Lec25 : Programming - Emergence of Connectedness

In [12]:

```
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy as np
```

In [2]:

```
#Add n nodes in the graph and return it
def add_nodes(n):
    G = nx.Graph()
    G.add_nodes_from(range(n))
    return G
```

In [3]:

```
add_nodes(10)
```

Out[3]:

```
<networkx.classes.graph.Graph at 0x7f328464f210>
```

In [4]:

```
#Experiment1
G = add_nodes(10)
print G.number_of_nodes()
print nx.is_connected(G)
```

```
10
False
```

In [5]:

```
#Add 1 random edge
def add_random_edge(G):
    v1 = rd.choice(G.nodes())
    v2 = rd.choice(G.nodes())
    if v1 != v2:
        G.add_edge(v1, v2)
    return G
```

In [6]:

```
G = add_random_edge(G)
print G.number_of_edges()
print G.edges()
```

```
1
[(8, 9)]
```

In [7]:

```
#Keeps adding random edges in G till it becomes connected
def add_till_connectivity(G):
    while(not nx.is_connected(G)):
        G = add_random_edge(G)
    return G
```

In [8]:

```
#Makes the graph connected in nlogn edges
print nx.info(G)
G = add_till_connectivity(G)
print nx.info(G)
```

```
Name:
Type: Graph
Number of nodes: 10
Number of edges: 1
Average degree:   0.2000
Name:
Type: Graph
Number of nodes: 10
Number of edges: 13
Average degree:   2.6000
```

In [9]:

```
#Creates an instance of the entire process.
#It takes input as the number of nodes
#It returns the number of nodes to make the Graph connected
def create_instance(n):
    G = add_nodes(n)
    G = add_till_connectivity(G)
    return G.number_of_edges()
```

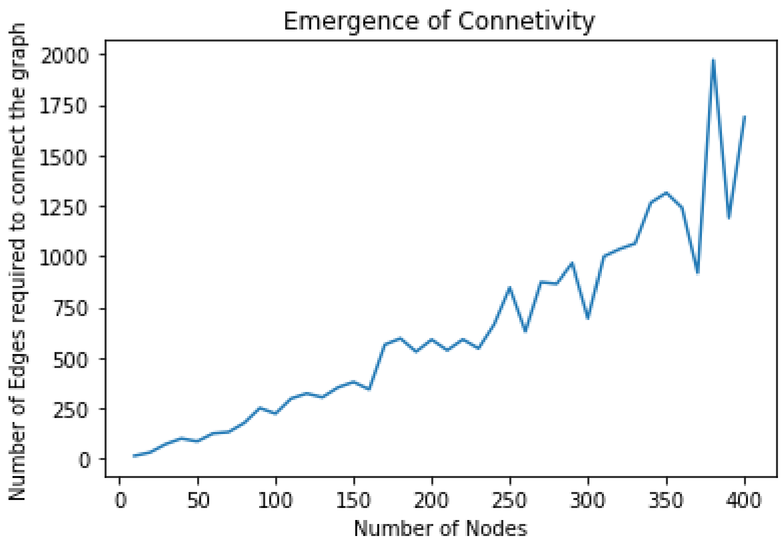
In [10]:

```
#Experiment
print create_instance(10)
print create_instance(20)
print create_instance(100)
print create_instance(1000)
```

```
19
20
203
3439
```

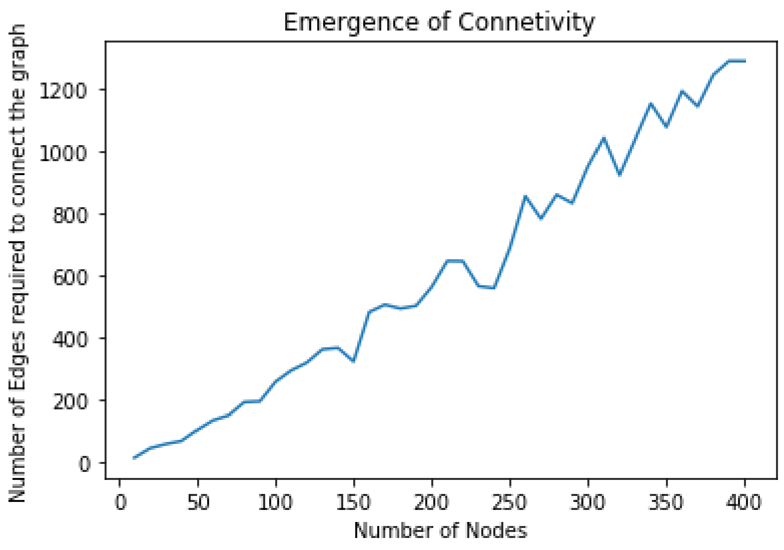
In [11]:

```
#Plot the desired for different number of edges
def plot_connectivity():
    x = []
    y = []
    i = 10 #i is the number of nodes
    while(i <= 400):
        x.append(i)
        y.append(create_instance(i))
        i += 10
    plt.title("Emergence of Connectivity")
    plt.xlabel("Number of Nodes")
    plt.ylabel("Number of Edges required to connect the graph")
    plt.plot(x, y)
    plt.show()

plot_connectivity()
```



In [14]:

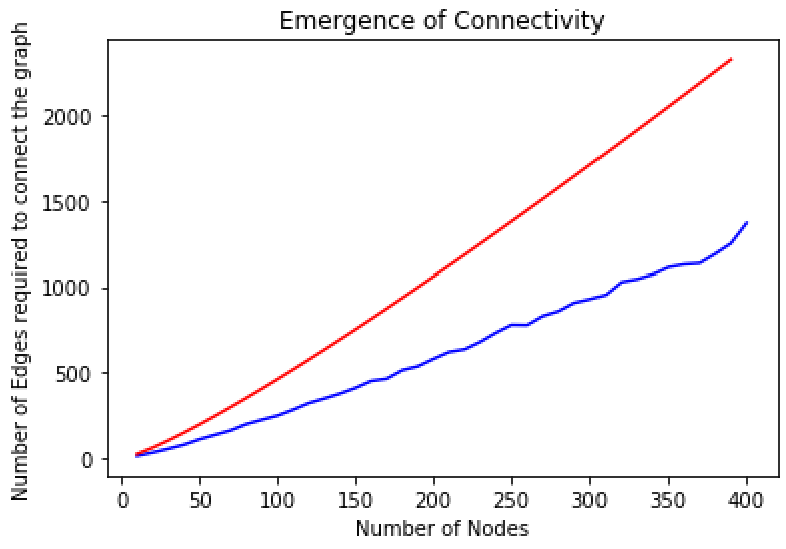
```
#Average it over 4 instances to make the curve smoother
def create_avg_instance(n):
    list1 = []
    for i in range(4):
        list1.append(create_instance(n))
    return np.average(list1)

#Plot the desired for different number of edges
def plot_connectivity():
    x = []
    y = []
    i = 10 #i is the number of nodes
    while(i <= 400):
        x.append(i)
        y.append(create_avg_instance(i))
        i += 10
    plt.title("Emergence of Connectivity")
    plt.xlabel("Number of Nodes")
    plt.ylabel("Number of Edges required to connect the graph")
    plt.plot(x, y)
    plt.show()

plot_connectivity()
#Graph => smoother
```



In [16]:

```
###!!!!!!!!!!!!!Takes more time to run!
#Lets compare the number of edges required to nlogn
#Average it over 100 instances to make the curve extra smoother
def create_avg_instance(n):
    list1 = []
    for i in range(100):
        list1.append(create_instance(n))
    return np.average(list1)

#Plot the desired for different number of edges
def plot_connectivity():
    x = []
    y = []
    i = 10 #i is the number of nodes
    while(i <= 400):
        x.append(i)
        y.append(create_avg_instance(i))
        print i,
        i += 10
    plt.title("Emergence of Connectivity")
    plt.xlabel("Number of Nodes")
    plt.ylabel("Number of Edges required to connect the graph")
    plt.plot(x, y, 'b')

    x2 = []
    y2 = []
    i = 10
    while(i < 400):
        x2.append(i)
        y2.append(i*np.log(i))
        i += 10
    plt.plot(x2, y2, 'r')

    plt.show()

plot_connectivity()
```

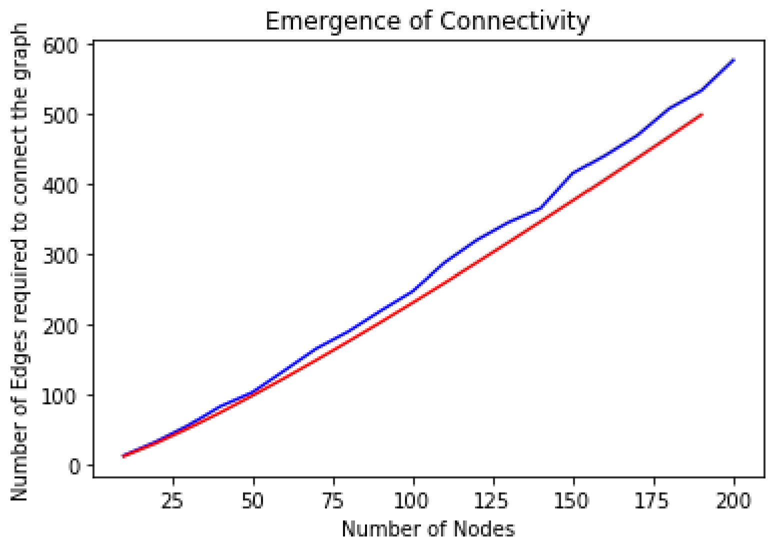
```
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 38
0 390 400
```



In [18]:

```
#Lets compare the number of edges required to nlogn
#Average it over 100 instances to make the curve extra smoother
def create_avg_instance(n):
    list1 = []
    for i in range(100):
        list1.append(create_instance(n))
    return np.average(list1)

#Plot the desired for different number of edges
def plot_connectivity():
    x = []
    y = []
    i = 10 #i is the number of nodes
    while(i <= 200):
        x.append(i)
        y.append(create_avg_instance(i))
        print i,
        i += 10
    plt.title("Emergence of Connectivity")
    plt.xlabel("Number of Nodes")
    plt.ylabel("Number of Edges required to connect the graph")
    plt.plot(x, y, 'b')

    x2 = []
    y2 = []
    i = 10
    while(i < 200):
        x2.append(i)
        y2.append(i * float(np.log(i))/2) #nlogn/2 = O(nlogn)
        i += 10
    plt.plot(x2, y2, 'r')

    plt.show()

plot_connectivity()
```

```
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
```



Lec26 Summary

- Datasets
- Emergence of Connectedness
- Importance of sythetic datasets

In [20]:

```
print nx.read_weighted_edgelist?
```

In [21]:

```
print nx.read_edgelist?
```

In [22]:

```
print nx.read_adjlist?
```