ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων

(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2018-19)

ΕΡΓΑΣΙΑ 2 – Χωρικά Δεδομένα

Όνοματεπώνυμο: Γεώργιος-Παναγιώτης Ταξιάρχου **ΑΜ** : 2552

Μέρος 1:

Το πρόγραμμά μας , έχει υλοποιηθεί με δυο βασικές κλάσεις Cell και Grid.

```
class Cell():
    def __init__(self,gridx,gridy,minX,minY,maxX,maxY):
        self.gridx = gridx
        self.gridy = gridy
        self.minX = minX
        self.maxX = maxX
        self.minY = minY
        self.maxY = maxY
        cellpoints = []
        self.cellpoints = cellpoints
    def getCellPoints(self):
        return self.cellpoints
    def addPoint(self,point):
        self.cellpoints.append(point)
```

```
class Grid():
    def __init__(self,minX,maxX,minY,maxY):
        self.minX = minX
        self.maxX = maxX
        self.minY = minY
        self.maxY = maxY

        xstep = (maxX-minX)/10
        ystep = (maxY-minY)/10
        grid = []
        self.grid = grid

        for x in range(0,10):
            for y in range(0,10):
                cellminX = minX + x*xstep
                cellminY = minY + y*ystep
                cellmaxX = cellminX + xstep
                cellmaxY = cellminY + ystep
                cell = Cell(x,y,cellminX,cellminY,cellmaxX,cellmaxY)
                grid.append(cell)
```

Στην main() , φτιάχνουμε ένα πίνακα για τα κελία και έναν πίνακα για τα σημεία εστιατορίων στο χάρτη διαβάζοντας απο το αρχείο beijing_restaurants.txt . Στην συνέχεια , βρίσκουμε το ελαχιστο και μέγιστο για την κάθε διάσταση χ,ψ : minX,minY,maxX,maxY , χωρίζουμε το grid σε κελιά με βάση το βήμα dx , dy και βρίσκουμε την θέση ενός σημείου με τον υπολογισμό του xposition , yposition

```
for point in coordsmatrix:
    xposition = (point[0] - minX) / dx
    yposition = (point[1] - minY) / dy

    cellX =int( xposition )
    cellY =int( yposition )

    for cell in grid.getGrid():
        if(cellX==cell.gridx and cellY==cell.gridy):
            cell.addPoint(point)
            pointsmatrix.append(point)
            pointsadded+=1
        elif(cellX==10 and cell.gridx==9 and cellY==cell.gridy):
            cell.addPoint(point)
            pointsmatrix.append(point)
            pointsadded+=1
        elif(cellY==10 and cell.gridy==9 and cellX==cell.gridx):
            cell.addPoint(point)
            pointsmatrix.append(point)
            pointsadded+=1
        elif(cellX==10 and cellY==10 and cell.gridx==9 and cell.gridy==9):
            cell.addPoint(point)
            pointsmatrix.append(point)
            pointsadded+=1
griddir.write(minX.__str__()+" "+maxX.__str__()
              +" "+minY.__str__()+" "+maxY.__str__()+"\n")
grdline = 0
for cell in grid.getGrid():
    if(len(cell.cellpoints)!=0):
        griddir.write(cell.gridx.__str__()+" "+cell.gridy.__str__()
                      +" "+grdline.__str__()+" "+len(cell.cellpoints).__str__()+"\n")
    for cellpoint in cell.cellpoints:
        gridgrd.write(cellpoint[2].__str__()+" "+cellpoint[0].__str__()+" "
                      +cellpoint[1].__str__()+"\n")
        grdline+=1
```

Τέλος, γράφουμε τα αποτελέσματα σε δυο αρχεία. Ένα για τα κελιά και ένα για τα σημεία εστιατορίων αντίστοιχα.

Μέρος 2:

Το πρόγραμμά μας , έχει μια μέθοδο **getCellPos(x,y,limitsarray)** , που επιστρέφει τις συντεταγμένες ενός σημείου , κανονικοποιημένες για τις συντεταγμένες του Grid απο το 0 εως το 10.

Αρχικά , βρίσκει όλα τα κελιά τα οποία είναι ολόκληρα η μερικώς μέσα στο window query. Στη συνέχεια ελέγχει ποια είναι μερικώς μέσα στο παράθυρο και ποια είναι ολόκληρα για να επιστρέψει τα σημεία τους χωρίς έλεγχο.

```
firstcell = getCellPos(x_low,y_low,limitsarray)
secondcell = getCellPos(x_high,y_high,limitsarray)

cellsminX = firstcell[2]
cellsminY = firstcell[3]
cellsmaxX = secondcell[2]
cellsmaxY = secondcell[3]

intersectingcellsarray = []
for i in range(firstcell[0],secondcell[0]+1):
    for j in range(firstcell[1],secondcell[1]+1):
        if(i<10 and j<10 and i>=0 and j>=0):
            intersectingcellsarray.append([i,j])

for cell in intersectingcellsarray:
    if(cell[0]>=cellsminX and cell[0]+1<=cellsmaxX
        and cell[1]>=cellsminY and cell[1]+1<=cellsmaxY):
        cell.append("full")
    else:
        cell.append("check")
```

Στη συνέχεια , με βάση το τι επέστρεψε ο προηγούμενος έλεγχος , βάζει όλα τα σημεία για τα κελιά που καλύπτονται πλήρως απο το παράθυρο και κάνει έλεγχο για τα σημεία των κελιών που καλύπτονται μερικώς , βάζοντας μόνο αυτά που βρίσκονται μέσα στα όρια του παραθύρου μας.

```
numberofpointsadded = 0
for cell in intersectingcellsarray:
    for cellmatrixcell in cellmatrix:
        if(cellmatrixcell[0]==cell[0] and cellmatrixcell[1]==cell[1]):
            if(cell[2]=="full"):
                for i in range(cellmatrixcell[3]):
                    queryanswer.append(pointsmatrix[cellmatrixcell[2]+i])
                    numberofpointsadded+=1
            elif(cell[2]=="check"):
                for i in range(cellmatrixcell[3]):
                    point = pointsmatrix[cellmatrixcell[2]+i]
                    pointposition = getCellPos(point[1],point[2],limitsarray)
                    pointx = pointposition[2]
                    pointy = pointposition[3]
                    if(pointx>=cellsminX and pointx<=cellsmaxX
                        and pointy>=cellsminY and pointy<=cellsmaxY):
                        queryanswer.append(point)
                        numberofpointsadded+=1

writeToFile(queryanswer)
```

Μέρος 3:

Το πρόγραμμά μας , έχει μια μέθοδο **mindist(q,cell,limitsarray)** , η οποία βρίσκει αρχικά την σχετική θέση ενός σημείου με ένα κελί και στη συνέχεια υπολογίζει και επιστρέφει ανάλογα την απόσταση του σημείου μας από το κελί.

```
def mindist(q,cell,limitsarray):
    qpointposition = getCellPos(q[1],q[2],limitsarray)
    dist = 0
    #cell is on top
    if(qpointposition[0] == cell[0] and cell[1]>qpointposition[1]):
        dqy = abs(qpointposition[3])
        dy = cell[1]
        dist = (dqy-dy)**2
        dist = math.sqrt(dist)
    #cell is on bottom
    elif(qpointposition[0] == cell[0] and cell[1]>qpointposition[1]):
        dqy = abs(qpointposition[3])
        dy = cell[1]+1
        dist = (dqy-dy)**2
        dist = math.sqrt(dist)
    #cell is left
    elif(cell[1]==qpointposition[1] and cell[0]<qpointposition[0]):
        dqx = abs(qpointposition[2])
        dx = cell[0]+1
        dist = (dqx-dx)**2
        dist = math.sqrt(dist)
    #cell is right
    elif(cell[1]==qpointposition[1] and cell[0]>qpointposition[0]):
        dqx = abs(qpointposition[2])
        dx = cell[0]
        dist = (dqx-dx)**2
        dist = math.sqrt(dist)
    #cell is on top right
    elif(cell[1]>qpointposition[1] and cell[0]>qpointposition[0]):
        dqx = abs(qpointposition[2])
        dqy = abs(qpointposition[3])
        dx = cell[0]
        dy = cell[1]
        dist = (dqx-dx)**2 + (dqy-dy)**2
        dist = math.sqrt(dist)
    #cell is on top left
    elif(cell[1]>qpointposition[1] and cell[0]<qpointposition[0]):
        dqx = abs(qpointposition[2])
        dqy = abs(qpointposition[3])
        dx = cell[0]+1
        dy = cell[1]
        dist = (dqx-dx)**2 + (dqy-dy)**2
        dist = math.sqrt(dist)
    #cell is on bottom right
    elif(cell[1]<qpointposition[1] and cell[0]>qpointposition[0]):
        dqx = abs(qpointposition[2])
        dqy = abs(qpointposition[3])
        dx = cell[0]
        dy = cell[1]+1
        dist = (dqx-dx)**2 + (dqy-dy)**2
        dist = math.sqrt(dist)
    #cell is on bottom left
    elif(cell[1]<qpointposition[1] and cell[0]<qpointposition[0]):
        dqx = abs(qpointposition[2])
        dqy = abs(qpointposition[3])
        dx = cell[0]+1
        dy = cell[1]+1
        dist = (dqx-dx)**2 + (dqy-dy)**2
        dist = math.sqrt(dist)
    return dist
```

Επιπλέον, έχει μια μέθοδο **minpointsdist(q1,q2,limitsarray)** που επιστρέφει την απόσταση μεταξύ δυο σημείων και μια μέθοδο **idItem(item)** που αναγνωρίζει εαν ένα στοιχείο είναι κελί ή σημείο.

Οι δυο κύριες μέθοδοι του προγράμματός είναι οι:

appendedqueue(queue,item,point,limitsarray,appendedcells) , που ελέγχει εαν το στοιχείο που θέλουμε να εισάγουμε στην ουρά μας είναι κελί η σημείο και το εισάγει στην ουρά με δυο επιπλέον πεδία , type και distance.

```
def appendedqueue(queue,item,point,limitsarray,appendedcells):
    if(idItem(item)=="cell" and (item not in appendedcells)):
        appendedcells.append(item)
        distance = mindist(point,item,limitsarray)
        tempitem = item
        tempitem.append("cell")
        tempitem.append(distance)
        queue.append(tempitem)
    elif(idItem(item)=="point"):
        distance = minpointsdist(point,item,limitsarray)
        tempitem = item
        tempitem.append("point")
        tempitem.append(distance)
        queue.append(tempitem)
    queue = sortedQueue(queue)
    return queue
```

getNearestNeighbor(queue,point,limitsarray,pointsmatrix,cellmatrix,appendedcells), που κοιτάει εαν το item στην κορυφή της ουράς είναι κελί και το βγάζει , εισάγοντας τα σημεία που περίεχει και τα γειτονικά κελιά του στην ουρά και που ταξινομείται αυτόματα με τη χρήση της **appendqueue()**. Εαν το item , στην κορυφή της ουράς είναι σημείο , τότε το βγάζει και το επιστρέφει σαν τον κοντινότερο γείτονα.

```
def getNearestNeighbor(queue,point,limitsarray,pointsmatrix,cellmatrix,appendedcells):
    while True:
        if(queue == [] ):
            print "NO MORE NEIGHBORS"
            exit(0)
        if(queue[0][-2]=="cell"):
            c = queue[0]
            neighborcells = []
            for i in range(-1,2):
                for j in range(-1,2):
                    if(c[0]+i==c[0] and c[1]+j==c[1]):
                        # print "keepoooo"
                        continue
                    elif(c[0]+i>=0 and c[0]+i<=9 and c[1]+j>=0 and c[1]+j<=9):
                        for cell in cellmatrix:
                            if(cell[0]==c[0]+i and cell[1]==c[1]+j):
                                neighborcells.append(cell)
            queue.pop(0)
            for i in range(c[3]):
                queue = appendedqueue(queue,pointsmatrix[c[2]+i],point,limitsarray,appendedcells)
            for neighborcell in neighborcells:
                queue = appendedqueue(queue,neighborcell,point,limitsarray,appendedcells)
        elif(queue[0][-2]=="point"):
            p = queue[0]
            queue.pop(0)
            yield p
```