

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

Ονοματεπώνυμο: Evangelos Iliadis 3117 (Ευάγγελος Ηλιάδης 3117)

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

Όλα τα δεδομένα που παρατίθενται παρακάτω αφορούν το αρχείο “tiger_roads.csv”. Ωστόσο ο κώδικας έχει αναπτυχθεί έτσι ώστε να μπορεί να δουλέψει για οποιαδήποτε αρχεία έχουν τις ίδιες λεπτομέρειες σύνταξης με αυτό, αρκεί να αλλάξει κάθε φορά η μεταβλητή path στο αντίστοιχο μονοπάτι της επιθυμητής εισόδου.

Για παράδειγμα εάν θέλαμε να το τρέξουμε σε έναν από τους υπολογιστές του εργαστηρίου του τμήματός μας, θα ήταν κάπως έτσι:

```
path =  
'/usr/home/students/stud15/cse53117/Desktop/ddtry/tiger_roads.csv'
```

Προκειμένου κάποιος να τρέξει τα εν λόγω προγράμματα στον υπολογιστή του, πρέπει να προσαρμόσει τα μονοπάτια στην αρχή των προγραμμάτων.

Μέσα στο πρόγραμμα υπάρχουν επιπλέον επεξηγηματικά σχόλια για την κατανόηση του ρόλου που επιτελούν τα διάφορα blocks κώδικα, από έναν απλό αναγνώστη που δεν εμπλέκεται με την ανάπτυξη του εν λόγω κώδικα.

Μέρος 1^ο

```
#Paths. Should be changed accordingly. For grd and dir paths: These  
#are the paths of the files that will be created when the code runs.  
path =  
'/usr/home/students/stud15/cse53117/Desktop/ddtry/tiger_roads.csv'  
grdPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.grd'  
dirPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.dir'
```

Σε αυτό το κομμάτι βάζουμε τα μονοπάτια των αρχείων. Έπειτα έχουμε τις δομές που πρόκειται να χρησιμοποιήσουμε:

```
records = [] #Here will be stored the csv data in the expected format  
MBRIDList = [[list() for i in range(10)] for j in range(10)] #Here  
#will be stored the road ID's that belong in each cell  
sortedMBRIDList = [] #MBRIDList after the sorting
```

Πρώτο βήμα για το μέρος 1 της εργασίας είναι η απομόνωση των στοιχείων από το csv αρχείο:

```
#Data extraction
#Extract the elements from the road csv file
def extract_data():
    with open(path, mode = 'r') as inf: #Open csv file for reading.
        reader = csv.reader(inf, delimiter = ',') #Determine the
        delimiter that separates each cell.
        firstRow = next(reader) #Get first row of the opened csv file

        counter = 1 #Line ID
        for row in reader:
            singleRecord = [] #Each line that will be stored into
            records
            singleRecord.append(counter) #First cell should be the
            object ID

            rowCellStorage = [] #Cells will be stored here
            cellCounter = 0 #Index of columns
            for cell in row:
                splitCell = cell.split() #Extract the two "string"
                numbers from each cell
                floatSplitCell =
                [float(splitCell[0]),float(splitCell[1])] #Make them float and store
                them

                #Find MBRmin/ MBRmax
                if cellCounter == 0: #Assign first cell coordinates
                as MBR min/ max for the comparison. This will probably change with
                each iteration.

                    MBRminX = float(splitCell[0])
                    MBRminY = float(splitCell[1])
                    MBRmaxX = float(splitCell[0])
                    MBRmaxY = float(splitCell[1])
                else: #If current cell coordinates are higher/ lower
                than the current MBR min/ max replace them.
                    if float(splitCell[0]) < MBRminX:
                        MBRminX = float(splitCell[0])
                    if float(splitCell[0]) > MBRmaxX:
                        MBRmaxX = float(splitCell[0])
                    if float(splitCell[1]) < MBRminY:
                        MBRminY = float(splitCell[1])
                    if float(splitCell[1]) > MBRmaxY:
                        MBRmaxY = float(splitCell[1])
                rowCellStorage.append(floatSplitCell)
                cellCounter += 1

            #Create the MBR matrix. It will be consisted of two sub
            matrices, the MBRmin/ MBRMax.
            MBRmin = [MBRminX,MBRminY]
            MBRmax = [MBRmaxX,MBRmaxY]
            MBR = []
            MBR.append(MBRmin)
            MBR.append(MBRmax)
            singleRecord.append(MBR) #Put MBR's array as the 2nd
```

```

element
    singleRecord.append(rowCellStorage) #Put the cells array
as the 3rd element
    records.append(singleRecord) #Add each record line in the
records data structure
    counter += 1

```

Έπειτα στόχος μας είναι η δημιουργία του πλέγματος grid 10x10.

Βήμα 1^ο: Βρίσκουμε τα min, max MBR για να ξέρουμε τις ελάχιστες/ μέγιστες συντεταγμένες του πλέγματός μας.

```

#Dataset min/ max
#We need to find the min/ max of MBR's in order to create the grid.
def find_min_max():
    counter = 1
    for record in records:
        if counter == 1: #1st iteration. Set min/ max as the 1st
element's min/ max.
            datasetMinX = record[1][0][0]
            datasetMinY = record[1][0][1]
            datasetMaxX = record[1][1][0]
            datasetMaxY = record[1][1][1]
        else:
            if record[1][0][0] < datasetMinX:
                datasetMinX = record[1][0][0]
            if record[1][0][1] < datasetMinY:
                datasetMinY = record[1][0][1]
            if record[1][1][0] > datasetMaxX:
                datasetMaxX = record[1][1][0]
            if record[1][1][1] > datasetMaxY:
                datasetMaxY = record[1][1][1]
        counter += 1
    return datasetMinX, datasetMinY, datasetMaxX, datasetMaxY

```

Βήμα 2^ο: Βρίσκουμε το πλάτος, ύψος των κελιών του πλέγματος

```

#Calculate cell width/ height
def calculateCellDimensions():
    cellWidth = (datasetMinMax[2] - datasetMinMax[0])/10
    cellHeight = (datasetMinMax[3] - datasetMinMax[1])/10

    return cellWidth,cellHeight

```

Βήμα 3^ο: Υπολογίζουμε τις συντεταγμένες των κελιών του πλέγματος, χρησιμοποιώντας σε μία δομή επανάληψης τα ελάχιστα MBR και τις διαστάσεις των κελιών που βρήκαμε.

```

#Calculate the x/ y coordinates of each cell edge using width/ height
def calculateBoundaries(axis):
    boundaries = []
    if axis == 'x':
        for i in range(11):
            boundaries.append(datasetMinMax[0] + i*cellDimensions[0])
    elif axis == 'y':
        for i in range(11):
            boundaries.append(datasetMinMax[1] + i*cellDimensions[1])

```

```

else:
    print('Axis does not exist')
    return boundaries

```

Βήμα 4^ο: Τέλος δημιουργούμε το grid

```

#Create grid
def createGrid():
    grid = []
    y = 0
    for i in reversed(range(10)):
        x = 0
        for j in range(10):
            #Each cell will contain the bottom left and top right
            corner coordinates
            cell =
            [xBoundaries[x],yBoundaries[y],xBoundaries[x+1],yBoundaries[y+1]]
            grid.append(cell)
            x += 1
        y += 1
    return grid

```

Πλέον αφού έχουμε το grid, ήρθε η ώρα να βρούμε που ανήκει το κάθε αντικείμενο.

```

def findCells(): #Now we need to find where each record belongs
    for record in records: #For each record
        counter = xIndex = yIndex = 0 #Counter increased with each
        cell. Helpful to set xIndex/ yIndex which are our position in the
        grid.
        xMinIndex = yMinIndex = xMaxIndex = yMaxIndex = -1 #min/ max
        indices represent the first/ last cell that each record route
        occupies.
        minFlag = maxFlag = 1

        for cell in grid: #For each grid cell
            #Compare the upper lower edges of each cell with the
            record MBR's to see if it belongs in it.
            if (record[1][0][0] >= cell[0] and record[1][0][0] <=
            cell[2]) and (record[1][0][1] >= cell[1] and record[1][0][1] <=
            cell[3]):
                if minFlag == 1: #Do this only once.
                    xMinIndex = xIndex
                    yMinIndex = yIndex
                    minFlag = 0
                if (record[1][1][0] >= cell[0] and record[1][1][0] <=
            cell[2]) and (record[1][1][1] >= cell[1] and record[1][1][1] <=
            cell[3]):
                    if maxFlag == 1: #Do this only once.
                        xMaxIndex = xIndex
                        yMaxIndex = yIndex
                        maxFlag = 0

        #Set x/y indices in the grid

```

```

        counter += 1
        if counter % 10 == 0:
            xIndex = 0
            yIndex += 1
        else:
            xIndex += 1
        addToIDList(record[0], xMinIndex, yMinIndex, xMaxIndex,
yMaxIndex)

```

Αφού βρούμε το αρχικό και το καταληκτικό κελί, πρέπει να συμπληρώσουμε σε όλα τα κελιά το ID του αντικειμένου.

```

#For each cell between the min/max cells add the record ID
def addToIDList(id,xMin,yMin,xMax,yMax):
    for y in range(10):
        for x in range(10):
            if (x>=xMin and x<=xMax) and (y>=yMin and y<=yMax):
                MBRIDList[y][x].append(id)

```

Έπειτα την ταξινομούμε ανάλογα με το πώς μας ζητείται να εμφανίζεται

```

#Sort the MBRList in the specific order that is request
def sortMBRIDList():
    listStorage = []
    for y in reversed(range(10)):
        for x in range(10):
            listStorage.append(MBRIDList[x][y])

```

Για τη δημιουργία του του “grid.grd” αρχείου έχουμε την εξής συνάρτηση:

```

#Create grd file
def createGrdFile():
    if(os.path.isfile(grdPath)): #If it already exists delete it.
Cause it may need an update
        os.remove("grid.grd")
    f = open("grid.grd", "w") #Create file for writing
    for y in range(10):
        for x in range(10):
            for id in MBRIDList[x][y]:
                record = records[id-1]
                strRecord = convertTupleToString(record) + "\n"
                f.write(strRecord)

```

Ωστόσο επειδή τα στοιχεία δεν είναι στην κατάλληλη μορφή, χρησιμοποιούμε την παρακάτω συνάρτηση για να τα μετατρέψουμε σε string.

```

#Convert tuple to string so it can be added to the file
def convertTupleToString(tuple):
    strTuple = ""
    for item in tuple:
        if isinstance(item, int):
            strTuple += str(item)+','
        else:
            strTuple += str(item)
    return strTuple

```

Για τη δημιουργία του “grid.dir” αρχείου έχουμε:

```
#Create dir file
def createDirFile():
    if(os.path.isfile(dirPath)): #If it already exists delete it.
    Cause it may need an update
        os.remove("grid.dir")
    f = open("grid.dir", "w") #Create file for writing
    f.write("%d %f %f %f %f\n"
%(1,datasetMinMax[0],datasetMinMax[2],datasetMinMax[1],datasetMinMax[
3]))
    linenummer = 2
    for y in range(10):
        for x in range(10):
            f.write("%d %d %d %d\n"
%(linenummer,y,x,len(MBRIDList[x][y])))
            linenummer += 1
```

Τέλος, έχουμε τις κλήσεις των συναρτήσεων

```
#Main
extract_data()
datasetMinMax = find_min_max()
cellDimensions = calculateCellDimensions()
xBoundaries = calculateBoundaries('x')
yBoundaries = calculateBoundaries('y')
grid = createGrid()
findCells()
createGrdFile()
createDirFile()
```

Τα αρχεία που δημιουργήθηκαν κατά την εκτέλεση του εν λόγω μέρους θα χρησιμοποιηθούν παρακάτω για τα μέρη 2,3. Παρακάτω παρουσιάζεται το αποτέλεσμα όταν τρέχουμε την εντολή “python3 Assignment2Part1.py” στους υπολογιστές του τμήματός μας:

```
cse53117@d1380ws03:~/Desktop/ddtry$ python3 Assignment2Part1.py
cse53117@d1380ws03:~/Desktop/ddtry$ ls
Assignment2Part1.py  Assignment2Part3.py  grid.grd      results.txt
Assignment2Part2.py  grid.dir             queries.txt   tiger_roads.csv
cse53117@d1380ws03:~/Desktop/ddtry$
```

Όπως βλέπουμε όντως δημιουργούνται τα αρχεία “grid.grd” και “grid.dir” στον φάκελο.

Preview για Grid.dir:

1	1	-87.752641	-85.565306	31.963678	33.517448
2	2	0	0	0	
3	3	0	1	0	
4	4	0	2	0	
5	5	0	3	0	
6	6	0	4	0	
7	7	0	5	0	
8	8	0	6	4	
9	9	0	7	4	
10	10	0	8	4	
11	11	0	9	0	
12	12	1	0	0	
13	13	1	1	1	
14	14	1	2	2	
15	15	1	3	5	
16	16	1	4	1	
17	17	1	5	0	
18	18	1	6	9	
19	19	1	7	4	
20	20	1	8	4	

Preview για Grid.grd:

1	35664, [[-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, 33.033884], [-87.737479, 33.036270], [-87.736166, 33.038656], [-87.734853, 33.041042], [-87.733540, 33.043428], [-87.732227, 33.045814], [-87.730914, 33.048200], [-87.729601, 33.050586], [-87.728288, 33.052972], [-87.726975, 33.055358], [-87.725662, 33.057744], [-87.724349, 33.060130], [-87.723036, 33.062516], [-87.721723, 33.064902], [-87.720410, 33.067288], [-87.719097, 33.069674], [-87.717784, 33.072060], [-87.716471, 33.074446], [-87.715158, 33.076832], [-87.713845, 33.079218], [-87.712532, 33.081604], [-87.711219, 33.083990], [-87.709906, 33.086376], [-87.708593, 33.088762], [-87.707280, 33.091148], [-87.705967, 33.093534], [-87.704654, 33.095920], [-87.703341, 33.098306], [-87.702028, 33.100692], [-87.700715, 33.103078], [-87.699402, 33.105464], [-87.698089, 33.107850], [-87.696776, 33.110236], [-87.695463, 33.112622], [-87.694150, 33.115008], [-87.692837, 33.117394], [-87.691524, 33.119780], [-87.690211, 33.122166], [-87.688898, 33.124552], [-87.687585, 33.126938], [-87.686272, 33.129324], [-87.684959, 33.131710], [-87.683646, 33.134096], [-87.682333, 33.136482], [-87.681020, 33.138868], [-87.679707, 33.141254], [-87.678394, 33.143640], [-87.677081, 33.146026], [-87.675768, 33.148412], [-87.674455, 33.150798], [-87.673142, 33.153184], [-87.671829, 33.155570], [-87.670516, 33.157956], [-87.669203, 33.160342], [-87.667890, 33.162728], [-87.666577, 33.165114], [-87.665264, 33.167500], [-87.663951, 33.169886], [-87.662638, 33.172272], [-87.661325, 33.174658], [-87.660012, 33.177044], [-87.658699, 33.179430], [-87.657386, 33.181816], [-87.656073, 33.184202], [-87.654760, 33.186588], [-87.653447, 33.188974], [-87.652134, 33.191360], [-87.650821, 33.193746], [-87.649508, 33.196132], [-87.648195, 33.198518], [-87.646882, 33.200904], [-87.645569, 33.203290], [-87.644256, 33.205676], [-87.642943, 33.208062], [-87.641630, 33.210448], [-87.640317, 33.212834], [-87.639004, 33.215220], [-87.637691, 33.217606], [-87.636378, 33.219992], [-87.635065, 33.222378], [-87.633752, 33.224764], [-87.632439, 33.227150], [-87.631126, 33.229536], [-87.629813, 33.231922], [-87.628500, 33.234308], [-87.627187, 33.236694], [-87.625874, 33.239080], [-87.624561, 33.241466], [-87.623248, 33.243852], [-87.621935, 33.246238], [-87.620622, 33.248624], [-87.619309, 33.251010], [-87.617996, 33.253396], [-87.616683, 33.255782], [-87.615370, 33.258168], [-87.614057, 33.260554], [-87.612744, 33.262940], [-87.611431, 33.265326], [-87.610118, 33.267712], [-87.608805, 33.270098], [-87.607492, 33.272484], [-87.606179, 33.274870], [-87.604866, 33.277256], [-87.603553, 33.279642], [-87.602240, 33.282028], [-87.600927, 33.284414], [-87.599614, 33.286800], [-87.598301, 33.289186], [-87.596988, 33.291572], [-87.595675, 33.293958], [-87.594362, 33.296344], [-87.593049, 33.298730], [-87.591736, 33.301116], [-87.590423, 33.303502], [-87.589110, 33.305888], [-87.587797, 33.308274], [-87.586484, 33.310660], [-87.585171, 33.313046], [-87.583858, 33.315432], [-87.582545, 33.317818], [-87.581232, 33.320204], [-87.579919, 33.322590], [-87.578606, 33.324976], [-87.577293, 33.327362], [-87.575980, 33.329748], [-87.574667, 33.332134], [-87.573354, 33.334520], [-87.572041, 33.336906], [-87.570728, 33.339292], [-87.569415, 33.341678], [-87.568102, 33.344064], [-87.566789, 33.346450], [-87.565476, 33.348836], [-87.564163, 33.351222], [-87.562850, 33.353608], [-87.561537, 33.355994], [-87.560224, 33.358380], [-87.558911, 33.360766], [-87.557598, 33.363152], [-87.556285, 33.365538], [-87.554972, 33.367924], [-87.553659, 33.370310], [-87.552346, 33.372696], [-87.551033, 33.375082], [-87.549720, 33.377468], [-87.548407, 33.379854], [-87.547094, 33.382240], [-87.545781, 33.384626], [-87.544468, 33.387012], [-87.543155, 33.389398], [-87.541842, 33.391784], [-87.540529, 33.394170], [-87.539216, 33.396556], [-87.537903, 33.398942], [-87.536590, 33.401328], [-87.535277, 33.403714], [-87.533964, 33.406100], [-87.532651, 33.408486], [-87.531338, 33.410872], [-87.530025, 33.413258], [-87.528712, 33.415644], [-87.527399, 33.418030], [-87.526086, 33.420416], [-87.524773, 33.422802], [-87.523460, 33.425188], [-87.522147, 33.427574], [-87.520834, 33.429960], [-87.519521, 33.432346], [-87.518208, 33.434732], [-87.516895, 33.437118], [-87.515582, 33.439504], [-87.514269, 33.441890], [-87.512956, 33.444276], [-87.511643, 33.446662], [-87.510330, 33.449048], [-87.509017, 33.451434], [-87.507704, 33.453820], [-87.506391, 33.456206], [-87.505078, 33.458592], [-87.503765, 33.460978], [-87.502452, 33.463364], [-87.501139, 33.465750], [-87.499826, 33.468136], [-87.498513, 33.470522], [-87.497200, 33.472908], [-87.495887, 33.475294], [-87.494574, 33.477680], [-87.493261, 33.480066], [-87.491948, 33.482452], [-87.490635, 33.484838], [-87.489322, 33.487224], [-87.488009, 33.489610], [-87.486696, 33.491996], [-87.485383, 33.494382], [-87.484070, 33.496768], [-87.482757, 33.499154], [-87.481444, 33.501540], [-87.480131, 33.503926], [-87.478818, 33.506312], [-87.477505, 33.508698], [-87.476192, 33.511084], [-87.474879, 33.513470], [-87.473566, 33.515856], [-87.472253, 33.518242], [-87.470940, 33.520628], [-87.469627, 33.523014], [-87.468314, 33.525400], [-87.467001, 33.527786], [-87.465688, 33.530172], [-87.464375, 33.532558], [-87.463062, 33.534944], [-87.461749, 33.537330], [-87.460436, 33.539716], [-87.459123, 33.542102], [-87.457810, 33.544488], [-87.456497, 33.546874], [-87.455184, 33.549260], [-87.453871, 33.551646], [-87.452558, 33.554032], [-87.451245, 33.556418], [-87.449932, 33.558804], [-87.448619, 33.561190], [-87.447306, 33.563576], [-87.445993, 33.565962], [-87.444680, 33.568348], [-87.443367, 33.570734], [-87.442054, 33.573120], [-87.440741, 33.575506], [-87.439428, 33.577892], [-87.438115, 33.580278], [-87.436802, 33.582664], [-87.435489, 33.585050], [-87.434176, 33.587436], [-87.432863, 33.589822], [-87.431550, 33.592208], [-87.430237, 33.594594], [-87.428924, 33.596980], [-87.427611, 33.599366], [-87.426298, 33.601752], [-87.424985, 33.604138], [-87.423672, 33.606524], [-87.422359, 33.608910], [-87.421046, 33.611296], [-87.419733, 33.613682], [-87.418420, 33.616068], [-87.417107, 33.618454], [-87.415794, 33.620840], [-87.414481, 33.623226], [-87.413168, 33.625612], [-87.411855, 33.627998], [-87.410542, 33.630384], [-87.409229, 33.632770], [-87.407916, 33.635156], [-87.406603, 33.637542], [-87.405290, 33.639928], [-87.403977, 33.642314], [-87.402664, 33.644700], [-87.401351, 33.647086], [-87.400038, 33.649472], [-87.398725, 33.651858], [-87.397412, 33.654244], [-87.396099, 33.656630], [-87.394786, 33.659016], [-87.393473, 33.661402], [-87.392160, 33.663788], [-87.390847, 33.666174], [-87.389534, 33.668560], [-87.388221, 33.670946], [-87.386908, 33.673332], [-87.385595, 33.675718], [-87.384282, 33.678104], [-87.382969, 33.680490], [-87.381656, 33.682876], [-87.380343, 33.685262], [-87.379030, 33.687648], [-87.377717, 33.690034], [-87.376404, 33.692420], [-87.375091, 33.694806], [-87.373778, 33.697192], [-87.372465, 33.699578], [-87.371152, 33.701964], [-87.369839, 33.704350], [-87.368526, 33.706736], [-87.367213, 33.709122], [-87.365900, 33.711508], [-87.364587, 33.713894], [-87.363274, 33.716280], [-87.361961, 33.718666], [-87.360648, 33.721052], [-87.359335, 33.723438], [-87.358022, 33.725824], [-87.356709, 33.728210], [-87.355396, 33.730596], [-87.354083, 33.732982], [-87.352770, 33.735368], [-87.351457, 33.737754], [-87.350144, 33.740140], [-87.348831, 33.742526], [-87.347518, 33.744912], [-87.346205, 33.747298], [-87.344892, 33.749684], [-87.343579, 33.752070], [-87.342266, 33.754456], [-87.340953, 33.756842], [-87.339640, 33.759228], [-87.338327, 33.761614], [-87.337014, 33.763999], [-87.335701, 33.766385], [-87.334388, 33.768771], [-87.333075, 33.771157], [-87.331762, 33.773543], [-87.330449, 33.775929], [-87.329136, 33.778315], [-87.327823, 33.780701], [-87.326510, 33.783087], [-87.325197, 33.785473], [-87.323884, 33.787859], [-87.322571, 33.790245], [-87.321258, 33.792631], [-87.319945, 33.795017], [-87.318632, 33.797403], [-87.317319, 33.799789], [-87.316006, 33.802175], [-87.314693, 33.804561], [-87.313380, 33.806947], [-87.312067, 33.809333], [-87.310754, 33.811719], [-87.309441, 33.814105], [-87.308128, 33.816491], [-87.306815, 33.818877], [-87.305502, 33.821263], [-87.304189, 33.823649], [-87.302876, 33.826035], [-87.301563, 33.828421], [-87.300250, 33.830807], [-87.298937, 33.833193], [-87.297624, 33.835579], [-87.296311, 33.837965], [-87.295, 33.840351], [-87.293687, 33.842737], [-87.292374, 33.845123], [-87.291061, 33.847509], [-87.289748, 33.849895], [-87.288435, 33.852281], [-87.287122, 33.854667], [-87.285809, 33.857053], [-87.284496, 33.859439], [-87.283183, 33.861825], [-87.281870, 33.864211], [-87.280557, 33.866597], [-87.279244, 33.868983], [-87.277931, 33.871369], [-87.276618, 33.873755], [-87.275305, 33.876141], [-87.273992, 33.878527], [-87.272679, 33.880913], [-87.271366, 33.883299], [-87.270053, 33.885685], [-87.268740, 33.888071], [-87.267427, 33.890457], [-87.266114, 33.892843], [-87.264801, 33.895229], [-87.263488, 33.897615], [-87.262175, 33.899999], [-87.260862, 33.902385], [-87.259549, 33.904771], [-87.258236, 33.907157], [-87.256923, 33.909543], [-87.255610, 33.911929], [-87.254297, 33.914315], [-87.252984, 33.916701], [-87.251671, 33.919087], [-87.250358, 33.921473], [-87.249045, 33.923859], [-87.247732, 33.926245], [-87.246419, 33.928631], [-87.245106, 33.931017], [-87.243793, 33.933403], [-87.242480, 33.935789], [-87.241167, 33.938175], [-87.239854, 33.940561], [-87.238541, 33.942947], [-87.237228, 33.945333], [-87.235915, 33.947719], [-87.234602, 33.950105], [-87.233289, 33.952491], [-87.231976, 33.954877], [-87.230663, 33.957263], [-87.229350, 33.959649], [-87.228037, 33.962035], [-87.226724, 33.964421], [-87.225411, 33.966807], [-87.224098, 33.969193], [-87.222785, 33.971579], [-87.221472, 33.973965], [-87.220159, 33.976351], [-87.218846, 33.978737], [-87.217533, 33.981123], [-87.216220, 33.983509], [-87.214907, 33.985895], [-87.213594, 33.988281], [-87.212281, 33.990667], [-87.210968, 33.993053], [-87.209655, 33.995439], [-87.208342, 33.997825], [-87.207029, 34.000211], [-87.205716, 34.002597], [-87.204403, 34.004983], [-87.203090, 34.007369], [-87.201777, 34.009755], [-87.200464, 34.012141], [-87.199151, 34.014527], [-87.197838, 34.016913], [-87.196525, 34.019299], [-87.195212, 34.021685], [-87.193899, 34.024071], [-87.192586, 34.026457], [-87.191273, 34.028843], [-87.189960, 34.031229], [-87.188647, 34.033615], [-87.187334, 34.035999], [-87.186021, 34.038385], [-87.184708, 34.040771], [-87.183395, 34.043157], [-87.182082, 34.045543], [-87.180769, 34.047929], [-87.179456, 34.050315], [-87.178143, 34.052701], [-87.176830, 34.055087], [-87.175517, 34.057473], [-87.174204, 34.059859], [-87.172891, 34.062245], [-87.171578, 34.064631], [-87.170265, 34.067017], [-87.168952, 34.069403], [-87.167639, 34.071789], [-87.166326, 34.074175], [-87.165013, 34.076561], [-87.163700, 34.078947], [-87.162387, 34.081333], [-87.161074, 34.083719], [-87.159761, 34.086105], [-87.158448, 34.088491], [-87.157135, 34.090877], [-87.155822, 34.093263], [-87.154509, 34.095649], [-87.153196, 34.098035], [-87.151883, 34.100421], [-87.150570, 34.102807], [-87.149257, 34.105193], [-87.147944, 34.107579], [-87.146631, 34.109965], [-87.145318, 34.112351], [-87.144005, 34.114737], [-87.142692, 34.117123], [-87.141379, 34.119509], [-87.140066, 34.121895], [-87.138753, 34.124281], [-87.137440, 34.126667], [-87.136127, 34.129053], [-87.134814, 34.131439], [-87.133501, 34.133825], [-87.132188, 34.136211], [-87.130875, 34.138597], [-87.129562, 34.140983], [-87.128249, 34.143369], [-87.126936, 34.145755], [-87.125623, 34.148141], [-87.124310, 34.150527], [-87.122997, 34.152913], [-87.121684, 34.155299], [-87.120371, 34.157685], [-87.119058, 34.160071], [-87.117745, 34.162457], [-87.116432, 34.164843], [-87.115119, 34.167229], [-87.113806, 34.169615], [-87.112493, 34.171999], [-87.111180, 34.174385], [-87.109867, 34.176771], [-87.108554, 34.179157], [-87.107241, 34.181543], [-87.105928, 34.183929], [-87.104615, 34.186315], [-87.103302, 34.188701], [-87.101989, 34.191087], [-87.100676, 34.193473], [-87.099363, 34.195859], [-87.098050, 34.198245], [-87.096737, 34.200631], [-87.095424, 34.203017], [-87.094111, 34.205403], [-87.092798, 34.207789], [-87.091485, 34.210175], [-87.090172, 34.212561], [-87.088859, 34.214947], [-87.087546, 34.217333], [-87.086233, 34.219719], [-87.084920, 34.222105], [-87.083607, 34.224491], [-87.082294, 34.226877], [-87.080981, 34.229263], [-87.079668, 34.231649], [-87.078355, 34.234035], [-87.077042, 34.236421], [-87.075729, 34.238807], [-87.074416, 34.241193], [-87.073103, 34.243579], [-87.071790, 34.245965], [-87.070477, 34.248351], [-87.069164, 34.250737], [-87.067851, 34.253123], [-87.066538, 34.25
---	---	---

Μέρος 2^ο:

Μεγάλο μέρος του κώδικα επαναλαμβάνεται από το πρώτο μέρος γι' αυτό και παραλείπεται. Αντ' αυτού θα επικεντρωθούμε στα νέα κομμάτια.

```
#Paths. Should be changed accordingly
path =
'/usr/home/students/stud15/cse53117/Desktop/ddtry/tiger_roads.csv'
grdPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.grd'
dirPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.dir'
queriesPath =
'/usr/home/students/stud15/cse53117/Desktop/ddtry/queries.txt'
```

Εξαγωγή δεδομένων από το grd αρχείο με τη βοήθεια του dir αρχείου

```
#Extract data
def extract_from_grd():
    with open(dirPath, mode='r') as dir, open(grdPath, mode='r') as
grd: # Open files for reading.
        firstLine = dir.readline() #Skip first line
        for line in dir:
            splitLine = line.split() #Split dir line contents where
there are spaces " "

            for i in range(int(splitLine[3])): #splitLine[3]
represents the number of roads that belong to each cell
                #So we need to append the next int(splitLine[3])
elements, from grid.grd, into the correct list cell

            add_to_list(int(splitLine[1]),int(splitLine[2]),grd.readline())
#grd.readline() reads the next grd line
```

Τοποθέτηση αυτών των δεδομένων σε μία νέα δομή δεδομένων

```
#Add the data from the grd file to the record data structure
def add_to_list(x,y,record):
    splitRecord = [ "{}".format(x) for x in
list(csv.reader([record], delimiter=',', quotechar='"'))[0] ]
    records[x][y].append(splitRecord)
```

Για την ανάγνωση των queries από το αρχείο “queries.txt” έχουμε:

```
#Read queries from the queries.txt file
def read_queries():
    queries = []
    with open(queriesPath, mode='r') as que: #Open file for reading
        for line in que:
            query = []
```

```

        line = line.split(',')[1] #extract each query coordinate
by splitting each line with the ',' delimiter
        for x in line.split():
            query.append(float(x)) #Since it's string cast it as
float so we can use it for numeric operations
            queries.append(query)
    return queries

```

Για την ανίχνευση του της τομής του MBR με το παράθυρο του query έχουμε τον ακόλουθο κώδικα, ο οποίος είναι και ο πυρήνας του part 2:

```

#Detect the intersection between the MBR of an object and the query
window
def detect_intersection():
    for query in queries: #For each query
        counter = xIndex = yIndex = 0 #Helper counters. Help with the
position while traversing through the grid
        cellIntersections = 0 #Count number of cells intersected by
the query window
        intersectedObjects = [] #List of the intersected objects
        windowCo = [query[0],query[2],query[1],query[3]] #Query
window coordinates. They have been "swapped" around a bit so that the
order is xmin,ymin,xmax,ymax
        for cell in grid:
            if(windowCo[0] <= cell[2] and windowCo[2] >= cell[0] and
windowCo[1] <= cell[3] and windowCo[3] >= cell[1]): #Compare window/
cell bottom left and top right edges
                cellIntersections += 1 #If they intersect increase
counter by 1
                for i in range(len(records[yIndex][xIndex])):
#len(records[yIndex][xIndex]) is the number of objects that belong in
each grid
                    objectCo =
originalRecords[int(records[yIndex][xIndex][i][0].strip('"'))-1][1]
                    #objectCo broken down step by step.
                    #1. records[yIndex][xIndex][i][0] is the ID of
each one of the objects that belong in each cell.
                    #2. records[yIndex][xIndex][i][0].strip('"')
removes the excess ' "' that were added by the conversion from grd
to data structure
                    #3.
int(records[yIndex][xIndex][i][0].strip('"'))-1. Since it's string
make it int. -1 since the the object with ID = n, is in the n-1
position of the list
                    #4.
originalRecords[int(records[yIndex][xIndex][i][0].strip('"'))-1][1].
We use original records since the conversion form grd messes up with
the record syntax.
                    if(windowCo[0] <= objectCo[1][0] and windowCo[1]
<= objectCo[1][1]): #Compare object MBR and window bottom-left/top-
right edge coordinates
                        if(windowCo[2] >= objectCo[0][0] and
windowCo[3] >= objectCo[0][1]):
                            if(originalRecords[int(records[yIndex][xIndex][i][0].strip('"'))-
1][0] not in intersectedObjects): #Only add the ID if it's the first
time detecting it.Filter out the duplicates
                                intersectedObjects.append(originalRecords[int(records[yIndex][xIndex][i][0].strip('"'))-1][0])

```

```

        #Set our position in the grid
        counter += 1
        if counter % 10 == 0:
            yIndex = 0
            xIndex += 1
        else:
            yIndex += 1
        queryObjectResults.append(intersectedObjects)
        queryCellResults.append(cellIntersections)

```

Τέλος έχει μείνει η εκτύπωση των αποτελεσμάτων με τη μορφή που μας ζητήθηκε:

```

#Print results in the requested format
def print_results():
    for i in range(len(queries)):
        print('Query',i+1 , 'results:')
        print(*queryObjectResults[i], sep = ' ')
        print('Cells:',queryCellResults[i])
        print('Results:',len(queryObjectResults[i]))
        print('-----')

```

Η main για την κλήση των συναρτήσεων:

```

#Main
extract_from_grd()
queries = read_queries()
queryObjectResults = []
queryCellResults = []
extract_data()
datasetMinMax = find_min_max()
cellDimensions = calculateCellDimensions()
xBoundaries = calculateBoundaries('x')
yBoundaries = calculateBoundaries('y')
grid = createGrid()
detect_intersection()
print_results()

```

Τα αποτελέσματα όταν κάποιος τρέξει το command “python3 Assignment2Part2.py” στους υπολογιστές του τμήματός μας:

```
scylla.cs.uoi.gr - PuTTY
cse53117@dl1380ws03:~/Desktop/ddtry$ python3 Assignment2Part2.py
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
cse53117@dl1380ws03:~/Desktop/ddtry$
```

Μέρος 3°

```
#Paths. Should be changed accordingly
path =
'/usr/home/students/stud15/cse53117/Desktop/ddtry/tiger_roads.csv'
grdPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.grd'
dirPath = '/usr/home/students/stud15/cse53117/Desktop/ddtry/grid.dir'
queriesPath =
'/usr/home/students/stud15/cse53117/Desktop/ddtry/queries.txt'
```

Όπως και στο part 2 έτσι και εδώ το μεγαλύτερο κομμάτι κώδικα αποτελεί αντίγραφο των δύο προηγούμενων. Όλη η ουσία του μέρους 3 είναι η ανίχνευση των αντικειμένων που όντως τέμνονται από το παράθυρο του query και όχι μόνο το MBR τους.

```

#This function detects the objects that are truly intersected by the
query window.
def detect_true_intersection():
    turn = 0 #Query turns
    for query in queries:
        windowCo = [query[0],query[2],query[1],query[3]] #Window
coordinates
        windowEdges =
[[windowCo[0],windowCo[1]], [windowCo[2],windowCo[1]], [windowCo[0],win
dowCo[3]], [windowCo[2],windowCo[3]]] #Tuple that contains the
coordinates of each query window edge
        windowSides =
[[windowEdges[0],windowEdges[1]], [windowEdges[0],windowEdges[2]], [win
dowEdges[2],windowEdges[3]], [windowEdges[1],windowEdges[3]]] #Tuple
that contains the coordinates of each query window side
        truelyIntersepted = [] #Here will be added the objects that
are truly intersected
        for objID in queryObjectResults[turn]: #Scan only the objects
whose MBR's are intersepted
            #First simple case: The window x/y axis completely cover
up the respective object MBR axis
            if((windowCo[0] <= originalRecords[objID-1][1][0][0] and
windowCo[2] >= originalRecords[objID-1][1][1][0]) or\
               (windowCo[1] <= originalRecords[objID-1][1][0][1] and
windowCo[3] >= originalRecords[objID-1][1][1][1])):
                if(objID not in truelyIntersepted):
                    truelyIntersepted.append(objID)

            #Second case: Gotta check individually each line segment
for intersections
            points = originalRecords[objID-1][2] #Tuple that has all
of the line segment points
            for line in windowSides: #For each window side
                for i in range(len(points)-1): #For each line segment
point
                    #Calculate t,u using the formulas provided by the
assignment
                    tNumerator = (points[i][0] -
line[0][0])*(line[0][1] - line[1][1]) - (points[i][1] -
line[0][1])*(line[0][0] - line[1][0])
                    tDenominator = (points[i][0] -
points[i+1][0])*(line[0][1] - line[1][1]) - (points[i][1] -
points[i+1][1])*(line[0][0] - line[1][0])
                    if(tDenominator != 0):
                        t = tNumerator/tDenominator
                    uNumerator = (points[i][0] -
line[0][0])*(points[i][1] - points[i+1][1]) - (points[i][1] -
line[0][1])*(points[i][0] - points[i+1][0])
                    uDenominator = (points[i][0] -
points[i+1][0])*(line[0][1] - line[1][1]) - (points[i][1] -
points[i+1][1])*(line[0][0] - line[1][0])
                    if(uDenominator != 0):
                        u = uNumerator/uDenominator
                    if(0 <= t <= 1 and 0 <= u <= 1): #If true there's
an intersection
                        if (objID not in truelyIntersepted):
                            truelyIntersepted.append(objID)

        turn += 1
    queryTruelyInterseptedObjects.append(truelyIntersepted)

```

Η φόρμουλες που χρησιμοποιήθηκαν ήταν αυτές που παρήχθησαν μαζί με την εκφώνηση της άσκησης.

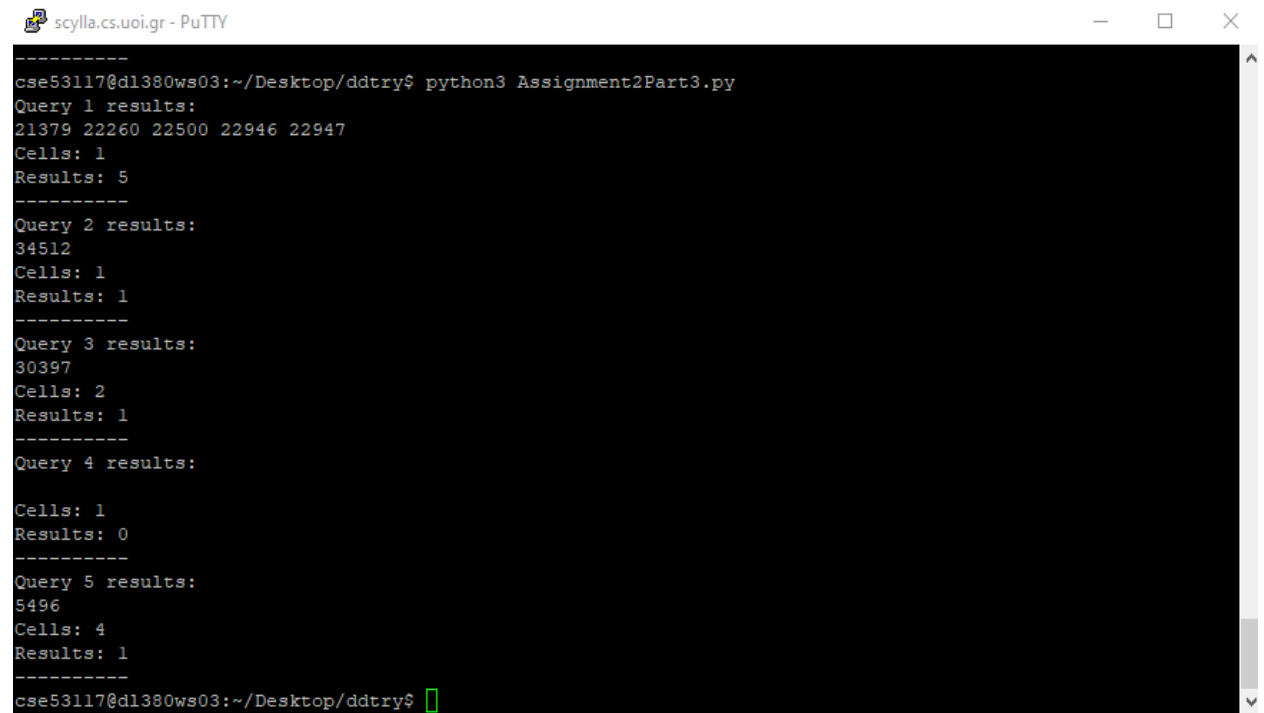
Τέλος, εκτυπώνουμε τα αποτελέσματα:

```
#Print the results of the refined intersection stage
def print_refined_results():
    for i in range(len(queries)):
        print('Query',i+1 , 'results:')
        print(*queryTruelyInterseptedObjects[i], sep = ' ')
        print('Cells:',queryCellResults[i])
        print('Results:',len(queryTruelyInterseptedObjects[i]))
        print('-----')
```

Η main για την κλήση των συναρτήσεων:

```
#Main
extract_from_grd()
queries = read_queries()
queryObjectResults = []
queryCellResults = []
extract_data()
datasetMinMax = find_min_max()
cellDimensions = calculateCellDimensions()
xBoundaries = calculateBoundaries('x')
yBoundaries = calculateBoundaries('y')
grid = createGrid()
detect_intersection()
queryTruelyInterseptedObjects = []
detect_true_intersection()
print_refined_results()
```

Για την εκτέλεση τρέχουμε “python3 Assignment2Part3.py” στο τερματικό:



The screenshot shows a PuTTY terminal window titled "scylla.cs.uoi.gr - PuTTY". The terminal displays the output of running the command "python3 Assignment2Part3.py". The output consists of five queries, each with its results, the number of cells, and the number of results. The terminal text is as follows:

```
-----  
cse53117@dl1380ws03:~/Desktop/ddtry$ python3 Assignment2Part3.py  
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: 4  
Results: 1  
-----  
cse53117@dl1380ws03:~/Desktop/ddtry$
```