In [1]:
```python
from gerrychain import Graph
```

In [2]:
```python
# Read Alabama county graph from the json file "COUNTY_01.json"

filename = 'COUNTY_01.json'

# GerryChain has a built-in function for reading graphs of this type:
G = Graph.from_json( filename )
```

In [3]:
```python
# For each node, print the node #, county name, and its population
for node in G.nodes:
    name = G.nodes[node]["NAME10"]
    population = G.nodes[node]['TOTPOP']
    x_coordinate = G.nodes[node]['C_X']
    y_coordinate = G.nodes[node]['C_Y']
    print("Node",node,"is",name,"County, which has population",population,"and is cente
```

```
Node 0 is Barbour County, which has population 27457 and is centered at ( -85.3931969717
4619 , 31.869603217898423 )
Node 1 is Clay County, which has population 13932 and is centered at ( -85.8605470594419
4 , 33.269085886198276 )
Node 2 is Marengo County, which has population 21027 and is centered at ( -87.7895192852
3522 , 32.24761342168653 )
Node 3 is Houston County, which has population 101547 and is centered at ( -85.302519754
26631 , 31.15318422009409 )
Node 4 is Cherokee County, which has population 25989 and is centered at ( -85.603793825
52829 , 34.175955543616034 )
Node 5 is Baldwin County, which has population 182265 and is centered at ( -87.749844694
53685 , 30.6609738924624 )
Node 6 is Conecuh County, which has population 13228 and is centered at ( -86.9936813437
0104 , 31.42926727158543 )
Node 7 is Cleburne County, which has population 14972 and is centered at ( -85.518767919
86888 , 33.67455784656897 )
Node 8 is Jefferson County, which has population 658466 and is centered at ( -86.8964902
800165 , 33.55431475658097 )
Node 9 is Henry County, which has population 17302 and is centered at ( -85.241406300126
81 , 31.51469440693819 )
Node 10 is Crenshaw County, which has population 13906 and is centered at ( -86.31354765
901722 , 31.731486183501875 )
Node 11 is Madison County, which has population 334811 and is centered at ( -86.55020653
641006 , 34.76308970093465 )
Node 12 is Limestone County, which has population 82782 and is centered at ( -86.9813695
0223683 , 34.81007650885704 )
Node 13 is Dallas County, which has population 43820 and is centered at ( -87.1064787764
9095 , 32.325974538937146 )
Node 14 is Covington County, which has population 37765 and is centered at ( -86.4512478
8997703 , 31.248492845464902 )
Node 15 is Shelby County, which has population 195085 and is centered at ( -86.660648630
41424 , 33.2642778204166 )
Node 16 is Bibb County, which has population 22915 and is centered at ( -87.126439083023
29 , 32.998644313019 )
Node 17 is Butler County, which has population 20947 and is centered at ( -86.6802892156
2546 , 31.752433942560142 )
Node 18 is Macon County, which has population 21452 and is centered at ( -85.69266850175
498 , 32.38597359119281 )
Node 19 is Autauga County, which has population 54571 and is centered at ( -86.642757249
25504 , 32.53492069021946 )
Node 20 is Franklin County, which has population 31704 and is centered at ( -87.84381198
716348 , 34.441670632318164 )
```

```
Node 21 is Marshall County, which has population 93019 and is centered at ( -86.30663696
580778 , 34.36696193383658 )
Node 22 is Talladega County, which has population 82291 and is centered at ( -86.1659069
1520001 , 33.38005742632039 )
Node 23 is Walker County, which has population 67023 and is centered at ( -87.2973574325
2813 , 33.803334781108035 )
Node 24 is Lauderdale County, which has population 92709 and is centered at ( -87.654019
46382268 , 34.90114017896794 )
Node 25 is Morgan County, which has population 119490 and is centered at ( -86.852927813
64645 , 34.45347499534302 )
Node 26 is Pickens County, which has population 19746 and is centered at ( -88.088692609
80207 , 33.28087688365432 )
Node 27 is Etowah County, which has population 104430 and is centered at ( -86.034762204
99286 , 34.045254128332594 )
Node 28 is Tuscaloosa County, which has population 194656 and is centered at ( -87.52502
994601569 , 33.289549847357435 )
Node 29 is Dale County, which has population 50251 and is centered at ( -85.611031961504
93 , 31.431819470587527 )
Node 30 is Tallapoosa County, which has population 41616 and is centered at ( -85.797502
22941729 , 32.8624049444577 )
Node 31 is Geneva County, which has population 26790 and is centered at ( -85.8389848908
043 , 31.095016702449865 )
Node 32 is Mobile County, which has population 412992 and is centered at ( -88.197529576
3405 , 30.685146910580293 )
Node 33 is Wilcox County, which has population 11670 and is centered at ( -87.3081990893
2153 , 31.98924196018949 )
Node 34 is Sumter County, which has population 13763 and is centered at ( -88.1987913473
9342 , 32.591014077360576 )
Node 35 is Lawrence County, which has population 34339 and is centered at ( -87.31104472
929573 , 34.521650917020324 )
Node 36 is Calhoun County, which has population 118572 and is centered at ( -85.82603486
914782 , 33.77142782210847 )
Node 37 is Jackson County, which has population 53227 and is centered at ( -85.999300681
75878 , 34.77941470747724 )
Node 38 is Marion County, which has population 30776 and is centered at ( -87.8871384347
8739 , 34.13654972020097 )
Node 39 is DeKalb County, which has population 71109 and is centered at ( -85.8041414499
2365 , 34.45977281989643 )
Node 40 is Pike County, which has population 32899 and is centered at ( -85.940920740428
94 , 31.802715045380932 )
Node 41 is Perry County, which has population 10591 and is centered at ( -87.29440084137
56 , 32.63846568673937 )
Node 42 is Colbert County, which has population 54428 and is centered at ( -87.805296766
72091 , 34.70025404740263 )
Node 43 is Elmore County, which has population 79303 and is centered at ( -86.1491463397
6642 , 32.59664788700988 )
Node 44 is Washington County, which has population 17581 and is centered at ( -88.207876
81246034 , 31.407603796087137 )
Node 45 is Chambers County, which has population 34215 and is centered at ( -85.39204294
373201 , 32.9143722172032 )
Node 46 is Clarke County, which has population 25833 and is centered at ( -87.8308098505
0333 , 31.676659580123818 )
Node 47 is St. Clair County, which has population 83593 and is centered at ( -86.3146877
4720817 , 33.71570290572818 )
Node 48 is Chilton County, which has population 43643 and is centered at ( -86.718813730
83016 , 32.84785313410404 )
Node 49 is Bullock County, which has population 10914 and is centered at ( -85.715697201
64433 , 32.10055420532796 )
Node 50 is Greene County, which has population 9045 and is centered at ( -87.95223434205
52 , 32.853137408728344 )
Node 51 is Blount County, which has population 57322 and is centered at ( -86.5673709599
9279 , 33.98086739482323 )
Node 52 is Coosa County, which has population 11539 and is centered at ( -86.24765921602
953 , 32.936226896286094 )
Node 53 is Winston County, which has population 24484 and is centered at ( -87.373683521
54319 , 34.14922498766865 )
Node 54 is Russell County, which has population 52947 and is centered at ( -85.184962597
87717 , 32.288380056218536 )
Node 55 is Coffee County, which has population 49948 and is centered at ( -85.9882069708
7443 , 31.40262716953991 )
Node 56 is Monroe County, which has population 23068 and is centered at ( -87.3654308965
5846 , 31.570836608980876 )
Node 57 is Lamar County, which has population 14564 and is centered at ( -88.09689885856
152 , 33.77920579845164 )
Node 58 is Choctaw County, which has population 13859 and is centered at ( -88.263201266
21178 , 32.01961076277522 )
Node 59 is Cullman County, which has population 80406 and is centered at ( -86.867619397
44032 , 34.13194106743531 )
Node 60 is Montgomery County, which has population 229363 and is centered at ( -86.20761
429347573 , 32.22025793851669 )
Node 61 is Escambia County, which has population 38319 and is centered at ( -87.16161983
379405 , 31.126122544455463 )
Node 62 is Randolph County, which has population 22913 and is centered at ( -85.45907017
313048 , 33.29378740802176 )
Node 63 is Fayette County, which has population 17241 and is centered at ( -87.738861962
61972 , 33.72120614857144 )
Node 64 is Lee County, which has population 140247 and is centered at ( -85.355562714324
5 , 32.601136470581956 )
Node 65 is Lowndes County, which has population 11299 and is centered at ( -86.650107703
8513 , 32.15475001138599 )
Node 66 is Hale County, which has population 15760 and is centered at ( -87.629117736684
48 , 32.76266425862635 )
```

In [12]:

```
# pip install geopy

!pip install geopy
```

```
Collecting geopy
  Downloading geopy-2.1.0-py3-none-any.whl (112 kB)
Collecting geographiclib<2,>=1.49
  Downloading geographiclib-1.50-py3-none-any.whl (38 kB)
Installing collected packages: geographiclib, geopy
Successfully installed geographiclib-1.50 geopy-2.1.0
```

In [13]:

```
import geopy
```

In [14]:

```
# what is the "distance" between Barbour County (node 0), Shelby County (node 15), and
from geopy.distance import geodesic

# Store centroid location as ( long, lat )
Barbour = ( G.nodes[0]['C_Y'],  G.nodes[0]['C_X'] )
Shelby = ( G.nodes[15]['C_Y'], G.nodes[15]['C_X'] )
Walker = ( G.nodes[23]['C_Y'], G.nodes[23]['C_X'] )

# Print the distance in miles
print("Barbour -> Shelby:",geodesic(Barbour, Shelby).miles)
print("Shelby -> Walker:",geodesic(Shelby, Walker).miles)
print("Walker -> Barbour:",geodesic(Walker, Barbour).miles)
```

```
Barbour -> Shelby: 121.2644116391872
Shelby -> Walker: 52.255718525283626
Walker -> Barbour: 173.27867565607963
```

In [15]:

```
# create distance dictionary
dist = dict()
```

```python
for i in G.nodes:
    for j in G.nodes:
        loc_i = ( G.nodes[i]['C_Y'],  G.nodes[i]['C_X'] )
        loc_j = ( G.nodes[j]['C_Y'],  G.nodes[j]['C_X'] )
        dist[i,j] = geodesic(loc_i,loc_j).miles
```

In [16]:

```python
# check the dictionary by printing the Barbour County -> Shelby County distance
print("Barbour -> Shelby:",dist[0,15])
```

```
Barbour -> Shelby: 121.2644116391872
```

In [17]:

```python
# Let's impose a 1% population deviation (+/- 0.5%)
deviation = 0.01

import math
k = 7          # number of districts
total_population = sum(G.nodes[node]['TOTPOP'] for node in G.nodes)

L = math.ceil((1-deviation/2)*total_population/k)
U = math.floor((1+deviation/2)*total_population/k)
print("Using L =",L,"and U =",U,"and k =",k)
```

```
Using L = 679406 and U = 686233 and k = 7
```

In [18]:

```python
import gurobipy as gp
from gurobipy import GRB

# create model
m = gp.Model()

# create x[i,j] variable which equals one when county i
#    is assigned to (the district centered at) county j
x = m.addVars(G.nodes, G.nodes, vtype=GRB.BINARY)
```

```
Academic license - for non-commercial use only - expires 2021-06-24
Using license file C:\Users\Louisa\gurobi.lic
```

In [19]:

```python
# objective is to minimize the moment of inertia: d^2 * p * x
m.setObjective( gp.quicksum( dist[i,j]*dist[i,j]*G.nodes[i]['TOTPOP']*x[i,j]
                            for i in G.nodes for j in G.nodes), GRB.MINIMIZE )
```

In [20]:

```python
# add constraints saying that each county i is assigned to one district
m.addConstrs( gp.quicksum(x[i,j] for j in G.nodes) == 1 for i in G.nodes)

# add constraint saying there should be k district centers
m.addConstr( gp.quicksum( x[j,j] for j in G.nodes ) == k )

# add constraints that say: if j roots a district, then its population is between L and
m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] * x[i,j] for i in G.nodes) >= L * x[j,j
m.addConstrs( gp.quicksum( G.nodes[i]['TOTPOP'] * x[i,j] for i in G.nodes) <= U * x[j,j

# add coupling constraints saying that if i is assigned to j, then j is a center.
m.addConstrs( x[i,j] <= x[j,j] for i in G.nodes for j in G.nodes )

m.update()
```

In [21]:
```
# Add contiguity constraints

import networkx as nx
DG = nx.DiGraph(G)

# Add variable f[j,u,v] which equals the amount of flow (originally from j) that is sen
f = m.addVars( DG.nodes, DG.edges, vtype=GRB.CONTINUOUS)
M = DG.number_of_nodes()-1

# Add constraint saying that node j cannot receive flow of its own type
m.addConstrs( gp.quicksum( f[j,u,j] for u in DG.neighbors(j) ) == 0 for j in DG.nodes )

# Add constraints saying that node i can receive flow of type j only if i is assigned t
m.addConstrs( gp.quicksum( f[j,u,i] for u in DG.neighbors(i)) <= M * x[i,j] for i in DG

# If i is assigned to j, then i should consume one unit of j flow.
#    Otherwise, i should consume no units of j flow.
m.addConstrs( gp.quicksum( f[j,u,i] - f[j,i,u] for u in DG.neighbors(i)) == x[i,j] for

m.update()
```

In [22]:
```
# solve, making sure to set a 0.00% MIP gap tolerance(!)
m.Params.MIPGap = 0.0
m.optimize()
```

```
Changed value of parameter MIPGap to 0.0
   Prev: 0.0001  Min: 0.0  Max: inf  Default: 0.0001
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 13602 rows, 27403 columns and 99280 nonzeros
Model fingerprint: 0x8806bbad
Variable types: 22914 continuous, 4489 integer (4489 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+05]
  Objective range  [4e+06, 4e+10]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 7e+00]
Warning: Model contains large objective coefficients
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Presolve removed 1027 rows and 2760 columns
Presolve time: 1.12s
Presolved: 12575 rows, 24643 columns, 90388 nonzeros
Variable types: 20420 continuous, 4223 integer (4223 binary)

Root relaxation: objective 5.661390e+09, 1722 iterations, 0.37 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 5.6614e+09    0  175          - 5.6614e+09      -     -    2s
     0     0 5.7467e+09    0  241          - 5.7467e+09      -     -    3s
     0     0 5.7636e+09    0  250          - 5.7636e+09      -     -    4s
     0     0 5.7637e+09    0  250          - 5.7637e+09      -     -    4s
     0     0 5.7992e+09    0  241          - 5.7992e+09      -     -    5s
     0     0 5.8036e+09    0  254          - 5.8036e+09      -     -    5s
     0     0 5.8284e+09    0  258          - 5.8284e+09      -     -    5s
     0     0 5.8447e+09    0  278          - 5.8447e+09      -     -    6s
     0     0 5.8449e+09    0  279          - 5.8449e+09      -     -    6s
     0     0 5.8597e+09    0  282          - 5.8597e+09      -     -    6s
     0     0 5.8641e+09    0  280          - 5.8641e+09      -     -    7s
     0     0 5.8671e+09    0  250          - 5.8671e+09      -     -    7s
     0     0 5.8673e+09    0  256          - 5.8673e+09      -     -    7s
     0     0 5.8686e+09    0  264          - 5.8686e+09      -     -    7s
     0     0 5.8688e+09    0  260          - 5.8688e+09      -     -    7s
     0     0 5.8689e+09    0  262          - 5.8689e+09      -     -    7s
     0     0 5.8689e+09    0  264          - 5.8689e+09      -     -    7s
     0     0 5.8898e+09    0  306          - 5.8898e+09      -     -    8s
     0     0 5.8916e+09    0  321          - 5.8916e+09      -     -    8s
     0     0 5.8921e+09    0  321          - 5.8921e+09      -     -    8s
     0     0 5.8921e+09    0  321          - 5.8921e+09      -     -    8s
     0     0 5.8921e+09    0  322          - 5.8921e+09      -     -    8s
     0     0 5.9102e+09    0  294          - 5.9102e+09      -     -    8s
     0     0 5.9117e+09    0  308          - 5.9117e+09      -     -    9s
     0     0 5.9119e+09    0  310          - 5.9119e+09      -     -    9s
     0     0 5.9158e+09    0  284          - 5.9158e+09      -     -    9s
     0     0 5.9158e+09    0  285          - 5.9158e+09      -     -    9s
     0     0 5.9191e+09    0  284          - 5.9191e+09      -     -   10s
     0     0 5.9191e+09    0  288          - 5.9191e+09      -     -   10s
     0     0 5.9200e+09    0  271          - 5.9200e+09      -     -   10s
     0     0 5.9207e+09    0  296          - 5.9207e+09      -     -   11s
     0     0 5.9213e+09    0  287          - 5.9213e+09      -     -   11s
     0     0 5.9215e+09    0  284          - 5.9215e+09      -     -   11s
     0     0 5.9215e+09    0  285          - 5.9215e+09      -     -   11s
     0     0 5.9238e+09    0  306          - 5.9238e+09      -     -   11s
     0     0 5.9241e+09    0  314          - 5.9241e+09      -     -   12s
     0     0 5.9241e+09    0  314          - 5.9241e+09      -     -   12s
     0     0 5.9248e+09    0  316          - 5.9248e+09      -     -   12s
     0     0 5.9248e+09    0  316          - 5.9248e+09      -     -   12s
     0     0 5.9249e+09    0  326          - 5.9249e+09      -     -   13s
     0     0 5.9251e+09    0  332          - 5.9251e+09      -     -   13s
     0     0 5.9251e+09    0  317          - 5.9251e+09      -     -   13s
     0     0 5.9255e+09    0  333          - 5.9255e+09      -     -   14s
     0     0 5.9255e+09    0  331          - 5.9255e+09      -     -   14s
     0     0 5.9262e+09    0  310          - 5.9262e+09      -     -   14s
     0     0 5.9263e+09    0  316          - 5.9263e+09      -     -   15s
     0     0 5.9263e+09    0  308          - 5.9263e+09      -     -   15s
     0     0 5.9305e+09    0  333          - 5.9305e+09      -     -   15s
     0     0 5.9306e+09    0  331          - 5.9306e+09      -     -   15s
     0     0 5.9331e+09    0  317          - 5.9331e+09      -     -   16s
     0     0 5.9336e+09    0  317          - 5.9336e+09      -     -   16s
     0     2 5.9336e+09    0  317          - 5.9336e+09      -     -   20s
   108   117 6.6698e+09   21   22          - 5.9663e+09      -   158   25s
   620   466 7.2493e+09   46   12          - 5.9818e+09      -  88.6   30s
  1044   729 6.5538e+09   18  317          - 6.0132e+09      -  85.1   37s
  1046   730 8.2062e+09   35  175          - 6.0132e+09      -  85.0   40s
  1056   737 6.5616e+09   23  295          - 6.0132e+09      -  84.2   45s
  1066   744 8.4801e+09   35  316          - 6.0132e+09      -  83.4   50s
  1069   746 6.9744e+09   18  351          - 6.0132e+09      -  83.1   59s
  1070   749 6.0132e+09   15  281          - 6.0132e+09      -   5.5   66s
  1072   753 6.0304e+09   16  267          - 6.0132e+09      -   6.3   70s
  1131   803 6.5993e+09   24   19          - 6.0303e+09      -  18.1   75s
  1415   902 6.0668e+09   18  209          - 6.0313e+09      -  27.4   80s
  1490   954 6.1919e+09   27   19          - 6.0313e+09      -  36.0   85s
  1892  1163 6.1273e+09   22  133          - 6.0669e+09      -  43.3   90s
  2203  1253 6.5566e+09   32   20          - 6.0669e+09      -  48.7   95s
  2613  1461 6.2920e+09   24   61          - 6.0766e+09      -  55.5  100s
  3244  1715 7.6653e+09   55    6          - 6.0965e+09      -  59.5  105s
  3670  2079 6.3004e+09   22  111          - 6.1107e+09      -  64.1  110s
* 3804   672              39    6.613137e+09 6.1170e+09  7.50%  65.4  111s
H 3957   297                    6.409756e+09 6.1170e+09  4.57%  65.4  112s
  4140   300 6.3664e+09   26  131 6.4098e+09 6.1340e+09  4.30%  66.5  115s
  4564   313     cutoff   20       6.4098e+09 6.1958e+09  3.34%  68.8  120s
  5005   261 6.3430e+09   26  131 6.4098e+09 6.2382e+09  2.68%  68.7  125s
  5512     0 6.3370e+09   26  215 6.4098e+09 6.3107e+09  1.55%  68.7  130s

Cutting planes:
  Cover: 94
  MIR: 18
  StrongCG: 8
  Flow cover: 123
  GUB cover: 10
  Zero half: 6
  Mod-K: 2
  Network: 8
  RLT: 1

Explored 5718 nodes (485813 simplex iterations) in 130.86 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 6.40976e+09 6.61314e+09

Optimal solution found (tolerance 0.00e+00)
Best objective 6.409755608475e+09, best bound 6.409755608475e+09, gap 0.0000%
```

In [23]:

```python
print("The moment of inertia objective is",m.objval)

# retrieve the districts and their populations
centers = [j for j in G.nodes if x[j,j].x > 0.5 ]
districts = [ [i for i in G.nodes if x[i,j].x > 0.5] for j in centers]
district_counties = [ [ G.nodes[i]["NAME10"] for i in districts[j] ] for j in range(k)]
district_populations = [ sum(G.nodes[i]["TOTPOP"] for i in districts[j]) for j in range

# print district info
for j in range(k):
    print("District",j,"has population",district_populations[j],"and contains counties"
```

```
The moment of inertia objective is 6409755608.475345
District 0 has population 681381 and contains counties ['Jefferson', 'Bibb']
District 1 has population 680437 and contains counties ['Madison', 'Limestone', 'Frankli
n', 'Lauderdale', 'Jackson', 'Marion', 'Colbert']
District 2 has population 684669 and contains counties ['Cherokee', 'Cleburne', 'Marshal
l', 'Morgan', 'Etowah', 'Lawrence', 'DeKalb', 'St. Clair', 'Blount', 'Cullman']
District 3 has population 683280 and contains counties ['Dallas', 'Shelby', 'Walker', 'P
ickens', 'Tuscaloosa', 'Sumter', 'Perry', 'Chilton', 'Greene', 'Winston', 'Lamar', 'Choc
taw', 'Fayette', 'Hale']
District 4 has population 684512 and contains counties ['Clay', 'Macon', 'Autauga', 'Tal
ladega', 'Tallapoosa', 'Calhoun', 'Elmore', 'Chambers', 'Bullock', 'Coosa', 'Russell',
'Randolph', 'Lee']
District 5 has population 682766 and contains counties ['Marengo', 'Baldwin', 'Mobile',
'Washington', 'Clarke', 'Monroe']
District 6 has population 682691 and contains counties ['Barbour', 'Houston', 'Conecuh',
'Henry', 'Crenshaw', 'Covington', 'Butler', 'Dale', 'Geneva', 'Wilcox', 'Pike', 'Coffe
e', 'Montgomery', 'Escambia', 'Lowndes']
```

In [24]:

```python
# Let's draw it on a map
import geopandas as gpd
```

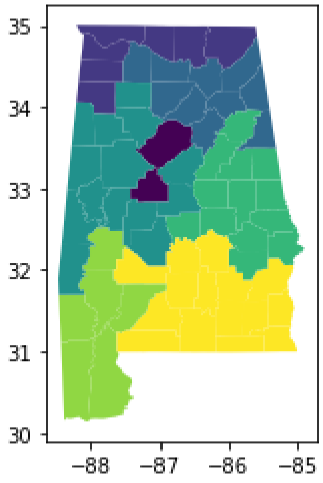
In [25]:

```python
# Read Alabama county shapefile from "OK_county.shp"

filename = 'AL_counties.shp'

# Read geopandas dataframe from file
df = gpd.read_file( filename )
```

In [26]:

```python
# Which district is each county assigned to?
```

```
assignment = [ -1 for u in G.nodes ]

# for each district j
for j in range(len(districts)):

    # for each node i in this district
    for i in districts[j]:

        # What is its GEOID?
        geoID = G.nodes[i]["GEOID10"]

        # Need to find this GEOID in the dataframe
        for u in G.nodes:
            if geoID == df['GEOID10'][u]: # Found it
                assignment[u] = j # Node u from the dataframe should be assigned to dis

# Now add the assignments to a column of the dataframe and map it
df['assignment'] = assignment
my_fig = df.plot(column='assignment').get_figure()
```



```
In [ ]:
```