ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων
(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2022-23)

ΕΡΓΑΣΙΑ 2 - Χωρικά Δεδομένα

Προθεσμία: 2 Μαΐου 2023

Περιγραφή

Στόχος της εργασίας είναι η ανάπτυξη δεικτοδότησης χωρικών δεδομένων με την τεχνική σχάρας και στην συνέχεια η αποδοτική αναζήτηση των δεδομένων.

Μέρος 1 (30%, δημιουργία ευρετηρίου)

Κατεβάστε το αρχείο **tiger_roads.csv** από το ecourse. Το αρχείο περιέχει τις συντεταγμένες **35667 αντικειμένων τεθλασμένων γραμμών** οι οποίες είναι δρόμοι στις ΗΠΑ (κάποιοι δρόμοι υπάρχουν παραπάνω από 1 φορά ο καθένας στο dataset, παρόλα αυτά θεωρούμε κάθε αντίγραφο unique). Η πρώτη γραμμή του αρχείου περιέχει το πλήθος των τεθλασμένων γραμμών (**linestrings**). Από την επόμενη γραμμή και μετά, κάθε γραμμή έχει την ακολουθία των συντεταγμένων (X Y) ενός δρόμου, χωρισμένες με κόμμα. Ζητείται να γράψετε ένα πρόγραμμα το οποίο θα φτιάχνει ένα απλό χωρικό ευρετήριο βασισμένο σε σχάρα (grid). Η σχάρα χωρίζει το χώρο που καλύπτουν τα σημεία σε 10*10 = 100 ισομεγέθη ορθογώνια (κελιά).

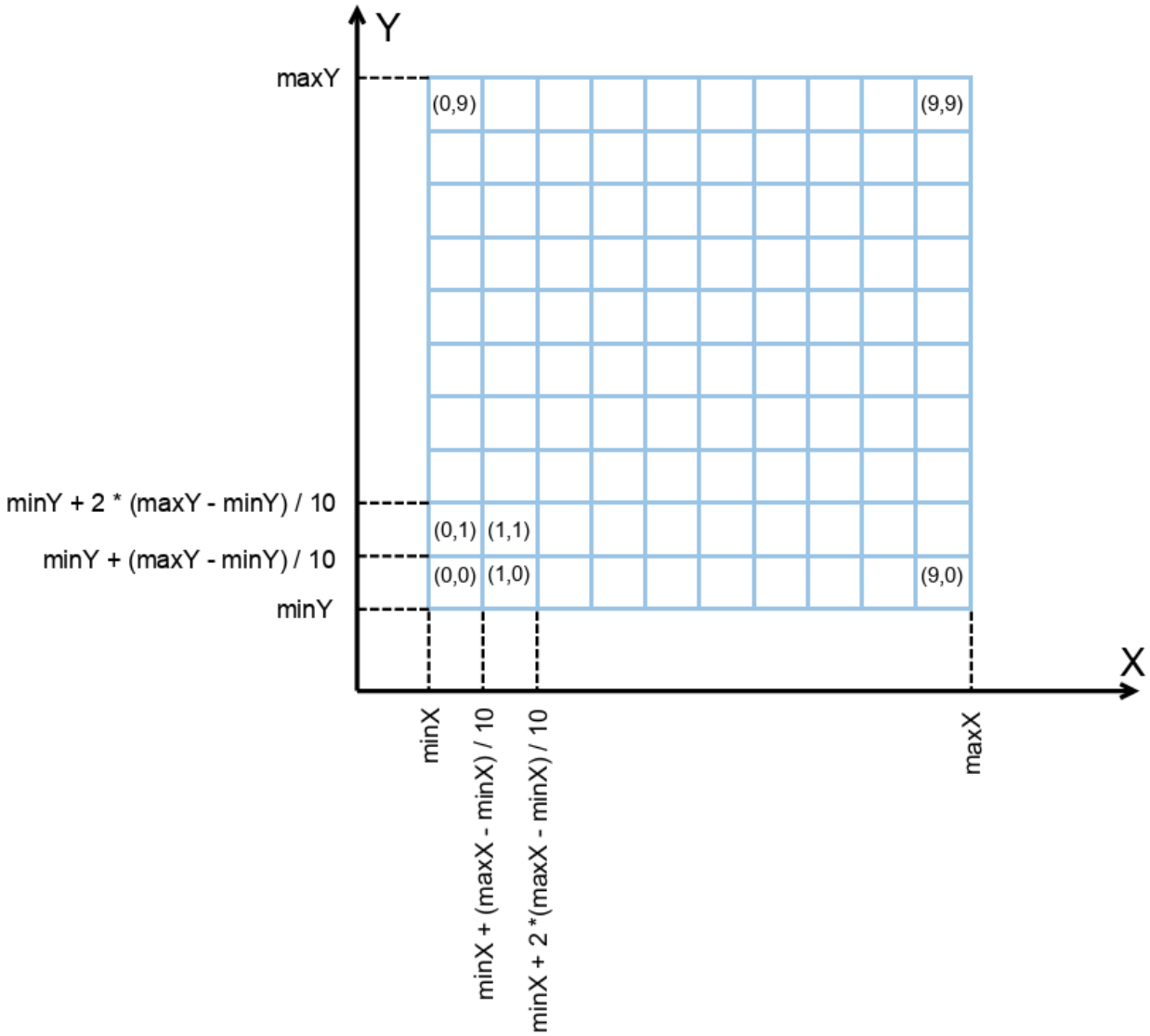
Θα πρέπει να διαβάσετε τα αντικείμενα (records) από το αρχείο και για κάθε ένα να αποθηκεύσετε α) ένα αναγνωριστικό αριθμό (identifier), β) τα σημεία min,max του MBR του, και γ) τα σημεία του linestring. Το identifier θα αντιστοιχεί στη γραμμή στην οποία βρίσκεται το linestring στο αρχείο tiger_roads.csv. Π.χ. το linestring <-86.449493 32.425501,-86.449391 32.425472,...> θα έχει αναγνωριστικό 1, επειδή βρίσκεται στη γραμμή 1 του αρχείου (χωρίς να μετράμε την γραμμή με το πλήθος των αντικειμένων), κλπ. Μπορείτε να αποθηκεύσετε τα δεδομένα σε δομή της αρεσκείας σας. Π.χ. σε python, μπορείτε να χρησιμοποιήσετε μία λίστα από τριπλέτες, όπου κάθε μία είναι της μορφής:

[ID, [[MBRminX MBRminY],[MBRmaxX MBRmaxY]],[[x1,y1],[x2,y2]...]]

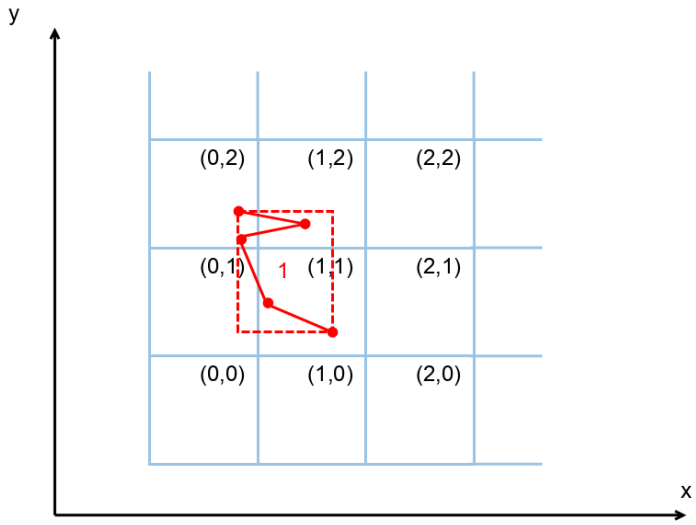
ενώ σε Java/C++ μπορείτε να δημιουργήσετε κλάση Record με τα αντίστοιχα 3 πεδία και στην συνέχεια να αποθηκεύσετε τα αντικείμενα σε ένα hash map (για γρήγορη ανάκτηση μέσω του ID). Μην ξεχάσετε επίσης να υπολογίσετε και να αποθηκεύσετε τις ελάχιστες και μέγιστες συντεταγμένες για κάθε άξονα ολόκληρου του dataset.

Κατόπιν, για να δημιουργήσετε την σχάρα, χωρίστε το εύρος τιμών σε κάθε συντεταγμένη σε 10 ίσα διαστήματα τιμών δημιουργώντας έτσι 100 κελιά. Μετά, για κάθε MBR των αντικειμένων, υπολογίστε ποια κελιά της σχάρας τέμνει και αναθέστε το σε αυτά τα κελιά. Π.χ. μπορείτε να χρησιμοποιήσετε δομή τύπου hash map όπου το κλειδί θα είναι οι συντεταγμένες του κελιού (π.χ. (0,0)) και τιμή θα είναι μία λίστα από τα IDs των αντικειμένων των οποίων το MBR τέμνει αυτό το κελί. Μερικά MBRs ενδέχεται να ακουμπάνε πάνω από 1 κελί της σχάρας, οπότε φροντίστε να δημιουργήσετε αντίγραφά τους και να τα αποθηκεύσετε σε κάθε κελί που ακουμπάνε.

<u>Παράδειγμα</u> Παρακάτω φαίνεται ένα παράδειγμα μιας 10 επί 10 σχάρας και πως υπολογίζονται οι συντεταγμένες των μεμονωμένων κελιών της. Όπου minX,minY και maxX,maxY είναι τα ελάχιστα και μέγιστα X,Y του dataset.



Παράδειγμα Το αντικείμενο 1 της παρακάτω εικόνας θα ανατεθεί στα κελιά (0,1),(0,2),(1,1) και (1,2), επειδή το MBR του τέμνεται με αυτά. Για να τα υπολογίσουμε, βρίσκουμε τα κελιά στα οποία πέφτουν τα min,max σημεία του MBR (κάτω αριστερά και πάνω δεξιά) και βρίσκουμε επαναληπτικά όλα τα ενδιάμεσα τους σε κάθε άξονα. Αν εξετάζαμε τα σημεία του linestring για το ποια κελιά ακουμπάνε, θα χάναμε το κελί (0,1) το οποίο όμως τέμνεται από ακμή του linestring ενώ το να επιχειρήσουμε να βρούμε ποια κελιά τέμνονται από το linestring με μαθηματικές πράξεις τομής ευθειών, είναι υπολογιστικά ακριβό.



Γράψτε τα περιεχόμενα των κελιών ταξινομημένα (πρώτα τα περιεχόμενα του (0,0), μετά του (0,1) κ.ο.κ) σε ένα αρχείο grid.grd όπου κάθε γραμμή είναι της μορφής

<identifier,<MBRminX MBRminY,MBRmaxX MBRmaxY>,<x1 y1,x2 y2,...>>

(χωρίς τα <>). Παράλληλα δημιουργείστε και ένα άλλο αρχείο grid.dir (directory), το οποίο θα έχει στην πρώτη γραμμή του την ελάχιστη και τη μέγιστη τιμή σε κάθε άξονα X και Y. Στις επόμενες γραμμές, για κάθε κελί θα υπάρχει η συντεταγμένη του κελιού (π.χ. (0,0), (0,1) κλπ.) και ο αριθμός των αντικειμένων στο κελί. Έτσι, χρησιμοποιώντας τα δεδομένα του αρχείου grid.dir, κάποιος θα μπορεί να προσπελάσει από την αρχή το αρχείο grid.grd και να διαβάζει για κάθε κελί όλα τα περιεχόμενα του με την σειρά. Καθώς η δημιουργία της σχάρας αποτελεί pre-processing, χρησιμοποιούμε αυτά τα δύο ξεχωριστά αρχεία ώστε να έχουμε την σχάρα μας κατασκευασμένη στον δίσκο για μελλοντική χρήση.

Παράδειγμα Οι πρώτες 15 γραμμές του αρχείου grid.dir φαίνονται παρακάτω

```
 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
```

Ενώ οι πρώτες 4 εγγραφές στο αρχείο grid.grd αφορούν τα αντικείμενα με αναγνωριστικά **35664,35665,35666 και 35667** (φυσικά με τα MBR και linestrings τους) και το κελί (0,6), αφού τα προηγούμενα κελιά στην σειρά είναι άδεια.

Μέρος 2 (30%, ερωτήματα επιλογής, MBR φίλτρο με duplicate detection)

Ζητείται να γράψετε ένα πρόγραμμα το οποίο θα χρησιμοποιεί το grid που φτιάξατε στο 1ο μέρος ώστε να αποτιμήσετε ερωτήματα επιλογής παραθύρου (window selection queries).

Αρχικά, το πρόγραμμα θα διαβάζει εξ ολοκλήρου το αρχείο grid.dir και θα χρησιμοποιεί τα περιεχόμενά του για να φορτώσει τα περιεχόμενα των κελιών από το αρχείο grid.grd σε μία δομή στην μνήμη. Χρησιμοποιήστε την ίδια δομή που χρησιμοποιήσατε και στο Μέρος 1 για την κατασκευή της σχάρας, πριν την αποθήκευση.

Κατόπιν το πρόγραμμα θα διαβάζει ερωτήσεις παραθύρου (window queries) από το αρχείο queries.txt και, για κάθε ερώτηση, θα υπολογίζει και θα τυπώνει στην έξοδο το πλήθος των αντικειμένων και τα IDs τους, των οποίων το MBR τέμνεται με το παράθυρο (δηλαδή έχει τουλάχιστον ένα κοινό σημείο με το παράθυρο).

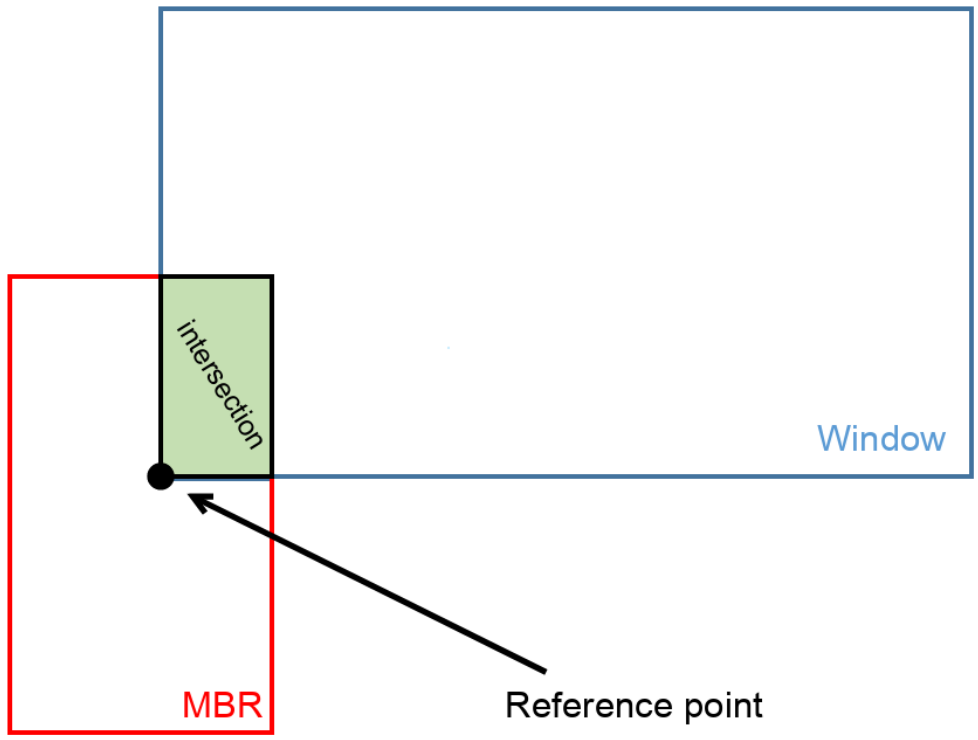
Για κάθε ερώτηση, το πρόγραμμα θα εξετάζει τα περιεχόμενα κάθε κελιού που τέμνεται με το παράθυρο (θα αγνοεί εντελώς κελιά που δεν έχουν καμία επικάλυψη με το παράθυρο). Το πρόγραμμα θα εντοπίζει στην δομή που χρησιμοποιείται στην μνήμη τα δεδομένα του κελιού και θα ελέγχει τα MBR τους για το αν επικαλύπτονται με το παράθυρο επιλογής.

Προσέξτε ότι καθώς μερικά αντικείμενα μπορεί να υπάρχουν ως αντίγραφα σε πάνω από ένα κελιά, μπορεί να καταμετρηθούν πάνω από μία φορά το καθένα στα τελικά αποτελέσματα. Για να το αποφύγουμε αυτό, αναθέστε ως σημείο αναφοράς του κάθε αντικειμένου (reference point) το κάτω αριστερά σημείο της τομής του MBR του με το παράθυρο του ερωτήματος. Αυτό μπορείτε να το υπολογίσετε βρίσκοντας τις μέγιστες τιμές για το x και το y από τις ελάχιστες των δύο (MBR και παράθυρο). Στην συνέχεια, κάθε ξεχωριστό αντικείμενο θα αναφέρεται ως αποτέλεσμα μόνο εάν το σημείο αναφοράς του βρίσκεται μέσα στο κελί το οποίο εξετάζεται εκείνη την δεδομένη στιγμή.

Το πρόγραμμά σας θα πρέπει να τυπώνει τον αριθμό των κελιών που ελέγχθηκαν καθώς και το συνολικό πλήθος των αποτελεσμάτων και τα IDs των αντικειμένων.

Παράδειγμα 2.1

Παρακάτω φαίνεται ένα παράδειγμα του reference point.



Παράδειγμα 2.2 (έξοδος προγράμματος)

```
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: 3
Results: 1
----------
```

Μέρος 3 (40%, ερωτήματα επιλογής, refinement stage)

Το ότι ένα MBR ενός αντικειμένου τέμνεται με το παράθυρο του ερωτήματος επιλογής δεν σημαίνει απαραίτητα ότι και κάποιο κομμάτι του linestring θα εμπεριέχεται σίγουρα στο παράθυρο. Ζητείται να γράψετε κώδικα ο οποίος θα εντοπίζει αν ένα linestring εμπεριέχεται όντως, έστω και τμηματικά στο παράθυρο επιλογής. Αυτός ο κώδικας θα πραγματοποιεί τον έλεγχο αποκλειστικά και μόνο για κάθε αντικείμενο του οποίου το MBR βρέθηκε να τέμνεται με το παράθυρο επιλογής στο φίλτρο του προηγούμενου ερωτήματος και όχι για όλα τα αντικείμενα που βρίσκονται μέσα σε ένα υπό-έλεγχο κελί.

Για να εντοπίσετε αν ένα αντικείμενο (του οποίου το MBR έχουμε ήδη επιβεβαιώσει ότι τέμνεται από το παράθυρο επιλογής) βρίσκεται έστω και τμηματικά μέσα στο παράθυρο επιλογής, θα πρέπει πρώτα να ελέγξετε αν το MBR του υπερκαλύπτεται εντελώς σε τουλάχιστον μία από τις δύο διαστάσεις x,y από το παράθυρο επιλογής. Αν ναι, τότε σίγουρα και το ίδιο linestring θα εμπεριέχεται έστω μερικώς στο παράθυρο επιλογής. Αν όχι, θα πρέπει να προχωρήσετε σε έλεγχο των μεμονωμένων ευθυγράμμων τμημάτων (line segments) του linestring και να εξετάσετε αν κάποιο τέμνει τις πλευρές του παραθύρου.

Για να ελέγξετε αν δυο ευθύγραμμα τμήματα τέμνονται μπορείτε να χρησιμοποιήσετε την φόρμουλα που περιγράφεται στο εξής άρθρο και φαίνεται στην παρακάτω φωτογραφία. Χρησιμοποιώντας για κάθε ευθύγραμμο τμήμα τα σημεία που το ορίζουν στον x και y άξονα.

Given two points on each line segment [edit]

See also: Intersection_(geometry) § Two_line_segments

The intersection point above is for the infinitely long lines defined by the points, rather than the line segments between the points, and can produce an intersection point not contained in either of the two line segments. In order to find the position of the intersection in respect to the line segments, we can define lines L_1 and L_2 in terms of first degree Bézier parameters:

$$L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}, \qquad L_2 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + u \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix}$$

(where t and u are real numbers). The intersection point of the lines is found with one of the following values of t or u, where

$$t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

and

$$u = \frac{\begin{vmatrix} x_1 - x_3 & x_1 - x_2 \\ y_1 - y_3 & y_1 - y_2 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_1 - y_2) - (y_1 - y_3)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)},$$

with

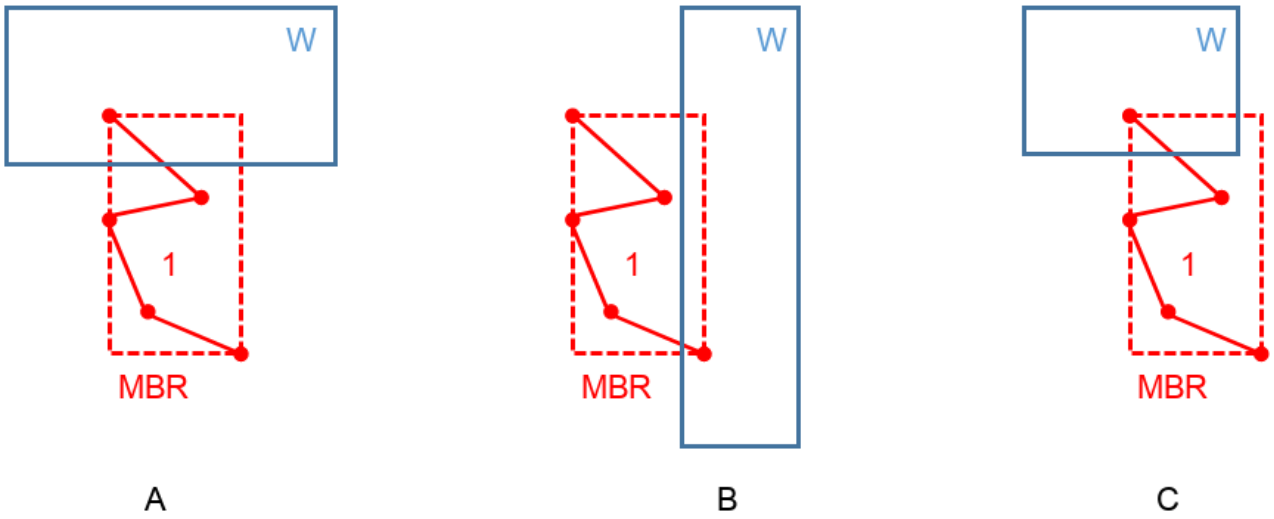
$$(P_x, P_y) = \big(x_1 + t(x_2 - x_1),\ y_1 + t(y_2 - y_1)\big) \quad \text{or} \quad (P_x, P_y) = \big(x_3 + u(x_4 - x_3),\ y_3 + u(y_4 - y_3)\big)$$

There will be an intersection if $0 \le t \le 1$ and $0 \le u \le 1$. The intersection point falls within the first line segment if $0 \le t \le 1$, and it falls within the second line segment if $0 \le u \le 1$. These inequalities can be tested without the need for division, allowing rapid determination of the existence of any line segment intersection before calculating its exact point.[2]

Γράψτε ένα πρόγραμμα, το οποίο επεκτείνει το πρόγραμμα του Μέρους 2 ώστε να εκτελεί και το refinement step για όλες τις ερωτήσεις του αρχείου queries.txt. Όπως και στο Μέρος 2, το πρόγραμμα θα πρέπει να τυπώνει πόσα και ποια αντικείμενα είναι αποτελέσματα του ερωτήματος (μετά το refinement step) καθώς και τα κελιά που χρησιμοποιήθηκαν.

Παράδειγμα 3.1

Παρακάτω, φαίνονται οι περιπτώσεις όπου το refinement εντοπίζει πραγματική τομή του παραθύρου W με το linestring 1. Οι A και B πέφτουν στην περίπτωση όπου εντοπίζεται επικάλυψη μέσω της ολικής επικάλυψης του MBR στους άξονες X και Y αντίστοιχα, ενώ η περίπτωση C εντοπίζει επικάλυψη μέσω της ανίχνευσης τομής της μίας πλευράς του παραθύρου με το linestring.



Παράδειγμα 3.2 (έξοδος προγράμματος)

```
Query  1 results:
21379 22260 22500 22946 22947
Cells: 1
Results: 5
----------
Query  2 results:
34512
Cells: 1
Results: 1
----------
Query  3 results:
30397
Cells: 2
Results: 1
----------
Query  4 results:

Cells: 1
Results: 0
----------
Query  5 results:
5496
Cells: 3
Results: 1
----------
```

Υποδείξεις. Μπορείτε να χρησιμοποιήσετε λογισμικό visualization μεμονωμένων σημειών/ευθειών όπως αυτό για να εξετάσετε πιο λεπτομερώς τις περιπτώσεις του refinement που φτιάξατε. Κάθε μέρος

θα εξεταστεί ξεχωριστά, οπότε συστήνεται να γίνει η υλοποίηση της άσκησης σε 3 διαφορετικά αρχεία. Προφανώς το Μέρος 3 πρέπει να επεκτείνει το Μέρος 2, οπότε ο κώδικας κατά κύρια βάση θα είναι ίδιος με την διαφορά ότι το πρόγραμμα του Μέρους 3 θα συνεχίζει και στο refinement. Το Μέρος 1 αποτελεί pre-processing, δηλαδή το τρέχουμε μία φορά για ένα σύνολο δεδομένων και δεν χρειάζεται ξανά, ενώ μπορούμε να τρέχουμε τα Μέρη 2 και 3 απεριόριστες φορές στο ήδη κατασκευασμένο grid.

Ερωτήματα. Στο αρχείο queries.csv δίνονται 5 ερωτήματα για να τρέξετε και στο αρχείο results.txt δίνονται οι απαντήσεις τους (αριθμός κελιών και identifiers αποτελεσμάτων) για να ελέγξετε την ορθότητα του προγράμματος σας. Συστήνεται να πειραματιστείτε και με δικά σας ερωτήματα. Το πρόγραμμά σας θα πρέπει αφού φορτώνει το πλέγμα και τα δεδομένα στην δομή που έχετε φτιάξει, να διαβάζει τα ερωτήματα του αρχείου και να τα τρέχει το ένα μετά το άλλο τυπώνοντας τα αποτελέσματά τους.

Παραδοτέα.

Βάλτε σέ ένα zip αρχείο τα προγράμματά σας και ένα PDF αρχείο, το οποίο θα περιέχει πληροφορίες και οδηγίες εκτέλεσης για τα προγράμματά σας. Υποβάλετε το zip αρχείο σας μέσω turnin στο assignment2@mye041

Οδηγίες για τις υποβολές:

1) Αν χρησιμοποιήσετε Java, το πρόγραμμά σας θα πρέπει να γίνεται compile και να τρέχει και εκτός Eclipse στους υπολογιστές του εργαστηρίου. **Μην χρησιμοποιείτε packages**.
2) Αν χρησιμοποιήσετε Python, μην χρησιμοποιήστε τη βιβλιοθήκη pandas και μην υποβάλετε κώδικα για interactive programming (π.χ. ipython)
3) Υποβάλετε τις εργασίες σας σε ένα **zip** αρχείο (**όχι rar**) το οποίο πρέπει να περιλαμβάνει όλους τους κώδικες καθώς και ένα αρχείο τεκμηρίωσης το οποίο να περιγράφει τη μεθοδολογία σας και να περιλαμβάνει το PDF αρχείο. **Μην υποβάλετε αρχεία δεδομένων**.
4) Μην ξεχνάτε να βάζετε το όνομά σας (σε greeklish) και το ΑΜ σε κάθε αρχείο που υποβάλετε.
5) Ο έλεγχος των προγραμμάτων σας μπορεί να γίνει σε άλλα αρχεία εισόδου από αυτά που σας δίνονται, άρα θα πρέπει ο κώδικάς σας να μην εξαρτάται από τα συγκεκριμένα αρχεία εισόδου που σας δίνονται.
6) Μπορείτε να απευθύνεστε στον βοηθό Θανάση Γεωργιάδη (geor.thanasi@gmail.com) για τυχόν απορίες.