ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ

Φοιτητές
Καρβούνης Σπυρίδων - 3928
Νικόλαος Βογιατζής - 3952

Γενικά

Η εργασία αποτελείται από 16 κλάσεις. Παρακάτω περιγράφονται οι κλάσεις καθώς και βασικές λειτουργίες τους μέσω των μεθόδων τους. Δεν περιγράφονται τελείως αναλυτικά στην αναφορά αυτή καθώς υπάρχει και το απαραίτητο JavaDoc καθώς και inline σχόλια στον κώδικα. Σημειώνεται ότι ο κώδικας τρέχει μόνο για περιπτώσεις που το πλήθος των διαστάσεων είναι 2.

Κλάση Main

Η κλάση αυτή είναι η κύρια κλάση διαχείρισης του project στην οποία υλοποιείται το UI καθώς και όλες οι απαραίτητες κλήσεις για τις λειτουργίες τις εργασίας.
Πιο συγκεκριμένα από την κλάση αυτή, καλούνται συναρτήσεις επεξεργασίας του OSM αρχείου, δημιουργίας του datafile.txt , δημιουργίας του R*tree και του καταλόγου του και αποθήκευση του τελευταίου στο indexfile.txt. Έπειτα υλοποιούνται λειτουργίας εισαγωγής και διαγραφής νέου στοιχείου καθώς και εκτέλεση όλων των ερωτημάτων (ερωτήματα περιοχών, κ-γειτόνων, κορυφογραμμής και μαζικής κατασκευής του δέντρου). Το κάθε ερώτημα εκτελείται σε συγκεκριμένη μέθοδο ώστε να είναι πιο ευανάγνωστος ο κώδικας αλλά και να ελεγχθεί το κάθε ερώτημα ανεξάρτητα, εύκολα.

Κλάση FileManager

Στην κλάση αυτή υλοποιούνται όλες οι λειτουργίες που έχουν να κάνουν με δημιουργία, διάβασμα ή γράψιμο από/σε αρχείο. Πιο συγκεκριμένα, υλοποιούνται οι παρακάτω μέθοδοι που περιγράφονται περιληπτικά:

- Μέθοδος **create_datafile():** Η μέθοδος αυτή φορτώνει δεδομένα από ένα OSM αρχείο με σημεία 2 διαστάσεων με σκοπό να αποθηκευτούν στο αρχείο datafile.txt ως μια σειρά από bytes. Κάθε εισαγωγή περιλαμβάνει 24 bytes,

άρα για να δημιουργήσουμε blocks με χωρητικότητα 32KB πρέπει να έχουμε 1365 εισαγωγές. Χρησιμοποιώντας έναν DocumentBuilder διαβάζουμε το .osm αρχείο και φιλτράρουμε τις εγγραφές του με την κλάση NodeList, ώστε να κρατήσουμε αυτές με tag_name "node" . Μαζί με τις τιμές id, lat, lon αποθηκεύουμε τα δεδομένα ως μία σειρά από bytes στο datafile.txt. Παράλληλα εισάγονται τα σημεία στο R*tree και τέλος καλείται η επόμενη μέθοδος για την αποθήκευση του καταλόγου στο indexfile.txt.

- Μέθοδος **write_tree_in_indexfile(Tree myRtree):** Η συγκεκριμένη μέθοδος καλείται αμέσως μετά τη δημιουργία του datafile.txt και είναι υπεύθυνη για τη δημιουργία και αποθήκευση του καταλόγου του R*tree. Με ένα FileOutputStream ανοίγουμε το αρχείο indexfile και τέλος αποθηκεύει τον κατάλογο στο αρχείο ως αντικείμενο χρησιμοποιώντας έναν ObjectOutputStream.

- Μέθοδος **read_tree_from_indexfile():** Η μέθοδος αυτή καλείται από την main κατά την εκκίνηση του προγράμματος ώστε να φορτώσει και να επιστρέψει τον κατάλογο του R*tree.

- Μέθοδος **add_location(Location location, ArrayList<Location> locations):**
  Οι παράμετροι που χρησιμοποιούνται είναι οι: location που αναφέρεται σε μία νέα τοποθεσία προς εισαγωγή και η λίστα με όλες τις υπάρχουσες εισαγωγές – locations. Χρησιμοποιείται από τη main κατά τη λειτουργία εισαγωγής νέου στοιχείου.

- Μέθοδος **parse_points():** Η μέθοδος αυτή σκανάρει το αρχείο osm και αποθηκεύει τα σημεία σε μία λίστα point_list. Χρησιμοποιείται από την κλάση BottomUp κατά τη λειτουργία μαζικής κατασκευής του R*tree.

Κλάση Data

Αυτή η κλάση χρησιμοποιείται για να φορτώσουμε και να διαβάσουμε το ήδη υπάρχον αρχείο datafile. Υλοποιούνται οι δύο παρακάτω συναρτήσεις:

- Μέθοδος **read_datafile():** Η συνάρτηση αυτή αποτελεί διαχειριστική συνάρτηση καθώς χρησιμοποιείται για να διαβάσει τις τοποθεσίες από το datafile.txt και να τις επιστρέψει σε μία λίστα.
- Μέθοδος **get_block_loc():** Η συνάρτηση που παίρνει ως ορίσματα ένα συγκεκριμένο block_id και slot_id και επιστρέφει την τοποθεσία στην οποία δείχνουν.

Κλάση Point

Η κλάση αυτή αναπαριστά ένα σημείο στις δύο διαστάσεις. Οι ιδιότητες της κλάσης είναι οι block_id, slot_id, το γεωγραφικό πλάτος της τοποθεσίας lat, το γεωγραφικό μήκος lon και την απόσταση distance. Υλοποιούνται συναρτήσεις εύρεσης της απόστασης Manhattan και δύο συναρτήσεις σύγκρισης των lat και lon συντεταγμένων.

Κλάση Location

Η κλάση αυτή αναπαριστά μια τοποθεσία με τις τιμές της lat και lon που παίρνουμε από το map.osm αρχείο. Περιλαμβάνει κι άλλες ιδιότητες όπως το location_id και distance και υλοποιεί μία συνάρτηση που υπολογίζει την απόσταση Manhattan μεταξύ δύο σημείων και μια που συγκρίνει δύο τοποθεσίες βάσει των αποστάσεών τους.

Κλάση Rectangle

Η κλάση αυτή αναπαριστά ένα ορθογώνιο. Ως μεταβλητές χρησιμοποιεί τις μικρότερες και μεγαλύτερες τιμές στον άξονα x και y. Περιέχει μεθόδους οι οποίες έχουν τις εξής λειτουργίες: α) επιστρέφει όλες τις τιμές που χαρακτηρίζουν το ορθογώνιο, β) υπολογισμούς για εμβαδό , γ) υπολογισμούς για περίμετρο , δ) νέο εμβαδό όταν μπει καινούργιο σημείο στο ορθογώνιο , ε) ενημέρωση διαστάσεων όταν εισαχθεί νέο σημείο, ζ) αλλαγή συντεταγμένων

ορθογωνίου λόγω εισαγωγής νέου κόμβου, η) επικάλυψη μεταξύ δυο ορθογωνίων .

Κλάση TreeNode

Η κλάση αυτή αναπαριστά έναν κόμβο στο R*tree. Ως ιδιότητες χρησιμοποιεί 2 λίστες, μία για τα παιδιά ενός non-leaf κόμβου και μία λίστα με κόμβους φύλλα. Ακόμη, χρησιμοποιείται μία Rectangle μεταβλητή, ο μέγιστος αριθμός εισόδων για έναν κόμβο (max) και τέλος μία μεταβλητή που δείχνει την απόσταση ενός κόμβου από ένα σημείο. Οι μέθοδοι που υλοποιούνται είναι οι εξής (παραλείπονται setters και getters) : Μία που επιστρέφει αν ένας κόμβος είναι φύλλο ή όχι (isLeaf()), μια που προσθέτει έναν καινούργιο κόμβο ως παιδί (add_new_child_node()), μια που προσθέτει ένα σημείο (add_new_point()) και μια που συγκρίνει δύο κόμβους σύμφωνα με τις τιμές των ορθογωνίων τους ως προς x και μια ως προς y (CompareLat() και CompareLon()).

Κλάση Tree

Η κλάση αυτή είναι υπεύθυνη για την δημιουργία, εισαγωγή και διαγραφή του R*tree. Χρησιμοποιούνται οι παρακάτω ιδιότητες:
root – η ρίζα του δέντρου
max - η μέγιστη χωρητικότητα ενός κόμβου
min – η ελάχιστη χωρητικότητα ενός κόμβου
node_path – το μονοπάτι που ακολουθήθηκε στο δέντρο κατά την εισαγωγή ενός σημείου.
Οι μέθοδοι της κλάσης αυτής είναι οι παρακάτω:

- **ChooseSubtree():** Υλοποιήθηκε με βάση το Paper που δόθηκε στο μάθημα. Έχει ως παραμέτρους έναν κόμβο του δέντρου και ένα σημείο που θέλουμε να προσθέσουμε στο δέντρο. Στην περίπτωση που ο κόμβος είναι φύλλο επιστρέφεται ο ίδιος και τερματίζεται η διαδικασία. Διαφορετικά, καλείται αναδρομικά η μέθοδος μέχρι να φτάσει σε κόμβο φύλλο. Κατά την αναδρομική διαδικασία υπολογίζεται η επέκταση του ορθογωνίου και επιλέγεται αυτό με την μικρότερη επικάλυψη.
- **ChooseSplitAxis():** Δέχεται ως παράμετρο έναν κόμβο τον οποίο διαχωρίζει και επιστρέφει μέσω της λίστας splitted_node. Υλοποιήθηκε σύμφωνα με το paper του μαθήματος αλλά υπάρχουν και inline σχόλια.

- **insert_in_Rtree():** Η μέθοδος αυτή υλοποιεί την διαδικασία εισαγωγής ενός νέου σημείου στο R*tree. Περιέχονται inline σχόλια για την περιγραφή της διαδικασίας που ακολουθείται.
- **delete_from_tree():** Στη μέθοδο αυτή υλοποιείται η διαδικασία διαγραφής ενός κόμβου από το R*tree βάσει του id του.

Τέλος για την εκτέλεση και την επεξεργασία των ερωτημάτων περιοχής και k γειτόνων, χρησιμοποιούνται οι κλάσεις QueriesWithoutIndex (για τα ερωτήματα χωρίς τη χρήση καταλόγου) και QueriesWithIndex (για τα ερωτήματα που κάνουν χρήση του καταλόγου). Για τα ερωτήματα κορυφογραμμής χρησιμοποιείται η κλάση Skyline και SkylineNode καθώς και οι MaxHeap και MinHeap. Δεν αναλύονται ενδελεχώς στην παρούσα αναφορά καθώς το Javadoc είναι αρκετό. Παρακάτω υπάρχουν αποτελέσματα εκτέλεσης του κώδικα για διάφορες περιπτώσεις.

Παραδείγματα εκτέλεσης

1. **Εισαγωγή:** Κατά την εισαγωγή ενός νέου σημείου υπολογίζεται η μικρότερη επικάλυψη μεταξύ κόμβων και επιτυγχάνεται η ισορροπία στο δέντρο.

```
----------------->   Εισαγωγή νέου στοιχείου <----------------------
    id:35836525   latitude= 38.444235   longitude= 23.853263  distance= 1.7976931348623157E308
Η νέα τοποθεσία προστέθηκε επιτυχώς!
Ο χρόνος που χρειάστηκε: 29ms
-----------------------------------------------------------------------
```

2. **Διαγραφή:** Δοκιμή διαγραφής του κόμβου που προστέθηκε προηγουμένως.

```
----------------->   Διαγραφή του στοιχείου  <----------------------
    id:35836525   latitude= 38.444235   longitude= 23.853263  distance= 1.7976931348623157E308
Η τοποθεσία διαγράφηκε επιτυχώς!
Ο Χρόνος που χρειάστηκε: 1585ms
-----------------------------------------------------------------------
```

3. **Ερωτήματα περιοχής:** Τα ερωτήματα περιοχής υλοποιούνται με αλλά και χωρίς τη χρήση του καταλόγου.

```
-----------------> Ερώτημα περιοχής χωρίς χρήση καταλόγου! <-----------------------
    range: 5.0 lat: 38.444235 lon: 23.853263
Ο χρόνος που χρειάστηκε το ερώτημα: 1ms

-----------------> Ερώτημα περιοχής με χρήση καταλόγου!    <-----------------------
        range: 5.0 lat: 38.444235 lon: 23.853263
Ο χρόνος που χρειάστηκε το ερώτημα: 826ms
```

4. **Ερωτήματα kn-γειτόνων:** Κι εδώ η εκτέλεση των ερωτημάτων γίνεται με και χωρίς κατάλογο.

```
------------------>  Ερώτημα εύρεσης k κοντινότερων γειτόνων χωρίς κατάλογο για k: 5 lat: 38.444235 lon: 23.853263  <---------------
id:2259883238  latitude= 38.4028363  longitude= 23.8002353  distance= 0.09442639999999969
id:2259883190  latitude= 38.4027962  longitude= 23.8002161  distance= 0.0944856999999999
id:2259883200  latitude= 38.4027373  longitude= 23.8001821  distance= 0.0945785999999984
id:2259883160  latitude= 38.4026722  longitude= 23.8001311  distance= 0.09469469999999802
id:2259883153  latitude= 38.4026115  longitude= 23.800059  distance= 0.0948274999999974
Ο χρόνος που χρειάστηκε το ερώτημα: 4ms

------------------>  Ερώτημα εύρεσης k κοντινότερων γειτόνων με κατάλογο για k: 5 lat: 38.444235 lon: 23.853263  <---------------
id:2259883153  latitude= 38.4026115  longitude= 23.800059  distance= 0.0948274999999974
id:2259883160  latitude= 38.4026722  longitude= 23.8001311  distance= 0.09469469999999802
id:2259883200  latitude= 38.4027373  longitude= 23.8001821  distance= 0.0945785999999984
id:2259883190  latitude= 38.4027962  longitude= 23.8002161  distance= 0.0944856999999999
id:2259883238  latitude= 38.4028363  longitude= 23.8002353  distance= 0.09442639999999969
Ο χρόνος που χρειάστηκε το ερώτημα: 5ms
```

5. **Ερώτημα κορυφογραμμής (Skyline):**

```
---------------------------->   Ερώτημα κορυφογραμμής!  <----------------------------
 point : BlockID=1 slotID=34
 point : BlockID=1 slotID=30
 point : BlockID=1 slotID=567
 point : BlockID=1 slotID=569
 point : BlockID=1 slotID=48
 point : BlockID=1 slotID=61
 point : BlockID=1 slotID=49
 point : BlockID=1 slotID=575
 point : BlockID=1 slotID=578
Ο χρόνος που χρειάστηκε το ερώτημα: 8ms
```

6. **Μαζική κατασκευή του δέντρου (BottomUp):**

```
----------------------> Υλοποίηση μαζικής κατασκευής δέντρου BottomUp  <------------------
Ο χρόνος που χρειάστηκε το ερώτημα: 268ms
```

ΣΥΓΚΕΝΤΡΩΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Εκτέλεση ερωτημάτων κοντινότερων γειτόνων με και χωρίς τη βοήθεια του καταλόγου:

	Με κατάλογο	Χωρίς κατάλογο
k=5	5	4
k=20	16	15
k=50	32	20
k=100	62	21
k=250	113	33

Εκτέλεση ερωτημάτων περιοχής με και χωρίς τη βοήθεια του καταλόγου:

	Με χρήση καταλόγου	Χωρίς κατάλογο
range=5	825	50
range=20	885	52
range=25	766	54
range=50	791	53
range=100	803	51

Σχολιασμός Αποτελεσμάτων

Εξετάζοντας τους χρόνους εκτέλεσης, είναι σαφές ότι τα ερωτήματα που εκτελούνται χωρίς τη βοήθεια του καταλόγου χρειάζονται πολύ λιγότερο χρόνο από εκείνα που εκτελούνται με τη χρήση του καταλόγου. Αυτό μπορεί να οφείλεται στον μεγάλο όγκο δεδομένων που προκύπτει. Με έναν κατάλογο, είναι πολύ πιθανό ότι καθώς αυξάνεται ο αριθμός των αποτελεσμάτων, αυξάνεται και ο αριθμός των προσβάσεων σε μπλοκ που πληρώνω, με αποτέλεσμα να αυξάνονται οι χρόνοι εκτέλεσης για κάθε ερώτημα.