Μυλωνάκης Αλέξανδρος

ΑΜ: 3045

Εργασία 2η Διαχείριση Σύνθετων δεδομένων

Γλώσσα προγραμματισμού : Python

Μέρος 1ο:

Κατέβασα το αρχείο tiger.csv και δημιούργησα την κλάση csv_manager.py για να διατρέχω το αρχείο και να αρχικοποιώ την λίστα της οποίας το κάθε αρχείο περιέχει στην 1η θέση το identifier του αντικειμένου, στην 2η τα min_max του αρχείου και στην 3η το linestring

```
def create_mylist(self):
    road_list = [[] for i in range(3)]

    with open(self.csv_path) as csvfile:
        reader = csv.reader(csvfile)
        identifier = 1
        for row in reader:
            if len(row) > 1:
                road_list[0].append(identifier)
                road_list[1].append(self.find_line_minmax(row))
                road_list[2].append(row)
                identifier+=1
    return road_list
```

Η κλάση αυτή επίσης είναι υπεύθυνη να για να βρίσκει τα ελάχιστα και μέγιστα X και Y από όλα τα linestrings έτσι ώστε να φτιάξουμε το grid

```
def __init__(self, csv_path):
    self.csv_path= csv_path
    self.minX =float('inf')
    self.minY=float('inf')
    self.maxX=float('-inf')
    self.maxY=float('-inf')

def find_line_minmax(self,row):
    minX =float('inf')
    minY=float('inf')
    maxX=float('-inf')
    maxY=float('-inf')
    for column in row:
        values = column.split()
        x = float(values[0])
        y = float(values[1])
        if(minX > x):
            minX=x
            if(x<self.minX):
                self.minX=x
        if(minY > y):
            minY=y
            if(y<self.minY):
                self.minY=y
        if(maxX < x):
            maxX=x
            if(x>self.maxX):
                self.maxX=x
        if(maxY < y):
            maxY=y
            if(y>self.maxY):
                self.maxY=y
    s1 = '{} {}'.format(minX, minY)
    s2 = '{} {}'.format(maxX, maxY)
    s3 = [s1,s2]
    return s3
```

Αυτό το πετυχαίνουμε αρχικοποιώντας τόσο τα τοπικά όσο και τα ολικά μέγιστα και ελάχιστα σε τιμές -άπειρο και +άπειρο αντίστοιχα.

Στη συνέχεια για να δημιουργήσω τη σχάρα έφτιαξα την κλάση my_grid.py η στην οποία αρχικοποιώ τα minmax στοιχεία το υψος και το παχος τον κελιών

```
def __init__(self,road_list,csv_manager,number_of_rows):
    self.road_list =road_list
    self.csv_manager = csv_manager
    self.minX = csv_manager.getMinX()
    self.minY = csv_manager.getMinY()
    self.maxX = csv_manager.getMaxX()
    self.maxY = csv_manager.getMaxY()
    self.cell_width = (self.maxX - self.minX) / number_of_rows
    self.cell_height = (self.maxY - self.minY) / number_of_rows
    self.grid_map = {}
    self.number_of_rows =number_of_rows
    for y_coord in range(number_of_rows):
        for x_coord in range(number_of_rows):
            coord = '({},{})'.format(y_coord,x_coord)
            self.grid_map[coord] = []
```

Και τη μέθοδο create grid για την δημιουργία του grid και των αρχείων grid.dir και grid.grd

```
def create_grid(self):

    for row in range(len(self.road_list[0])):
        self.find_MBR_cells(self.split_minXY_maxXY(self.road_list[1][row]),self.road_list[0][row])

    with open('C:\\Users\\Alekos\\Desktop\\projects\\Diaxeirisi\\assignment2\\grid.grd', 'w') as f:
        for key in self.grid_map:
            value = self.grid_map[key]
            if len(value)>0:
                for element in range(len(value)):
                    x =value[element]
                    identifier =self.road_list[0][x-1]
                    mbr_str = ' '.join(map(str, self.road_list[1][x-1]))
                    vertex_str = ','.join([''.join(map(str, vertex)) for vertex in self.road_list[2][x-1]])
                    f.write('{},{},{}\n'.format(identifier,mbr_str,vertex_str))

    with open('C:\\Users\\Alekos\\Desktop\\projects\\Diaxeirisi\\assignment2\\grid.dir', 'w') as f:

        f.write('{} {} {} {}'.format(self.minX,self.maxX, self.minY,self.maxY ))
        for key in self.grid_map:
            elements = len(self.grid_map[key])
            x_coord=(key[1])
            y_coord=(key[3])
            f.write('\n{} {} {}'.format(x_coord,y_coord,elements))
```

Η συνάρτηση find_MBR_cells είναι υπεύθυνη για να βρίσκει τα όρια του MBR του

```
def find_MBR_cells(self, elem_minXY_maxXY, index):
    bottom_left = self.find_vertice_cell( elem_minXY_maxXY[0], elem_minXY_maxXY[1])
    top_right = self.find_vertice_cell( elem_minXY_maxXY[2], elem_minXY_maxXY[3])
    for row in range(bottom_left[1],top_right[1]+1):
        for col in range(bottom_left[0],top_right[0]+1):
            map_key =  '({},{})'.format(row,col)
            self.grid_map[map_key].append(index)
```

Επίσης έχω υλοποιήσει την συνάρτηση fid_vertice_cell που είναι υπεύθυνη για να βρίσκει το κελί που βρίσκεται μια κορυφή

```
def find_vertice_cell(self,x,y):
    row = int((y - self.minY) // self.cell_height)
    col = int((x - self.minX) // self.cell_width)
    if(col == self.number_of_rows):
        col -=1
    if(row == self.number_of_rows):
        row -=1
    cell_value = [row,col]
    return cell_value
```

Μέρος 2ο

Όσο αναφορά το 2ο μέρος δημιούργησα μια κλάση manage_files.py για να διαβάσω τα αρχεία grid.dir

```
    def read_file_and_split(self,file_name):
        with open(file_name, 'r') as f:
            contents = f.read()
            lines = contents.splitlines()
            return lines
```

και grid.grd

```
def read_grd_file(self,file_name):
    with open(file_name) as f:
        my_list = []
        for line in f:
            fields = line.strip().split(",")
            my_grid_to_list = []
            my_grid_to_list.append(fields[0])
            my_grid_to_list.append(fields[1])
            my_grid_to_list.append(fields[2:])
            my_list.append(my_grid_to_list)
    return my_list
```

Η κλάση αυτή επίσης περιέχει δυο ακόμη μεθόδους την read-queries για να διαβάζει το αρχείο queries.txt

```
def read_queries(self,file_name):
    with open(file_name, 'r') as f:
        contents = f.read()
        lines = contents.splitlines()
        for i in range(len(lines)):
            lines[i] = lines[i][2:]
    return lines
```

Και μία μέθοδο για να κάνει split lines με 4 στοιχεία

```
def split_line(self,line):
    n1,n2,n3,n4 =line.split()
    n1 = float(n1)
    n2 = float(n2)
    n3 = float(n3)
    n4= float(n4)
    return  n1,n2,n3,n4
```

Στη συνέχεια δημιούργησα την κλάση selection_queries.py στην οποία αρχικοποιώ πάλι τα min_max to grid map μου καθώς και την συνάρτηση find_vertice_cell

```
def __init__(self,number_of_rows):
    self.minX =float('inf')
    self.minY=float('inf')
    self.maxX=float('-inf')
    self.maxY=float('-inf')
    self.cell_width = 0
    self.cell_height = 0
    self.file_manager = manage_files()
    self.grid_map = {}
    self.number_of_rows = number_of_rows
    for row in range(number_of_rows):
        for col in range(number_of_rows):
            coord = '({},{})'.format(row,col)
            self.grid_map[coord] = []

def find_vertice_cell(self,x,y):
    row = int((y - self.minY) // self.cell_height)
    col = int((x - self.minX) // self.cell_width)
    if(col == self.number_of_rows):
        col -=1
    if(row == self.number_of_rows):
        row -=1
    cell_value = [col,row]
    return cell_value
```

Έφτιαξα μια συνάρτηση create_structure για τη δημιουργία της δομής από την επεξεργασία των αρχείων grid.dir και grid.grd

Η δομή που χρησιμοποίησα όπως και στην δημιουργία του grid στο πρώτο μέρος (ένα dictionary)

```
def create_structure(self,my_grid,my_grid_dir):
    flag = False
    counter=0
    for line in my_grid_dir:
        if not flag:
            flag = True
            self.minX,self.maxX,self.minY,self.maxY = self.file_manager.split_line(line)
            self.cell_width = (self.maxX - self.minX) / self.number_of_rows
            self.cell_height = (self.maxY - self.minY) / self.number_of_rows
            continue
        y,x,elements = line.split()
        key = '({},{})'.format(y,x)
        if int(elements)==0:
            continue
        for i in range(int(elements)):
            self.grid_map[key].append(my_grid[counter])
            counter += 1
```

Μέρος 2ο και 3ο

Έφτιαξα την μέθοδο selection_query για την αρχικοποίηση του παραθύρου(από το query) και για την επεξεργασία τους

```
def selection_query(self, wx_min , wx_max, wy_min,  wy_max):
    bottom_left =  self.find_vertice_cell(wx_min, wy_min)
    top_right = self.find_vertice_cell(wx_max, wy_max)
    identifiers =[]
    refined_identifiers = []
    cells = 0
    for row in range(bottom_left[0],top_right[0]+1):
        for col in range(bottom_left[1],top_right[1]+1):
            coord =  '({},{})'.format(row, col)
            elements = self.grid_map.get(coord)
            if len(elements)==0:
                continue
            cells += 1
            for element in elements:
                mbrminX,mbrminY,mbrmaxX,mbrmaxY = self.file_manager.split_line(element[1])
                if not self.check_window_intersection(wx_min , wy_min, wx_max, wy_max,mbrminX,mbrminY,mbrmaxX,mbrmaxY):
                        continue
                ref_x,ref_y =self.find_refernce_point(mbrminX,mbrminY,mbrmaxX,mbrmaxY)
                if self.point_in_window(ref_x,ref_y,row,col):
                    if self.refinement_step(wx_min , wx_max, wy_min, wy_max,element[1],element[2]):
                        refined_identifiers.append(element[0])
                    identifiers.append(element[0])
    return identifiers,refined_identifiers,cells
```

Για να ελέγξω αν επικαλύπτεται το παράθυρο από κάποιο

MBR αρχικά δημιουργώ την μέθοδο check_window_intersection

```
def check_window_intersection(self,wx_min , wy_min, wx_max, wy_max, mbrx_min ,mbry_min , mbrx_max , mbry_max):
    if wx_max < mbrx_min or wx_min > mbrx_max:
        return False
    if wy_max < mbry_min or wy_min > mbry_max:
        return False
    return True
```

Για να βρω το reference_point έφτιαξα την

find_reference_point

```
def find_refernce_point(self,mbrX,mbrY,windowX,windowY):
    if mbrX>windowX:
        x = mbrX
    else:
        x = windowX
    if mbrY>windowY:
        y= mbrY
    else:
        y=windowY
    return x,y
```

Για να δω αν κάποιο σημείο βρίσκεται μέσα στο παράθυρο

```
def point_in_window(self,ref_x,ref_y,row,col):
    cell_value = self.find_vertice_cell(ref_x,ref_y)
    if cell_value[0]==row:
        if cell_value[1]==col:
            return True
    return False
```

Τέλος για το refinement_stage έφτιαξα την refinement_step η οποία επιστρέφει μια Boolean τιμή αν στο παράθυρο περιέχεται ένα linestring

```
def refinement_step(self,wx_min , wx_max, wy_min,  wy_max,mbrminmax,linestring):
    mbrminX,mbrminY,mbrmaxX,mbrmaxY = self.file_manager.split_line(mbrminmax)
    if wx_min< mbrminX and  wx_max > mbrmaxX:
        return True
    if wy_min < mbrminY and wy_max > mbrmaxY:
        return True
    if self.line_points_interesction(wx_min , wx_max, wy_min,  wy_max,linestring):
        return True
    return False
```

Αυτό το πετυχαίνει με την βοήθεια των μεθόδων line_points_intersection και check_intersection για την περίπτωση που δεν είναι επικαλυπτόμενα σε κάποιον άξονα τα παράθυρα εκ των οποίων η check_intersection ελέγχει αν υπάρχει σημείο τομής μεταξύ μιας πλευράς και του linestring και η line_points_intersection διατρέχει το linestring.

```
def check_intersection(self,x1,y1,x2,y2,point3,point4):
    x3,y3 = point3.split()
    x4,y4 = point4.split()
    x3 = float(x3)
    y3 = float(y3)
    x4 = float(x4)
    y4 = float(y4)
    t=-1
    u=-1
    if (x1-x2)*(y3-y4)-(y1-y2)*(x3-x4)!=0:
        t = ((x1-x3)*(y3-y4)-(y1-y3)*(x3-x4))/((x1-x2)*(y3-y4)-(y1-y2)*(x3-x4))
    if (x1-x2)*(y3-y4)-(y1-y2)*(x3-x4) !=0:
        u = ((x1-x3)*(y1-y2)-(y1-y3)*(x1-x2))/((x1-x2)*(y3-y4)-(y1-y2)*(x3-x4))
    if t>=0 and t<=1 and u>=0 and u<=1:
        return True
    return False

def line_points_interesction(self,wx_min , wx_max, wy_min,  wy_max,linestring):
    prev_point =linestring[0]
    for point in linestring:
        if point == linestring[0]:
            continue
        if self.check_intersection(wx_min,wy_min,wx_min,wy_max,prev_point,point):
            return True
        if self.check_intersection(wx_min,wy_min,wx_max,wy_min,prev_point,point):
            return True
        if self.check_intersection(wx_max,wy_max,wx_min,wy_max,prev_point,point):
            return True
        if self.check_intersection(wx_max,wy_max,wx_max,wy_min,prev_point,point):
            return True
        prev_point = point
    return False
```

Testing:

Για τον έλεγχο των αποτελεσμάτων αρχικά χρησιμοποίησα το αρχείο queries.txt και τα αποτελέσματα

Pre-refinement

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

Post-refinement:

```
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
----------
```

Και κάποια δικά μου τεστάκια:

Στα queries

```
6, -87 -86.9 32.9 33
7,-87.752641 -87.752641 31.963678 31.963678
8, -86 -86 32 32
```

```
Query 6 results:
2800 2801 2802 2824 2825 2826 2869 2881 2904 2905 2907 2918 2919 2920 2921 2922 2923 2924 2925 2926 2927 2928 2929 2930 2931 2932 2933 2957 2972 3116 3147 3148 3217 3218 3239 3240 3241 3242 3243 3244 3245 32
46 3247 3248 3249 3250 3251 3252 3253 3255 3256 3257 3291 3295 3428 3452 3458 3459 3503 3504 3505 3515 3516 3521 3524 3525 3567 3568 3569 3570 3571 3572 3573 3574 3575 3576 3577 3578 3579 3580 3581 3582 3583
 3584 3585 3586 3589 3600 3761 3784 3785 3786 3787 3848 3849 3850 3851 3870 3871 3883 3884 3885 3886 3887 3888 3889 3890 3891 3892 3893 3894 3895 3896 3897 3898 3900 3901 3904 3905 3916 3917 3929 3931 4092 4
095 4138 4141 4162 4189 4190 4210 4211 4212 4213 4214 4215 4216 4217 4219 4220 4222 4249 4254 4260 4269 4466 4467 4482 4483 4484 4501 4528 4529 4541 4542 4543 4547 4548 4549 4550 4551 4552 4553 4554 4558 476
8 4807 4848 4849 4897 4901 4903 4918 4919 4921 4922 4923 4924 4925 4926 4927 4931 4933 4934 4935 4936 4937 4938 4940 4941 4942 4946 4963 5123 5155 5156 5182 5183 5185 5195 5208 5220 5233 5234 5235 5236 5237
5238 5239 5240 5241 5242 5243 5245 5246 5251 5268 5269
Cells: 1
Results: 218
----------
Query 7 results:

Cells: 0
Results: 0
----------
Query 8 results:

Cells: 1
Results: 0
----------
```

Και 1 για όλο τον πίνακα

```
6,-87.752641 -85.565306 31.963678 33.517448
```

```
4013 34014 34032 34033 34034 34042 34043 34047 34048 34069 34091 34138 34139 34148 34149 34151 34162 34163 34164 34175 34176 34179 34181 34219 34220 34261 34262 34285 34286 34319 34320 34325 34326 34329 3433
7 34351 34352 34353 34354 34357 34370 34371 34372 34379 34380 34383 34492 34494 34501 34520 34521 34522 34523 34524 34525 34546 34547 34548 34549 34636 34640 34643 34648 34649 34656 34657 34658 34659 34660 3
4661 34663 34681 34691 34692 34693 34703 34704 34741 34755 34790 34791 34792 34794 34795 34796 34797 34804 34805 34818 34819 34820 34843 34892 34916 34926 34931 34932 34933 34934 34957 34958 34998 35081 3508
5 35090 35177 35206 35225 35426 34384 34694 5493 5494 5495 5497 5498 5499 5500 5501 5502 5503 5504 5505 5506 5507 5508 5509 5510 5511 5512 5513 23977 24791 24805 24812 24844 25578 25581 28763 28790 28791 288
22 28898 29563 29570 29578 29579 29697 23174 23175 23176 23177 23178 23180 23181 23182 23183 23184 23185 23186 23187 23188 23190 23191 23192 23193 23195 23196 23197 23200 23201 23203 23204 23206 23208 23209
23212 23214 23215 23216 23217 23218 23219 23220 23221 23224 23225 23226 23227 23228 23229 23230 23232 23233 23234 23236 23237 23238 23239 23241 23242 23243 23244 23245 23246 23247 23248 23249 23250 23251 232
52 23253 23255 23256 23257 23258 23259 23261 24049 24051 24052 24869 24883 25589 25603 25605 25651 26470 27224 27302 27303 28094 28218 28854 28855 28875 28876 29592 29617 29973 17493 17571 17610 17645 17647
17648 17855 17959 18030 18035 18074 18075 18076 18077 18078 18079 18106 18107 18110 18531 18534 18538 18539 18540 18541 18589 18590 18627 18821 19026 19027 19032 19033 19083 19084 19085 19087 19131 19132 191
33 19134 19315 19436 19501 19519 19520 19522 19524 19563 19566 19567 19570 19612 19832 20039 20040 20041 20043 20044 20048 20049 20090 20125 20126 20127 20418 20419 20484 20502 20503 20504 20558 20559 20560
20561 20600 20602 20932 21022 21025 21027 21076 21077 21103 21104 21105 21106 21109 21313 23202 23213 17304 17306 17307 17308 17309 17381 17391 17398 17403 17404 17405 17406 17407 17408 17409 17410 17411 174
12 17413 17414 17415 17416 17417 17418 17419 17420 17421 17422 17423 17424 17425 17435 17436 17437 17441 17860 17868 17869 17870 17872 17874 17875 17876 17877 17878 17879 17880 17882 17883 17884 17885 17886
17887 17895 17896 18325 18328 18331 18332 18333 18334 18336 18343 18348 18349 18350 18351 18352 18353 18354 18373 18374 18377 18378 18379 18380 18381 18382 18388 18837 18847 18851 18852 18858 18860 18861 188
62 18863 18864 18866 18867 18868 18869 19334 19335 19342 19343 19344 19345 19347 19348 19353 19354 19355 19356 19357 19358 19359 19360 19361 19362 19363 19364 19374 19375 19376 19377 19378 19379 19384 19523
19838 19839 19845 19848 19849 19850 19853 19854 19855 19856 19857 19858 19859 19877 19878 19883 20322 20323 20324 20325 20327 20328 20333 20334 20341 20342 20343 20344 20345 20346 20347 20348 20349 20350 203
51 20360 20361 20362 20363 20366 20506 20819 20820 20821 20822 20823 20824 20827 20828 20830 20833 20834 20835 20836 20837 20838 20862 21275 21276 21317 11961 17873 17893 18297 18323 18346 18347 18360 18361
18843 18875 18876 18877 18882 19341 19352 19368 20337 20338 20339 20340 20832 20850 20851 20861 35092 35095 35113 35118 35119 35120 35121 35122 35123 35124 35125 35126 35129 35130 35131 35132 35133 35134 351
35 35136 35144 35146 35147 35158 35159 35161 35162 35163 35164 35165 35166 35167 35180 35181 35186 35187 35188 35189 35190 35192 35193 35194 35195 35196 35197 35198 35199 35200 35201 35211 35212 35213 35214
35215 35216 35217 35218 35228 35229 35235 35240 35252 35253 35254 35255 35256 35257 35258 35259 35260 35262 35263 35265 35266 35267 35268 35269 35270 35279 35280 35281 35283 35294 35301 35302 35303 35304 353
07 35308 35309 35314 35315 35320 35321 35325 35326 35327 35328 35329 35330 35331 35332 35339 35351 35357 35358 35361 35362 35363 35364 35372 35373 35374 35379 35380 35381 35382 35383 35384 35385 35386 35387
35388 35389 35390 35392 35395 35396 35398 35409 35410 35415 35416 35419 35420 35421 35422 35435 35439 35440 35441 35442 35450 35451 35454 35455 35456 35457 35458 35459 35460 35461 35462 35465 35466 35468 354
77 35485 35486 35487 35494 35500 35501 35504 35505 35508 35509 35511 35522 35523 35524 35525 35526 35527 35528 35529 35530 35531 35532 35535 35536 35537 35538 35539 35540 35541 35542 35550 35551 35553 35560
35561 35567 35568 35570 35571 35572 35574 35575 35576 35577 35578 35583 35587 35590 35591 35593 35595 35600 35601 35602 35603 35604 35606 35607 35608 35609 35610 35611 35612 35616 35629 35636 35637 35639 356
40 35641 35642 35643 35654 35657 35659 35091 35093 35094 35096 35097 35098 35099 35100 35101 35102 35103 35104 35105 35106 35107 35108 35109 35110 35111 35112 35114 35115 35117 35127 35128 35139 35140 35142
35143 35145 35148 35149 35152 35153 35154 35155 35156 35157 35160 35168 35169 35170 35171 35173 35174 35175 35178 35179 35182 35184 35185 35191 35202 35203 35204 35208 35209 35210 35219 35221 35222 35223 352
24 35227 35230 35231 35234 35236 35237 35238 35239 35241 35242 35243 35244 35245 35246 35247 35248 35249 35250 35251 35261 35264 35271 35274 35275 35276 35278 35282 35285 35286 35287 35288 35289 35290 35291
35295 35296 35297 35299 35300 35305 35306 35310 35311 35312 35316 35317 35318 35319 35322 35323 35324 35334 35340 35341 35342 35345 35346 35347 35348 35349 35352 35353 35354 35355 35356 35359 35360 35365 353
66 35369 35371 35375 35376 35377 35378 35393 35394 35397 35399 35400 35401 35403 35405 35406 35407 35408 35411 35412 35413 35414 35423 35424 35425 35427 35428 35429 35430 35432 35433 35434 35436 35437 35438
35445 35447 35448 35449 35453 35463 35464 35467 35471 35472 35473 35475 35476 35478 35479 35480 35481 35482 35483 35488 35489 35490 35491 35492 35493 35495 35496 35497 35498 35499 35502 35503 35506 35507 355
12 35513 35515 35516 35517 35518 35519 35520 35521 35533 35534 35543 35544 35549 35552 35554 35555 35556 35557 35558 35559 35562 35563 35564 35565 35566 35569 35573 35579 35584 35585 35586 35588 35589 35592
35596 35597 35598 35599 35605 35614 35615 35618 35623 35624 35625 35626 35627 35628 35630 35631 35632 35633 35634 35635 35638 35644 35645 35646 35648 35649 35650 35651 35652 35653 35655 35656 35658 35660 356
61 35662 35663 9517 9518 9519 9520 9521 9522 9523 9543 9551 9552 9554 9555 9562 9565 9568 9569 9570 9571 9572 9594 9595 9602 9614 9615 9616 9617 9618 9626 9627 9635 9637 9638 9639 9650 9651 9652 9653 9654 96
55 9669 9670 9674 9680 9689 9690 9691 9692 9693 9721 9725 9726 9737 9740 9741 9742 9743 9744 9745 9746 9747 9772 9776 9777 9778 9792 9793 9794 9808 9813 9822 9823 9824 9839 9840 9841 9842 9843 9861 9862 9872
 9873 35116 35141 35150 35151 35176 35183 35205 35207 35220 35226 35273 35284 35293 35298 35335 35336 35337 35338 35343 35350 35391 35402 35469 35470 35484 35514 35545 35546 35547 35548 35619 35620 35621 356
22 9524 5496 23194 23223 23240 35452 9567
Cells: 85
Results: 35668
```

Τέλος για να τρέξω τον κώδικα μου έχω υλοποιήσει μια main.py

```
manager1 = csv_manager("C:\\Users\\Alekos\\Desktop\\projects\\Diaxeirisi\\assignment2\\tiger_roads.csv")
road_list = manager1.create_mylist()
grid1 = my_grid(road_list,manager1,10)
this_grid = grid1.create_grid()

my_selection_queries = selection_queries(grid1.get_number_of_rows())
my_manager = manage_files()
my_grid_dir = my_manager.read_file_and_split('grid.dir')
the_grid = my_manager.read_grd_file('grid.grd')
my_selection_queries.create_structure(the_grid,my_grid_dir)

queries=my_manager.read_queries('queries.txt')
counter = 1

for query in queries:
    minX,maxX,minY,maxY = my_manager.split_line(query)
    identifiers,refined_identifiers,cells = my_selection_queries.selection_query(minX,maxX,minY,maxY)
    """
    identifiers_str = ' '.join(map(str,identifiers))
    #print("Pre refinement stage results")
    print("Query {} results:".format(counter))
    print(identifiers_str)
    print("Cells:",cells)
    print("Results:",len(identifiers))
    print('----------')
    #print("Post refinement stage results")
    """
    ref_identifiers_str = ' '.join(map(str, refined_identifiers))
    print("Query {} results:".format(counter))
    print(ref_identifiers_str)
    print("Cells:",cells)
    print("Results:",len(refined_identifiers))
    print('----------')
    counter += 1
```

Στην οποία τα print τα οποία βρίσκονται μέσα σε σχόλια αφορούν τον κώδικα πριν το refinement step και τα υπόλοιπα είναι οι κλήσεις τον μεθόδων. Καθώς μου ζητήθηκε στην προηγουμένη άσκηση να μην δίνετε δυναμικά το path του αρχείου κράτησα την manual υλοποίηση

```
manager1 = csv_manager("C:\\Users\\Alekos\\Desktop\\projects\\Diaxeirisi\\assignment2\\tiger_roads.csv")
```

Στην οποία πρέπει να δοθεί το πλήρες path του csv αρχείου από τον προγραμματιστή στο δικό του μηχάνημα.