Collaboration Networks in Physics

In this exercise we will explore a co-author network constructed from research papers on High Energy Physics Theory in the pre-print server Arxiv (https://arxiv.org/). The nodes in this network are authors of papers. Two authors that have published at least one paper together have an edge between them. This is an undirected network.

Initializing NetworkX

We will now use the NetworkX package to analyse this network. As the first step we import NetworkX in to the work environment

In [1]:
```
import networkx as nx
```

Constructing the network

The data for the network is stored in the file **ca-HepTh.txt** in the work folder. It is in **edgelist** format, which means that each line contains two numbers separated by a tab. Each of these numbers corresponds to a author. Thus each line corresponds to an edge between two authors and the file contains the list of all edges in the network.

Now, we will load this file in to a NetworkX network object.

In [4]:
```
#arg1 - path to the file
#arg2 - create a undirected network
#arg3 - delimiter used to separate the author identifiers when defining an edge
g = nx.read_edgelist("ca-HepTh.txt", create_using=nx.Graph(), delimiter='\t')
```

Size of the network

Let's try find out the number of nodes (authors) and edges(collaborations) in this network.

Number of Nodes

In [5]:
```
g.number_of_nodes()
```

Out[5]:
```
9877
```

Number of Edges

In [6]:
```
g.number_of_edges()
```

Out[6]:
```
25998
```

Diameter of the network

Now, let's try to get some understanding about the overall connectivitiy of this network. We'll try to find the diameter of the network, which the highest distance between any pair of nodes in the network.

In [7]:
```
nx.diameter(g)
```

```
---------------------------------------------------------------------------
NetworkXError                             Traceback (most recent call last)
<ipython-input-7-2b8a3cef05a2> in <module>()
----> 1 nx.diameter(g)

~/anaconda/envs/py36/lib/python3.6/site-packages/networkx/algorithms/distance_measures.py in diameter(G, e)
     94     """
     95     if e is None:
---> 96         e=eccentricity(G)
     97     return max(e.values())
     98

~/anaconda/envs/py36/lib/python3.6/site-packages/networkx/algorithms/distance_measures.py in
eccentricity(G, v, sp)
     61         if L != order:
     62             msg = "Graph not connected: infinite path length"
---> 63             raise networkx.NetworkXError(msg)
     64
     65         e[n]=max(length.values())

NetworkXError: Graph not connected: infinite path length
```

What does this mean?

We got an error because **the network is not connected**. Generally, this means that there are at least 2 components in the network. In such a case diamter for the whole network is infinite because there's no path from nodes of one component to nodes of another.

We can check if the network is connected directly with the following command.

In [8]:
```
nx.is_connected(g)
```

Out[8]:
```
False
```

What do you think about the result for the diameter? Is it meaningful?

In reality, it depends on the network. Imagine a situation where there are only components in the network; a isolated node and the rest of the nodes in a single component. Diameter would still be infinity.

Network Components

Let's now find out some information about the components present in this network. Each component is a subset of nodes where there's a path between each pair of nodes in the subset and no links from any of its nodes to remaining nodes in the network

Number of Components

In [9]:
```
nx.number_connected_components(g)
```

Out[9]:
```
429
```

Component Membership

We can use NetworkX to find which nodes belong to each of these components.

In [20]:
```
#Find the set of components in g.
#Returns a generator object you can iterate (but not a list)
components = nx.connected_components(g)
#Iterate the generator and load the components in to a list.
#Each component is a set of nodes
components = [component for component in components]
```

Let's take a look at one of the components

In [21]:
```
components[10]
```

Out[21]:
```
{'13245', '20801'}
```

Is there a Giant Component?

Now that we know what the components are, let's try to figure out if there's a component that represents a significant fraction of the nodes in the network. To do this we will sort the components in the descending order of their sizes.

In [23]:
```
#parameters of sorted
#  arg1 - the list of components
#  arg2 - the property of each list item by which to sort - len gives the size of the s
et
#  arg3 - reverse=True to sort in the descending order
components = sorted(components, key=len, reverse=True)
```

The largest component is in the first position (index 0)

In [ ]:
```
components[0]
```

Let's compare the size of this component with the size of the network. We'll calculate the size of the component as a percentage of the number of nodes in the network.

In [28]:
```
round(100*len(components[0])/g.number_of_nodes(),2)
```

Out[28]:
```
87.46
```

Let's compare this with the 2nd largest component.

In [29]:
```
round(100*len(components[1])/g.number_of_nodes(),2)
```

Out[29]:
```
0.21
```

This shows us that majority of the authors in this domain belonged to a giant component, while there rest of the authors belonged to a relatively large number of comparatively very small groups.

Properties of the Giant Component

Now we will explore the properties of the giant component. To do this we will extract the giant component as sub-network from our network

In [30]:
```
#the subgraph method takes a list/set of nodes
#and returns a new graph from the original graph
#, that contains only nodes in that list
giant_subgraph = g.subgraph(components[0])
```

Diameter

Let's find the diameter of the giant component

In [31]:
```
nx.diameter(giant_subgraph)
```

Out[31]: 18

Can we say this network is a small world network?

We can't say this just the diameter of the network. The diameter is the worst-case distance between a pair of nodes. We can look at the mean distance in the network to get a better idea.

Average Distance between node pairs in the giant component

In [34]:
```
round(nx.average_shortest_path_length(giant_subgraph),2)
```

Out[34]: 5.95

Exercise

Now you can try out these steps with a new network. We will use the co-author network in General Relativity (introduced by Einstein) from Arxiv. Try to answer each of the questions in order by using the code cells immediately below the question. We've completed the first step for you.

Construct the network

In [35]:
```
g2 = nx.read_edgelist("ca-GrQc.txt", create_using=nx.Graph(), delimiter='\t')
```

How many nodes does the network have?

In [ ]:
```

```

How many edges does the network have?

In [ ]:
```

```

What is the diameter of the network?

In [ ]:
```

```

Verify whether the network is connected or not

In [ ]:
```

```

How many components does the network have?

In [ ]:

What fraction of the network is in the largest component?

In [ ]:

Compare the size of the largest component with that second largest component

In [ ]:

Would you say the network has a giant component?

In [ ]:

What is the diameter of the giant component?

In [ ]:

What is the mean distance between node pairs in the giant component?

In [ ]: