Γειτόνων αν υπάρχει ισοπαλία στις τελευταίες θέσεις πρέπει να επιστραφούν όλα τα ισόπαλα σημεία).

Τέλος πρέπει να κρατάει στοιχεία όπως το ύψος του δένδρου και το πόσα σημεία είναι αποθηκευμένα στη δομή αυτή.

Δομές και τεχνικές λεπτομέρειες της υλοποίησης

Όλοι η δομή και η λειτουργία του δένδρου περιγράφεται αναλυτικά στο JavaDoc το οποίο παρατίθεται. Ακολουθεί μια συνοπτική περιγραφή των τεχνικών λεπτομερειών. Για περαιτέρω λεπτομέρειες ανατρέξτε στην αναλυτική περιγραφή του JavaDoc.

Οι κόμβοι:

Το δένδρο αποτελείται από εσωτερικούς κόμβους και κόμβους-φύλλα. Οι εσωτερικοί κόμβοι είναι οι τυπικοί, έχοντας μόνο ένα πίνακα με τις καταχωρήσεις (entries) του και το ελάχιστο περικλείον ορθογώνιο (minimum bounding rectangle) που περιέχει τις καταχωρήσεις αυτές. Τα φύλλα από τη άλλη πλευρά, έχουν επιπρόσθετα έναν αριθμό σελίδας, ο οποίος υποδηλώνει σε ποια σελίδα κείτονται τα σημεία τα οποία του έχουν ανατεθεί. Πρέπει να σημειωθεί ότι για οποιαδήποτε ενέργεια στα φύλλα, πρέπει πρωτίστως να φορτωθεί η σελίδα (ευθύνη του Διαχειριστή Δίσκου), να εκτελεστούν οι ενέργειας και έπειτα να αποδοθεί η σελίδα στο σύστημα.

Η εισαγωγή σημείου:

Κατά την εισαγωγή, χρησιμοποιούνται δύο στοίβες με σκοπό την ανοδική (ή προς τα πίσω) διάσχιση. Στη μία εξ αυτών αποθηκεύεται ο κόμβος-γονιός ενώ στην άλλη αποθηκεύεται ένας ακέραιος ο οποίος υποδηλώνει την θέση του πίνακα καταχωρήσεων στον κόμβο-γονίο η οποία χρησιμοποιήθηκε για να ανακτηθεί ο κόμβος-παιδί.

Οι διασπάσεις κόμβων και η αναπροσαρμογή του δένδρου εκτελούνται ακριβώς όπως περιγράφονται στο άρθρο του Guttman και δεν θα σχολιαστούν περαιτέρω.

Ερώτημα Περιοχής:

Στην λειτουργία αυτή, χρησιμοποιούνται μια λίστα (ArrayList) για την προσωρινή αποθήκευση των σημείων που έχουν ανακτηθεί έως τώρα καθώς και μία στοίβα για την αποθήκευση των κόμβων που απομένουν να επισκεφτούν. Στο τέλος της διαδικασίας η λίστα μετατρέπεται σε πίνακα για ευκολότερη μεταχείριση.

Στο ερώτημα αυτό εφαρμόζεται η πρακτική του κλαδέματος κόμβων με βάση την μικρότερη απόσταση που απέχουν από το σημείο αναζήτησης.

Ερώτημα Πλησιέστερων Γειτόνων:

Στην λειτουργία αυτή, χρησιμοποιείται μια ταξινομημένη λίστα (ArrayList) για την αποθήκευση των κόμβων που απομένουν να εξεταστούν και οι οποίοι έχουν περάσει τις στρατηγικές κλαδέματος του δένδρου όπως περιγράφονται στο άρθρο του Ρουσσόπουλου.

Ομοίως χρησιμοποιείται πάλι μια ταξινομημένη λίστα (ArrayList) για την αποθήκευση των σημείων που έχουν ανακτηθεί και τα οποία δεν έχουν διαγραφεί από τις στρατηγικές κλαδέματος του δένδρου.

Για την χρήση των λιστών δημιουργήθηκαν δύο ψευδό-κλάσεις. Η μία φιλοξενεί τα σημεία που έχουν ανακτηθεί μαζί με την απόσταση τους από το κέντρο της έρευνας. Η άλλη φιλοξενεί τους κόμβους που απομένουν να εξεταστούν καθώς και τις δύο μετρικές αποστάσεις όπως περιγράφονται στο άρθρο του Ρουσσόπουλου.

Ακόμη δημιουργήθηκε και μια κλάση σύγκρισης για την δυνατότητα ταξινόμησης των παραπάνω μέσα στις λίστες.

Στο τέλος της διαδικασίας αυτής μετατρέπεται η λίστα των αποτελεσμάτων σε πίνακα για ευκολότερη μεταχείριση.

Στατιστικά στοιχεία:

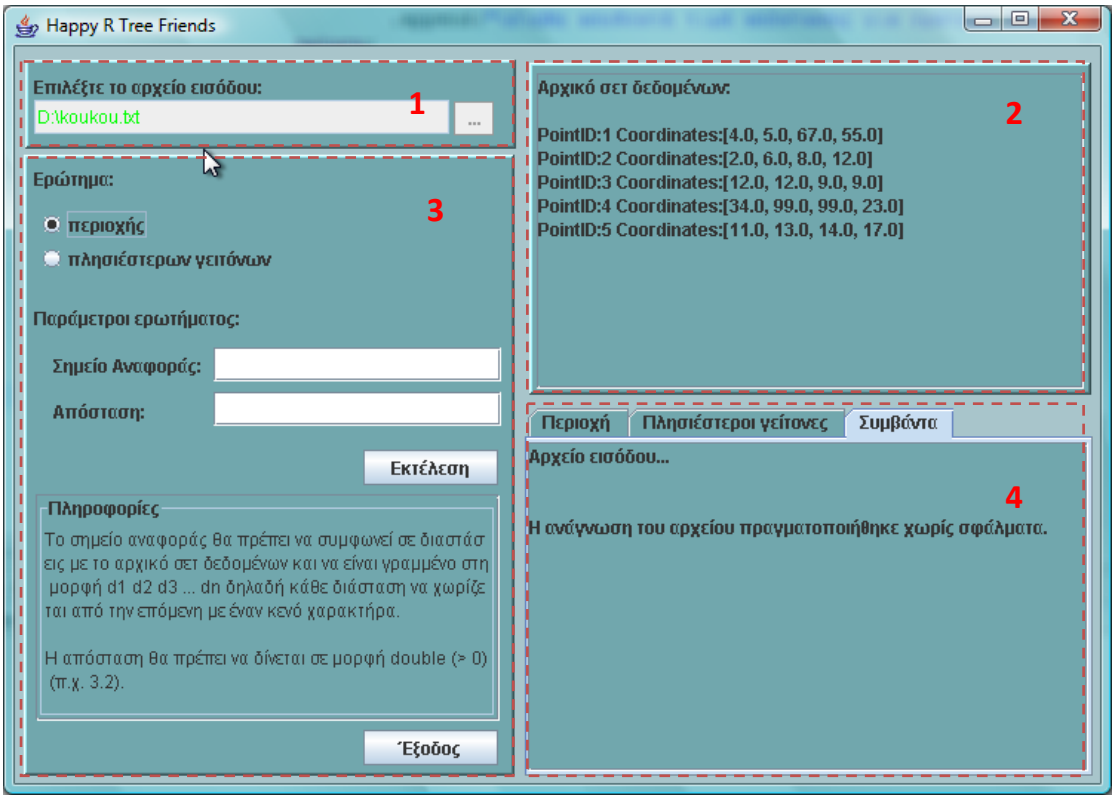
Η δομή αυτή δοκιμάστηκε με την εισαγωγή 3000 σημείων, εκτέλεσης 3000 ερωτημάτων περιοχής και 500 ερωτημάτων εύρεσης των 5-πλησιέστερων γειτόνων. Τα σημεία που χρησιμοποιήθηκαν ακολουθούσαν n-διάστατη κατανομή Gauss με μέση τιμή το 0.0 και τυπική απόκλιση 500. Ακολουθεί ο συγκεντρωτικός πίνακας.

Αριθμός διαστάσεων:	**2**	**5**	**15**	**30**
Εισαγωγή	2.688 sec	3.953 sec	2.672sec	4.578 sec
Ερώτημα περιοχής	28.203 sec	45.359 sec	26.610sec	9.553 sec
Ερώτημα Πλησιέστερων γειτόνων	21.800 sec	29.813 sec	22.125sec	24.360 sec

Γραφικό περιβάλλον

Το GUI αποτελεί ένα κομμάτι της εργασίας που έχει ως στόχο την οπτικοποίηση των λειτουργιών που αναπτύχθηκαν δηλαδή του κτησίματος του RTree από το πρόγραμμα και των ερωτήσεων Περιοχής (Range) και Γειτνίασης (Nearest Neighbours).

Παρακάτω μπορεί κανείς να δει το παράθυρο στην πλήρη του μορφή:



Το παράθυρο στην πλήρη του μορφή αποτελείται από τις 4 σημασμένες περιοχές που φαίνονται παραπάνω. Κάθε μια από αυτές αποτελεί ένα panel. Ας δούμε εν συντομία τη λειτουργία κάθε μιας από τις περιοχές αυτές.

Panel επιλογής αρχείου (είσοδος) (1):

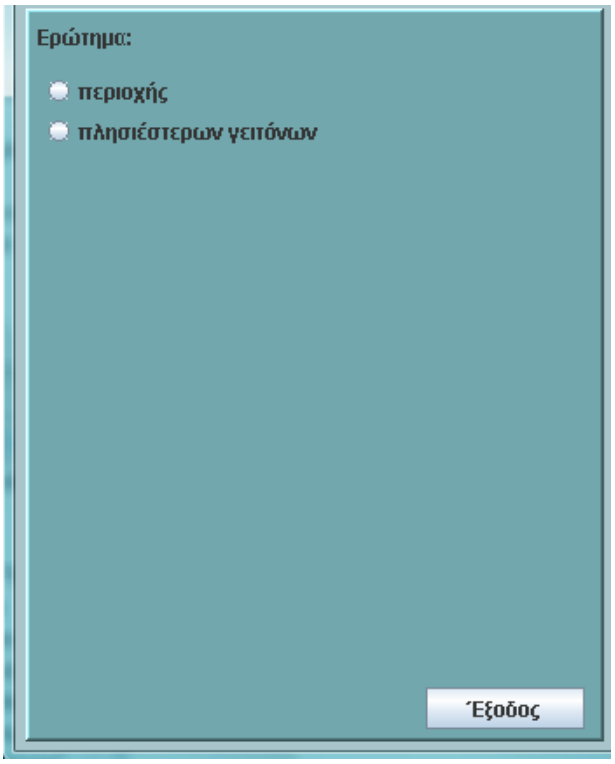
Ο χρήστης καλείται να επιλέξει το αρχείο που περιέχει τα σημεία με βάση τα οποία θα χτισθεί (αρχικοποιηθεί) το RTree. Υπάρχει ένα κουμπί στα δεξιά το οποίο βοηθά τον χρήστη να επιλέξει το επιθυμητό αρχείο. Μόλις αυτός το κάνει και εφόσον το αρχείο δεν εμφανίζει fatal errors (π.χ. δεν υπάρχει ή είναι άδειο) τότε εμφανίζεται η διαδρομή του στο πλαίσιο κειμένου ενώ το κουμπί παγώνει κλειδώνοντας στην επιλογή του χρήστη και μη επιτρέποντας εκ νέου επιλογή. Για να συμβεί κάτι τέτοιο ο χρήστης οφείλει να τρέξει εκ νέου την εφαρμογή.

Panel εμφάνισης αρχικών σημείων (2):

Εδώ εμφανίζεται (εφόσον δεν υπήρξε πρόβλημα με το αρχείο εισόδου) το αρχικό σετ δεδομένων (σημεία) με βάση το οποίο δομείται αρχικά το RTree.

Panel ερωτημάτων (3):

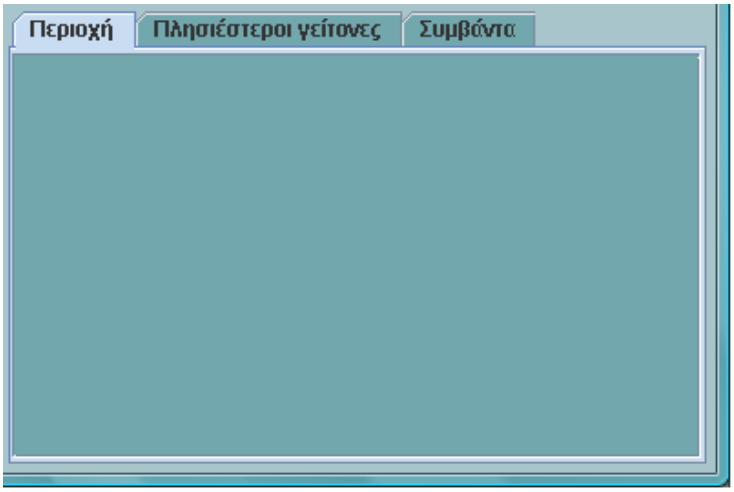
Εδώ ο χρήστης μπορεί να επιλέξει μεταξύ των δύο διαθέσιμων ερωτημάτων (περιοχής και γειτνίασης). Αρχικά το Panel περιέχει μόνον τα κουμπιά επιλογής μεταξύ ερωτημάτων.



Αφού ο χρήστης κάνει την επιλογή του εμφανίζονται τα απαραίτητα πλαίσια για το σχηματισμό του ερωτήματος καθώς και οδηγίες για τις παραδοχές που πρέπει να ακολουθηθούν σε σχέση με τον ορισμό των σημείων αναφοράς καθώς και της επιθυμητής περιοχής ή του αριθμού των γειτόνων.

Panel αποτελεσμάτων & Καταγραφής συμβάντων (4):

Πρόκειται για το πιο σημαντικό τμήμα του γραφικού περιβάλλοντος. Αποτελείται από τρεις καρτέλες.



Περιοχή:

Στην καρτέλα αυτή φιλοξενούνται τα αποτελέσματα της αναζήτησης με βάση ένα σημείο αναφοράς και μια περιοχή ενδιαφέροντος γύρω από αυτό.

Πλησιέστεροι γείτονες:

Αντίστοιχα εδώ φιλοξενούνται τα αποτελέσματα της αναζήτησης με βάση ένα σημείο αναφοράς και έναν αριθμό πλησιέστερων γειτόνων σε αυτό.

Συμβάντα:

Λειτουργεί ως log καταγράφοντας οτιδήποτε συμβαίνει κατά τη λειτουργία του προγράμματος. Ξεκινά με την αρχικοποίηση του RTree ενημερώνοντας τον χρήστη αν το αρχείο διαβάστηκε σωστά, αν υπήρχαν σφάλματα και τι είδους ήταν αυτά ή αν κάτι χρειάζεται να ορισθεί εκ νέου.

Στη συνέχεια παρακολουθεί της κινήσεις του όσον αφορά τα ερωτήματα και εμφανίζει μηνύματα λάθους οδηγώντας το χρήστη να δομήσει σωστά τα ερωτήματά του.

Αξίζει να σημειωθεί ότι στις δύο πρώτες καρτέλες εμφανίζεται ένα πλήθος σημείων ή μήνυμα αποτυχίας αν δεν ήταν δυνατή η εύρεση σημείων που να επαληθεύουν το ερώτημα.

Παράρτημα – Javadoc

Package rtree

Class Summary	
Rtree	A modified implementation of the R-Tree data structure.

rtree

Class Rtree

```
java.lang.Object
  └ rtree.Rtree
```

```
public class Rtree
extends Object
```

A modified implementation of the R-Tree data structure. It supports Points (instead of Rectangles as in Guttman's original paper). Points can be inserted at any time. It also supports a Range Query based on a Point and a radius and k-Nearest Neighbours Query.

The Range Query is based on minDistance as described in Roussopoulos paper and the methodology as described in Guttman's paper.

The k-Nearest Neighbours Query is based on Roussopoulos paper,with a minor efficiency change on calculating MinMaxDistance

---

Nested Class Summary

private class	`Rtree.NNeighbourEntry` This is a wrapper of an entry for the Nearest Neighbour Query.
private class	`Rtree.NNeighbourEntryComparator` A comparator for the Rtree.NNeighbourEntry.
private class	`Rtree.NNeighbourRectangle` This is a wrapper of an Node/Rectangle for the Nearest Neighbour Query.

Field Summary

private int	`height` The height of the tree.
private `Stack<Integer>`	`parentEntryIndex` A stack that specifies the index that was followed, in the parent's entries array, in order to reach this node.
private `Stack<InnerNode>`	`parents` A stack that is used when the tree is traversed and stores the parents of each visited node.
private long	`pointCount` The number of points currently in the tree.
private `Node`	`root` The root of the tree.

Constructor Summary

`Rtree()` Constructs a R-Tree and initializes all the data structures in order for the tree to function.

Method Summary

private `Node`	`adjustTree(Node leaf, Node newLeaf)` Ascend from a leaf node L to the root, adjusting MinimumBoundaryRectangles and propagating node splits as necessary.Used by insert(Point) (Also taken from Guttman's paper).
private `LeafNode`	`chooseLeaf(Point point)` Chooses a leaf to add the rectangle to.Used by insert(Point) (Also taken from Guttman's paper).
void	`insert(Point point)` Inserts a point to the Tree-Structure.
private	`minmaxDistance(Rectangle minimumBoundryRectangle, Point centre)`

double	Calculates the MinMaxDistance as described to Roussopoulos paper.
`Point[]`	`nearestNeighbours`(`Point` centre, int noNeighbours) Implements a nearest Neighbours search from a point of origin, according to Roussopoulos paper.
`Point[]`	`rangeQuery`(`Point` centre, double distance) Calculates a range query.Given the point of origin and the distance finds all the points within this area.

Methods inherited from class java.lang.`Object`
`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

root

`private Node` **`root`**

The root of the tree.

It is an InnerNode unless the tree is consisted only by this node;

In that case it's LeafNode.

---

height

`private int` **`height`**

The height of the tree.

---

pointCount

`private long` **`pointCount`**

The number of points currently in the tree.

---

parents

`private Stack<InnerNode>` **`parents`**

A stack that is used when the tree is traversed and stores the parents of each visited node.

See Also:

Stack

---

parentEntryIndex

```
private Stack<Integer> parentEntryIndex
```

A stack that specifies the index that was followed, in the parent's entries array, in order to reach this node.

See Also:

parents, Stack

Constructor Detail

Rtree

```
public Rtree()
```

Constructs a R-Tree and initializes all the data structures in order for the tree to function.

Method Detail

nearestNeighbours

```
public Point[] nearestNeighbours(Point centre,
                                 int noNeighbours)
```

Implements a nearest Neighbours search from a point of origin, according to Roussopoulos paper. It applies all the pruning strategies (both downward and upward).
The result might have less size than noNeighbours in case the tree contains less Points than noNeighbours.

Parameters:

centre - The point of origin.

noNeighbours - The number of the required Neighbours.

Returns:

A Point[] containing the results; null in case noNeighbours <= 0.

---

minmaxDistance

```
private double minmaxDistance(Rectangle minimumBoundryRectangle,
                              Point centre)
```

Calculates the MinMaxDistance as described to Roussopoulos paper.
It has linear cost (depending on the dimensions of course)

Parameters:

minimumBoundryRectangle - The rectangle to calculate the MinMaxDistance from the centre.

centre - The point from which the distance is calculated

Returns:

The distance calculated.

---

rangeQuery

```
public Point[] rangeQuery(Point centre,
                          double distance)
```

Calculates a range query.Given the point of origin and the distance finds all the points within this area.

Parameters:

centre - The point of origin.

distance - The distance.

Returns:

A Point[] containing the results. The array might have lenght == 0 in case no results are found.

---

insert

```
public void insert(Point point)
```

Inserts a point to the Tree-Structure.
It invokes chooseLeaf(Point) to find the appropriate leaf to insert to. Checks whether the leaf is full and in that case invokes LeafNode.splitNode(Entry). Finally it propagates the changes upwards using adjustTree(Node, Node) and if a root split has occurred, it creates a new root.(Taken from Guttman's paper)

Parameters:

point - The point to be inserted

See Also:

chooseLeaf(Point), adjustTree(Node, Node), LeafNode.splitNode(Entry)

---

adjustTree

```
private Node adjustTree(Node leaf,
                        Node newLeaf)
```

Ascend from a leaf node L to the root, adjusting MinimumBoundaryRectangles and propagating node splits as necessary.Used by insert(Point)
(Also taken from Guttman's paper).

Parameters:

leaf - The leaf that the point was insert.

newLeaf - The new leaf if a split has occurred;null otherwise.

Returns:

The new root if a root split occurred;else null.

---

chooseLeaf

```
private LeafNode chooseLeaf(Point point)
```

Chooses a leaf to add the rectangle to.Used by insert(Point)
(Also taken from Guttman's paper).

Package rtree.util

Class Summary

DiskHandler	This class implements a handler for the static data inside our disk.
Entry	This class represents an entry of our R-tree implementation.
InnerNode	The representation of an InnerNode inside an R-Tree.
LeafNode	The representation of an LeafNode inside an R-Tree.
Node	This class represents the Nodes of the tree.
Point	
Rectangle	This is a basic representation of a multidimensional rectangle.

rtree.util

Class DiskHandler

```
java.lang.Object
  └ rtree.util.DiskHandler
```

```
public abstract class DiskHandler
extends Object
```

This class implements a handler for the static data inside our disk. It retrieves and stores the contents of the R-tree into the appropriate files.

Author:

Sskalist

Constructor Summary

`DiskHandler()`

Method Summary

static `Point[]`	`loadFromDisk`(long pageNumber, int entryCount, int maxEntries) It loads a page from the disk and returns it to a Point array.
static void	`writeToDisk`(`Point[]` entries, int entryCount, long pageNumber) It writes a page to a file on the disk.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DiskHandler

```
public DiskHandler()
```

Method Detail

loadFromDisk

```
public static Point[] loadFromDisk(long pageNumber,
                                   int entryCount,
                                   int maxEntries)
```

It loads a page from the disk and returns it to a Point array.

Parameters:

pageNumber - The number of the page-target.

maxEntries - The maximum entries.

Returns:

An maxEntries-lengthed array containing the points loaded.
it has from start to entryCount -1 the points loaded and until the end null.

---

writeToDisk

```
public static void writeToDisk(Point[] entries,
                               int entryCount,
                               long pageNumber)
```

It writes a page to a file on the disk.

Parameters:

entries - The entries which are about to be written to the file.

entryCount - The number of entries in the Page/Node

pageNumber - The number that shows the index of the page.

rtree.util

Class Entry

java.lang.Object
  └ **rtree.util.Entry**

Direct Known Subclasses:

Node, Point

---

public abstract class **Entry**
extends Object

This class represents an entry of our R-tree implementation. It can be either Point or Node

Author:

Sskalist

Field Summary

protected long	entryID The id number of the entry.
private static long	maxUsedEntryId The maximum Id that has been used
private static long	noEntries The entry count.
private static Stack<Long>	unusedEntriesStack A stack that holds all the unused Ids.

Constructor Summary

Entry() Default constructor.

Method Summary

protected	finalize()

void	Finalizes the Entry.Inserts its entryID to the stack so it can be used later on.
long	`getEntryID`() Gets the id of the entry.
static long	`getNextEntryId`() Gets the next available Id.

Methods inherited from class java.lang.`Object`
`clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

entryID
`protected long` **`entryID`**
The id number of the entry.

---

unusedEntriesStack
`private static Stack<Long>` **`unusedEntriesStack`**
A stack that holds all the unused Ids.

---

noEntries
`private static long` **`noEntries`**
The entry count.

---

maxUsedEntryId
`private static long` **`maxUsedEntryId`**
The maximum Id that has been used

Constructor Detail

Entry
`public` **`Entry`**`()`
Default constructor. Assigns entryID by calling getNextEntryId()

Method Detail

getNextEntryId

```
public static long getNextEntryId()
```

Gets the next available Id.

Returns:

++maxUsedEntryId if the unusedEntriesStack is empty else the top of the stack.

---

getEntryID

```
public long getEntryID()
```

Gets the id of the entry.

Returns:

the entryID

---

finalize

```
protected void finalize()
                throws Throwable
```

Finalizes the Entry.Inserts its entryID to the stack so it can be used later on.

Overrides:

`finalize` in class `Object`

Throws:

`Throwable`

See Also:

Object.finalize()

rtree.util

Class InnerNode

```
java.lang.Object
  └ rtree.util.Entry
      └ rtree.util.Node
          └ rtree.util.InnerNode
```

```
public class InnerNode
extends Node
```

The representation of an InnerNode inside an R-Tree.

It has InnerNodes or LeafNodes as entries. It supports all the basic functionality of a Node.

Some of it is implemented in Node's level.
Among these Add/Remove an entry & split the node.

Author:

Sskalist

Field Summary

Fields inherited from class rtree.util.`Node`
`entries`, `entryCount`, `MaxEntries`, `MinEntries`, `minimumBoundaryRectangle`

Fields inherited from class rtree.util.`Entry`
`entryID`

Constructor Summary

`InnerNode()` Constructor.

Method Summary

void	`addEntry(Entry toBeAdded)` Adds the specified entry.
`Node`	`getEntry(int index)` Gets the specified entry.
boolean	`isLeaf()` Because it's not a leaf it always returns false.
`InnerNode`	`splitNode(Entry splitGuilty)`

	Splits this innerNode.

Methods inherited from class rtree.util.`Node`
`getEntryCount`, `getMinimumBoundryRectangle`, `isFull`, `performSplit`, `removeEntry`, `removeLastEntry`

Methods inherited from class rtree.util.`Entry`
`finalize`, `getEntryID`, `getNextEntryId`

Methods inherited from class java.lang.`Object`
`clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

InnerNode

`public` **`InnerNode`**`()`

Constructor. Calls the Node.Node() and initializes the Node.entries array.

Method Detail

addEntry

`public void` **`addEntry`**`(Entry toBeAdded)`

Adds the specified entry.

Specified by:

`addEntry` in class `Node`

Parameters:

toBeAdded - The entry to add.

See Also:

Node.addEntry(Entry)

---

getEntry

`public Node **getEntry**(int index)`

Gets the specified entry.

Overrides:

`getEntry` in class `Node`

Parameters:

index - The specified index

Returns:

The specified Node-entry.

See Also:

Node.getEntry(int)

---

splitNode

`public InnerNode **splitNode**(Entry splitGuilty)`

Splits this innerNode.

Specified by:

`splitNode` in class `Node`

Parameters:

splitGuilty - The entry that caused the split.

Returns:

The new InnerNode created.

See Also:

Node.splitNode(rtree.util.Entry), Node.performSplit(Node, Node, Entry)

---

isLeaf

`public boolean **isLeaf**()`

Because it's not a leaf it always returns false.

Specified by:

`isLeaf` in class `Node`

Returns:

false.

See Also:

Node.isLeaf()

rtree.util

Class LeafNode

```
java.lang.Object
  └ rtree.util.Entry
      └ rtree.util.Node
          └ rtree.util.LeafNode
```

```
public class LeafNode
extends Node
```

The representation of an LeafNode inside an R-Tree.

It has Points as entries. It supports all the basic functionality of a Node. Some of it is implemented in Node's level.

Among these Add/Remove an entry & split the node.

Author:

Sskalist

Field Summary

private boolean	`pageChanged` States whether the page has changed or not.
private static long	`pageCount` Keeps track of how many pages have been created.
private boolean	`pageLoaded` States whether the page is loaded or not.
private long	`pageNumber` The number of the page assigned to the this LeafNode.

Fields inherited from class rtree.util.`Node`

entries, entryCount, MaxEntries, MinEntries, minimumBoundaryRectangle

Fields inherited from class rtree.util.Entry
entryID

Constructor Summary

LeafNode() Constructor.

Method Summary

void	addEntry(Entry toBeAdded) Adds an entry to the leaf.
Point	getEntry(int index) Gets the entry at the specified index.
boolean	isLeaf() Because it's a leaf it always returns true.
private boolean	isPageLoaded() Check whether the page is loaded or not.
void	loadPage() Loads page of this leaf.
Entry	removeLastEntry() Removes the last entry.
LeafNode	splitNode(Entry splitGuilty) Splits a leafNode.
void	unloadPage() Unloads the page.

Methods inherited from class rtree.util.Node
getEntryCount, getMinimumBoundryRectangle, isFull, performSplit, removeEntry

Methods inherited from class rtree.util.Entry
finalize, getEntryID, getNextEntryId

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

pageCount

```
private static long pageCount
```

Keeps track of how many pages have been created.

---

pageNumber

```
private long pageNumber
```

The number of the page assigned to the this LeafNode.

---

pageLoaded

```
private boolean pageLoaded
```

States whether the page is loaded or not.

---

pageChanged

```
private boolean pageChanged
```

States whether the page has changed or not.

Constructor Detail

LeafNode

`public` **`LeafNode`**`()`

Constructor. Calls the Node.Node() and initializes the Node.entries array. Also assigns a page number to the leaf and allocates the necessary space to the disk.

Method Detail

addEntry

`public void` **`addEntry`**`(Entry toBeAdded)`

Adds an entry to the leaf.

The page must be loaded! Otherwise a runtime error is thrown.

Specified by:

`addEntry` in class `Node`

Parameters:

toBeAdded - The entry to add.

See Also:

Node.addEntry(Entry)

---

splitNode

`public LeafNode` **`splitNode`**`(Entry splitGuilty)`

Splits a leafNode. Alters the current leaf and return the newly created.

Specified by:

`splitNode` in class `Node`

Parameters:

splitGuilty - The entry that causes the leaf to split.

Returns:

The leafNode created.

---

removeLastEntry

`public Entry` **`removeLastEntry`**`()`

Removes the last entry.

The page must be loaded! Otherwise a runtime error is thrown.

Overrides:

`removeLastEntry` in class `Node`

Returns:

The removed entry.

See Also:

Node.removeLastEntry()

---

loadPage

`public void` **`loadPage`**`()`

Loads page of this leaf.

---

unloadPage

`public void` **`unloadPage`**`()`

Unloads the page. If it hasn't changed or it isn't loaded it has no effect.

---

isPageLoaded

`private boolean` **`isPageLoaded`**`()`

Check whether the page is loaded or not.

Returns:

pageLoaded

---

getEntry

`public Point` **`getEntry`**`(int index)`

Gets the entry at the specified index.

The page must be loaded! Otherwise a runtime error is thrown.

Overrides:

`getEntry` in class `Node`

Parameters:

index - The index of the required entry.

Returns:

the specified entry.

See Also:

Node.getEntry(int)

---

isLeaf

`public boolean isLeaf()`

Because it's a leaf it always returns true.

Specified by:

`isLeaf` in class `Node`

Returns:

true.

See Also:

Node.isLeaf()

rtree.util

Class Node

```
java.lang.Object
  └ rtree.util.Entry
      └ rtree.util.Node
```

Direct Known Subclasses:

`InnerNode`, `LeafNode`

`public abstract class Node`
`extends Entry`

This class represents the Nodes of the tree.

They can be either InnerNode or LeafNode.

It provides basic functionality that is common between the above classes.

Author:

Sskalist

See Also:

InnerNode, LeafNode

Field Summary

protected Entry[]	`entries` The entries of the Node...
protected	`entryCount`

int	The number of entries currently in the Node
static int	`MaxEntries` The maximum entries that a Node can host.
static int	`MinEntries` The minimum entries that a Node can host.
protected `Rectangle`	`minimumBoundaryRectangle` The minimum boundary rectangle for the entries of the Node

Fields inherited from class rtree.util.`Entry`
`entryID`

Constructor Summary

`Node()` The default constructor.

Method Summary

abstract void	`addEntry(Entry toBeAdded)` Adds an entry.
`Entry`	`getEntry(int index)` The entry at index
int	`getEntryCount()` Gets the number of entries.
`Rectangle`	`getMinimumBoundryRectangle()` Gets The minimumBoundryRectangle.
boolean	`isFull()` Checks if the Node is Full
abstract	`isLeaf()`

boolean	Checks whether the Node is a leaf or not.
protected static void	`performSplit`(Node originalNode, Node newNode, `Entry` splitGuilty) Performs the split.
private static void	`pickNext`(Node originalNode, Node newNode, int assignedToOriginal) Picks and assigns the next entry to be assigned to a group during a node split.
private static void	`pickSeeds`(Node originalNode, Node newNode, `Entry` splitGuilty) Guttman's pick Seeds.
`Entry`	`removeEntry`(int index) Removes the entry at index.
`Entry`	`removeLastEntry`() Removes the last entry.
abstract Node	`splitNode`(`Entry` splitGuilty) Splits the node.

Methods inherited from class rtree.util.`Entry`
`finalize`, `getEntryID`, `getNextEntryId`

Methods inherited from class java.lang.`Object`
`clone`, `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

entries

`protected Entry[]` **`entries`**

The entries of the Node... An InnerNode has Nodes as entries; a LeafNode has Points as entries;

---

entryCount

`protected int` **`entryCount`**

The number of entries currently in the Node

---

minimumBoundaryRectangle

`protected Rectangle` **`minimumBoundaryRectangle`**

The minimum boundary rectangle for the entries of the Node

MaxEntries

`public static final int` **`MaxEntries`**

The maximum entries that a Node can host. It is calculated based on Main.PAGE_SIZE and the Point#DIMENSIONS.

MinEntries

`public static final int` **`MinEntries`**

The minimum entries that a Node can host. Equals to MaxEntries/2

Constructor Detail

Node

`public` **`Node`**`()`

The default constructor. It calls the above constructor to assign the EntryId which is inherited and initializes the table of Entries with length equal to MaxEntries

Method Detail

addEntry

`public abstract void` **`addEntry`**`(Entry toBeAdded)`

Adds an entry. The way is handled lower on the class hierarchy.

Parameters:

toBeAdded - The entry to add.

---

splitNode

`public abstract Node` **`splitNode`**`(Entry splitGuilty)`

Splits the node. Alters the current and returns the new one.The way is handled lower on the class hierarchy.

Parameters:

splitGuilty - The entry that caused the node to split.

Returns:

The new Node

---

isLeaf

```
public abstract boolean isLeaf()
```

Checks whether the Node is a leaf or not.

Returns:

True if LeafNode else false.

See Also:

LeafNode.isLeaf(), InnerNode.isLeaf()

---

performSplit

```
protected static void performSplit(Node originalNode,
                                   Node newNode,
                                   Entry splitGuilty)
```

Performs the split. Removes entries from the orginalNode and assigns them to the newNode

The split is performed according to the original Guttman's algorithm with linear cost.

Parameters:

originalNode - The Node that is full.

newNode - The new Node that will host some of the originalNode entries.

splitGuilty - The entry that caused originalNode to split.

---

pickSeeds

```
private static void pickSeeds(Node originalNode,
                              Node newNode,
                              Entry splitGuilty)
```

Guttman's pick Seeds. It picks two entries (one for each node) and assigns them.

Parameters:

originalNode - The Node that is full.

newNode - The new Node that will host some of the originalNode entries.

splitGuilty - The entry that caused originalNode to split.

---

pickNext

```
private static void pickNext(Node originalNode,
                             Node newNode,
                             int assignedToOriginal)
```

Picks and assigns the next entry to be assigned to a group during a node split.

---

removeEntry

```
public Entry removeEntry(int index)
```

Removes the entry at index.

Parameters:

index - The specified index

Returns:

If index <= entryCount returns the removed Entry; otherwise null.

---

removeLastEntry

```
public Entry removeLastEntry()
```

Removes the last entry.

Returns:

The removed entry.

---

getEntry

```
public Entry getEntry(int index)
```

The entry at index

Parameters:

index - The specified index

Returns:

If is inside of bounds, the specified entry else throws RuntimeException

---

getMinimumBoundryRectangle

```
public Rectangle getMinimumBoundryRectangle()
```

Gets The minimumBoundryRectangle.

Returns:

the minimumBoundryRectangle

---

isFull

`public boolean isFull()`

Checks if the Node is Full

Returns:

true in case is full; else false.

---

getEntryCount

`public int getEntryCount()`

Gets the number of entries.

Returns:

the entryCount.

---

rtree.util

Class Point

```
java.lang.Object
  └ rtree.util.Entry
      └ rtree.util.Point
```

`public class Point`
`extends Entry`

Author:

Sskalist

Field Summary

private double[]	`coordinates` An array containing the coordinate for each of the #DIMENSIONS
private long	`pointID` The pointID.

Fields inherited from class rtree.util.`Entry`

`entryID`

Constructor Summary

`Point`(double[] coordinates) Constructs a point with the given coordinates having pointID equal to Entry.entryID.
`Point`(long pointID, double[] coordinates) Constructs a point with the given coordinates and the given pointID.

Method Summary

double	`distance`(`Point` from) Calculates the Euclidian Distance between this Point and the Point passes as parameter.
double	`getCoordinate`(int index) Gets the coordinate at the specified index.
double[]	`getCoordinates`() Clones the coordinates
double[]	`getCoordinatesNoClone`() Gets the coordinates array without cloning it.
long	`getPointID`() Gets the id of the point.
`String`	`toString`() Creates a string containing the pointID and using Arrays.toString(double[]) to transform the coordinates array.The String representation is in the form: PointID:10 Coordinates:[1.2,3.4]

Methods inherited from class rtree.util.`Entry`
`finalize`, `getEntryID`, `getNextEntryId`

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

coordinates

```
private double[] coordinates
```

An array containing the coordinate for each of the #DIMENSIONS

---

pointID

```
private long pointID
```

The pointID. If not specified it is assigned the value of Entry.entryID

Constructor Detail

Point

```
public Point(double[] coordinates)
```

Constructs a point with the given coordinates having pointID equal to Entry.entryID.

Parameters:

coordinates - The coordinates of the point.

Point

```
public Point(long pointID,
             double[] coordinates)
```

Constructs a point with the given coordinates and the given pointID.

Parameters:

pointID - The desired pointID.

coordinates - The coordinates of the point.

Method Detail

getCoordinates

```
public double[] getCoordinates()
```

Clones the coordinates

Returns:

A clone of the coordinates

---

getCoordinate

`public double` **`getCoordinate`**`(int index)`

Gets the coordinate at the specified index.

Parameters:

index -

Returns:

The coordinate at index.

---

getCoordinatesNoClone

`public double[]` **`getCoordinatesNoClone`**`()`

Gets the coordinates array without cloning it.

Returns:

the coordinates

---

getPointID

`public long` **`getPointID`**`()`

Gets the id of the point.

Returns:

the pointID

---

distance

`public double` **`distance`**`(Point from)`

Calculates the Euclidian Distance between this Point and the Point passes as parameter.

Parameters:

from - The end-point.

Returns:

The Euclidian Distance.

toString

`public String toString()`

Creates a string containing the pointID and using Arrays.toString(double[]) to transform the coordinates array.The String representation is in the form:

PointID:10 Coordinates:[1.2,3.4]

Overrides:

`toString` in class `Object`

Returns:

A String representation of the point.

See Also:

Arrays.toString(double[])

rtree.util

Class Rectangle

```
java.lang.Object
  └ rtree.util.Rectangle
```

`public class Rectangle`
`extends Object`

This is a basic representation of a multidimensional rectangle. It consists of two arrays, min and max, containing the minimum and max values respectively.

Author:

Sskalist

Field Summary

private double[]	`max` Array containing the maximum value for each dimension
private double[]	`min` Array containing the minimum value for each dimension

Constructor Summary

`Rectangle()` Default Constructor.
`Rectangle(double[] min, double[] max)` Constructor.

Method Summary

void	`add(Entry entry)` Adds the entry.
private void	`add(Point toBeAdded)` Computes the union of this rectangle and the passed point, storing the result in this rectangle.
private void	`add(Rectangle rectangle)` Computes the union of this rectangle and the passed rectangle, storing the result in this rectangle.
double	`area()` Compute the area of this rectangle.
boolean	`contains(Rectangle rectangle)` Determine whether this rectangle contains the specified rectangle
double	`distance(Point origin)` Return the distance between this rectangle and the passed point.
double	`enlargement(Entry entry)` Calculates the how much the rectangle will grow if the entry would be added.
double	`enlargement(Point point)` Calculate the area by which this rectangle would be enlarged if the passed point was added to it.
double	`enlargement(Rectangle rectangle)` Calculate the area by which this rectangle would be enlarged if added to the passed rectangle.
boolean	`equals(Object obgject)`

	Determine whether this rectangle is equal to a given object.
double	`getMaximum`(int index) Gets the specified value of the max array
double[]	`getMaximumArray`() Gets the max array.
double	`getMininimum`(int index) Gets the specified value of the min array
double[]	`getMininimumArray`() Gets the min array.
void	`set`(double[] min, double[] max) Sets the size of the rectangle.
`String`	`toString`() Return a string representation of this rectangle, in the form: [1.2, 3.4],[5.6, 7.8]

Methods inherited from class java.lang.`Object`
`clone`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

min

`private double[] min`

Array containing the minimum value for each dimension

---

max

`private double[] max`

Array containing the maximum value for each dimension

Constructor Detail

Rectangle

```
public Rectangle()
```

Default Constructor.

Initializes the min array with the Double.POSITIVE_INFINITY value and the max array with the Double.NEGATIVE_INFINITY value.

Rectangle

```
public Rectangle(double[] min,
                 double[] max)
```

Constructor. Constructs an new Rectangle according to the passed arrays using set(double[], double[])

Parameters:

min - array containing the minimum value for each dimension; eg { min(x), min(y) }

max - array containing the maximum value for each dimension; eg { max(x), max(y) }

Method Detail

set

```
public void set(double[] min,
                double[] max)
```

Sets the size of the rectangle.

Parameters:

min - Array containing the minimum value for each dimension

max - Array containing the maximum value for each dimension

Throws:

`RuntimeException` - if the length of either **min** or **max** differs from Rectangle#DIMENSIONS

---

distance

```
public double distance(Point origin)
```

Return the distance between this rectangle and the passed point. If the rectangle contains the point, the distance is zero.

Parameters:

origin - Point to find the distance to

Returns:

distance between this rectangle and the passed point.

---

enlargement

```
public double enlargement(Entry entry)
```

Calculates the how much the rectangle will grow if the entry would be added. The rectangle is not altered.

Parameters:

entry - The entry that would be added

Returns:

The enlargement. In case of error returns Double.POSITIVE_INFINITY

See Also:

enlargement(Point), enlargement(Rectangle)

---

enlargement

```
public double enlargement(Point point)
```

Calculate the area by which this rectangle would be enlarged if the passed point was added to it. The rectangle is not altered.

Parameters:

point - Point to be added to this rectangle, in order to compute the enlargement.

---

enlargement

```
public double enlargement(Rectangle rectangle)
```

Calculate the area by which this rectangle would be enlarged if added to the passed rectangle. Neither rectangle is altered.

Parameters:

rectangle - Rectangle to be added to this rectangle, in order to compute the enlaregment.

---

area

```
public double area()
```

Compute the area of this rectangle.

Returns:

The area of this rectangle

---

add

```
public void add(Entry entry)
```

Adds the entry. In case the entry is not an instance of Point or Node throws an Exception.

Parameters:

entry - The entry to be added.

Throws:

`RuntimeException`

See Also:

add(Point), add(Rectangle)

---

add

```
private void add(Rectangle rectangle)
```

Computes the union of this rectangle and the passed rectangle, storing the result in this rectangle.

Parameters:

rectangle - Rectangle to add to this rectangle

---

add

```
private void add(Point toBeAdded)
```

Computes the union of this rectangle and the passed point, storing the result in this rectangle.

Parameters:

toBeAdded - Rectangle to add to this rectangle

---

getMininimum

```
public double getMininimum(int index)
```

Gets the specified value of the min array

Parameters:

index - The index of the required value.

Returns:

The required value

---

getMininimumArray

`public double[] getMininimumArray()`

Gets the min array.

Returns:

The min array.

---

getMaximum

`public double getMaximum(int index)`

Gets the specified value of the max array

Parameters:

index - The index of the required value.

Returns:

The required value.

---

getMaximumArray

`public double[] getMaximumArray()`

Gets the max array.

Returns:

The max array.

---

equals

`public boolean equals(Object obgject)`

Determine whether this rectangle is equal to a given object. Equality is determined by the bounds of the rectangle.

Overrides:

`equals` in class `Object`

Parameters:

obgject - - The object to compare with this rectangle.

---

contains

`public boolean contains(Rectangle rectangle)`

Determine whether this rectangle contains the specified rectangle

Parameters:

rectangle - The rectangle that might be contained by this rectangle

Returns:

true if this rectangle contains the passed rectangle; otherwise false

---

toString

`public String` **`toString`**`()`

Return a string representation of this rectangle, in the form: [1.2, 3.4],[5.6, 7.8]

Overrides:

`toString` in class `Object`

Returns:

String String representation of this rectangle.