



# 2<sup>η</sup> Εργασία

**Διαχείριση Σύνθετων Δεδομένων**  
02/05/2023 – Εαρινό Εξάμηνο 2023

By

ΤΟΔΡΙ ΑΓΓΕΛΟ (ΑΜ. 3090)  
Καθηγητής: Ν. Μαμουλής

## ΠΕΡΙΕΧΟΜΕΝΑ

Χωρικά Δεδομένα – Λίγα λόγια σχετικά με αυτά .....	2
<b>Σημείωση για τον αξιολογητή.....</b>	<b>2</b>
Μέρος 1: Δημιουργία ευρετηρίου – indexing spatial data.....	2
Εξήγηση προγράμματος για το μέρος 1.....	3
Ανάγνωση εγγράφων από αρχείο CSV – ΟΝΟΜΑ ΜΕΘΟΔΟΥ <b>readRecordsFromFile</b> .....	4
Δημιουργία χωρικού ευρετηρίου - ΟΝΟΜΑ ΜΕΘΟΔΟΥ: <b>createGrid</b> .....	5
Υπολογισμός των ορίων του πλέγματος – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: <b>getGridLimits</b> .....	6
Δημιουργία αρχείων ευρετηρίων – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: <b>writeGridToFile</b> .....	7
Μέρος 2 - 3: Επιλογή παραθύρων ερωτημάτων σε δομές πλέγματος .....	8
Known issues – Γνωστά προβλήματα.....	8
Εξήγηση προγράμματος για το μέρος 2.....	8
Main μέθοδος.....	9
Φόρτωση καταλόγου πλέγματος – ΟΝΟΜΑΤΑ ΜΕΘΟΔΩΝ: <b>loadGrid</b> – <b>loadGridDirectory</b> .....	10
Διάβασμα αρχείου ερωτημάτων παραθύρου ( window query ) – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: <b>readQueryfile</b> .....	12
Εκτέλεση ερωτημάτων παραθύρου ( window query ) – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: <b>performQuery</b> .....	13
Ιδιαιτερότητες και edge cases που υλοποιήθηκαν για το μέρος 2 – Ζητήθηκαν απο την εκφώνηση .....	14
Μέρος 3 – Σχολιασμός & Περιγραφή & Κώδικας.....	14
Εκτελέσιμα output για το μέρος 2 & 3 της άσκησης και προεπισκοπήσεις των grid output files .....	16


## ΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ – ΛΙΓΑ ΛΟΓΙΑ ΣΧΕΤΙΚΑ ΜΕ ΑΥΤΑ

Τα χωρικά δεδομένα αναφέρονται σε οποιαδήποτε δεδομένα που έχουν γεωγραφικό ή τοπικό στοιχείο. Αυτό σημαίνει ότι τα δεδομένα συνδέονται με ένα συγκεκριμένο μέρος ή τοποθεσία στην επιφάνεια της Γης. Τα χωρικά δεδομένα μπορούν να περιλαμβάνουν ένα ευρύ φάσμα πληροφοριών, όπως χάρτες, δορυφορικές εικόνες, συντεταγμένες GPS και άλλους τύπους γεωγραφικών δεδομένων.

Τα χωρικά δεδομένα συνήθως αναπαρίστανται σε διάφορες μορφές, συμπεριλαμβανομένων διανυσματικών δεδομένων (όπως σημεία, γραμμές και πολύγωνα), δεδομένων ράστερ (όπως δορυφορικές εικόνες ή δεδομένα υψομέτρων) και δεδομένων σε μορφή πίνακα (όπως δεδομένα που είναι οργανωμένα σε πίνακα με γεωγραφικές συντεταγμένες).

Τα χωρικά δεδομένα χρησιμοποιούνται σε ένα ευρύ φάσμα εφαρμογών, όπως ο αστικός σχεδιασμός, η διαχείριση φυσικών πόρων, ο σχεδιασμός μεταφορών, η διαχείριση εκτάκτων αναγκών και πολλοί άλλοι τομείς. Η ανάλυση χωρικών δεδομένων μπορεί να μας βοηθήσει να κατανοήσουμε καλύτερα τα πρότυπα και τις σχέσεις στο φυσικό και δομημένο περιβάλλον και μπορεί να μας βοηθήσει να λάβουμε τεκμηριωμένες αποφάσεις σχετικά με τον τρόπο διαχείρισης και χρήσης των πόρων μας.

## ΣΗΜΕΙΩΣΗ ΓΙΑ ΤΟΝ ΑΞΙΟΛΟΓΗΤΗ

 **READ ME FIRST:** Για δικιά σας διευκόλυνση όλος ο κώδικας της άσκησης και για τα 3 μέρη της εργασίας εξηγείται αναλυτικά και επιδεικνύεται στα παρακάτω Screenshots. **Πρακτικά μπορείτε εάν το επιθυμείτε να παραλείψετε** την αξιολόγηση των αρχείων .java καθώς τα screenshots καλύπτουν το μεγαλύτερο μέρος της έκτασης του κώδικα και για τα 3 αρχεία αναλυτικά. Αυτό το PDF δημιουργήθηκε με μεγάλη προσοχή και προσπάθεια για να δοθεί έμφαση στις λεπτομέρειες. Για του λόγου το αληθές μπορείτε να το εξακριβώσετε αυτό και οι ίδιοι. Καλή ανάγνωση/αξιολόγηση!

## ΜΕΡΟΣ 1: ΔΗΜΙΟΥΡΓΙΑ ΕΥΡΕΤΗΡΙΟΥ – INDEXING SPATIAL DATA

- Στο πρώτο μέρος της άσκησης, ζητήθηκε η υλοποίηση ενός χωρικού ευρετηρίου σε Java. Το ευρετήριο δημιουργείται με βάση ένα σύνολο εγγραφών, όπου κάθε εγγραφή αναπαριστά μια γραμμή σε ένα δισδιάστατο χώρο. Τα linestring αποθηκεύονται σε ένα αρχείο CSV (tiger\_roads.csv) με τη μορφή "x1 y1,x2 y2,...,xn yn", όπου x και y είναι οι συντεταγμένες κάθε σημείου στο linestring.
- Ο χωρικός δείκτης διαιρεί το χώρο σε ένα πλέγμα κελιών και αναθέτει κάθε linestring στα κελιά που επικαλύπτονται με το ελάχιστο ορθογώνιο οριοθέτησής του (MBR). Το MBR είναι το μικρότερο ορθογώνιο που περιέχει το linestring.

- Το πλέγμα (grid) αναπαρίσταται ως ένας χάρτης κλειδιών κελιών. Το κλειδί κελιού είναι μια συμβολοσειρά που αντιπροσωπεύει τις συντεταγμένες του κελιού στο πλέγμα, π.χ. "(0,0)", "(0,1)", κ.λπ.
- Ο κώδικας διαβάζει το αρχείο CSV, δημιουργεί το πλέγμα και γράφει τα αρχεία πλέγματος και καταλόγου στο δίσκο σε δυο αρχεία *grid.grd* και *grid.dir*.
- Τα αρχεία αυτά θα χρησιμοποιηθούν σε επόμενο μέρος της άσκησης για δημιουργία window selection query (ερωτήματα επιλογής παραθύρου)
- Το classpath που έχει οριστεί στα παραδόμενα αρχεία Java είναι του μηχανήματος εργασίας. Για λόγους διευκόλυνσης η άσκηση δουλεύεται μέσα στο περιβαλλον (project) IntelliJ άλλης άσκησης του μαθήματος του εξαμήνου.

```
"C:\\Users\\a.todhri\\IdeaProjects\\LyriSearch-
Lucene\\src\\main\\java\\assign2\\";
```

## ΕΞΗΓΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΓΙΑ ΤΟ ΜΕΡΟΣ 1

Το πρόγραμμα χρησιμοποιεί τη μέθοδο "main" και τέσσερις επιπλέον μεθόδους, καθώς και μια κλάση με το όνομα "Record", προκειμένου να επιτελέσει τη λειτουργικότητα που απαιτείται για το μέρος 1. Παρακάτω παρέχονται screenshots και περαιτέρω εξηγήσεις για τα διάφορα κομμάτια του κώδικα, όπου είναι απαραίτητο.

Συνολικά, δημιουργήθηκαν 5 μέθοδοι:

1. Μέθοδος main – κλήση επιμέρους μεθόδων
2. **readRecordsFromFile** – ανάγνωση εγγράφων από αρχείο csv
3. **createGrid** – δημιουργία χωρικού ευρετηρίου (spatial index)
4. **getGridLimits** – υπολογισμός των ορίων του πλέγματος (grid)
5. **writeGridToFile** – δημιουργία αρχείων ευρετηρίων

Η κλάση *Record* που δημιουργήθηκε έχει τα εξής πεδία (attributes):

```
public int identifier;
public double mbr_min_x;
public double mbr_min_y;
public double mbr_max_x;
public double mbr_max_y;
public List<double[]> points;
```

## ΑΝΑΓΝΩΣΗ ΕΓΓΡΑΦΩΝ ΑΠΟ ΑΡΧΕΙΟ CSV – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: READRECORDSFROMFILE

```
public static List<Record> readRecordsFromFile(String filename) throws IOException {
    int num_linestrings;
    List<Record> records = new ArrayList<Record>();

    // Open the file and read the number of linestrings
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    num_linestrings = Integer.parseInt(reader.readLine().trim());

    // Loop through the linestrings and parse them
    for (int i = 0; i < num_linestrings; i++) {
        String line = reader.readLine();
        String[] points_str = line.split(",");
        List<double[]> points = new ArrayList<double[]>();
        double mbr_min_x = Double.POSITIVE_INFINITY;
        double mbr_min_y = Double.POSITIVE_INFINITY;
        double mbr_max_x = Double.NEGATIVE_INFINITY;
        double mbr_max_y = Double.NEGATIVE_INFINITY;

        // Loop through the points in the linestring and parse them
        for (int j = 0; j < points_str.length; j++) {
            String[] point_str = points_str[j].split(" ");
            double x = Double.parseDouble(point_str[0]);
            double y = Double.parseDouble(point_str[1]);
            points.add(new double[]{x, y});

            // Update the MBR of the linestring
            if (x < mbr_min_x) mbr_min_x = x;
            if (y < mbr_min_y) mbr_min_y = y;
            if (x > mbr_max_x) mbr_max_x = x;
            if (y > mbr_max_y) mbr_max_y = y;
        }

        // Create a new Record object and add it to the list of records
        Record record = new Record(i, mbr_min_x, mbr_min_y, mbr_max_x, mbr_max_y, points);
        records.add(record);
    }
    reader.close();

    // Return the list of Records
    return records;
}
```

Εικόνα 1 – μέθοδος readRecordsFromFile ( csv )

### Σχόλια:

- Ο κώδικας διαβάζει ένα αρχείο CSV που περιέχει ένα σύνολο γραμμοσειρών (linestring) σε δισδιάστατο χώρο. Το filename ορίζεται στην κορυφή της κλάσης ( static final – it can be configurable )

- Κάθε γραμμή αναπαρίσταται ως ένας κατάλογος σημείων με τις συντεταγμένες x και y τους.
- Παρσάρει κάθε linestring στο αρχείο και αποθηκεύει τα σημεία της.
- Υπολογίζει το ελάχιστο ορθογώνιο περίβλημα (MBR) για κάθε linestring και το αποθηκεύει στο αντίστοιχο αντικείμενο Record.
- Επιστρέφει μια λίστα με όλες τις εγγραφές linestring στο αρχείο.

## ΔΗΜΙΟΥΡΓΙΑ ΧΩΡΙΚΟΥ ΕΥΡΕΤΗΡΙΟΥ - ΟΝΟΜΑ ΜΕΘΟΔΟΥ: CREATEGRID

```

/**
 * Creates a grid index for the set of Records.
 *
 * @param records The set of Records to index.
 * @param gridLimits The limits of the grid to create.
 * @return A Map from cell key to list of Record identifiers that intersect the cell.
 */
1 usage
2 public static Map<String, List<Integer>> createGrid(List<Record> records, double[] gridLimits) {
3     // Calculate the cell size of the grid
4     double cell_size_x = (gridLimits[2] - gridLimits[0]) / 10.0;
5     double cell_size_y = (gridLimits[3] - gridLimits[1]) / 10.0;
6
7     // Initialize the grid with empty lists
8     Map<String, List<Integer>> grid = new HashMap<String, List<Integer>>();
9
10    // Loop through each Record and assign it to the appropriate cells in the grid
11    for (Record record : records) {
12        // Calculate the range of cells that the Record intersects
13        int min_cell_x = (int) Math.floor((record.mbr_min_x - gridLimits[0]) / cell_size_x);
14        int min_cell_y = (int) Math.floor((record.mbr_min_y - gridLimits[1]) / cell_size_y);
15        int max_cell_x = (int) Math.floor((record.mbr_max_x - gridLimits[0]) / cell_size_x);
16        int max_cell_y = (int) Math.floor((record.mbr_max_y - gridLimits[1]) / cell_size_y);
17
18        // Add the Record identifier to each intersecting cell in the grid
19        for (int i = min_cell_x; i <= max_cell_x; i++) {
20            for (int j = min_cell_y; j <= max_cell_y; j++) {
21                String key = String.format("(%d,%d)", i, j);
22                List<Integer> ids = grid.get(key);
23                if (ids == null) {
24                    ids = new ArrayList<>();
25                    grid.put(key, ids);
26                }
27                ids.add(record.identifier);
28            }
29        }
30    }
31
32    // Return the grid index
33    return grid;
34 }

```

Εικόνα 2 – μέθοδος createGrid

#### Σχόλια:

- Δημιουργεί έναν χωρικό ευρετήριο για το σύνολο των εγγραφών Record, αποθηκεύεται στην δομή grid τύπου Map<String, List<Integer>>
- Υπολογίζει το μέγεθος των κελιών του χωρικού ευρετηρίου με βάση τα όρια του ευρετηρίου σχάρας, cell\_size\_x, cell\_size\_y.
- Επεξεργάζεται κάθε εγγραφή Record και την αντιστοιχεί στα κατάλληλα κελιά του χωρικού ευρετηρίου.
- Υπολογίζει τα κελιά που ταυτίζονται με το ορθογώνιο περίβλημα (MBR) κάθε Record, min\_cell\_x, min\_cell\_y, max\_cell\_x, max\_cell\_y.
- Επιστρέφει έναν χωρικό ευρετήριο από το κλειδί κελιού στη λίστα αναγνωριστικών εγγραφών που ταυτίζονται με το κελί.

### ΥΠΟΛΟΓΙΣΜΟΣ ΤΩΝ ΟΡΙΩΝ ΤΟΥ ΠΛΕΓΜΑΤΟΣ – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: GETGRIDLIMITS

```
1 usage
public static double[] getGridLimits(List<Record> records) {
    // Initialize the minimum and maximum x and y values to positive and negative infinity, respectively
    double min_x = Double.POSITIVE_INFINITY;
    double min_y = Double.POSITIVE_INFINITY;
    double max_x = Double.NEGATIVE_INFINITY;
    double max_y = Double.NEGATIVE_INFINITY;

    // Iterate through all records and update the minimum and maximum x and y values
    for (Record record : records) {
        if (record.mbr_min_x < min_x) min_x = record.mbr_min_x;
        if (record.mbr_min_y < min_y) min_y = record.mbr_min_y;
        if (record.mbr_max_x > max_x) max_x = record.mbr_max_x;
        if (record.mbr_max_y > max_y) max_y = record.mbr_max_y;
    }

    // Return an array of doubles containing the computed limits of the grid
    return new double[]{min_x, min_y, max_x, max_y};
}
```

Εικόνα 3- getGridLimits

#### Σχόλια:

- Λαμβάνει μια λίστα από αντικείμενα Record ως είσοδο και επιστρέφει έναν πίνακα τεσσάρων αριθμών διπλής ακρίβειας που αντιπροσωπεύουν τα όρια του πλέγματος: τις ελάχιστες και μέγιστες τιμές των συντεταγμένων x και y όλων των εγγραφών.
- Αρχικοποιεί τις ελάχιστες και μέγιστες τιμές για τις συντεταγμένες x και y σε θετικό και αρνητικό άπειρο, αντίστοιχα.



- Διατρέχει όλες τις εγγραφές και ενημερώνει τις ελάχιστες και μέγιστες τιμές x και y εάν μια νέα εγγραφή έχει μικρότερη ή μεγαλύτερη τιμή για οποιαδήποτε συντεταγμένη.
- Επιστρέφει έναν πίνακα διπλής ακρίβειας (doubles) που περιέχει τα τέσσερα υπολογισμένα όρια του πλέγματος.

## ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΕΙΩΝ ΕΥΡΕΤΗΡΙΩΝ – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: **WRITEGRIDTOFILE**

```

1 usage
public static void writeGridToFile(Map<String, List<Integer>> grid, double[] gridLimits, List<Record> records) throws IOException {
    // Write directory to file
    // Create a BufferedWriter to write to the directory file
    BufferedWriter dirWriter = new BufferedWriter(new FileWriter(CLASSPATH + "grid.dir"));
    // Write the grid limits to the directory file
    dirWriter.write(String.format(Locale.ENGLISH, "%f %f %f %f\n", gridLimits[0], gridLimits[2], gridLimits[1], gridLimits[3]));
    // Iterate over each cell in the grid and write the corresponding key-value pair to the directory file
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            String key = String.format("(%d,%d)", i, j);
            // Use requestedWriteFormat to write the cell indices to the file
            String requestedWriteFormat = String.format("%d %d", i, j);
            List<Integer> ids = grid.get(key);
            if (ids == null) {...} else {
                // Write the number of records in the cell to the file if the cell is not empty
                dirWriter.write(String.format("%s %d\n", requestedWriteFormat, ids.size()));
            }
        }
    }
    // Close the directory file writer
    dirWriter.close();
}

//...
BufferedWriter gridWriter = new BufferedWriter(new FileWriter(CLASSPATH + "grid.grd"));
// Iterate over each cell in the grid and write the corresponding records to the grid file
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        String key = String.format("(%d,%d)", i, j);
        List<Integer> ids = grid.get(key);
        if (ids != null) {
            // Iterate over each record in the cell and write its details to the grid file
            for (int id : ids) {...}
        }
    }
}
// Close the grid file writer
gridWriter.close();
}

```

Εικόνα 4 - writeGridToFile

### Σχόλια:

- Αποθηκεύει την πλέγματική δομή σε αρχεία με τις κατάληξεις ".dir" και ".grd". Αυτό γίνεται με τα δύο αρχεία εξόδου, ένα για το directory και ένα για το grid, χρησιμοποιώντας αντικείμενα BufferedWriter.
- Για κάθε κελί στο πλέγμα (10x10), η μέθοδος γράφει το μορφοποιημένο string με τη μορφή "(i,j) numOfRecords" στο αρχείο directory.
- Αν η λίστα ids για ένα κελί δεν είναι κενή, τότε η μέθοδος για κάθε εγγραφή που αντιστοιχεί σε αυτό το κελί δημιουργεί ένα string που περιέχει τα όρια και τα σημεία και τα γράφει στο αρχείο grid.
- Η μέθοδος χρησιμοποιεί μια μεταβλητή CLASSPATH (αυτή μπορεί να αλλάξει ανάλογα με την εκτέλεση – στην άσκηση είναι classpath του μηχανήματος εργασίας ) για να καθορίσει τη διαδρομή αποθήκευσης των αρχείων.



## ΜΕΡΟΣ 2 - 3: ΕΠΙΛΟΓΗ ΠΑΡΑΘΥΡΩΝ ΕΡΩΤΗΜΑΤΩΝ ΣΕ ΔΟΜΕΣ ΠΛΕΓΜΑΤΟΣ

- Η λειτουργία αυτή επιτρέπει στο χρήστη να εισάγει ένα αρχείο ερωτημάτων (queries.txt) και να παραχθούν τα αποτελέσματα για κάθε ερώτημα.
- Κατά την εκτέλεση του προγράμματος, φορτώνονται τα δεδομένα του πλέγματος, δηλαδή τα αντικείμενα που περιέχει κάθε κελί, και υπολογίζονται τα κελιά του πλέγματος που τέμνονται από κάθε παράθυρο ερωτήματος.
- Στη συνέχεια ελέγχεται αν τα αντικείμενα σε κάθε κελί τέμνονται με το παράθυρο ερωτήματος, και αν η αναφορική θέση του αντικειμένου βρίσκεται εντός του κελιού.
- Τα αποτελέσματα των ερωτημάτων εμφανίζονται στο τέλος της εκτέλεσης του προγράμματος και περιλαμβάνουν τα αντικείμενα που εντοπίστηκαν, τα κελιά που επισκέφτηκε ο αλγόριθμος και το συνολικό αριθμό των αντικειμένων που βρέθηκαν.

### KNOWN ISSUES – ΓΝΩΣΤΑ ΠΡΟΒΛΗΜΑΤΑ.

- Για τη δημιουργία του ευρετηρίου στο μέρος 1, όλα τα identifiers των linestrings δημιουργούνται μειωμένα κατά 1 (-1). Ως αποτέλεσμα, για να αντιμετωπίσουμε αυτό το πρόβλημα και να έχουμε το ίδιο output με το αναμενόμενο (expected outcome files στο Ecourse), στην μέθοδο "loadGrid()" του μέρους 2 πρέπει να προσθέσουμε 1 στα IDs που φορτώνουν και απαντούν στα queries.
- Το μέρος 3 δημιουργήθηκε πειραματικά και δεν δοκιμάστηκε πλήρως, αν και είναι πλήρως λειτουργικό. Το output διαφέρει από το αναμενόμενο output που δόθηκε στο ecourse και το σφάλμα δεν έχει ακόμα βρεθεί.
- Κατά την φόρτωση του πλέγματος (grid) στο μέρος 2, δεν χρησιμοποιήθηκε η ίδια δομή ( *Record* ) που χρησιμοποιήθηκε στο μέρος 1.

### ΕΞΗΓΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΓΙΑ ΤΟ ΜΕΡΟΣ 2

Το πρόγραμμα χρησιμοποιεί τη μέθοδο "main" και τέσσερις (4) επιπλέον μεθόδους, καθώς και μια κλάση με το όνομα "*WindowQuery*", προκειμένου να επιτελέσει τη λειτουργικότητα που απαιτείται από το μέρος 2. Παρακάτω παρέχονται screenshots και περαιτέρω εξηγήσεις για τα διάφορα κομμάτια του κώδικα, όπου είναι απαραίτητο.

Οι 5 συνολικά μέθοδοι είναι οι εξής:

1. Μέθοδος main – κλήση επιμέρους μεθόδων
2. `readQueryFile` – ανάγνωση αρχείου ερωτήσεων
3. `performQuery` – εκτέλεση ερωτήσεων παραθύρου
4. `loadGrid & readGridDirectory` – φόρτωση καταλόγου πλέγματος

Η κλάση `WindowQuery` έχει τα εξής πεδία:

```
public final int queryId;  
public final double minX;  
public final double minY;  
public final double maxX;  
public final double maxY;
```

## MAIN ΜΕΘΟΔΟΣ – ΚΥΡΙΑ ΜΕΘΟΔΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
2 usages  
public static final String CLASSPATH = "C:\\Users\\a.todhri\\IdeaProjects\\LyrSearch-Lucene\\src\\main\\java\\assign2\\";  
9 usages  
public static double minX, maxX, minY, maxY, cellSizeX, cellSizeY;  
5 usages  
public static int numCellsPerAxis;  
1 usage  
public static Map<String, Long> cellCounts;  
  
public static void main(String[] args) throws IOException {  
  
    String queriesFile = CLASSPATH + "queries.txt";  
  
    Map<String, List<String[]>> gridData = loadGrid(CLASSPATH + "grid");  
  
    // Perform window queries  
    List<WindowQuery> queries = readQueryFile(queriesFile);  
    for (int i = 0; i < queries.size(); i++) {  
        WindowQuery query = queries.get(i);  
        performQuery(query, gridData);  
    }  
}
```

Εικόνα 5 – main method για το μέρος 2/3 της άσκησης

### Σχόλια:

- Η κύρια μέθοδος του προγράμματος δημιουργεί τα αναζητήσιμα παράθυρα (windowQueries) από το αρχείο "queries.txt".
- Φορτώνει τα δεδομένα του πλέγματος από το αρχείο "grid" μέσω της μεθόδου "loadGrid".
- Διατρέχει τα αναζητήσιμα παράθυρα (window selection queries) που έχουν διαβαστεί από το αρχείο "queries.txt" και εκτελεί τη μέθοδο "performQuery" για κάθε ένα από αυτά, δίνοντας το πλέγμα και το παράθυρο ως ορίσματα.

- Το output δημιουργείται απο την ίδια την performQuery
- Υπάρχει μέσα στο ίδιο αρχείο (*WindowSelectionQueries.java*) και η **performQueryPart3** που υλοποιεί τα query με τον τρόπο που ζητείται για το τρίτο μέρος (refinmenet).

## ΦΟΡΤΩΣΗ ΚΑΤΑΛΟΓΟΥ ΠΛΕΓΜΑΤΟΣ – ΟΝΟΜΑ ΜΕΘΟΔΩΝ: **LOAD GRID – LAOD GRID DIRECTORY**

```

usage
public static void readGridDirectory(String directoryFilename) throws IOException {
    BufferedReader reader = new BufferedReader(new FileReader(directoryFilename));

    // Parse the grid dimensions
    String[] line1 = reader.readLine().split(" ");
    double minX = Double.parseDouble(line1[0]);
    double maxX = Double.parseDouble(line1[1]);
    double minY = Double.parseDouble(line1[2]);
    double maxY = Double.parseDouble(line1[3]);

    // Calculate the size of each cell
    double cellSizeX = (maxX - minX) / 10;
    double cellSizeY = (maxY - minY) / 10;

    // Store the results in instance variables
    WindowSelectionQueries.minX = minX;
    WindowSelectionQueries.maxX = maxX;
    WindowSelectionQueries.minY = minY;
    WindowSelectionQueries.maxY = maxY;
    WindowSelectionQueries.numCellsPerAxis = 10;
    WindowSelectionQueries.cellSizeX = cellSizeX;
    WindowSelectionQueries.cellSizeY = cellSizeY;

    // Parse the cell counts
    Map<String, Long> cellCounts = new HashMap<>();
    String line;
    while ((line = reader.readLine()) != null) {
        String[] parts = line.split("\\s+");
        long count = Long.parseLong(parts[2]);
        String cellKey = parts[0] + "," + parts[1];
        cellCounts.put(cellKey, count);
    }

    reader.close();

    WindowSelectionQueries.cellCounts = cellCounts;
}

```

Εικόνα 6 – loadGridDirectory για το μέρος 2/3

### Σχόλια σχετικά με την loadGridDirectory:

- Γίνεται φόρτωση του αρχείου πλέγματος και οργάνωση των δεδομένων που φτιάχτηκαν απο το indexing των χωρικών δεδομένων στο πρώτο μέρος της άσκησης.

- Αρχικοποιούνται μερικές σημαντικές μεταβλητές για την συνέχεια του προγράμματος και την επιτέλεση των Windows Selection Queries, όπως τα minx, minY boundaries, τα μεγέθη του κάθε κελιού κτλπ. Οι παρακάτω μεταβλητές αρχικοποιούνται:

```
WindowSelectionQueries.minX = minX;
WindowSelectionQueries.maxX = maxX;
WindowSelectionQueries.minY = minY;
WindowSelectionQueries.maxY = maxY;
WindowSelectionQueries.numCellsPerAxis = 10;
WindowSelectionQueries.cellSizeX = cellSizeX;
WindowSelectionQueries.cellSizeY = cellSizeY;
WindowSelectionQueries.cellCounts = cellCounts;
```

- Διαβάζει πληροφορίες για τον αριθμό των εγγραφών σε κάθε κελί του πλέγματος και τις αποθηκεύει στο αντίστοιχο κλειδί του HashMap cellCounts.

```
public static Map<String, List<String[]>> loadGrid(String gridFilename) throws IOException {
    // Read the grid directory to get the number of cells and their counts
    readGridDirectory(gridFilename + ".dir");

    // Initialize an empty map to hold the record data for each cell
    Map<String, List<String[]>> cellDataMap = new HashMap<>();
    for (int i = 0; i < numCellsPerAxis; i++) {
        for (int j = 0; j < numCellsPerAxis; j++) {
            String cellKey = i + "," + j;
            cellDataMap.put(cellKey, new ArrayList<>());
        }
    }

    // Open the grid file and read each line
    BufferedReader reader = new BufferedReader(new FileReader(gridFilename + ".grd"));
    String line;
    while ((line = reader.readLine()) != null) {
        // Split the line into record ID and point data
        String[] parts = line.split(",");
        String id = parts[0];
        String[] points = Arrays.copyOfRange(parts, 1, parts.length);

        // Calculate the cell IDs that this record intersects
        String[] firstPoint = points[0].split(" ");
        double x = Double.parseDouble(firstPoint[0]);
        double y = Double.parseDouble(firstPoint[1]);
        int cellX = (int) ((x - minX) / cellSizeX); // Calculate the X cell ID that this record intersects
        int cellY = (int) ((y - minY) / cellSizeY); // Calculate the Y cell ID that this record intersects

        // Add the record data to each intersected cell's list
        for (int i = cellX; i <= cellX + 1 && i < numCellsPerAxis; i++) { // Iterate over the cells that this record intersects horizontally
            for (int j = cellY; j <= cellY + 1 && j < numCellsPerAxis; j++) { // Iterate over the cells that this record intersects vertically
                String cellKey = i + "," + j;
                cellDataMap.get(cellKey).add(new String[] { id, String.join(",", points) }); // Add the record data to the intersected cell's list
            }
        }
    }

    reader.close();

    return cellDataMap;
}
```

Εικόνα 7 – loadGrid για το μέρος 2/3

#### Σχόλια σχετικά με την loadGrid:

- Αρχικοποίηση ενός κενού χάρτη (Map) που θα χρησιμοποιηθεί στη συνέχεια για να αποθηκεύσει τα δεδομένα του grid σε κελιά. Κάθε κελί αρχικοποιείται με μια κενή λίστα από συμβολοσειρές που αντιπροσωπεύουν τα δεδομένα που βρίσκονται εντός του κελιού.
- Διαβάζεται το αρχείο grid.grd και γίνεται διαχωρισμός των εγγραφών του αρχείου σε επιμέρους τιμές, όπως το ID της εγγραφής και οι συντεταγμένες της.
- Υπολογίζονται τα κελιά του πλέγματος που επικαλύπτονται από κάθε εγγραφή, χρησιμοποιώντας τις συντεταγμένες της εγγραφής και τα όρια του κάθε κελιού.
- Προστίθενται τα δεδομένα της κάθε εγγραφής στις λίστες δεδομένων κάθε κελιού που επικαλύπτεται.

#### ΔΙΑΒΑΣΜΑ ΑΡΧΕΙΟΥ ΕΡΩΤΗΜΑΤΩΝ ΠΑΡΑΘΥΡΟΥ – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: READQUERYFILE

```
1 usage
public static List<WindowQuery> readQueryFile(String queryFilename) throws IOException {
    List<WindowQuery> queries = new ArrayList<>();

    // Open the query file and read each line
    BufferedReader reader = new BufferedReader(new FileReader(queryFilename));
    String line;
    while ((line = reader.readLine()) != null) {
        // Parse the query values
        String[] parts = line.split(",");
        int queryID = Integer.parseInt(parts[0]);
        String[] values = parts[1].split(" ");
        double queryMinX = Double.parseDouble(values[0]);
        double queryMaxX = Double.parseDouble(values[1]);
        double queryMinY = Double.parseDouble(values[2]);
        double queryMaxY = Double.parseDouble(values[3]);
        WindowQuery query = new WindowQuery(queryID, queryMinX, queryMinY, queryMaxX, queryMaxY);
        queries.add(query);
    }

    return queries;
}
```

Εικόνα 8 – readQuery method of Part2/3

#### Σχόλια:

- Διαβάζεται ένα αρχείο που περιέχει ερωτήματα παραθύρου.
- Κάθε γραμμή του αρχείου αντιστοιχεί σε ένα ερώτημα παραθύρου που αποθηκεύεται σε μια λίστα από WindowQuery αντικείμενα.
- Η κλάση WindowQuery έχει εξηγηθεί παραπάνω ( κλικ για μεταβαση στο αντίστοιχο σημείο του εγγράφου )

## ΕΚΤΕΛΕΣΗ ΕΡΩΤΗΜΑΤΩΝ ΠΑΡΑΘΥΡΟΥ – ΟΝΟΜΑ ΜΕΘΟΔΟΥ: **PERFORMQUERY**

```
1 usage
public static void performQuery(WindowQuery query, Map<String, List<String[]>> grid) {
    // Calculate grid cell IDs that intersect the query window
    int cellX1 = (int) ((query.minX - minX) / cellSizeX);
    int cellX2 = (int) ((query.maxX - minX) / cellSizeX);
    int cellY1 = (int) ((query.minY - minY) / cellSizeY);
    int cellY2 = (int) ((query.maxY - minY) / cellSizeY);

    // Keep track of results and visited cells
    Set<String> results = new HashSet<>();
    Set<String> visitedCells = new HashSet<>();

    // Iterate over all intersecting cells and objects
    for (int x = cellX1; x <= cellX2; x++) {...}

    // Convert Set to List and sort
    List<String> sortedResults = new ArrayList<>(results);
    Collections.sort(sortedResults);

    // Print query results
    System.out.println("Query " + query.queryId + " results:");
    System.out.println(String.join(" ", sortedResults));
    System.out.println("Cells: " + visitedCells.size());
    System.out.println("Results: " + results.size());
    System.out.println("-----");
}
```

Εικόνα 9 – perform query method of Part2 of the exercise

Η μέθοδος **performQuery** υλοποιεί το ερώτημα επιλογής που δίνεται σαν είσοδος και επιστρέφει τα αντικείμενα που βρίσκονται εντός του παραθύρου επιλογής. Μερικά σημαντικά σημεία για την λειτουργικότητα της μεθόδου είναι τα εξής:

- Υπολογίζει τα IDs των κελιών του πλέγματος που τέμνονται με το παράθυρο του ερωτήματος.
- Διατρέχει όλα τα τέμνονται κελιά του πλέγματος και αναζητά αντικείμενα που εμπίπτουν εντός του παραθύρου επιλογής και βρίσκονται στο κάθε κελί.
- Αποθηκεύει τα αντικείμενα που εμπίπτουν εντός του παραθύρου επιλογής σε μια συλλογή Set.
- Μετατρέπει τη συλλογή Set σε μια λίστα και ταξινομεί τα αποτελέσματα.
- Εμφανίζει τα αποτελέσματα του ερωτήματος, τα κελιά του πλέγματος που επισκέφθηκε και τον αριθμό των αποτελεσμάτων.

## ΙΔΙΑΙΤΕΡΟΤΗΤΕΣ ΚΑΙ EDGE CASES ΠΟΥ ΥΛΟΠΟΙΗΘΗΚΑΝ ΓΙΑ ΤΟ ΜΕΡΟΣ 2 – ΖΗΤΗΘΗΚΑΝ ΑΠΟ ΤΗΝ ΕΚΦΩΝΗΣΗ

Ιδιαιτερότητες και edge cases που υλοποιήθηκαν στο μέρος 2 και στην δημιουργία των ερωτημάτων παραθύρων:

- στον κώδικα της μεθόδου `performQuery` υπάρχει ο έλεγχος για τα κελιά που δεν έχουν κάποια επικάλυψη με το παράθυρο, καθώς αυτά τα κελιά αγνοούνται κατά την επεξεργασία των επικαλυπτόμενων κελιών.

```
// Check if cell intersects query window
if (visitedCells.contains(cellId)) {
    continue;
}
```

Εικόνα 10 – special check edge case

- στον κώδικα της μεθόδου `performQuery` υπάρχει αντίστοιχο κομμάτι κώδικα που χρησιμοποιεί το κάτω αριστερό σημείο του MBR ως σημείο αναφοράς για την κάθε εγγραφή. Συγκεκριμένα, στην επανάληψη που διανύει τα αντικείμενα του κάθε κελιού, το αντίστοιχο σημείο αναφοράς υπολογίζεται ως το μικρότερο  $x$  και  $y$  συντεταγμένες του MBR του αντικειμένου.

```
double refPointX = Math.max(minObjectX, query.minX);
double refPointY = Math.max(minObjectY, query.minY);
```

## ΜΕΡΟΣ 3 – ΣΧΟΛΙΑΣΜΟΣ & ΠΕΡΙΓΡΑΦΗ & ΚΩΔΙΚΑΣ

Το Μέρος 3 απαιτεί τροποποίηση/βελτιστοποίηση της μεθόδου **`performQuery`** του μέρους 2 της άσκησης για να ελέγχει εάν μια γραμμή είναι μερικώς ή πλήρως μέσα σε ένα παράθυρο ερωτήματος, ακόμη και αν μόνο η MBR του αντικειμένου τέμνει το παράθυρο.

Αυτό περιλαμβάνει την προσθήκη μιας νέας μεθόδου που ονομάσαμε **`isLinestringInsideWindow`** για να ελέγξουμε εάν μια γραμμή είναι μέσα σε ένα παράθυρο ερωτήματος και τον κλήση αυτής της μεθόδου για αντικείμενα των οποίων η MBR τέμνει το παράθυρο ερωτήματος στο προηγούμενο φίλτρο του ερωτήματος.

Η συνέχεια στην επόμενη σελίδα ...



Για χάρην απλότητας στο ίδιο αρχείο προσθέσαμε μια μέθοδο **performQueryPart3** όπου είναι πανομοιότυπη με την **performQuery** του μέρους 2 πλην της μίας διαφοράς που φαίνεται παρακάτω:

```
// Check if object reference point is within current cell
double refPointX = Math.max(minObjectX, query.minX);
double refPointY = Math.max(minObjectY, query.minY);
int refCellX = (int) ((refPointX - minX) / cellSizeX);
int refCellY = (int) ((refPointY - minY) / cellSizeY);
if (refCellX == x && refCellY == y) {
    if (isLineStringInsideWindow(minObjectX, minObjectY, maxObjectX, maxObjectY,
        query.minX, query.maxX, query.minY, query.maxY)) {
        results.add(objectId);
    }
}
```

Εικόνα 11 – code difference of performQuery between part2/part3

Η **isLineStringInsideWindow** ελέγχει εάν ένα τμήμα ευθείας που ορίζεται από δύο σημεία στον χώρο (x1, y1) και (x2, y2) είναι πλήρως εντός ενός ορθογωνίου που ορίζεται από τις συντεταγμένες (queryMinX, queryMaxX) και (queryMinY, queryMaxY). Πρώτα ελέγχει αν η Παραλληλογραμμική Ορθογώνιο (MBR) του τμήματος ευθείας είναι εντός του ορθογωνίου ερωτήματος σε τουλάχιστον μία διάσταση. Στη συνέχεια, ελέγχει εάν το τμήμα ευθείας τέμνει την περιοχή ερωτήματος σε κάποιο σημείο.

```
public static boolean isLineStringInsideWindow(double x1, double y1, double x2, double y2,
        double queryMinX, double queryMaxX, double queryMinY, double queryMaxY) {
    // Check if the object's MBR is completely overlapped in at least one dimension
    boolean isOverlapX = (x1 >= queryMinX && x1 <= queryMaxX) || (x2 >= queryMinX && x2 <= queryMaxX) ||
        (x1 < queryMinX && x2 > queryMaxX);
    boolean isOverlapY = (y1 >= queryMinY && y1 <= queryMaxY) || (y2 >= queryMinY && y2 <= queryMaxY) ||
        (y1 < queryMinY && y2 > queryMaxY);

    // If there is an overlap in at least one dimension, then check each line segment
    if (isOverlapX && isOverlapY) {
        double deltaX = x2 - x1;
        double deltaY = y2 - y1;

        // Check if the line segment intersects with any of the window sides
        if (deltaX != 0) {
            double slope = deltaY / deltaX;
            double yIntercept = y1 - slope * x1;
            double topY = slope * queryMaxX + yIntercept;
            double bottomY = slope * queryMinX + yIntercept;
            if ((topY >= queryMinY && topY <= queryMaxY) || (bottomY >= queryMinY && bottomY <= queryMaxY) ||
                (topY < queryMinY && bottomY > queryMaxY)) {...}
        }
        if (deltaY != 0) {
            double slope = deltaX / deltaY;
            double xIntercept = x1 - slope * y1;
            double leftX = slope * queryMinY + xIntercept;
            double rightX = slope * queryMaxY + xIntercept;
            if ((leftX >= queryMinX && leftX <= queryMaxX) || (rightX >= queryMinX && rightX <= queryMaxX) ||
                (leftX < queryMinX && rightX > queryMaxX)) {...}
        }
    }

    return false;
}
```

Εικόνα 12 – isLineStringInsideWindow

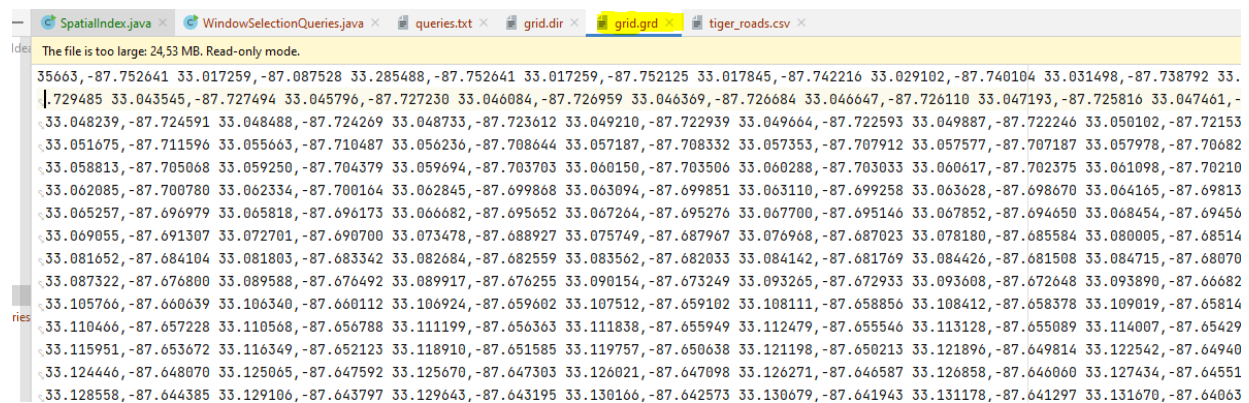
## ΠΡΑΓΜΑΤΙΚΑ ΕΚΤΕΛΕΣΙΜΑ ΤΗΣ ΑΣΚΗΣΗΣ ΚΑΙ ΠΡΟΕΠΙΣΚΟΠΗΣΕΙΣ ΤΩΝ ΚΑΤΑΛΟΓΩΝ ΠΛΕΓΜΑΤΟΣ



```
1 -87.752641 -85.565306 31.963678 33.517448
2 0 0 0
3 0 1 0
4 0 2 0
5 0 3 0
6 0 4 0
7 0 5 0
8 0 6 4
9 0 7 4
10 0 8 4
11 0 9 0
12 1 0 0
13 1 1 1
14 1 2 2
15 1 3 5
16 1 4 1
17 1 5 0
```

Εικόνα 13 – *grid.dir* file preview

Όπως φαίνεται παραπάνω το αρχείο με τον κατάλογο πλέγματος *grid.dir* δημιουργήθηκε σωστά και σύμφωνα με τις οδηγίες της εκφώνησης. Το ίδιο και το αρχείο *grid.grd* όπως διαφάνεται παρακάτω.



```
The file is too large: 24,53 MB. Read-only mode.
35663, -87.752641 33.017259, -87.087528 33.285488, -87.752641 33.017259, -87.752125 33.017845, -87.742216 33.029102, -87.740104 33.031498, -87.738792 33.
729485 33.043545, -87.727494 33.045796, -87.727230 33.046084, -87.726959 33.046369, -87.726684 33.046647, -87.726110 33.047193, -87.725816 33.047461, -
33.048239, -87.724591 33.048488, -87.724269 33.048733, -87.723612 33.049210, -87.722939 33.049664, -87.722593 33.049887, -87.722246 33.050102, -87.72153
33.051675, -87.711596 33.055663, -87.710487 33.056236, -87.708644 33.057187, -87.708332 33.057353, -87.707912 33.057577, -87.707187 33.057978, -87.70682
33.058813, -87.705068 33.059250, -87.704379 33.059694, -87.703703 33.060150, -87.703506 33.060288, -87.703033 33.060617, -87.702375 33.061098, -87.70210
33.062085, -87.700780 33.062334, -87.700164 33.062845, -87.699868 33.063094, -87.699851 33.063110, -87.699258 33.063628, -87.698670 33.064165, -87.69813
33.065257, -87.696979 33.065818, -87.696173 33.066682, -87.695652 33.067264, -87.695276 33.067700, -87.695146 33.067852, -87.694650 33.068454, -87.69456
33.069055, -87.691307 33.072701, -87.690700 33.073478, -87.688927 33.075749, -87.687967 33.076968, -87.687023 33.078180, -87.685584 33.080005, -87.68514
33.081652, -87.684104 33.081803, -87.683342 33.082684, -87.682559 33.083562, -87.682033 33.084142, -87.681769 33.084426, -87.681508 33.084715, -87.68070
33.087322, -87.676800 33.089588, -87.676492 33.089917, -87.676255 33.090154, -87.673249 33.093265, -87.672933 33.093608, -87.672648 33.093890, -87.66682
33.105766, -87.660639 33.106340, -87.660112 33.106924, -87.659602 33.107512, -87.659102 33.108111, -87.658856 33.108412, -87.658378 33.109019, -87.65814
33.110466, -87.657228 33.110568, -87.656788 33.111199, -87.656363 33.111838, -87.655949 33.112479, -87.655546 33.113128, -87.655089 33.114007, -87.65429
33.115951, -87.653672 33.116349, -87.652123 33.118910, -87.651585 33.119757, -87.650638 33.121198, -87.650213 33.121896, -87.649814 33.122542, -87.64940
33.124446, -87.648070 33.125065, -87.647592 33.125670, -87.647303 33.126021, -87.647098 33.126271, -87.646587 33.126858, -87.646060 33.127434, -87.64551
33.128558, -87.644385 33.129106, -87.643797 33.129643, -87.643195 33.130166, -87.642573 33.130679, -87.641943 33.131178, -87.641297 33.131670, -87.64060
```

Εικόνα 14 – *grid.grd* file preview

Οι εκτελέσεις του προγράμματος δημιουργίας ερωτημάτων παραθύρων:

#### μέρος 2

```
Query 1 results:
13151 15262 15774 16782 21379 22260 22500 22946 22947
Cells: 1
Results: 9
-----
Query 2 results:
33887 34512 34862
Cells: 1
Results: 3
-----
Query 3 results:
30397
Cells: 2
Results: 1
-----
Query 4 results:
1108
Cells: 1
Results: 1
-----
Query 5 results:
5496
Cells: 4
Results: 1
-----

Process finished with exit code 0
|
```

Εικόνα 15 – εκτέλεσιμο output μέρος 2

#### μέρος 3 - refinement

```
Query 1 results:
21379 22260 22500 22946 22947
Cells: 1
Results: 5
-----
Query 2 results:
Cells: 1
Results: 0
-----
Query 3 results:
30397
Cells: 2
Results: 1
-----
Query 4 results:
1108
Cells: 1
Results: 1
-----
Query 5 results:
5496
Cells: 4
Results: 1
-----

Process finished with exit code 0
```

Εικόνα 15 – output μέρος 3

**ΣΑΣ ΕΥΧΑΡΙΣΤΩ ΘΕΡΜΑ ΓΙΑ ΤΟΝ ΧΡΟΝΟ ΚΑΙ ΤΗΝ ΠΡΟΣΟΧΗ  
ΣΑΣ!!!**