

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

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

ΕΡΓΑΣΙΑ 2 – Χωρικά Δεδομένα

Βλαχόπουλος Λάμπρος, ΑΜ : 2948

Περιεχόμενα

- 1) Περιγραφή βασικών αλγορίθμων και δομών που χρησιμοποιήσα.**

A) Από την φόρτωση των δεδομένων παράγεται :

- `LinkedHashMap<Chell, List<Record>> mapPositionByChell ;`
- `LinkedHashMap<String, Chell> mapPositionByChellSearching;`

Έχω δηλαδή συνδέσει κάθε chell με ένα List από Records που περιέχει. Το δεύτερο map επιταχύνει την αναζήτηση καθώς είναι της μορφής

`(0,0 , Chell(0,0),`

`0,1 , Chell(0,1)).`

Για τον δημιουργία των chells χρησιμοποιήθηκε:

```
double xmin =Xmin + (i*(Xmax-Xmin))/this.gridSize;  
double xmax =Xmin + ((i+1)*(Xmax-Xmin))/this.gridSize;  
double ymin =Ymin + (j*(Ymax-Ymin))/this.gridSize;  
double ymax =Ymin + ((j+1)*(Ymax-Ymin))/this.gridSize;
```

Ενώ για να εξετάσω για ένα record σε ποιο chell ανήκει κάνω το παρακάτω

```
int xMinCell = (int) Math.floor((xmin - Xmin) / xCellSize);  
int yMinCell = (int) Math.floor((ymin - Ymin) / yCellSize);  
int xMaxCell = (int) Math.floor((xmax - Xmin) / xCellSize);  
int yMaxCell = (int) Math.floor((ymax - Ymin) / yCellSize);
```

Βρίσκω δηλαδή σε ποιο κελί είναι οι κορυφές και μετά με for I,j τα ενδιάμεσα. Εκμεταλλεύομαι δηλαδή την ιδιότητα που έχουν τα κελιά να έχουν ίδιο xChellSize μεταξύ τους αλλά και yCellSize.

B)

Για το δεύτερο μέρος αρχικά βρίσκω τα κελιά που περιλαμβάνει το window query με ίδιο τρόπο όπως πάνω δηλαδή βρίσκω

```
int xMinCell = (int) Math.floor((x1 - Xmin) / xCellSize);  
int yMinCell = (int) Math.floor((y1 - Ymin) / yCellSize);  
int xMaxCell = (int) Math.floor((x2 - Xmin) / xCellSize);  
int yMaxCell = (int) Math.floor((y2 - Ymin) / yCellSize);
```

Και στη συνέχεια για τα ενδιάμεσα

```
for (int i = xMinCell; i <= xMaxCell; i++) {  
    for (int j = yMinCell; j <= yMaxCell; j++) {  
        Chell chell = mapPositionByChellSearch.get(i+","+j);  
        listChells.add(chell);  
    }  
}
```

Στην συνέχεια

```
ArrayList<Record> mathesRecord =new ArrayList<Record>();
for(Chell chell : listChells) {
    //System.out.println(chell.getI()+" "+chell.getJ());
    for(Record record : chell.getRecordsInChell() ) {
        //System.out.println(record.getListMaxMinXYInRecord());
        List<Point> minMaxPoint= record.getListMaxMinXYInRecord();
        Point minPointInRecord = minMaxPoint.get(0);
        Point maxPointInRecord = minMaxPoint.get(1);

        double referenceX = Math.max(minPointInRecord.getX(), x1);
        double referenceY = Math.max(minPointInRecord.getY(), y1);
        //System.out.println(referenceX+" "+referenceY);

        if (minPointInRecord.getX() > x2 || maxPointInRecord.getX() < x1 || minPointInRecord.getY() > y2
            || maxPointInRecord.getY() < y1) {

            // System.out.println("μφκα");
        } else {

            if (referenceX <= chell.getXmax() && referenceY <= chell.getYmax()
                && referenceX >= chell.getXmin() && referenceY >= chell.getYmin()) {
                //System.out.println(record.getIdintifier() + "   few " + chell.getI() + " " + chell.getJ());
                mathesRecord.add(record);
            }
        }
    }
}
```

Διατρέχω την λίστα με τα chells που δημιούργησα παραπάνω και για κάθε chell διατρέχω την λίστα με τα Record που περιλαμβάνει ορίζω Reference point και τσεκάρω αν το record είναι εκτός του παραθύρου. Αν είναι μέσα στο παράθυρο τσεκάρω αν το referencePoint είναι εσωτερικό του κελιού που εξετάζω κάθε φορά σύμφωνα με τις υποδείξεις που δόθηκαν.

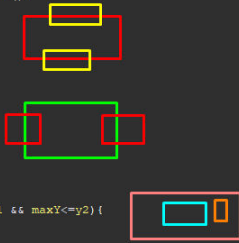
Γ)

Για το τρίτο μέρος ο κώδικας είναι ο ίδιος με το Β. με τις προσθήκες:

```
ArrayList<Record> intersect = new ArrayList<Record>();
for (Record record : mathesRecord) {

    List<Point> minMaxInRecord = record.getListMaxMinXYInRecord();
    double minX = minMaxInRecord.get(0).getX();
    double minY = minMaxInRecord.get(0).getY();
    double maxX = minMaxInRecord.get(1).getX();
    double maxY = minMaxInRecord.get(1).getY();

    // a3ona X
    if (minX >= x1 && maxX <= x2) {
        intersect.add(record);
        continue;
    }
    // a3ona Y
    else if (minY >= y1 && maxY <= y2) {
        intersect.add(record);
        continue;
    }
    //gihra epikaluth
    else if (minX<=x1 & maxX>=x2 && maxY>=y1 && minY<=y2){
        intersect.add(record);
        continue;
    }
}
}
```



Από το list με τα record που προκύπτει δηλαδή εξετάζω αρχικά τις επικαλύψεις στους άξονες αλλά και την πλήρη επικάλυψη με χρήση των Mbr.

Αν δεν υπάρχει επικάλυψη εκεί

```
private {
    //List<String> lineString = record.getLineString();
    List<Point> listPointInRecordList = record.getPointsList();

    for (int point = 0; point < listPointInRecordList.size()-1; point++) {
        // an AAAA & AAAAAA mono vale to !!!

        double x1a = listPointInRecordList.get(point).getX();
        double y1a = listPointInRecordList.get(point).getY();
        // AAAAAA
        double x1b = listPointInRecordList.get(point+1).getX();
        double y1b = listPointInRecordList.get(point+1).getY();

        Double[] pleura = new Double[4];
        pleura[0] = x1;
        pleura[1] = y1;
        pleura[2] = x2;
        pleura[3] = y2;
        Double[] lineStringXY = new Double[4];
        lineStringXY[0] = x1a;
        lineStringXY[1] = y1a;
        lineStringXY[2] = x1b;
        lineStringXY[3] = y1b;

        boolean result = haveIntersectionPoint(pleura, lineStringXY);
        if (result==true) {
            intersect.add(record);
            break;
            //continue;
        } else {
            //continue;

            // = AAAA

            pleura = new Double[4];
            pleura[0] = x1;
            pleura[1] = y2;
            pleura[2] = x2;
            pleura[3] = y2;
            lineStringXY = new Double[4];
            lineStringXY[0] = x1a;
            lineStringXY[1] = y1a;
            lineStringXY[2] = x1b;
            lineStringXY[3] = y1b;

            result = haveIntersectionPoint(pleura, lineStringXY);
            if (result==true) {
                intersect.add(record);
                break;
                //continue;
            } else {

                // y de3ia
                pleura = new Double[4];
                pleura[0] = x2;
                pleura[1] = y1;
                pleura[2] = x2;
                pleura[3] = y2;
                lineStringXY = new Double[4];
                lineStringXY[0] = x1a;
                lineStringXY[1] = y1a;
                lineStringXY[2] = x1b;
                lineStringXY[3] = y1b;

                result = haveIntersectionPoint(pleura, lineStringXY);
                if (result==true) {
                    intersect.add(record);
                    break;
                    //continue;
                } else {
                    // y AAAAAA
                }
            }
        }
    }
}
```

Δηλαδή παίρνω ανά 2 τα Point που έχω στη λίστα δημιουργώ ευθύγραμμο τμήμα και ελέγχω αν έχουν κοινό σημείο με το ευθύγραμμο τμήμα κάθε άξονα x, y . Αν βρω έστω ένα σημείο τομής έχω βρει ότι υπάρχει επικάλυψη.

List $[x,y] = (x_1 \ y_1, x_2 \ y_2, x_3 \ y_3, x_4 \ y_4)$

Ευθύγραμμο τμήμα $x_1y_1, x_2y_2, x_2y_2x_3y_3 \dots$ τα συγκρίνω με των αξόνων πχ $x_1y_1x_2y_1$.

Επίσης για την εύρεση σημείου τομής ευθυγράμμων τμημάτων χρησιμοποιήθηκε η φόρμουλα που υποδείχτηκε.

Για να τρέξετε το πρόγραμμα :

- 1) Για το πρώτο μέρος : javac (όλες τις κλάσεις με main την κλάση PreProcecingMain).
- 2) Για το δεύτερο μέρος javac(όλες τις κλάσεις χωρίς την Writer, PreProcecingMain. Με main QueriesMain).
- 3) Για το τρίτο μέρος ότι και στο 2^ο απλα βγάλτε την εντολή 61 { win.refinementQueries} από τα σχόλια και βάλτε σε σχόλια την εντολή 60 { εντολή win.filterQueries}
- 4) Εννοείται να έχετε στον ίδιο φάκελο και το αρχείο tiger.