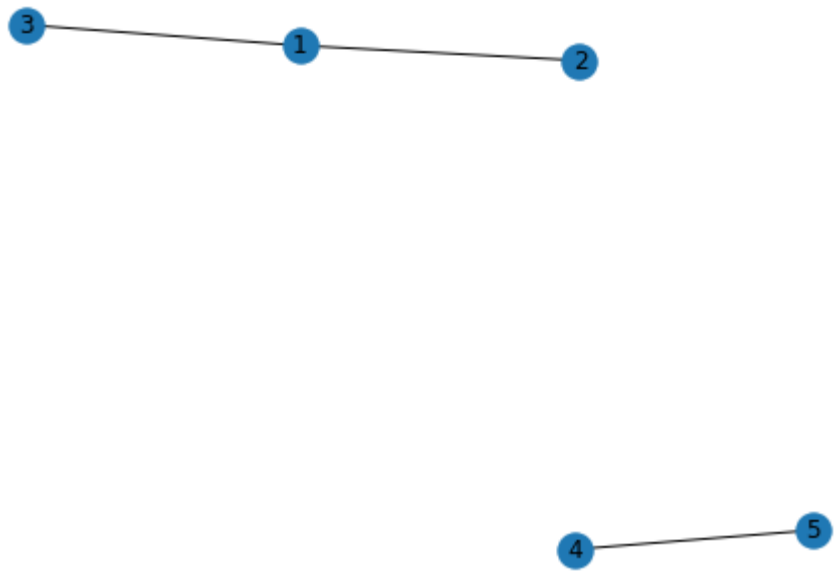
Tutorial do biblioteki networkx

In [ ]:
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
G = nx.Graph()

# lets add some nodes
G.add_node(1)
G.add_nodes_from([2, 3])
G.add_nodes_from([
    (4, {"color": "red"}),
    (5, {"color": "green"}),
])

# lets check the label for node 4
color=nx.get_node_attributes(G,'color')
print(color[4])

# adding some edges
G.add_edge(1, 2)
G.add_edges_from([(1, 3), (4, 5)])

nx.draw(G, with_labels=True)
plt.draw()
```

```
red
```



In [ ]:
```
# creating path graph H with 10 nodes and adding it as a subgraph to G
H = nx.path_graph(10)
G.add_node(H)

print(list(G.nodes))
nx.draw(G, with_labels=True)
plt.plot()
```

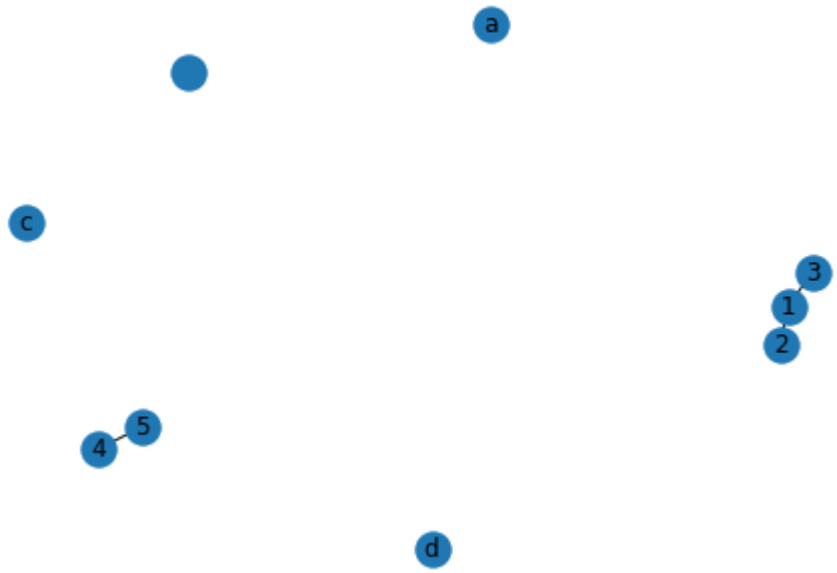
```
[1, 2, 3, 4, 5, <networkx.classes.graph.Graph object at 0x7f8d418d7e10>]
```

Out[ ]:
```
[]
```

In [ ]:
```
G.add_nodes_from("abcd")
G.remove_node('b')

# neighbours of vertex with label 1
print(G[1])

nx.draw(G, with_labels=True)
plt.plot()
```

```
{2: {}, 3: {}}
```

Out[ ]:
```
[]
```



In [ ]:
```
# lets create directed graph
DG = nx.DiGraph()
DG.add_weighted_edges_from([(1, 2, 0.5), (3, 1, 0.75)])
DG.out_degree(1, weight='weight')

print(DG.degree(1, weight='weight'))
print(list(DG.successors(1)))
```

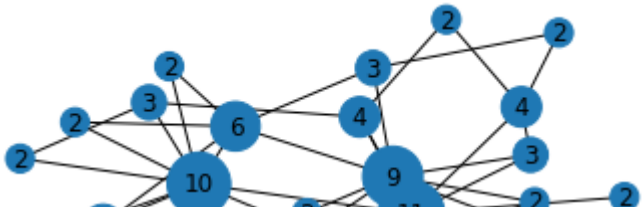
```
1.25
[2]
```

Model Barabási–Albert

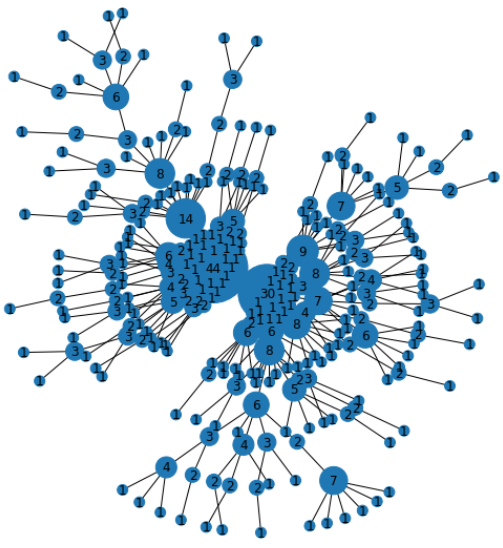
G(n, m)

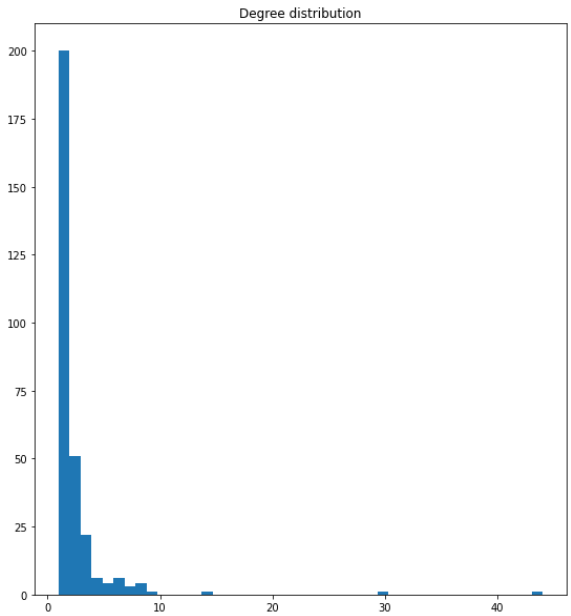
In [ ]:
```
n = 30   # 30 nodes
m = 2    # each node is joined with 2 neighbors

G = nx.barabasi_albert_graph(n, m)

pos = nx.kamada_kawai_layout(G)
node_sizes = []
node_labels = {}

# some properties
print("node | degree | clustering")
for v in nx.nodes(G):
    print(f"{str(v).ljust(4)} | {str(nx.degree(G, v)).ljust(6)} | {nx.clus

for v in nx.nodes(G):
  node_sizes.append(nx.degree(G, v) * 100)
  node_labels[v] = nx.degree(G, v)

# plotting network with node thickness proportional to its degree
nx.draw_networkx_labels(G, pos, node_labels)
nx.draw(G, pos, with_labels=False, node_size=node_sizes)
```

```
node | degree | clustering
0    | 6      | 0.26666666666666666
1    | 6      | 0.26666666666666666
2    | 10     | 0.15555555555555556
3    | 9      | 0.05555555555555555
4    | 4      | 0.3333333333333333
5    | 3      | 0.3333333333333333
6    | 11     | 0.09090909090909091
7    | 7      | 0.23809523809523808
8    | 3      | 0
9    | 3      | 0.3333333333333333
10   | 2      | 1.0
11   | 4      | 0
12   | 3      | 0.3333333333333333
13   | 2      | 1.0
14   | 4      | 0.16666666666666666
15   | 2      | 0
16   | 6      | 0.2
17   | 2      | 0
18   | 2      | 0
19   | 2      | 1.0
20   | 2      | 0
21   | 2      | 1.0
22   | 2      | 1.0
23   | 2      | 0
24   | 2      | 1.0
25   | 2      | 1.0
26   | 2      | 1.0
27   | 2      | 1.0
28   | 3      | 0
29   | 2      | 0
```



In [ ]:
```
n = 300   # 300 nodes
m = 1    # each node is joined with 1 neighbour

G = nx.barabasi_albert_graph(n, m)

pos = nx.kamada_kawai_layout(G)
node_sizes = []
node_labels = {}

for v in nx.nodes(G):
  node_sizes.append(nx.degree(G, v))
  node_labels[v] = nx.degree(G, v)


# network with node thickness proportional to its degree
plt.figure(figsize=(20,10))
plt.subplot(121)
nx.draw_networkx_labels(G, pos, node_labels)
nx.draw(G, pos, with_labels=False, node_size=[i*100 for i in node_sizes])

# distribution of node degrees
a = np.array(node_sizes)
plt.subplot(122)
plt.hist(a, bins = max(node_sizes))
plt.title("Degree distribution")
plt.show()
```



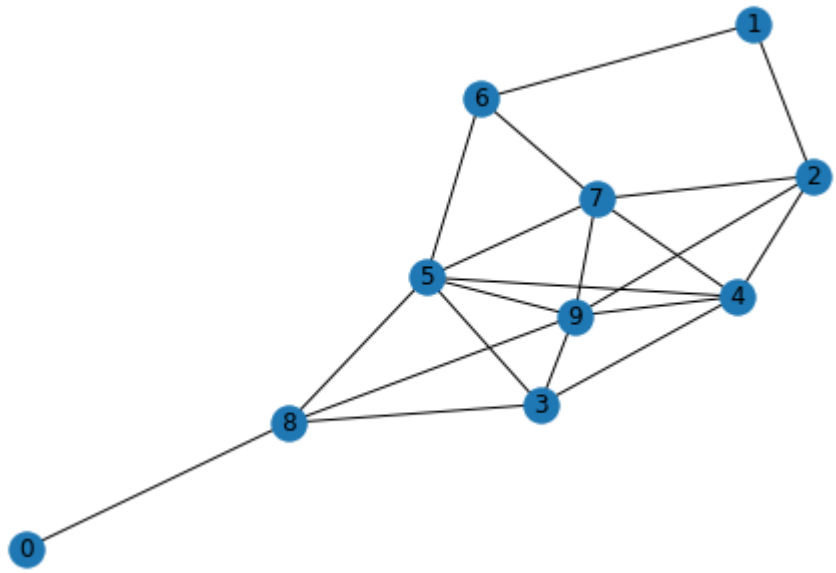


Model Erdős–Rényi

G(n, M)

In [ ]:
```
n = 10  # 10 nodes
m = 20  # 20 edges

G = nx.gnm_random_graph(n, m)

# some properties
print("node | degree | clustering")
for v in nx.nodes(G):
    print(f"{str(v).ljust(4)} | {str(nx.degree(G, v)).ljust(6)} | {nx.clus
```

```
node | degree | clustering
0    | 1      | 0
1    | 2      | 0
2    | 4      | 0.5
3    | 4      | 0.8333333333333334
4    | 5      | 0.7
5    | 6      | 0.5333333333333333
6    | 3      | 0.3333333333333333
7    | 5      | 0.6
8    | 4      | 0.5
9    | 6      | 0.6
```

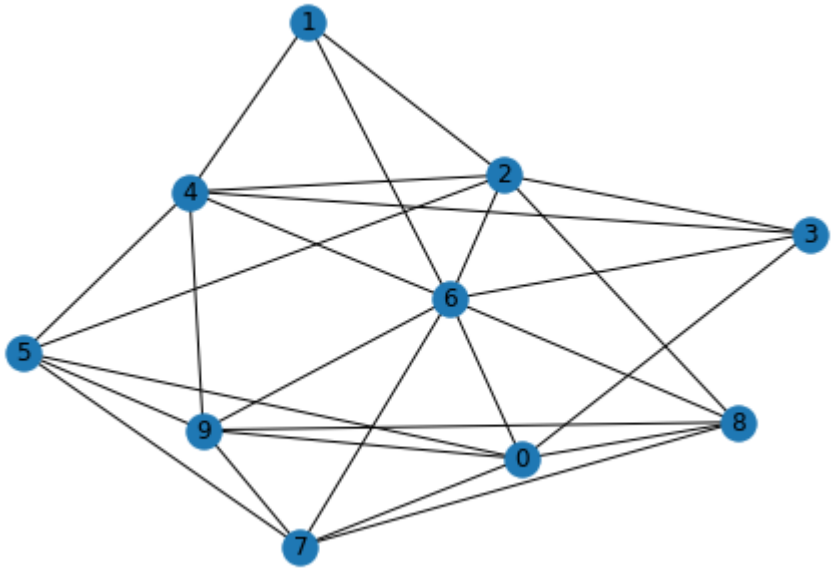


G(n, p)

In [ ]:
```
n = 10   # 10 nodes
p = 0.5  # 50% probability

G = nx.erdos_renyi_graph(n, p)

# some properties
print("node | degree | clustering")
for v in nx.nodes(G):
    print(f"{str(v).ljust(4)} | {str(nx.degree(G, v)).ljust(6)} | {nx.clus
```

```
node | degree | clustering
0    | 6      | 0.6
1    | 3      | 1.0
```

```
2    | 6       | 0.4666666666666667
3    | 4       | 0.6666666666666666
4    | 6       | 0.5333333333333333
5    | 5       | 0.5
6    | 8       | 0.5
7    | 5       | 0.8
8    | 5       | 0.7
```



Model Watts-Strogatz

In [ ]:

```
n = 20   # 20 nodes
k = 4    # each node is joined with 4 neighbors
p = 0.2  # 20% probability

G = nx.watts_strogatz_graph(n, k, p)

# some properties
print("node | degree | clustering")
for v in nx.nodes(G):
    print(f"{str(v).ljust(4)} | {str(nx.degree(G, v)).ljust(6)} | {nx.clus

nx.draw(G, with_labels=True)
plt.show()
```

```
node | degree | clustering
0    | 6      | 0.3333333333333333
1    | 6      | 0.13333333333333333
2    | 5      | 0.2
3    | 4      | 0.3333333333333333
4    | 4      | 0.16666666666666666
5    | 4      | 0.16666666666666666
6    | 4      | 0.3333333333333333
7    | 3      | 0
8    | 2      | 1.0
9    | 4      | 0.3333333333333333
10   | 4      | 0.5
11   | 5      | 0.4
12   | 4      | 0.3333333333333333
13   | 4      | 0.16666666666666666
14   | 4      | 0.16666666666666666
15   | 3      | 0
16   | 3      | 0.3333333333333333
17   | 4      | 0.3333333333333333
```

```
18    | 4       | 0.5
```