1_CodeNo(2 Novelty)

April 15, 2021

1 Novelty

1.0.1 Juhi Pandey 2018393

1.0.2 Ritik Malik 2018406

1.0.3 Yashraj 2018422

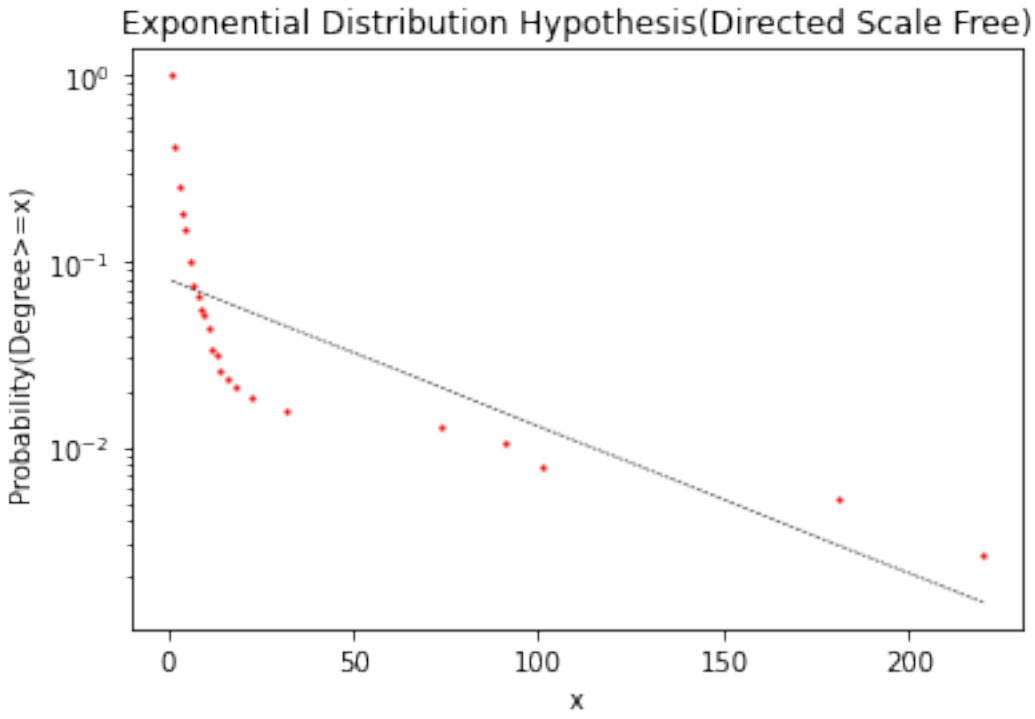
1. Compared the macaque brain degree distribution to other distributions to see what fits best

[1]:
```
import networkx as nx
from collections import Counter
import matplotlib.pyplot as plt
from numpy.polynomial.polynomial import polyfit
import numpy as np
no_of_nodes = 383
```

[2]:
```
g = nx.generators.directed.scale_free_graph(no_of_nodes)
degrees = [g.degree(n) for n in g.nodes()]
degrees = Counter(degrees)
degrees = dict(sorted(degrees.items()))
print(degrees)

no_of_nodes = sum(degrees.values())
for key in degrees.keys():
    degrees[key] /= no_of_nodes

cum_degree = {}
temp = 1
for key in degrees:
    cum_degree[key] = temp
    temp -= degrees[key]
```

```
{1: 225, 2: 60, 3: 28, 4: 13, 5: 19, 6: 10, 7: 3, 8: 4, 9: 1, 10: 3, 11: 4, 12:
1, 13: 2, 14: 1, 16: 1, 18: 1, 23: 1, 32: 1, 74: 1, 91: 1, 101: 1, 181: 1, 220:
1}
```

[3]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis(Directed Scale Free)")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.yscale("log")
b, m = polyfit(np.array(list(cum_degree.keys())), np.log10(np.
 ↪array(list(cum_degree.values()))), 1)
y_hat = (b + m * np.array(list(cum_degree.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(cum_degree.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
plt.scatter(cum_degree.keys(), cum_degree.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```

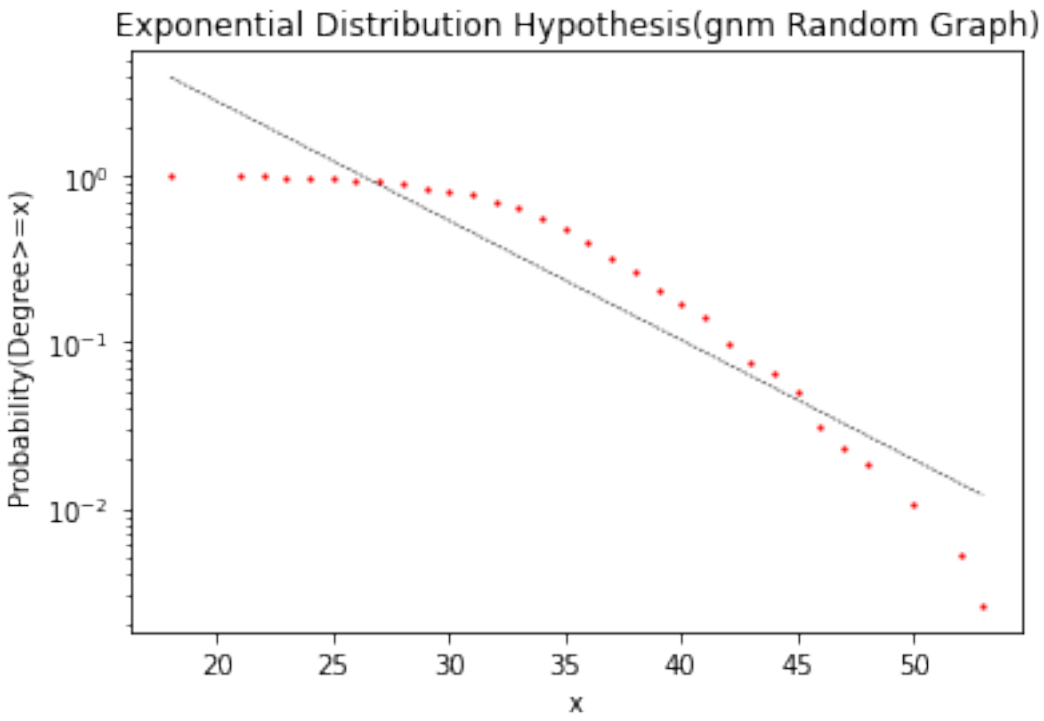


[4]:
```
g = nx.gnm_random_graph(no_of_nodes, 6602)
degrees = [g.degree(n) for n in g.nodes()]
degrees = Counter(degrees)
degrees = dict(sorted(degrees.items()))
print(degrees)

no_of_nodes = sum(degrees.values())
for key in degrees.keys():
```

```
    degrees[key] /= no_of_nodes

cum_degree = {}
temp = 1
for key in degrees:
    cum_degree[key] = temp
    temp -= degrees[key]
```

```
{18: 1, 21: 1, 22: 2, 23: 2, 24: 5, 25: 7, 26: 7, 27: 14, 28: 17, 29: 13, 30:
18, 31: 26, 32: 23, 33: 32, 34: 28, 35: 33, 36: 29, 37: 23, 38: 22, 39: 15, 40:
11, 41: 16, 42: 9, 43: 4, 44: 6, 45: 7, 46: 3, 47: 2, 48: 3, 50: 2, 52: 1, 53:
1}
```

[5]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis(gnm Random Graph)")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.yscale("log")
b, m = polyfit(np.array(list(cum_degree.keys())), np.log10(np.
 ↪array(list(cum_degree.values()))), 1)
y_hat = (b + m * np.array(list(cum_degree.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(cum_degree.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
plt.scatter(cum_degree.keys(), cum_degree.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```

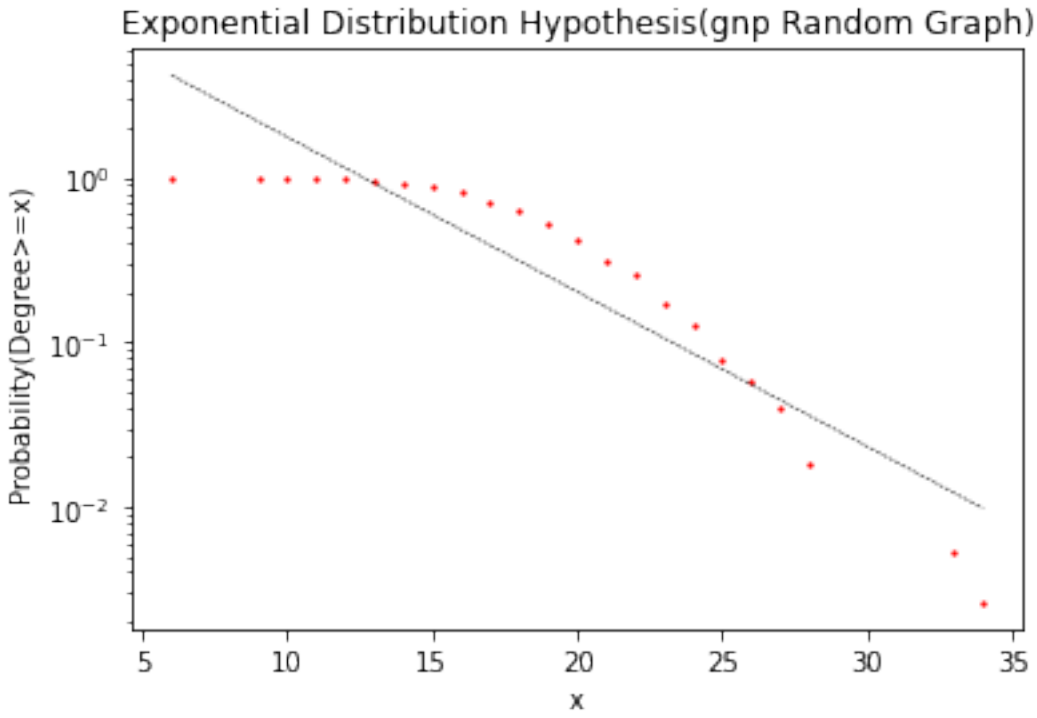


[6]:
```
g = nx.gnp_random_graph(no_of_nodes, 0.05)
degrees = [g.degree(n) for n in g.nodes()]
degrees = Counter(degrees)
degrees = dict(sorted(degrees.items()))
print(degrees)

no_of_nodes = sum(degrees.values())
for key in degrees.keys():
    degrees[key] /= no_of_nodes

cum_degree = {}
temp = 1
for key in degrees:
    cum_degree[key] = temp
    temp -= degrees[key]
```

```
{6: 2, 9: 3, 10: 2, 11: 3, 12: 6, 13: 15, 14: 12, 15: 29, 16: 40, 17: 32, 18:
40, 19: 37, 20: 42, 21: 22, 22: 33, 23: 16, 24: 19, 25: 8, 26: 7, 27: 8, 28: 5,
33: 1, 34: 1}
```

[7]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis(gnp Random Graph)")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
```

```
plt.yscale("log")
b, m = polyfit(np.array(list(cum_degree.keys())), np.log10(np.
 ↪array(list(cum_degree.values()))), 1)
y_hat = (b + m * np.array(list(cum_degree.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(cum_degree.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
plt.scatter(cum_degree.keys(), cum_degree.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```

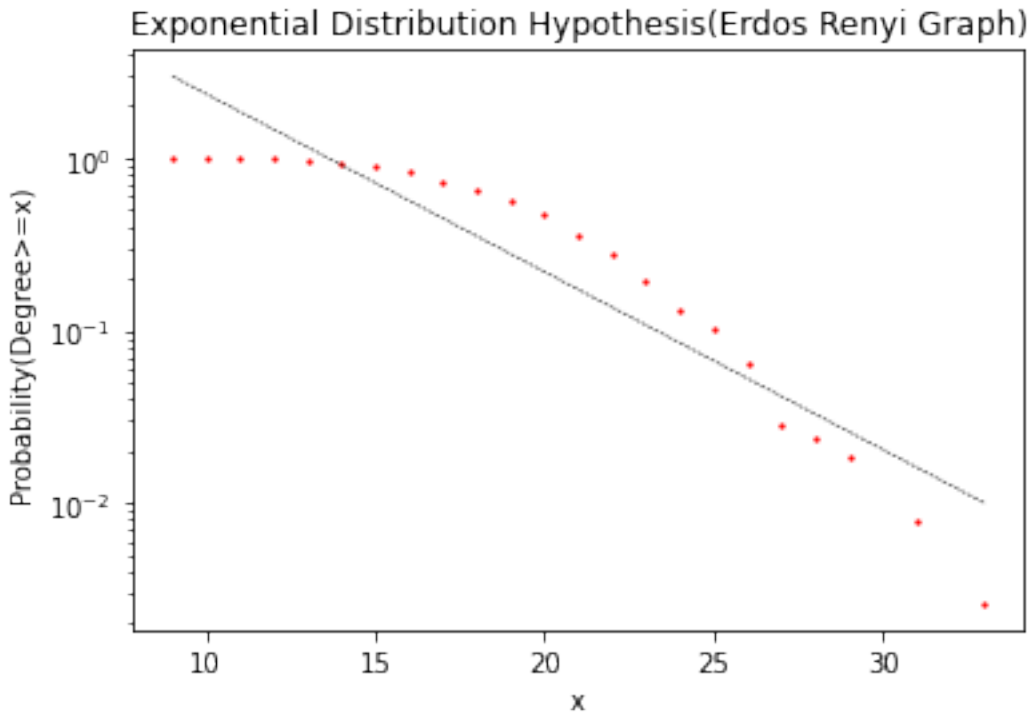


[8]:
```
g = nx.erdos_renyi_graph(no_of_nodes, 0.05)
degrees = [g.degree(n) for n in g.nodes()]
degrees = Counter(degrees)
degrees = dict(sorted(degrees.items()))
print(degrees)

no_of_nodes = sum(degrees.values())
for key in degrees.keys():
    degrees[key] /= no_of_nodes

cum_degree = {}
temp = 1
```

```
for key in degrees:
    cum_degree[key] = temp
    temp -= degrees[key]
```

```
{9: 1, 10: 3, 11: 3, 12: 8, 13: 14, 14: 15, 15: 23, 16: 36, 17: 34, 18: 29, 19:
38, 20: 43, 21: 30, 22: 31, 23: 24, 24: 12, 25: 14, 26: 14, 27: 2, 28: 2, 29: 4,
31: 2, 33: 1}
```

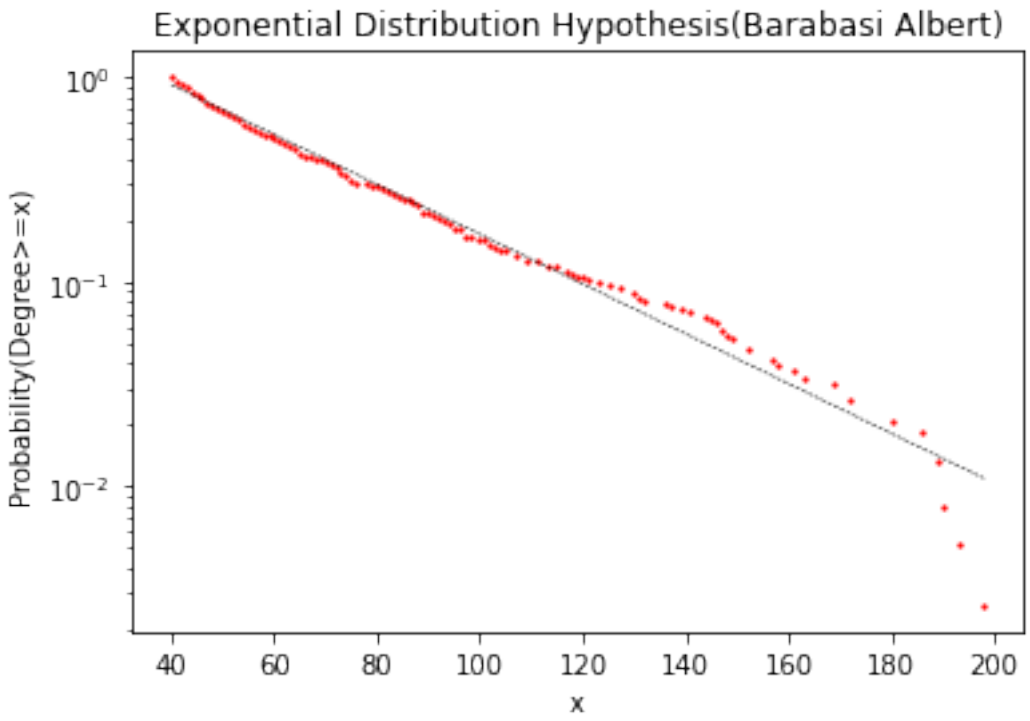
[9]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis(Erdos Renyi Graph)")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.yscale("log")
b, m = polyfit(np.array(list(cum_degree.keys())), np.log10(np.
 ↪array(list(cum_degree.values()))), 1)
y_hat = (b + m * np.array(list(cum_degree.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(cum_degree.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
plt.scatter(cum_degree.keys(), cum_degree.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```



[10]:
```
g = nx.barabasi_albert_graph(383, 40)
degrees = [g.degree(n) for n in g.nodes()]
degrees = Counter(degrees)
degrees = dict(sorted(degrees.items()))
print(degrees)

no_of_nodes = sum(degrees.values())
for key in degrees.keys():
    degrees[key] /= no_of_nodes

cum_degree = {}
temp = 1
for key in degrees:
    cum_degree[key] = temp
    temp -= degrees[key]
```

```
{40: 15, 41: 14, 42: 13, 43: 16, 44: 8, 45: 17, 46: 12, 47: 7, 48: 12, 49: 10,
50: 9, 51: 3, 52: 12, 53: 8, 54: 9, 55: 7, 56: 4, 57: 6, 58: 4, 59: 4, 60: 4,
61: 8, 62: 6, 63: 6, 64: 7, 65: 5, 66: 2, 67: 2, 68: 2, 69: 3, 70: 5, 71: 5, 72:
8, 73: 5, 74: 6, 75: 2, 76: 1, 78: 3, 79: 2, 80: 2, 81: 3, 82: 4, 83: 3, 84: 2,
85: 1, 86: 1, 87: 3, 88: 8, 89: 2, 90: 1, 91: 3, 92: 1, 93: 3, 94: 4, 95: 1, 96:
5, 97: 1, 98: 1, 100: 1, 101: 3, 102: 2, 103: 1, 104: 1, 105: 2, 107: 3, 109: 1,
111: 2, 113: 1, 115: 2, 117: 1, 118: 1, 119: 1, 120: 1, 121: 1, 123: 1, 125: 1,
127: 2, 130: 2, 131: 1, 132: 1, 136: 1, 137: 1, 139: 1, 141: 1, 144: 1, 145: 1,
146: 2, 147: 1, 148: 1, 149: 2, 152: 2, 157: 1, 158: 1, 161: 1, 163: 1, 169: 2,
172: 2, 180: 1, 186: 2, 189: 2, 190: 1, 193: 1, 198: 1}
```

[11]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis(Barabasi Albert)")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.yscale("log")
b, m = polyfit(np.array(list(cum_degree.keys())), np.log10(np.
 ↪array(list(cum_degree.values()))), 1)
y_hat = (b + m * np.array(list(cum_degree.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(cum_degree.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
plt.scatter(cum_degree.keys(), cum_degree.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```



2. Studied the assortativity and average shortest path length

[12]:
```
g = nx.DiGraph()
node_label = {}
f = open("NameList.txt")
line = f.readline()
while line:
    n1, n2 = line.split()
    node_label[int(n1)] = n2
    line = f.readline()
f.close()
# print("No of nodes:", len(node_label))
f = open("EdgeList.txt")
line = f.readline()
while line:
    n1, n2 = map(int, line.split())
    g.add_edge(node_label[n1], node_label[n2])
    line = f.readline()
f.close()
print("Average Shortest Path Length:", nx.average_shortest_path_length(g))
print("Degree Assortativity of Graph", nx.degree_assortativity_coefficient(g))
```

```
Average Shortest Path Length: 2.5541705354379447
Degree Assortativity of Graph -0.1147765924196248
```

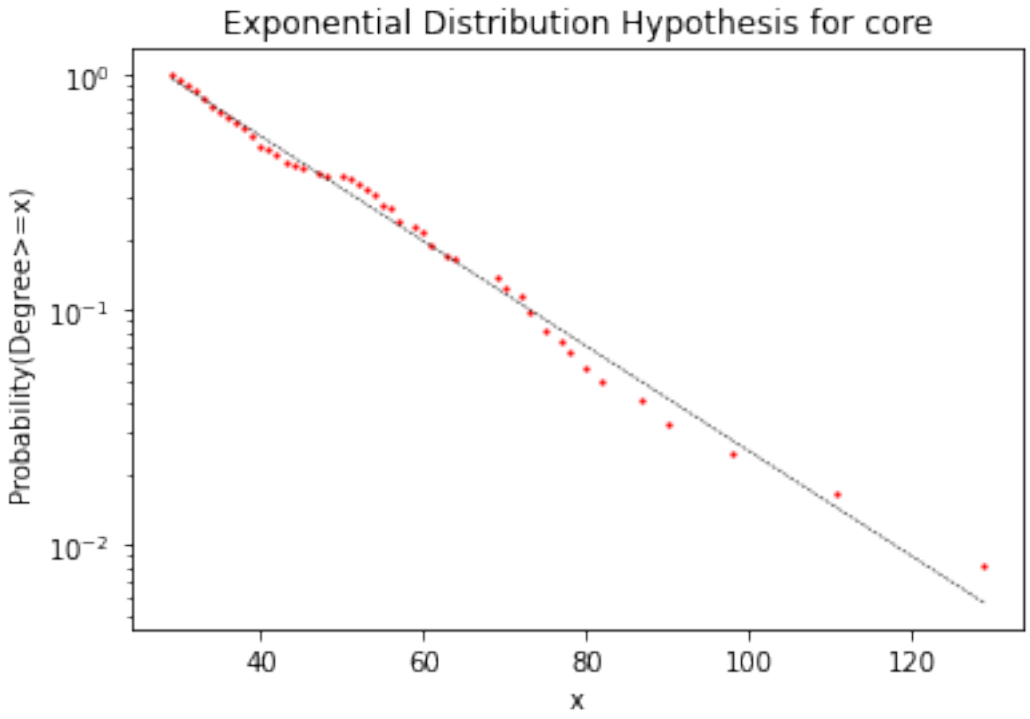
[13]:
```
core = g.copy()
d_core = [core.degree(n) for n in core.nodes()]
d_core = dict(sorted(Counter(d_core).items()))
for circle in range(1, 29):
    remove = []
    for node in core.nodes:
        if core.degree(node) <= circle:
            remove.append(node)
    core.remove_nodes_from(remove)

while True:
    remove = []
    for node in core.nodes:
        if core.degree(node) < 29:
            remove.append(node)
    core.remove_nodes_from(remove)
    if not remove:
        break

d_core = [core.degree(n) for n in core.nodes()]
print("Remaining nodes:", len(d_core))
d_core = dict(sorted(Counter(d_core).items()))
# print(d_core)
print("Average Shortest Path Length:", nx.average_shortest_path_length(core))
print("Degree Assortativity of Core", nx.degree_assortativity_coefficient(core))
```

```
Remaining nodes: 122
Average Shortest Path Length: 1.929006909632841
Degree Assortativity of Core 0.003719185906300945
```

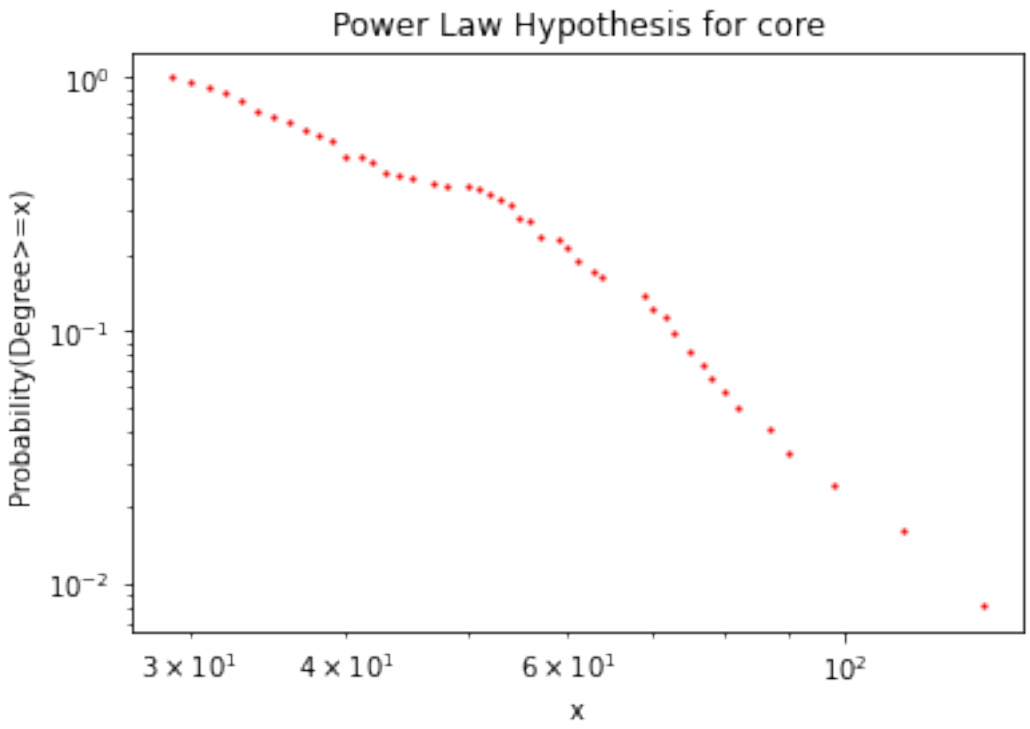
[14]:
```
d_corec = {}
temp  = 1
for i in sorted(d_core):
    d_corec[i] = temp
    temp -= (d_core[i]/122)
```

[15]:
```
plt.figure()
plt.title("Exponential Distribution Hypothesis for core")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.yscale("log")
b, m = polyfit(np.array(list(d_corec.keys())), np.log10(np.array(list(d_corec.
 ↪values()))), 1)
y_hat = (b + m * np.array(list(d_corec.keys())))
y_hat = [10**i for i in y_hat]
plt.plot(np.array(list(d_corec.keys())), y_hat, color="black", linestyle =␣
 ↪'--', linewidth = 0.5)
```

```
plt.scatter(d_corec.keys(), d_corec.values(), color="red", s=1.5)
plt.autoscale()
plt.show()
```



[16]:
```
plt.figure()
plt.title("Power Law Hypothesis for core")
plt.xlabel("x")
plt.ylabel("Probability(Degree>=x)")
plt.xscale("log")
plt.yscale("log")
plt.scatter(d_corec.keys(), d_corec.values(), color="red", s=1.5)
# print(len(cum_degree))

plt.autoscale()
plt.show()
```



3. Checked to see if our mapping follows a hierarchy

[17]:
```
hierarchy = nx.DiGraph()
```

[18]:
```
f = open("Mapping.txt")
line = f.readline()
while line:
    parent, child = map(int, line.split())
    hierarchy.add_edge(node_label[parent], node_label[child])
    line = f.readline()
f.close()
```

[19]:
```
print("Hierarchy:", nx.flow_hierarchy(hierarchy))
```

```
Hierarchy: 1.0
```