Untitled

May 4, 2022

[1]:
```
# Netzwerkanalyse
```

[2]:
```
#Ein Netzwerk kann man mathematisch als ein Graph darstellen.
# G=(Knoten, Kanten)
```

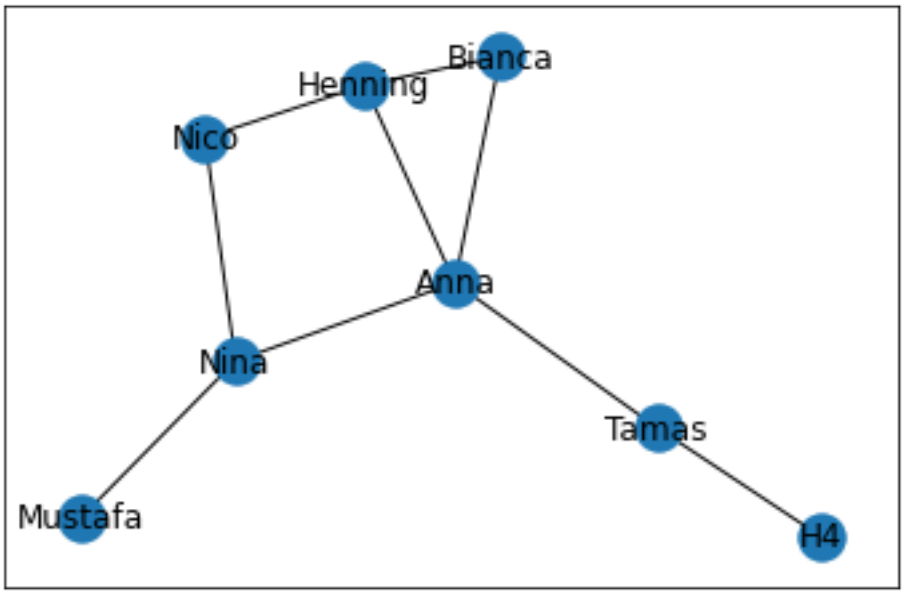
[3]:
```
!pip install networkx
```

```
Requirement already satisfied: networkx in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (2.6.3)
```

[6]:
```
import networkx as nx

Graph = nx.Graph() # Platzhalter
Graph.add_edge('Mustafa', 'Nina')
Graph.add_edge('Nina', 'Nico')
Graph.add_edge('Nina', 'Anna')
Graph.add_edge('Anna', 'Henning')
Graph.add_edge('Nico', 'Henning')
Graph.add_edge('Henning', 'Bianca')
Graph.add_edge('Bianca', 'Anna')
Graph.add_edge('Anna', 'Tamas')
Graph.add_edge('Tamas', 'H4')
```

[7]:
```
nx.draw_networkx(Graph)
```



[8]:
```
# die maximale Anzahln Beziehungen in einem Netzwerk ist N*(N-1)
```

[9]:
```
# Degree von einem Knote darstellen. Anzahl Nachbarn

nx.degree(Graph, 'Henning')
```

[9]: 3

[11]:
```
# Shortest Path zwischen Knoten

nx.shortest_path(Graph, 'Henning', 'Anna')
```

[11]: ['Henning', 'Anna']

[12]:
```
nx.shortest_path(Graph, 'Bianca', 'Mustafa')
```

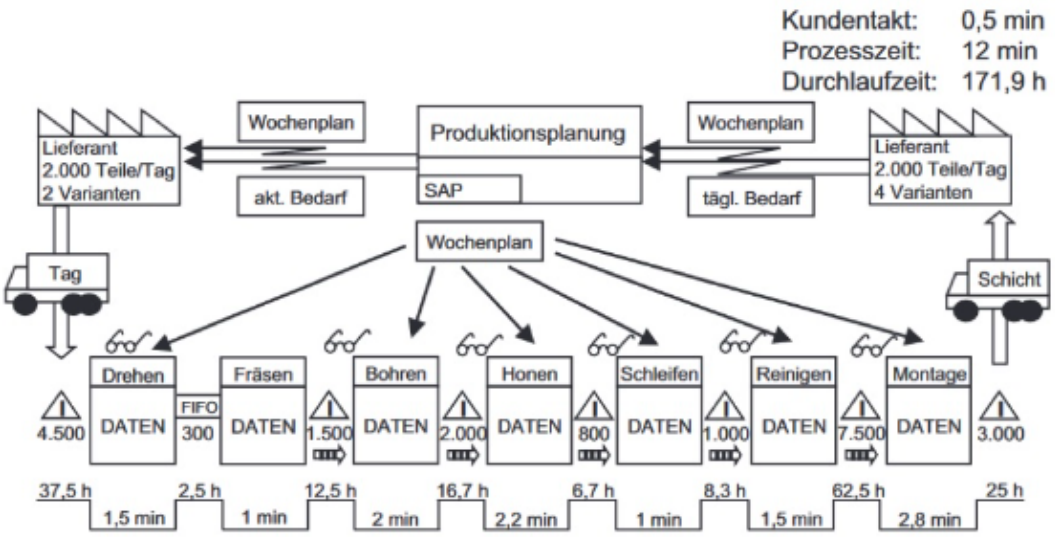
[12]: ['Bianca', 'Anna', 'Nina', 'Mustafa']

[13]:
```
# Beispiel Wertstrom
```

[17]:
```
from IPython.display import Image # Image aufzuladen

Image(filename= '/Users/h4/desktop/Wertstrom.jpeg')
```

[17]:



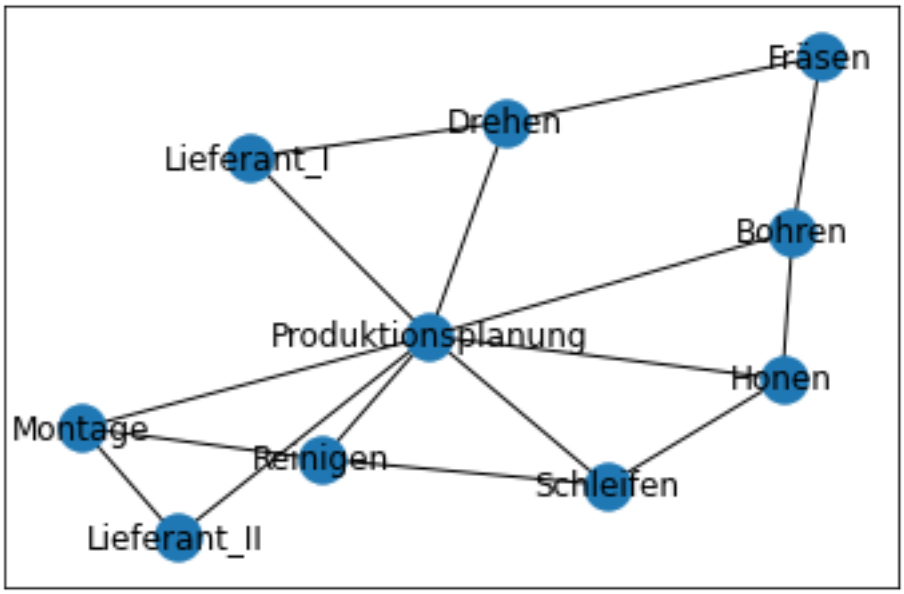
[19]:
```
import networkx as nx

Wertstrom=nx.Graph()

#Materialfluss
Wertstrom.add_edge('Drehen', 'Fräsen')
Wertstrom.add_edge('Fräsen', 'Bohren')
Wertstrom.add_edge('Bohren', 'Honen')
Wertstrom.add_edge('Honen', 'Schleifen')
Wertstrom.add_edge('Schleifen', 'Reinigen')
Wertstrom.add_edge('Reinigen', 'Montage')
Wertstrom.add_edge('Lieferant_I', 'Drehen')
Wertstrom.add_edge('Montage', 'Lieferant_II')

#Informationsfluss
Wertstrom.add_edge('Lieferant_I', 'Produktionsplanung')
Wertstrom.add_edge('Produktionsplanung', 'Lieferant_II')
Wertstrom.add_edge('Produktionsplanung', 'Drehen')
Wertstrom.add_edge('Produktionsplanung', 'Bohren')
Wertstrom.add_edge('Produktionsplanung', 'Honen')
Wertstrom.add_edge('Produktionsplanung', 'Schleifen')
Wertstrom.add_edge('Produktionsplanung', 'Reinigen')
Wertstrom.add_edge('Produktionsplanung', 'Montage')

nx.draw_networkx(Wertstrom)
```



[20]:
```
print(nx.info(Wertstrom))
```

```
Graph with 10 nodes and 16 edges
```

[21]:
```
nx.shortest_path(Wertstrom, 'Lieferant_II', 'Schleifen')
```

[21]:
```
['Lieferant_II', 'Produktionsplanung', 'Schleifen']
```

[22]:
```
# Shortest Path length des gesamten Netzwerks. Average Path Length

# APL = (1/N*(N-1))*summ(Alle Abstände Aller Knoten zueinander)

print(nx.average_shortest_path_length(Wertstrom))
```

```
1.7333333333333334
```

[24]:
```
# Book Recommendation. Network Science (Barabasi) 2016
```

[25]:
```
# Beispiel KARATE
```

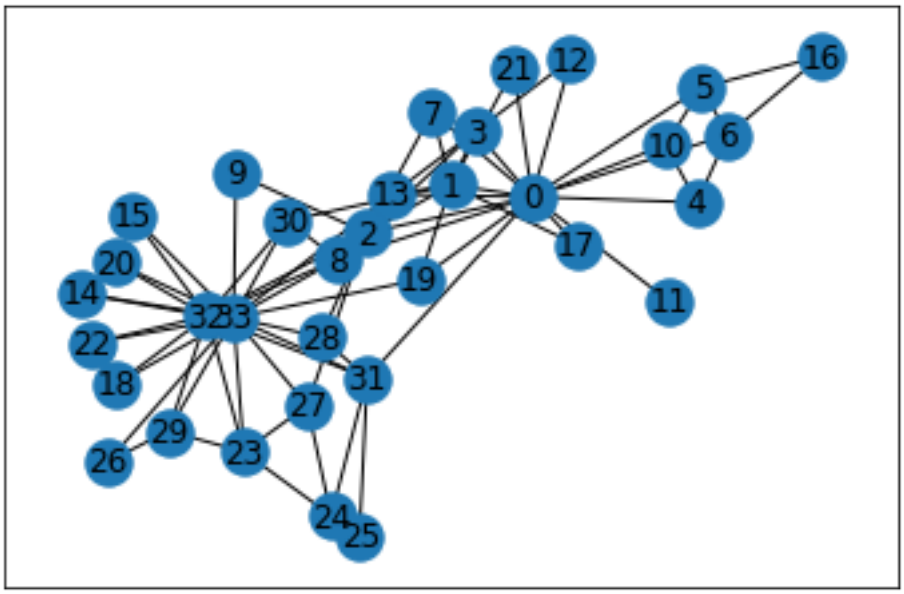
[26]:
```
ZKC_graph = nx.karate_club_graph()
```

[27]:
```
club_labels = nx.get_node_attributes(ZKC_graph, 'club')
```

[28]:
```
club_labels
```

[28]:
```
{0: 'Mr. Hi',
 1: 'Mr. Hi',
 2: 'Mr. Hi',
 3: 'Mr. Hi',
 4: 'Mr. Hi',
 5: 'Mr. Hi',
 6: 'Mr. Hi',
 7: 'Mr. Hi',
 8: 'Mr. Hi',
 9: 'Officer',
 10: 'Mr. Hi',
 11: 'Mr. Hi',
 12: 'Mr. Hi',
 13: 'Mr. Hi',
 14: 'Officer',
 15: 'Officer',
 16: 'Mr. Hi',
 17: 'Mr. Hi',
 18: 'Officer',
 19: 'Mr. Hi',
 20: 'Officer',
 21: 'Mr. Hi',
 22: 'Officer',
 23: 'Officer',
 24: 'Officer',
 25: 'Officer',
 26: 'Officer',
 27: 'Officer',
 28: 'Officer',
 29: 'Officer',
 30: 'Officer',
 31: 'Officer',
 32: 'Officer',
 33: 'Officer'}
```

[29]:
```
nx.draw_networkx(ZKC_graph)
```



[31]:
```
# Dichte ist ein Mass dafür, wie vollständig der GRaph ist
# (wie viele Kantenim Netzwerk vorhanden sind,
# vergliechen mit der gesamten Anzahl Kanten (N*(N-1)))

dichte = nx.density(ZKC_graph)

print('The edge density is: ' + str(dichte))
```

```
The edge density is: 0.13903743315508021
```

[32]:
```
# degree Distribution

# wie häufig die Anzahl NAchbarn im Netzwerk vorkommen.

degree = ZKC_graph.degree() # berechnen wir die Anzahl Nachbarn der Knoten

degree_list = [] # erzeugen wir eine leere Liste

for(n,d) in degree:
    degree_list.append(d) # liste wird ausgefüllt mit den Degrees von 'degree'

av_degree = sum(degree_list) / len(degree_list) # mittelwert der Anzahl Nachbarn
```

[33]:
```
degree_list
```

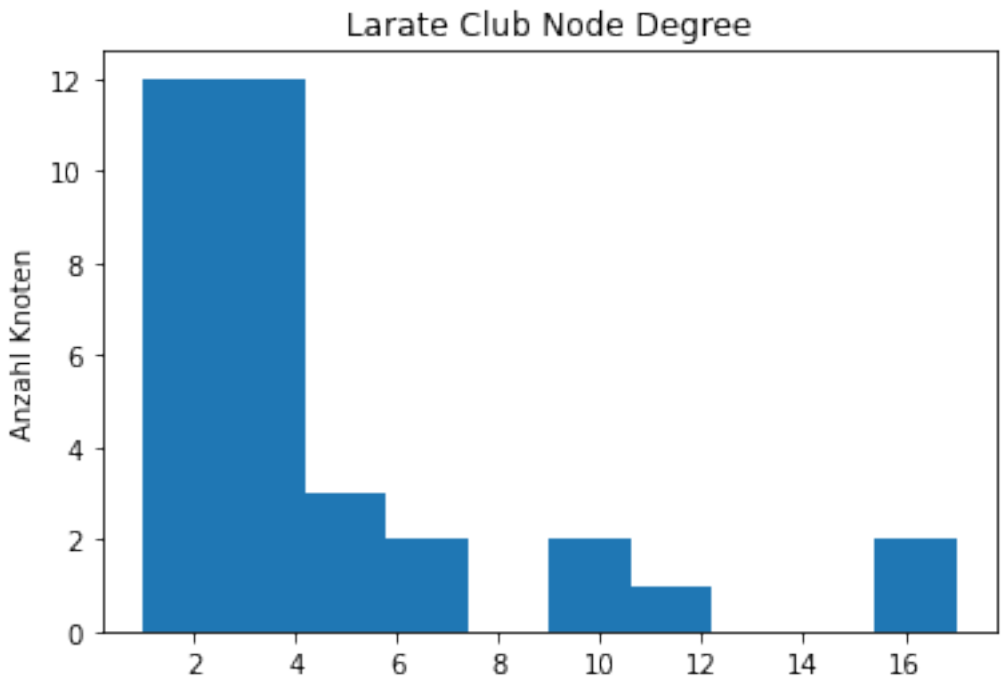
[33]:
```
[16,
 9,
```

```
10,
6,
3,
4,
4,
4,
5,
2,
3,
1,
2,
5,
2,
2,
2,
2,
2,
3,
2,
2,
2,
5,
3,
3,
2,
4,
3,
4,
4,
6,
12,
17]
```

[35]:
```
import matplotlib.pyplot as plt

plt.hist(degree_list, label='Degree Distribution')
plt.ylabel('Anzahl Knoten')
plt.title('Larate Club Node Degree')
```

[35]:
```
Text(0.5, 1.0, 'Larate Club Node Degree')
```


Larate Club Node Degree
Anzahl Knoten
0
2
4
6
8
10
12
2
4
6
8
10
12
14
16

[ ]: