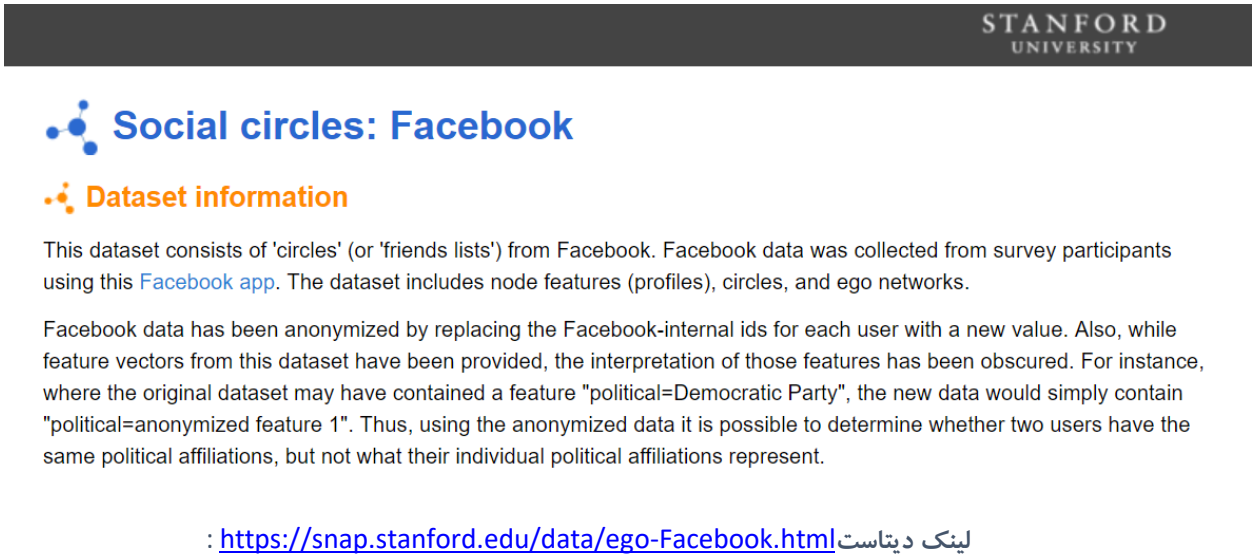
میلاد فرضعلی زاده

تمرین 3 : تحلیل گراف

برای تحلیل گراف از دیتاست فیسبوک که شامل دوستان یه کاربر در شبکه اجتماعی فیسبوک می باشد استفاد شده است . یال های گراف بدون جهت است



لینک دیتاست https://snap.stanford.edu/data/ego-Facebook.html :

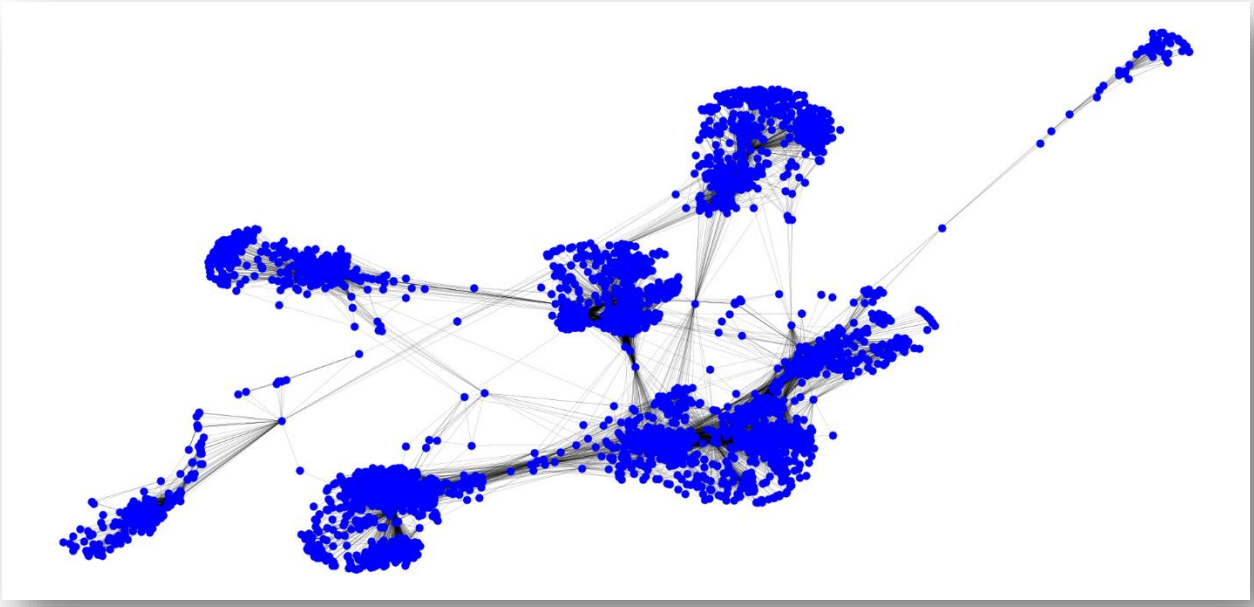
این دیتاست شامل اطلاعات زیر می باشد

```
print(nx.info(G))
```

```
[Out]: Name:
       Type: Graph
       Number of nodes: 4039
       Number of edges: 88234
       Average degree:  43.6910
```

خواندن دیتاست :

```
import networkx as nx
G =nx.read_edgelist("facebook_combined.txt", create_using = nx.Graph(), nodetype=int)
nx.draw(G)
```



شکل 1: گراف دیتاست

سوال 1 : پیدا کردن centrality در گراف :

روش Degree Centrality :

```
pos = nx.spring_layout(G1)
degCent = nx.degree_centrality(G)
node_color = [20000.0 * G.degree(v) for v in G]
node_size =  [v * 10000 for v in degCent.values()]
plt.figure(figsize=(15,15))
nx.draw_networkx(G, pos=pos, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')

```

خروجی روش :

Degree Centrality (node labels)	degree
107	1045
1684	792
1912	753
3437	547
0	340

```
sorted(degCent, key=degCent.get, reverse=True)[:5]
```

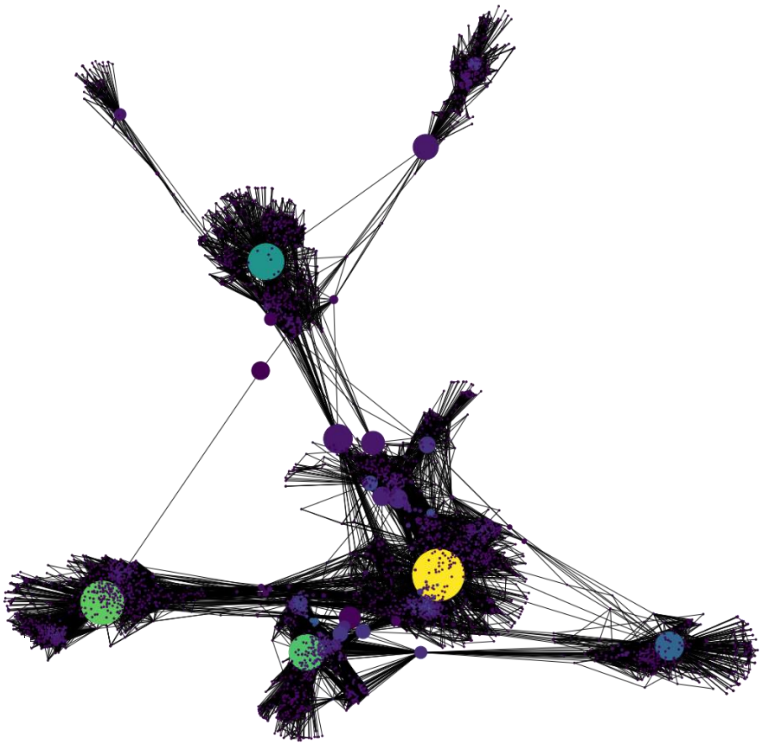
روش Betweenness Centrality :

```
pos = nx.spring_layout(G)
betCent = nx.betweenness_centrality(G, normalized=True, endpoints=True)
node_color = [20000.0 * G.degree(v) for v in G]
node_size =  [v * 10000 for v in betCent.values()]
plt.figure(figsize=(20,20))
nx.draw_networkx(G, pos=pos, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')
```

خروجی روش :

Betweenness Centrality (node labels)	degree
107	1045
1684	792
3437	547
1912	753
1085	66

```
sorted(betCent, key=betCent.get, reverse=True)[:5]
```



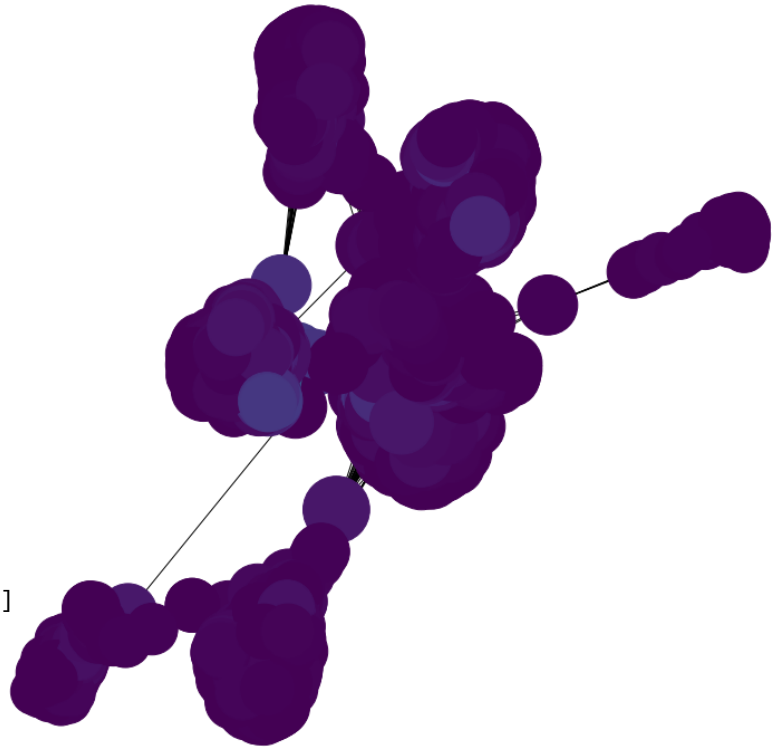
روش Closeness Centrality :

```
pos = nx.spring_layout(G)
cloCent = nx.closeness_centrality(G)
node_color = [20000.0 * G1.degree(v) for v in G]
node_size =  [v * 10000 for v in cloCent.values()]
plt.figure(figsize=(13,13))
nx.draw_networkx(G, pos=pos, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')
```

خروجی روش:

Closeness Centrality (node labels)	degree
107	1045
58	12
428	115
563	91
1684	792

```
sorted(cloCent , key=cloCent.get, reverse=True)[:5]
```

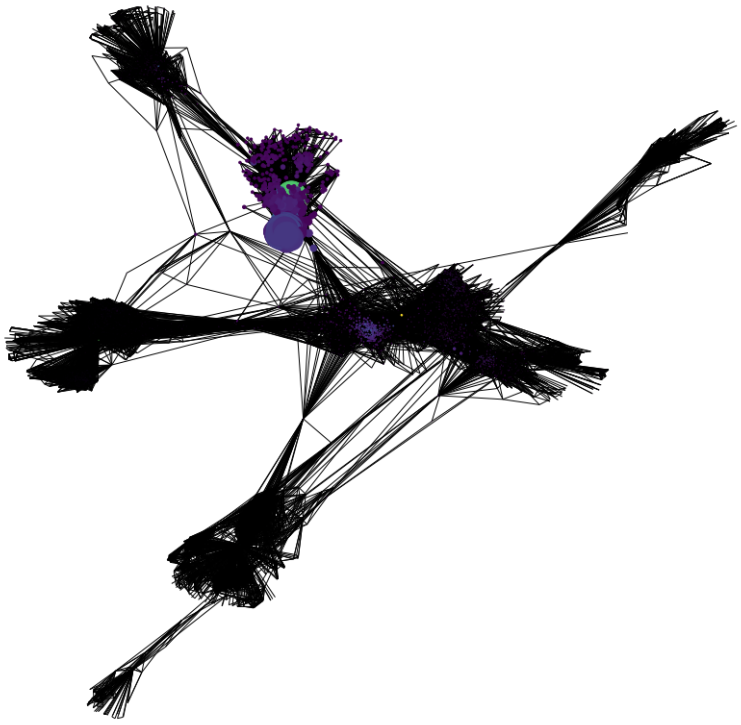


روش Eigenvector Centrality :

```
pos = nx.spring_layout(G)
eigCent = nx.eigenvector_centrality(G)
node_color = [20000.0 * G.degree(v) for v in G]
node_size =  [v * 10000 for v in eigCent.values()]
plt.figure(figsize=(15,15))
nx.draw_networkx(G, pos=pos, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')
```

خروجی روش:

Eigenvector Centrality (node labels)	degree
1912	753
2266	234
2206	210
2233	322
2464	202



سوال 2 : پیدا کردن کوتاه ترین مسیر بین نودها همراه با طول آنها

```
1. sources = [20,40,65,75]
2. targets = [650,802,920,1010]
3. for i in range(4):
4.     path = nx.shortest_path(G1,source=sources[i],target=targets[i])
5.     length = nx.shortest_path_length(G1,source=sources[i],target=targets[i],me
   thod='dijkstra')
6.     print("Shortest Path between Node ", str(sources[i])," ---
   > ", str(targets[i]), " is ",
7.           str(path), " ,Length = ", str(length))
```

خروجی :

```
Shortest Path between Node  20  --->  650  is  [20, 0, 34, 414, 650]  ,Length =  4
Shortest Path between Node  40  --->  802  is  [40, 0, 58, 1684, 860, 698, 686, 802]
,Length =  7
Shortest Path between Node  65  --->  920  is  [65, 0, 107, 920]  ,Length =  3
Shortest Path between Node  75  --->  1010  is  [75, 0, 107, 1010]  ,Length =  3
```

سوال 3 : یافتن کلیه همسایه های نودها

```
neigh = [1,20,40,65,75,90,1000,]
for i in range(len(neigh)):
    all_neighbors = list(nx.classes.function.all_neighbors(G1,1))
    print("All neighbors for Node ", str(neigh[i])," ---> ", str(all_neighbors))
```

خروجی : کلیه همسایه های 7 نود انتخابی

All neighbors for Node 1 ---> [0, 48, 53, 54, 73, 88, 92, 119, 126, 133, 194, 236, 280, 299, 315, 322, 346]

All neighbors for Node 20 ---> [0, 2, 14, 41, 44, 111, 115, 149, 162, 214, 226, 312, 326, 333, 343]

All neighbors for Node 40 ---> [0, 21, 25, 26, 29, 56, 67, 72, 77, 113, 132, 133, 141, 142, 158, 169, 172, 199, 200, 203, 212, 213, 224, 231, 232, 239, 257, 258, 265, 271, 272, 274, 277, 280, 298, 304, 307, 315, 317, 322, 325, 329, 332, 334]

All neighbors for Node 65 ---> [0, 7, 13, 25, 82, 118, 203, 252, 261, 297, 314, 339]

All neighbors for Node 75 ---> [0, 9, 56, 67, 85, 170, 188, 200, 258, 272, 274, 304, 322, 323]

All neighbors for Node 90 ---> [0, 179]

All neighbors for Node 1000 ---> [107, 924, 974, 985, 1010, 1127, 1134, 1228, 1304, 1474, 1640, 1667, 1703, 1725, 1759, 1840]

سوال 4 : یافتن درجه نودها همراه با تعداد درجه ها در گراف

```
from collections import Counter
deg = dict(G1.degree()).values()
Counter(deg )
```

خروجی :

Degree	Count
347	1
17	76
10	95
13	79
6	98
20	63

Degree	Count
65	20
11	81
68	14
5	93
44	21
7	98

سوال 5 : یافتن درجه هر نود

```
degrees = [(node,val) for (node, val) in G1.degree()]
for i in range(len(degrees)):
        print(degrees[i])
```

خروجی :

مثلا نود 107 درجه یالش 1045 می باشد

سوال 6 : یافتن clique در گراف (k=2)

```
list(nx.algorithms.community.k_clique_communities(G,k=2))
```

خروجی :

به علت پایین بودن قدرت سیستم مدت زمان پردازش بیش از اندازه طول کشید و خروجی نتونستم بگیرم

سوال 7 : یافتن چرخه در گراف

```
list(nx.selfloop_edges(G1, keys=True, data=True))
```

خروجی : هیچ چرخه ای در گراف وجود ندارد