Assignment

For this assignment you will calculate and plot the distribution of the path lengths of a graph. First we will generate a random graph which we will use for the assignment:

In [3]:

```
import networkx as nx

random_graph = nx.erdos_renyi_graph(1200, 0.008)
print(nx.info(random_graph))
```

```
Name: gnp_random_graph(1200,0.008)
Type: Graph
Number of nodes: 1200
Number of edges: 5695
Average degree:   9.4917
```

1. Finding path lengths

Networkx provides a shortest path length function (https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.shortest_paths.generic.shortest_path.html) which you can use to get the shortest path between nodes in a graph. Try out some of the examples in the documentation using the random graph we made above until you get familiar with how it works and what type of output it gives.

Next write a function that will return a list of the shortest path **lengths** between all pairs of nodes (make sure each pair is only counted once). This function should take a single argument, `G`, a networkx graph.

In [1]:

```
def all_shortest_path_lengths(G):
    shortest = []
    node_visited = []
    for node in G:
        try:
            Current = [node]
            node_visited.append(node)
            level = {}
            i = 1
            while Current:
                Next = []
                for c in Current:
                    for neb in G.neighbors_iter(c):
                        if neb not in level and neb not in node_visited:
                            level[neb] = i
                            Next.append(neb)
                Current = Next
                i += 1
            for key in level:
                shortest.append(level[key])
        except:
            pass
    return (shortest)
```

Apply your function to the `random_graph` and assign the list of shortest path lengths to a variable:

In [4]:

```
# your code here
shortest = all_shortest_path_lengths(random_graph)
```

2. Visualizing the results

Now that you have a list of the shortest paths for the graph, make a histogram for it. This can be done with matplotlibs histogram function (http://matplotlib.org/api/pyplot_api.html?highlight=hist#matplotlib.pyplot.hist)

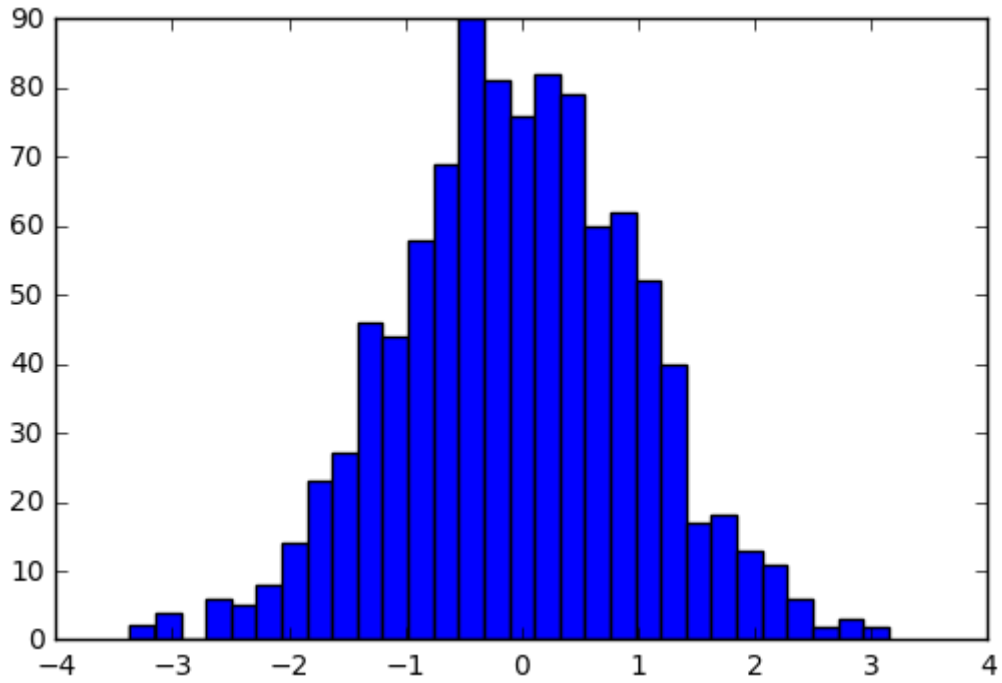
In [5]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

This function is fairly useful and lets us quickly visualize a distribution. For instance, if we create a small set of normally distributed random numbers we could use the histogram function to plot the bell curve:

In [6]:

```
import numpy as np
# Create dummy data
rvs = np.random.normal(size=1000)
# plot histogram
pdf, bins, patch = plt.hist(rvs, bins=30)
```

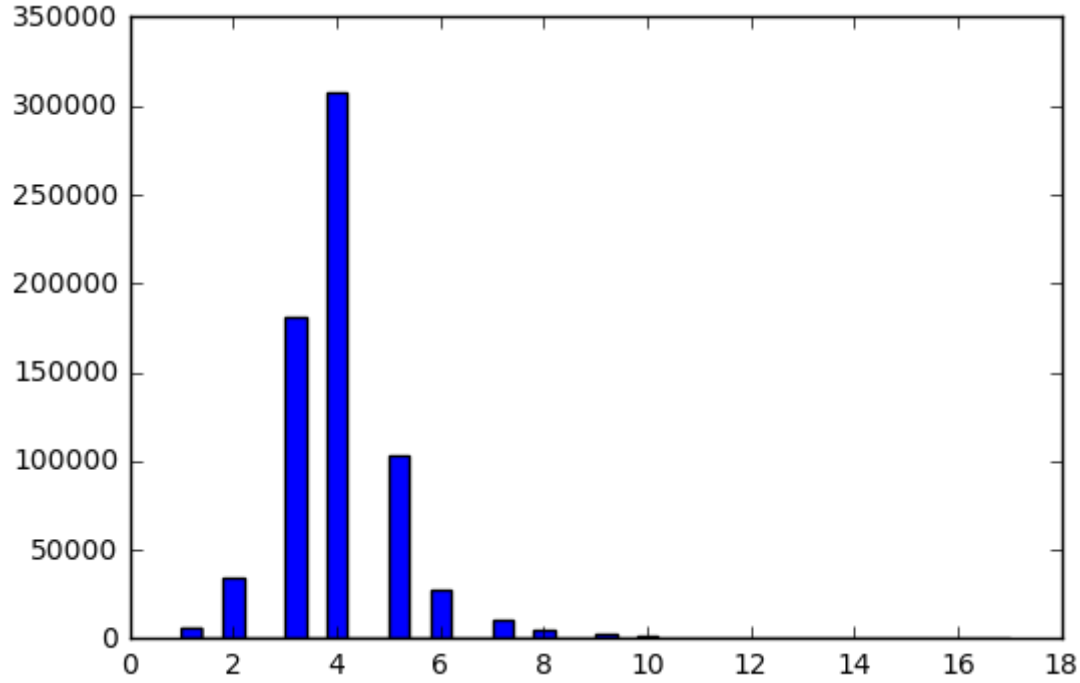


You may need to adjust the number of bins so that it is appropriate for the amount of data points you have.

Use the histogram function to make a plot of the results from your shortest paths length function:

In [7]:

```
# code here
pdf, bins, patch = plt.hist(shortest, bins=40)
```



Name your notebook: shortest_lastname_firstname.ipynb and submit to Canvas