CS4423 - Networks

Prof. Götz Pfeiffer
School of Mathematics, Statistics and Applied Mathematics
NUI Galway

Assignment 1

Provide answers to the problems in the boxes provided. Marks will be awarded for participation and engagement.

When finished, print this notebook into a **pdf** file and submit this to **blackboard**.

Deadline is next Monday at 5pm.

Setup

This is a `jupyter` notebook. Find an environment that allows you to work with it. You can either install `jupyter` as a python packag on your own laptop or PC. Or you can use a suitable website on the internet, such as [nbviewer (https://nbviewer.jupyter.org/github/cs4423)](https://nbviewer.jupyter.org/github/cs4423) and `binder` .

The following packages need to be loaded. In order to execute the code in a box, use the mouse or arrow keys to highlight the box and then press SHIFT-RETURN.

Should it ever happen that the notebook becomes unusable, start again with a fresh copy.

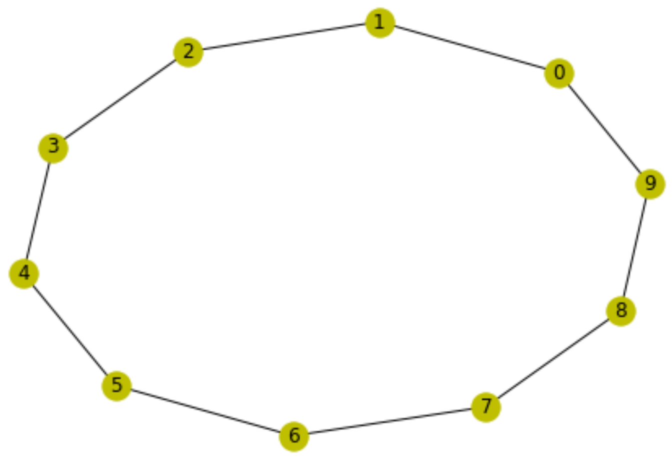
In [1]:
```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
```

1. Warmup.

The purpose of this task is to get you used to working with the `networkx` package in the `jupyter` notebook environment.

1. Define a new (simple) graph `G` on the vertex set $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$with edges $0 - 1$, $1 - 2$, $2 - 3$, $3 - 4$, $4 - 5$, $5 - 6$, $6 - 7$, $7 - 8$, $8 - 9$, and $9 - 0$. Draw the graph. Hence or otherwise determine its **order** (the number of nodes) and its **size** (the number of links).

In [2]:
```
G = nx.Graph(["01","12","23","34","45","56","67","78","89","90"])
nx.draw(G, with_labels=True, node_color='y')
```



1. Find the **adjacency matrix** `A` of the graph `G`. Then compute its square, A^2, and draw the graph `G2` that has A^2 as its adjacency matrix. What are the connected components of `G2` ?

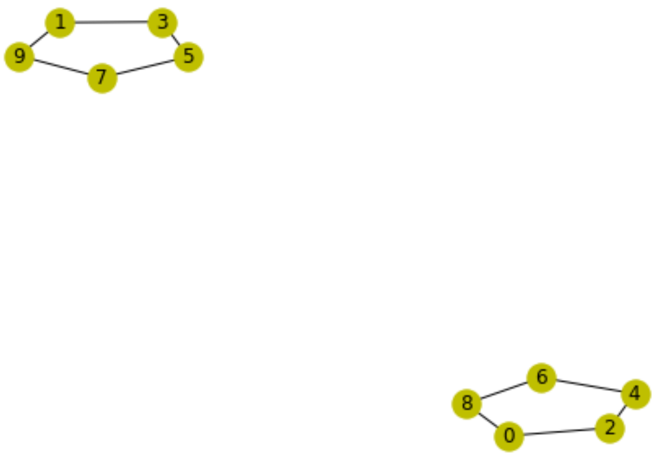
In [3]:
```
A = nx.adjacency_matrix(G)
print(A.todense())
```

```
[[0 1 0 0 0 0 0 0 0 1]
 [1 0 1 0 0 0 0 0 0 0]
 [0 1 0 1 0 0 0 0 0 0]
 [0 0 1 0 1 0 0 0 0 0]
 [0 0 0 1 0 1 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0]
 [0 0 0 0 0 1 0 1 0 0]
 [0 0 0 0 0 0 1 0 1 0]
 [0 0 0 0 0 0 0 1 0 1]
 [1 0 0 0 0 0 0 0 1 0]]
```

In [4]:
```
AA = A * A
print(AA.todense())
```

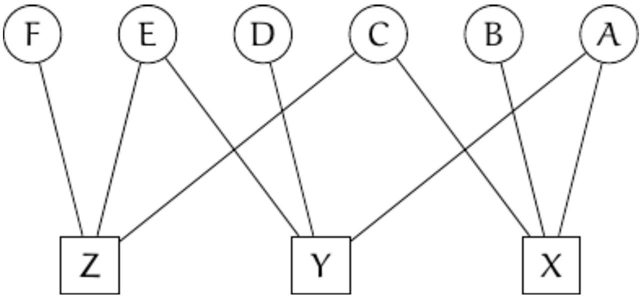
```
[[2 0 1 0 0 0 0 0 1 0]
 [0 2 0 1 0 0 0 0 0 1]
 [1 0 2 0 1 0 0 0 0 0]
 [0 1 0 2 0 1 0 0 0 0]
 [0 0 1 0 2 0 1 0 0 0]
 [0 0 0 1 0 2 0 1 0 0]
 [0 0 0 0 1 0 2 0 1 0]
 [0 0 0 0 0 1 0 2 0 1]
 [1 0 0 0 0 0 1 0 2 0]
 [0 1 0 0 0 0 0 1 0 2]]
```

In [5]:
```
GG = nx.from_numpy_matrix(AA.todense())
nx.draw(GG, with_labels=True, node_color='y')
```



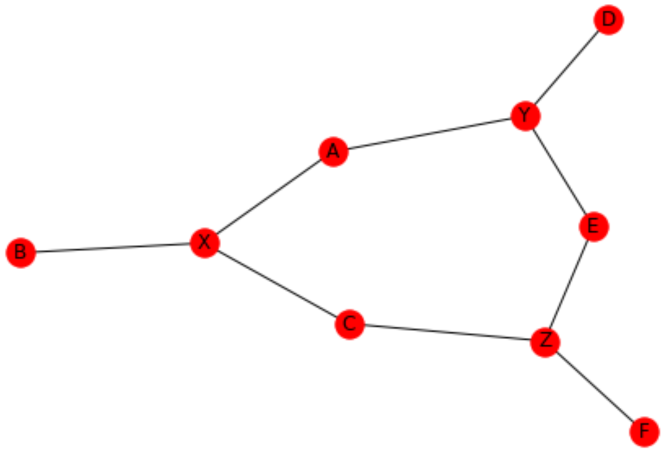
2. Projections

For the affilliation network below, with six people labelled A to F, and three foci labelled X, Y and Z, draw the projection on (just) the people, in which two people are joined by an edge if they have a common focus.

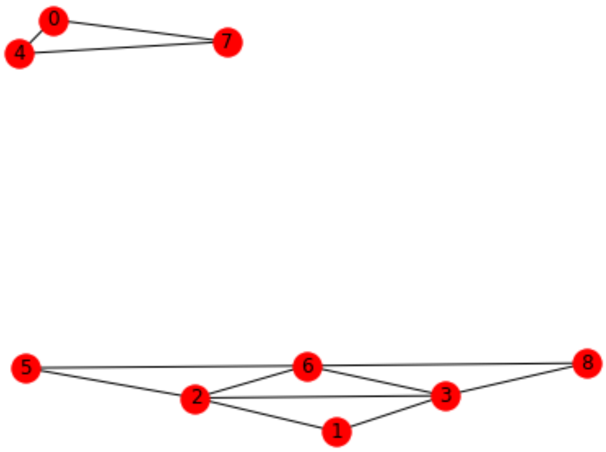


(Of course, one can do this easily by hand. It would be nice to get `networkx` to do it for you.)

In [6]:
```
G = nx.Graph(["ZF","ZE","ZC","YE","YD","YA","XC","XB","XA"])
nx.draw(G, with_labels=True)
```



In [7]:
```
A = nx.adjacency_matrix(G)
AA = A * A
GG = nx.from_numpy_matrix(AA.todense())
nx.draw(GG, with_labels=True)
```



3. A Collaborations Network

The **social graph** of a node x in a (social) network is the **induced subgraph** on the set of friends of x (that is the graph which has (only) the friends of x as its vertices, and between them all the edges from the original network). The **clustering coefficient** of x is the density $m/\binom{n}{2}$ of the social graph of x, the proportion its number of edges, m, and its potential number of edges, $\binom{n}{2} = \frac{1}{2}n(n-1)$, where n is its number of vertices.

[MathSciNet (http://www.ams.org/mathscinet)](http://www.ams.org/mathscinet) describes the social network of mathematical researchers defined by collaboration.

- **Pick** a (local) mathematician with at least 10 friends (i.e., co-authors), determine their social graph and hence compute their clustering coefficient.

In [8]:
```
node = 328305
```

In [9]:
```
from IPython.display import Markdown as md
mathscinet = "https://mathscinet.ams.org/mathscinet/search/author.html"
md(f'Picked node [{node}]({mathscinet}?mrauthid={node}) ...')
```

Out[9]: Picked node [328305 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=328305)](https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=328305) ...

... as node x from MathSciNet. Now

- find all its friends y;
- for each friend y find their friends z (one of which must be x) and store them in a file `social.adj` .

In [10]:
```
G = nx.read_adjlist("social.adj", nodetype=int)
nx.set_node_attributes(G, 'y', 'color')
```

In [11]:
```
n, m = G.order(), G.size()
n, m
```

Out[11]:
```
(548, 689)
```

- now find the neighbors of the picked node:

In [12]:
```
social = list(G.neighbors(node))
print(social)
```

```
[28105, 1001341, 272720, 631109, 350539, 250433, 272405, 86475, 618475, 78243
6, 268828, 362381, 225462, 664485, 1252628, 985127, 361517, 141715, 1032219,
149020, 337959, 329365, 653392, 619047]
```

In [13]:
```
text = ' '.join([f'[{node}]({mathscinet}?mrauthid={node})' for node in socia
l])
md(text)
```

Out[13]: 28105 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=28105) 1001341 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=1001341) 272720 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=272720) 631109 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=631109) 350539 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=350539) 250433 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=250433) 272405 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=272405) 86475 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=86475) 618475 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=618475) 782436 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=782436) 268828 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=268828) 362381 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=362381) 225462 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=225462) 664485 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=664485) 1252628 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=1252628) 985127 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=985127) 361517 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=361517) 141715 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=141715) 1032219 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=1032219) 149020 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=149020) 337959 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=337959) 329365 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=329365) 653392 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=653392) 619047 (https://mathscinet.ams.org/mathscinet/search/author.html?mrauthid=619047)

- construct the subgraph on that list of nodes (without the picked one):

In [14]:
```
GG = G.subgraph(list(social))
```

In [15]:
```
GG.order(), GG.size()
```

Out[15]:
```
(24, 25)
```

In [16]:
```
clustering = 2*25/24/23
clustering
```

Out[16]:
```
0.09057971014492754
```

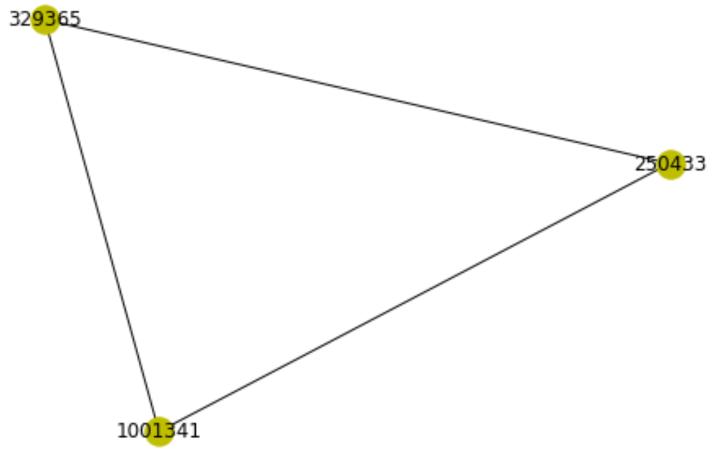
In [17]:
```
nx.clustering(G, node)
```

Out[17]:
```
0.09057971014492754
```

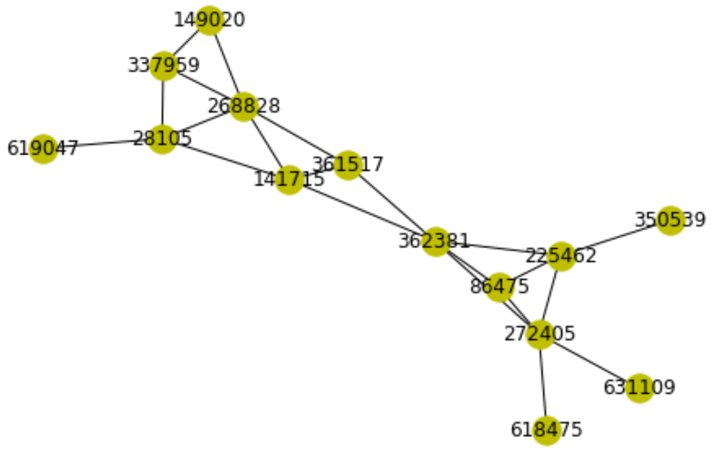
In [18]:
```
components = [GG.subgraph(c) for c in nx.connected_components(GG)]
[c.order() for c in components]
```

Out[18]:
```
[14, 1, 3, 1, 1, 1, 1, 2]
```

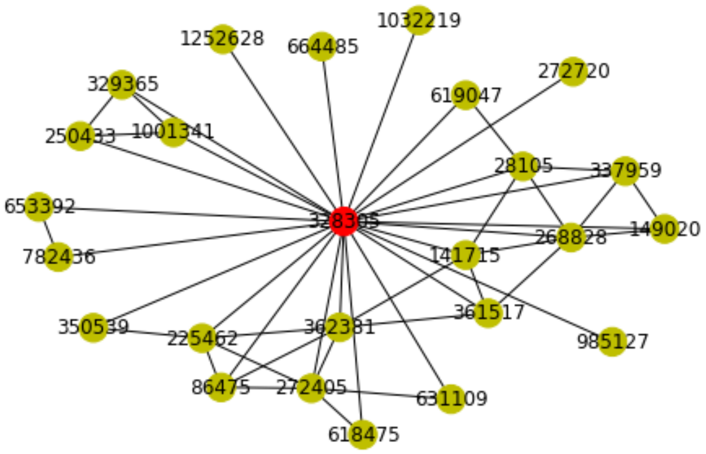
In [19]:
```
nx.draw(components[2], with_labels=True, node_color='y')
```



In [20]:
```
nx.draw(components[0], with_labels=True, node_color='y', figsize=(20, 10))
```



In [21]:
```
GP = G.subgraph([node] + social)
nx.set_node_attributes(GP, 'y', 'color')
GP.nodes[node]['color'] = 'r'
colors = nx.get_node_attributes(GP, 'color').values()
nx.draw(GP, with_labels=True, node_color=colors, figsize=(20, 10))
```



4. The Counties of Ireland.

Define a graph `I` on the 32 counties of Ireland by joining two counties whenever they have a common border. (A list of county names, suitable for cut-and-paste, can be found on the [internet (http://www.waterfordwebdesign.ie/alphabetical-list-32-counties-ireland/)](http://www.waterfordwebdesign.ie/alphabetical-list-32-counties-ireland/))

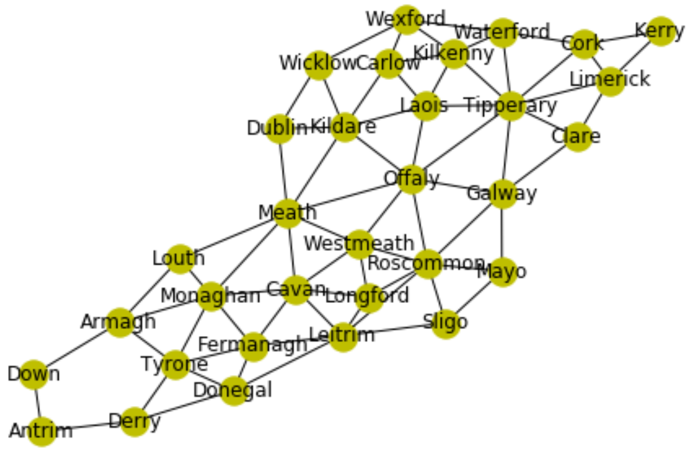
What is the order and the size of the resulting graph?

In terms of centrality measures, what are the 3 most central counties, for

1. degree centrality?
2. eigenvector centrality?
3. closeness centrality?
4. betweenness centrality?

In [22]:
```
I = nx.read_adjlist("ireland.adj")
```

In [23]:
```
nx.draw(I, with_labels=True, node_color='y')
```



In [24]:
```
data = dict(I.degree())
nx.set_node_attributes(I, data, 'degree')
max(data, key=data.get)
```

Out[24]: 'Tipperary'

In [25]:
```
data = nx.degree_centrality(I)
nx.set_node_attributes(I, data, '$C_i^D$')
max(data, key=data.get)
```

Out[25]: 'Tipperary'

In [26]:
```
data = nx.eigenvector_centrality(I)
nx.set_node_attributes(I, data, '$C_i^E$')
max(data, key=data.get)
```

Out[26]: 'Offaly'

In [27]:
```
data = nx.closeness_centrality(I)
nx.set_node_attributes(I, data, '$C_i^C$')
max(data, key=data.get)
```

Out[27]: 'Offaly'

In [28]:
```
data = nx.betweenness_centrality(I)
nx.set_node_attributes(I, data, '$C_i^B$')
max(data, key=data.get)
```

Out[28]: 'Meath'

In [29]:
```
pd.DataFrame.from_dict(dict(I.nodes(data=True)), orient='index').sort_values
('degree', ascending=False)
```

Out[29]:

	degree	C_i^D	C_i^E	C_i^C	C_i^B
Tipperary	8	0.258065	0.261930	0.387500	0.244031
Offaly	7	0.225806	0.340912	0.455882	0.281752
Meath	7	0.225806	0.294819	0.442857	0.296100
Roscommon	7	0.225806	0.285942	0.402597	0.166582
Leitrim	6	0.193548	0.222078	0.369048	0.130541
Kildare	6	0.193548	0.229157	0.397436	0.120517
Monaghan	6	0.193548	0.202733	0.382716	0.156100
Cavan	6	0.193548	0.262187	0.382716	0.063340
Tyrone	5	0.161290	0.119652	0.316327	0.057447
Fermanagh	5	0.161290	0.176225	0.333333	0.029978
Westmeath	5	0.161290	0.266356	0.397436	0.037166
Kilkenny	5	0.161290	0.159565	0.310000	0.020939
Laois	5	0.161290	0.216920	0.373494	0.043648
Galway	5	0.161290	0.215534	0.392405	0.102041
Donegal	4	0.129032	0.108654	0.306931	0.062597
Waterford	4	0.129032	0.118807	0.306931	0.023303
Cork	4	0.129032	0.099561	0.298077	0.034292
Wexford	4	0.129032	0.096008	0.279279	0.019304
Armagh	4	0.129032	0.087466	0.300971	0.081146
Limerick	4	0.129032	0.098276	0.292453	0.031371
Longford	4	0.129032	0.199572	0.348315	0.005030
Carlow	4	0.129032	0.135092	0.322917	0.020759
Sligo	3	0.096774	0.120881	0.322917	0.008602
Wicklow	3	0.096774	0.085187	0.310000	0.025392
Mayo	3	0.096774	0.119824	0.319588	0.006935
Louth	3	0.096774	0.112635	0.348315	0.045706
Dublin	3	0.096774	0.117284	0.340659	0.022107
Derry	3	0.096774	0.046390	0.264957	0.042151
Clare	3	0.096774	0.110850	0.316327	0.011220
Kerry	2	0.064516	0.038091	0.234848	0.000000
Down	2	0.064516	0.019274	0.238462	0.022584
Antrim	2	0.064516	0.012642	0.216783	0.004520

In [ ]: