

Auditorne 1

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
#pip install matplotlib
```

```
#pip install networkx
```

```
#pip install numpy
```

```
import networkx as nx
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import random
```

```
# # Inicijalizacija mreže
```

```
# Inicijalizacija neusmjerene mreže
```

```
# In[3]:
```

```
G = nx.Graph()
```

```
# Dodavanje čvorova
```

```
# In[4]:
```

```
G.add_node(1)  
G.add_node(2)  
G.add_nodes_from([3,4,5])
```

```
# Dodavanje veza
```

```
# In[5]:
```

```
G.add_edge(1,2)  
G.add_edges_from([(3,4),(2,4),(4,5),(3,5),(1,5)])
```

```
# Ili kraće
```

```
# In[74]:
```

```
G_con = nx.Graph([(1,2),(3,4),(2,4),(4,5),(3,5),(1,5)])
```

```
# Dodavanje i pristupanje atributima čvorova
```

```
# In[7]:
```

```
nx.set_node_attributes(G, "vrijednost", "atribut")
print(nx.get_node_attributes(G, "atribut"))
```

```
# Pristupanje i dodavanje atributa pojedinom čvoru
```

```
# In[8]:
```

```
G.nodes[1]["atribut"] = "nova vrijednost"
G.nodes[2]["atribut 2"] = "druga vrijednost"
```

```
# Dodavanje atributa čvora rječnikom rječnika
```

```
# In[9]:
```

```
node_attributes_dictionary = {1 : {"Atribut 1" : "Vrijednost 11"},
                             2 : {"Atribut 1" : "Vrijednost 12", "Atribut 2" : "Vrijednost 22"},
                             3 : {"Atribut 2" : "Vrijednost 23", "Atribut 1" : "Vrijednost 13"},
                             4 : {"Atribut 3" : "Vrijednost 34"}}
```

```
nx.set_node_attributes(G, node_attributes_dictionary)
print(nx.get_node_attributes(G, "Atribut 1"))
```

```
# Dodavanje i pristupanje atributima vezama
```

```
# In[10]:
```

```
nx.set_edge_attributes(G, "vrijednost", "atribut")
print(nx.get_edge_attributes(G, "atribut"))
```

```
# Pristupanje i dodavanje atributa pojedinoj vezi
```

```
# In[11]:
```

```
G.edges[(1,2)]["atribut"] = "nova vrijednost"
```

```
# Dodavanje atributa veza rječnikom rječnika
```

```
# In[12]:
```

```
edge_attributes_dictionary = {(1,2) : {"Atribut 1" : "Vrijednost 112"},
(2,4) : {"Atribut 1" : "Vrijednost 122", "Atribut 2" : "Vrijednost 222"},
(4,5) : {"Atribut 2" : "Vrijednost 245", "Atribut 1" : "Vrijednost 145"},
(3,5) : {"Atribut 3" : "Vrijednost 335"}}
```

```
nx.set_edge_attributes(G, edge_attributes_dictionary)
```

```
print(nx.get_edge_attributes(G, "Atribut 1"))
```

```
# Crtanje mreže
```

```
# In[13]:
```

```
plt.title('Mreža G')
```

```
nx.draw_networkx(G)
```

```
# Podešavanje parametara crtanja
```

```
# In[14]:
```

```
pos = nx.spring_layout(G_con, k = 0.1, seed = 7)
```

```
node_labels = nx.get_node_attributes(G, 'atribut')
```

```
# In[15]:
```

```
plt.title('Mreža G_con')
```

```
nx.draw_networkx(G_con, labels = node_labels, pos = pos)
```

```
# Inicijalizacija usmjerene mreže
```

```
# In[16]:
```

```
D = nx.DiGraph([(1,2),(2,1),(4,3),(2,4),(4,5),(3,5),(1,5)])
```

```
plt.title('Usmjerena mreža')
```

```
nx.draw_networkx(D)
```

```
# Bipartitna mreža
```

```
# In[17]:
```

```
B = nx.complete_bipartite_graph(4,5)
```

```
plt.title('Bipartitna mreža')
```

```
nx.draw_networkx(B)
```

```
# Ciklička mreža
```

```
# In[18]:
```

```
C = nx.cycle_graph(4)
```

```
plt.title('Ciklička mreža')
```

```
nx.draw_networkx(C)
```

```
# Lanac
```

```
# In[19]:
```

```
P = nx.path_graph(5)
```

```
plt.title('Lanac')
```

```
nx.draw_networkx(P)
```

```
# Zvijezda
```

```
# In[20]:
```

```
S = nx.star_graph(6)
```

```
plt.title('Zvijezda')
```

```
nx.draw_networkx(S)
```

```
# Stablo
```

```
# In[21]:
```

```
T = nx.generators.balanced_tree(2, h = 3)

plt.title('Stablo')
nx.draw_networkx(T)

# Inicijalizacija mreže s težinama

# In[22]:


D_w = nx.DiGraph()
D_w.add_weighted_edges_from([(1,2,3),(2,1,4),(4,3,1),(2,4,2),(4,5,2),(3,5,6),(1,5,1)])


# Crtanje mreže s težinama

# In[23]:


pos = nx.spring_layout(D_w, seed=7)
labels = nx.get_edge_attributes(D_w,'weight')

plt.title('Mreža D_w')
nx.draw_networkx(D_w,pos)
nx.draw_networkx_edge_labels(D_w, pos = pos, edge_labels = labels)
```

```
# Prolazak po čvorovima i vezama
```

```
# In[24]:
```

```
print(D_w.number_of_nodes())
print(D_w.number_of_edges())
```

```
# In[25]:
```

```
print(D_w.nodes())
print(D_w.edges())
```

```
# In[26]:
```

```
for node in D_w.neighbors(4):
    print(node)
```

```
# In[27]:
```

```
for node in D_w.predecessors(4):
    print(node)
```

```
# In[28]:
```

```
for node in D_w.successors(4):
    print(node)
```

```
# Podmreže
```

```
# In[29]:
```

```
D_w_subgraph = nx.subgraph(D_w, (3,4,5))
D_w_subgraph_V2 = D_w.subgraph((3,4,5))
```

```
# In[30]:
```

```
plt.subplot(131)
plt.title('Cijela mreža')
pos = nx.spring_layout(D_w, seed = 7)
labels = nx.get_edge_attributes(D_w, 'weight')
nx.draw_networkx(D_w, pos)
nx.draw_networkx_edge_labels(D_w, pos, labels)
```

```
plt.subplot(132)
```

```
plt.title('Podmreža')
```

```
pos = nx.spring_layout(D_w_subgraph, seed = 7)
labels = nx.get_edge_attributes(D_w_subgraph, 'weight')
nx.draw_networkx(D_w_subgraph, pos)
nx.draw_networkx_edge_labels(D_w_subgraph, pos, labels)
```

```
plt.subplot(133)
plt.title('Podmreža V2')
pos = nx.spring_layout(D_w_subgraph_V2, seed = 7)
labels = nx.get_edge_attributes(D_w_subgraph_V2, 'weight')
nx.draw_networkx(D_w_subgraph_V2, pos)
nx.draw_networkx_edge_labels(D_w_subgraph_V2, pos, labels)
```

```
# Klike
```

```
# In[31]:
```

```
G_clique_example = nx.Graph([(1,2),(1,3),(2,3),(3,4),(4,5),(4,6),(5,6)])
```

```
plt.title('Mreža G_clique_example')
nx.draw_networkx(G_clique_example, pos = nx.spring_layout(G_clique_example, seed = 7))
```

```
# In[32]:
```

```
G_cliques = nx.find_cliques(G_clique_example)
```

```
clique_n = 0
clique_colors = ['red', 'green']
for clique in G_cliques:
    print(clique)
    if len(clique) > 2:
        nx.set_node_attributes(G_clique_example.subgraph(clique), clique_colors[clique_n], "color")
    clique_n+=1

plt.title('Klike s više od 2 člana')

nx.draw_networkx(G_clique_example, node_color = list(nx.get_node_attributes(G_clique_example,
"color").values()), pos = nx.spring_layout(G_clique_example, seed = 7))

# Gustoća

# In[33]:


print(f"Gustoća mreže G : {nx.density(G)}")
print(f"Gustoća mreže D_w : {nx.density(D_w)}")

# Stupanj čvora

# In[34]:


print(f"Stupanj čvora 4 mreže G : {G.degree(4)}")
```

```
print(f"Stupnjevi svih čvorova mreže G : {G.degree()}")


G_nodes_degrees = G.degree()
nx.set_node_attributes(G, dict(G_nodes_degrees), "degree")


plt.title('Mreža G s naznačenim stupnjevima')
nx.draw_networkx(G, labels = nx.get_node_attributes(G, "degree"), pos = nx.spring_layout(G, seed = 7))

# Stupanj čvora za usmjerene mreže

# In[35]:


print(f"Stupanj čvora 4 mreže D : {D.degree(4)}")
print(f"Ulagni stupanj čvora 4 mreže D : {D.in_degree(4)}")
print(f"Izlazni stupanj čvora 4 mreže D : {D.out_degree(4)}")
print(f"Stupnjevi svih čvorova mreže D : {D.degree()}")
print(f"Ulagni stupnjevi svih čvorova mreže D : {D.in_degree()}")
print(f"Izlazni stupnjevi svih čvorova mreže D : {D.out_degree()}")


D_nodes_degrees = D.in_degree()
nx.set_node_attributes(D, dict(D_nodes_degrees), "in_degree")


plt.title('Mreža D s naznačenim stupnjevima')
nx.draw_networkx(D, labels = nx.get_node_attributes(D, "in_degree"), pos = nx.spring_layout(D, seed = 7))
```

```
# Prosječni stupanj čvora
```

```
# In[36]:
```

```
total_nodes = D.number_of_nodes()
```

```
total_degree = 0
```

```
total_in_degree = 0
```

```
total_out_degree = 0
```

```
for (node, degree) in D.degree():
```

```
    total_degree += degree
```

```
for (node, in_degree) in D.in_degree():
```

```
    total_in_degree += in_degree
```

```
for (node, out_degree) in D.out_degree():
```

```
    total_out_degree += out_degree
```

```
print(f"Prosječni stupanj čvora : {total_degree/total_nodes}")
```

```
print(f"Prosječni ulazni stupanj čvora : {total_in_degree/total_nodes}")
```

```
print(f"Prosječni izlazni stupanj čvora : {total_out_degree/total_nodes}")
```

```
# Mrežna reprezentacija
```

```
# In[37]:
```

```
print(f"Lista susjedstva za mrežu G : \n {list(nx.generate_adjlist(G))}")
print(f"Matrica susjedstva za mrežu G : \n {nx.adjacency_matrix(G).todense()}")
print(f"Lista veza za mrežu G : \n {list(nx.generate_edgelist(G))}")
```

In[38]:

```
print(f"Lista susjedstva za mrežu D_w : \n {list(nx.generate_adjlist(D_w))}")
print(f"Matrica susjedstva za mrežu D_w : \n {nx.adjacency_matrix(D_w).todense()}")
print(f"Lista veza za mrežu D_w : \n {list(nx.generate_edgelist(D_w))}")
```

Spremanje i učitavanje mreža pomoću liste veza

In[39]:

```
nx.write_edgelist(D_w, 'D_w.edgelist')
with open('D_w.edgelist') as f:
    head =[next(f) for x in range(5)]
print(head)
D_w_copy = nx.read_edgelist('D_w.edgelist')
```

Primjeri mreža

```
# Dodatni primjeri dostupni na: <br>
# https://snap.stanford.edu/data/ <br>
# http://vlado.fmf.uni-lj.si/pub/networks/data/
```

```
# Članova karate kluba
```

```
# In[40]:
```

```
G_karate = nx.karate_club_graph()
print(f"Broj čvorova : {G_karate.number_of_nodes()}")
print(f"Broj rubova : {G_karate.number_of_edges()}")
```

```
# In[41]:
```

```
plt.title('Mreža članova karate kluba')
nx.draw_networkx(G_karate, node_size = [v * 100 for v in dict(G_karate.degree).values()], pos =
nx.spring_layout(G_karate, seed = 7))
```

```
# Mreža interakcije proteina
```

```
# In[42]:
```

```
G_protein = nx.read_edgelist('protein_interaction.edgelist')
d = dict(G_protein.degree)
```

```
print(f"Broj čvorova : {G_protein.number_of_nodes()}")  
print(f"Broj rubova : {G_protein.number_of_edges()}")  
  
# In[43]:  
  
plt.title('Mreža interakcije proteina')  
nx.draw_networkx(G_protein, node_size = [v*100 for v in d.values()], pos = nx.spring_layout(G_protein, seed = 7))  
  
# Peer-to-peer mreža  
  
# In[44]:  
  
G_network = nx.read_edgelist('p2p-Gnutella08.txt')  
print(f"Broj čvorova : {G_network.number_of_nodes()}")  
print(f"Broj rubova : {G_network.number_of_edges()}")  
  
# Provjera povezanosti mreže i dobivanje dobivanje najveće komponente  
  
# In[45]:  
  
print(f"Mreža je povezana? {nx.is_connected(G_network)}")
```

```
G_network_components = sorted([G_network.subgraph(c).copy() for c in
nx.connected_components(G_network)], key= len, reverse = True)

for (index, network_components) in enumerate(G_network_components):
    print(f"Komponenta indexa {index} je povezana? {nx.is_connected(G_network_components[index])}")
```

```
# In[46]:
```

```
plt.figure(figsize = (30,30))
plt.title('Velika komponenta P2P mreže')

nx.draw_networkx(G_network_components[0], node_size = [v*100 for v in
dict(G_network_components[0].degree).values()], pos = nx.spring_layout(G_network_components[0], k
= 10) , alpha = 0.05, with_labels = False)
```

```
# In[47]:
```

```
plt.title('Druga komponenta P2P mreže')

nx.draw_networkx(G_network_components[1], pos = nx.spring_layout(G_network_components[1], k =
0.1), node_size = 300)
```

```
# Asortativnost
```

```
# In[48]:
```

```
print(f"Asortativnost karate : {nx.degree_assortativity_coefficient(G_karate)}")  
print(f"Asortativnost proteini : {nx.degree_assortativity_coefficient(G_protein)}")  
print(f"Asortativnost P2P mreža : {nx.degree_assortativity_coefficient(G_network)}")
```

Prosječan najkraći put

In[49]:

```
print(f"Prosječan najkraći put karate : {nx.average_shortest_path_length(G_karate)}")  
print(f"Prosječan najkraći put proteini : {nx.average_shortest_path_length(G_protein)}")  
print(f"Prosječan najkraći put najveće komponente P2P mreže :  
{nx.average_shortest_path_length(G_network_components[0])}")
```

Dijametar

In[50]:

```
print(f"Dijametar karate : {nx.diameter(G_karate)}")  
print(f"Dijametar proteini : {nx.diameter(G_protein)}")  
print(f"Dijametar najveće komponente P2P mreže : {nx.diameter(G_network_components[0])}")
```

Koeficijent klasteriranja čvorova

In[51]:

```
print(f"Koeficijent klasteriranja čvorova karate : {nx.average_clustering(G_karate)}")  
print(f"Koeficijent klasteriranja čvorova protein : {nx.average_clustering(G_protein)}")  
print(f"Koeficijent klasteriranja čvorova mreža : {nx.average_clustering(G_network)}")  
  
# Breadth first search  
  
# Prolazak rubova BFS-om počevsi od čvora 0  
  
# In[52]:  
  
for edge in nx.bfs_edges(G_karate, 0):  
    print(edge)  
  
# Konstrukcija stabla pomoću BFS-a  
  
# In[53]:  
  
some = nx.bfs_tree(G_karate, 0)  
  
plt.title('Stablo konstruirano pomoću BFS-a')  
nx.draw_networkx(some)
```

```
# Distribucija stupnja
```

```
# In[54]:
```

```
def degree_calc(G):  
    degrees = [val for (node, val) in G.degree()]  
    avg_degree = np.mean(degrees)  
    med_degree = np.median(degrees)  
  
    return degrees, avg_degree, med_degree
```

```
# In[55]:
```

```
def degree_distribution_plot(degree_list, avg_degree, med_degree, cumulative, title):  
    plt.hist(degree_list,label='Distribucija stupnja', cumulative = cumulative)  
    plt.axvline(avg_degree,color='r',linestyle='dashed',label='Prosječni stupanj')  
    plt.axvline(med_degree,color='g',linestyle='dashed',label='Medijan stupanj')  
    plt.legend()  
    plt.ylabel('Postotak čvorova')  
    plt.xlabel('Iznos stupnja')  
    plt.title(title)
```

```
# In[56]:
```

```
d_list_karate, avg_d_karate, med_d_karate = degree_calc(G_karate)
d_list_protein, avg_d_protein, med_d_protein = degree_calc(G_protein)
d_list_mreza, avg_d_mreza, med_d_mreza = degree_calc(G_network)
```

```
# In[57]:
```

```
degree_distribution_plot(d_list_karate, avg_d_karate, med_d_karate, False, 'Karate')
```

```
# In[58]:
```

```
degree_distribution_plot(d_list_protein, avg_d_protein, med_d_protein, False, 'Proteini')
```

```
# In[59]:
```

```
degree_distribution_plot(d_list_mreza, avg_d_mreza, med_d_mreza, False, 'P2P mreža')
```

```
# Distribucija bliskosti
```

```
# In[60]:
```

```
def closeness_calc(G):
```

```
closeness = [val for (node, val) in nx.closeness_centrality(G).items()]
avg_closeness = np.mean(closeness)
med_closeness = np.median(closeness)

return closeness, avg_closeness, med_closeness
```

```
# In[61]:
```

```
def closeness_distribution_plot(closeness_list, avg_closeness, med_closeness, cumulative, title):
    plt.hist(closeness_list,label='Distribucija bliskosti', cumulative = cumulative)
    plt.axvline(avg_closeness, color='r' ,linestyle='dashed', label = "Prosječna bliskost")
    plt.axvline(med_closeness, color='g' ,linestyle='dashed', label = "Medijan bliskosti")
    plt.legend()
    plt.ylabel('Postotak čvorova')
    plt.xlabel('Iznost bliskosti')
    plt.title(title)
```

```
# In[62]:
```

```
c_list_karate, avg_c_karate, med_c_karate = closeness_calc(G_karate)
c_list_protein, avg_c_protein, med_c_protein = closeness_calc(G_protein)
c_list_mreza, avg_c_mreza, med_c_mreza = closeness_calc(G_network)
```

```
# In[63]:
```

```
closeness_distribution_plot(c_list_karate, avg_c_karate, med_c_karate, False, 'Karate')
```

```
# In[64]:
```

```
closeness_distribution_plot(c_list_protein, avg_c_protein, med_c_protein, False, 'Proteini')
```

```
# In[65]:
```

```
closeness_distribution_plot(c_list_mreza, avg_c_mreza, med_c_mreza, False, 'P2P mreža')
```

```
# Distribucija međupoloženosti
```

```
# In[66]:
```

```
def calc_betweenness(G):
```

```
    betweenness = [val for (node, val) in nx.betweenness_centrality(G).items()]
```

```
    avg_betweenness = np.mean(betweenness)
```

```
    med_betweenness = np.median(betweenness)
```

```
    return betweenness, avg_betweenness, med_betweenness
```

```
# In[67]:
```

```
def betweenness_distribution_plot(betweenness_list, avg_betweenness, med_betweenness,
cumulative, title):

    plt.hist(betweenness_list,label='Distribucija međupoloženosti', cumulative = cumulative)
    plt.axvline(avg_betweenness,color='r',linestyle='dashed',label='Prosječna međupoloženost')
    plt.axvline(med_betweenness,color='g',linestyle='dashed',label='Medijan međupoloženosti')
    plt.legend()
    plt.ylabel('Postotak čvorova')
    plt.xlabel('Iznos međupoloženosti')
    plt.title(title)
```

```
# In[68]:
```

```
b_list_karate, avg_b_karate, med_b_karate = calc_betweenness(G_karate)
b_list_protein, avg_b_protein, med_b_protein = calc_betweenness(G_protein)
b_list_mreza, avg_b_mreza, med_b_mreza = calc_betweenness(G_network)
```

```
# In[69]:
```

```
betweenness_distribution_plot(b_list_karate, avg_b_karate, med_b_karate, False, 'Karate')
```

```
# In[70]:
```

```
betweenness_distribution_plot(b_list_protein, avg_b_protein, med_b_protein, False, 'Proteini')
```

```
# In[71]:
```

```
betweenness_distribution_plot(b_list_mreza, avg_b_mreza, med_b_mreza, False, 'P2P mreža')
```

```
# Heterogenost
```

```
# In[72]:
```

```
def calc_heterogeneity(g):  
    average_squared_degree = sum([degree**2 for (node, degree) in g.degree()])/g.number_of_nodes()  
    average_degree = sum(degree for (node, degree) in g.degree())/g.number_of_nodes()  
  
    heterogeneity = average_squared_degree/(average_degree**2)  
  
    return heterogeneity
```

```
# In[73]:
```

```
print(f"Karate heterogenost : {calc_heterogeneity(G_karate)}")  
print(f"Proteini heterogenost : {calc_heterogeneity(G_protein)}")  
print(f"P2P mreža heterogenost : {calc_heterogeneity(G_network)}")
```

Auditorne 2

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

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

```
# Robusnost
```

```
# In[2]:
```

```
g_karate = nx.karate_club_graph()  
g_protein = nx.read_edgelist('protein_interaction.edgelist')
```

```
# In[3]:
```

```
def get_giant_component_size(g):  
    return len(max(nx.connected_components(g), key = len))
```

```
# In[4]:
```

```
def failure(g, n_steps):

    c = g.copy()

    n_nodes_start = c.number_of_nodes()

    n_nodes_to_remove = max(n_nodes_start//n_steps, 1)

    #n_nodes_to_remove = np.ceil(n_node_start/n_steps)

    relative_giant_component_size = []

    relative_giant_component_size.append(1)

    relative_n_nodes_removed = []

    relative_n_nodes_removed.append(0)

    for step in range(1, n_steps + 1):

        if c.number_of_nodes() > n_nodes_to_remove:

            nodes_to_leave = random.sample(list(c.nodes), c.number_of_nodes() - n_nodes_to_remove)

            c = nx.subgraph(c, nodes_to_leave)

            giant_component_size = get_giant_component_size(c)

            relative_giant_component_size.append(giant_component_size/n_nodes_start)

            relative_n_nodes_removed.append(1 - (c.number_of_nodes()/n_nodes_start))

        else:

            relative_giant_component_size.append(0)
```

```
relative_n_nodes_removed.append(1)

return relative_giant_component_size, relative_n_nodes_removed

# In[5]:


def attack(g, n_steps):

    c = g.copy()

    n_nodes_start = c.number_of_nodes()

    n_nodes_to_remove = max(n_nodes_start//n_steps, 1)
    #n_nodes_to_remove = np.ceil(n_node_start/n_steps)

    relative_giant_component_size = []
    relative_giant_component_size.append(1)

    relative_n_nodes_removed = []
    relative_n_nodes_removed.append(0)

    for step in range(1, n_steps + 1):
        if c.number_of_nodes() > n_nodes_to_remove:
            nodes_to_leave = sorted(c.nodes, key = c.degree, reverse = True)[n_nodes_to_remove:]
            c = nx.subgraph(c, nodes_to_leave)

            giant_component_size = get_giant_component_size(c)

            relative_giant_component_size.append(giant_component_size/n_nodes_start)
```

```
relative_n_nodes_removed.append(1- (c.number_of_nodes()/n_nodes_start))

else:
    relative_giant_component_size.append(0)
    relative_n_nodes_removed.append(1)

return relative_giant_component_size, relative_n_nodes_removed
```

```
# In[6]:
```

```
def plot_comparison(n_nodes_removed, giant_component_failure, giant_component_attack):
    plt.plot(n_nodes_removed, giant_component_failure, color = 'b', label = "Kvar")
    plt.plot(n_nodes_removed, giant_component_attack, color = 'r', label = "Napad")
    plt.legend()
    plt.ylabel("Veličina gigantske komponente")
    plt.xlabel("Uklonjenih čvorova")
```

```
# In[7]:
```

```
giant_component_failure_karate, n_nodes_removed_karate = failure(g_karate, 30)
giant_component_attack_karate, g_nodes_removed_karate = attack(g_karate, 30)
```

```
# In[8]:
```

```
plot_comparison(g_nodes_removed_karate, giant_component_failure_karate,  
giant_component_attack_karate)
```

```
# In[9]:
```

```
giant_component_failure_protein, n_nodes_removed_protein = failure(g_protein, 30)  
giant_component_attack_protein, n_nodes_removed_protein = attack(g_protein, 30)
```

```
# In[10]:
```

```
plot_comparison(n_nodes_removed_protein, giant_component_failure_protein,  
giant_component_attack_protein)
```

```
# K-jezgrena dekompozicija
```

```
# In[11]:
```

```
max_core = max(list(nx.core_number(g_karate).values()))
```

```
# In[12]:
```

```
k_core_dict = {}  
k_shell_dict = {}  
  
for core in range(0, max_core + 2):  
    k_core_dict[core] = nx.k_core(g_karate, core)  
    k_shell_dict[core] = nx.k_shell(g_karate, core)
```

```
# In[13]:
```

```
pos = nx.spring_layout(g_karate, seed = 2)
```

```
plt.figure(figsize = (30, 30))  
plt.subplot(321)  
plt.title('0-jezgra')  
nx.draw_networkx(k_core_dict[0])
```

```
plt.subplot(322)  
plt.title('1-jezgra')  
nx.draw_networkx(k_core_dict[1])
```

```
plt.subplot(323)  
plt.title('2-jezgra')  
nx.draw_networkx(k_core_dict[2])
```

```
plt.subplot(324)  
plt.title('3-jezgra')
```

```
nx.draw_networkx(k_core_dict[3])
```

```
plt.subplot(325)
plt.title('4-jezgra')
nx.draw_networkx(k_core_dict[4])
```

```
plt.subplot(326)
plt.title('5-jezgra')
nx.draw_networkx(k_core_dict[5])
```

```
# In[14]:
```

```
plt.figure(figsize = (30, 30))
plt.subplot(321)
plt.title('0-ljuska')
nx.draw_networkx(k_shell_dict[0])
```

```
plt.subplot(322)
plt.title('1-ljuska')
nx.draw_networkx(k_shell_dict[1])
```

```
plt.subplot(323)
plt.title('2-ljuska')
nx.draw_networkx(k_shell_dict[2])
```

```
plt.subplot(324)
plt.title('3-ljuska')
```

```
nx.draw_networkx(k_shell_dict[3])
```

```
plt.subplot(325)
plt.title('4-ljuska')
nx.draw_networkx(k_shell_dict[4])
```

```
plt.subplot(326)
plt.title('5-ljuska')
nx.draw_networkx(k_shell_dict[5])
```

2.1. USMJERENE MREŽE SOCIJALNIH INTERAKCIJA

```
# U drugom djelu ove pokazne vježbe koristimo znanstvene Twitter podatke za stvaranje i istraživanje usmjerenih mreža društvenih interakcija.
```

```
# ## DATASET:
```

```
# In[15]:
```

```
import json
search_tweets = json.load(open('science_tweets.json'))
```

```
# Svaki tweet zapravo je jedna instanca Tweet objekta (https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet)
```

```
# In[16]:
```

```
search_tweets[:2]
```

```
# # Twitter retweetanje
```

```
# Temeljna interakcija u ekosustavu Twittera je "retweet" -- ponovno emitiranje tweeta drugog korisnika  
vašim pratiteljima.
```

```
# ## Filtriranje retweetova
```

```
# U našem skupu podataka nalaze se retweetovi. Objekt tweeta koji se nalazi u našem datasetu je  
retweet ako uključuje 'retweeted_status'. Napraviti ćemo novi skup podataka koji će se sastojati samo od  
retweetova:
```

```
# In[17]:
```

```
retweets = []
```

```
for tweet in search_tweets:
```

```
    if 'retweeted_status' in tweet:
```

```
        retweets.append(tweet)
```

```
len(retweets)
```

```
# ## Izrada DiGrafa
```

```
# Prikazat ćemo tweetove na ovom popisu retweetova u smjeru protoka informacija: od korisnika koji je  
retweetao do retweetara, korisnika čija je objava retweetana. Budući da korisnik može retweetati objave
```

drugog korisnika više puta, želimo da ovaj graf bude težinski, s brojem retweeta kao težinom - brojimo koliko je puta neki korisnik retweetao objave nekog drugog korisnika.

```
# In[18]:
```

```
import networkx as nx
```

```
D = nx.DiGraph() #inicijalizacija usmjerenog grafa
```

```
for retweet in retweets:
```

```
    retweeted_status = retweet['retweeted_status']
```

```
    retweeted_sn = retweeted_status['user']['screen_name'] #ime korisnika koji je retweetao
```

```
    retweeter_sn = retweet['user']['screen_name'] #ime ciji je tweet retweetan
```

```
    if D.has_edge(retweeted_sn, retweeter_sn):
```

```
        D.edges[retweeted_sn, retweeter_sn]['weight'] += 1
```

```
    else:
```

```
        D.add_edge(retweeted_sn, retweeter_sn, weight=1)
```

```
# In[19]:
```

```
D.edges
```

```
# Logika dodavanja bridova je povećati težinu brida za 1 ako brid postoji ili stvoriti brid s težinom 1 ako ne postoji.  
#  
# Kada pišete kod kao što je ovaj koji se više puta referira na isti usmjereni brid, pazite da budete u skladu  
sa smjerom brida.  
  
# ## Analiza grafa
```

```
# ### Najviše retweetani korisnik
```

```
# Budući da su bridovi u smjeru protoka informacija, out-degree nam daje broj korisnika koji retweetaju  
određenog korisnika. Možemo dobiti korisnika s najvišim stupnjem izlaza pomoću ugrađene max  
funkcije (korisnika čije se objave najviše retweetaju):
```

```
# In[20]:
```

```
max(D.nodes, key=D.out_degree)
```

```
# ali možemo dobiti i više informacija za "najboljih N" korisnika:
```

```
# In[21]:
```

```
from operator import itemgetter
```

```
sorted(D.out_degree(), key=itemgetter(1), reverse=True)[:5]
```

```
# U ovom dijelu koda koristimo činjenicu da D.out_degree() vraća niz (ime, stupanj) tuplova;  
key=itemgetter(1) govori sortiranoj funkciji da sortira ove tuplove prema njihovoj vrijednosti na indeksu  
1. Postavljanje reverse=True govori sortiranoj funkciji da to želimo u silaznom redoslijedu, a [:5] daje nam  
prvih 5 stavki s rezultirajuće liste.
```

```
#
```

```
# Međutim, ovo je težinski graf! Prema zadanim postavkama, out_degree() zanemaruje težine rubova.  
Možemo dobiti izlaznu težinu tako da kažemo funkciji out_degree() da uzme u obzir težinu bridova:
```

```
# In[22]:
```

```
sorted(D.out_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# U nekim će slučajevima ova dva rezultata biti ista, npr. ako niti jednog od ovih korisnika nije više puta  
retweetao isti korisnik. Ovisno o vašem slučaju upotrebe, možete ili ne morate uzeti težine u obzir.
```

```
# ### Detekcija anomalija
```

```
# Jedna vrsta manipulacije društvenih medija uključuje račune koji stvaraju vrlo malo originalnog  
sadržaja, umjesto toga "spammaju" retweetove svih sadržaja u određenom razgovoru. To su potencijalno  
korisnici koji puno više retweetaju od ostalih. Možemo li otkriti da neki korisnici znatno više retweetaju  
od ostalih? Pogledajmo N korisnika koji najčešće retweetaju:
```

```
# In[23]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# ### Povezanost
```

```
# Možemo se pitati predstavljaju li tweetovi jedan veliki razgovor ili mnogo malih razgovora; općenito govoreći, svaka slabo povezana komponenta predstavlja razgovor.
```

```
# In[24]:
```

```
nx.is_weakly_connected(D)
```

```
# Tweetovi definitivno ne predstavljaju jedan veliki razgovor, no ono što možemo očekivati je da postoji velik broj malih razgovora. Pogledajmo koliko:
```

```
# In[25]:
```

```
nx.number_weakly_connected_components(D)
```

```
# #### Crtanje grafa
```

```
# Možemo pokušati nacrtati ovaj graf s čvorovima veličine prema njihovoj izlaznoj snazi:
```

```
# In[26]:
```

```
node_sizes = [D.out_degree(n, weight='weight') * 50 for n in D.nodes] # množimo s 50 da bi nam čvorovi na slici izgledali veće
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
nx.draw(D, node_size=node_sizes)
```

```
# Imajte na umu da u ovom pojednostavljenom crtežu čvorovi s nultom vanjskom težinom nisu nacrtani na grafu jer je njihova veličina 0. To nam odgovara; ovdje su izvučeni samo korisnici koji su retweetani, ne i oni čije objave nitko nikad nije retweetao.
```

```
# # Twitter spominjanja
```

```
# Druga Twitter interakcija između korisnika događa se kada jedan korisnik spomene drugog u tweetu pod svojim @screen_name. Kao primjer, razmotrite sljedeći hipotetski tweet od @osome_iu:
```

```
#
```

```
# "Check out the new @IUSICE and @USC_ISI research https://..."
```

```
#
```

```
# Od ovog tweeta stvorili bismo dva brida:
```

```
#
```

```
# ('osome_iu', 'IUSICE')
```

```
# ('osome_iu', 'USC_ISI')
```

```
#
```

```
# Na nama je u kojem ćemo smjeru povući te rubove, ali moramo biti dosljedni. U ovom primjeru nacrtat ćemo rubove u smjeru toka pozornosti: @osome_iu posvećuje pozornost @IUSICE i @USC_ISI.
```

```
# ## Izrada DiGrafa
```

```
# Kao što smo na početku spomenuli, svaki tweet predstavljen je značajkom Tweet Object i svaki tweet ima Entitete (https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/entities#entitiesobject) koji uvijek sadržavaju popis 'user_mentions' pa čak i kad je taj popis prazan. Zbog toga nije potrebno filtrirati tweetove koji sadrže spominjanja.
```

```
# In[ ]:
```

```
import networkx as nx

D = nx.DiGraph()

for tweet in search_tweets:

    tweet_sn = tweet['user']['screen_name']

    for user_mention in tweet['entities']['user_mentions']:

        mentioned_sn = user_mention['screen_name']

        edge = (tweet_sn, mentioned_sn)

        if D.has_edge(*edge):

            D.edges[edge]['weight'] += 1

        else:

            D.add_edge(*edge, weight=1)

D.edges

# ## Analiza grafa

# ### Najpopularniji korisnici

# Budući da su ti bridovi u smjeru protoka pažnje, in-degree nam daje broj drugih korisnika koji spominju određenog korisnika. Možemo dobiti korisnika s najvišim stupnjem pomoću ugrađene max funkcije - korisnika koji se najčešće spominje od strane drugih:

# In[ ]:
```

```
max(D.nodes, key=D.in_degree)
```

ali opet možemo dobiti i više informacija za "najboljih N" korisnika - korisnika koji se najčešće spominju:

```
# In[ ]:
```

```
from operator import itemgetter
```

```
sorted(D.in_degree(), key=itemgetter(1), reverse=True)[:5]
```

Korištenjem weight='weight' možemo dobiti prvih 5 korisnika prema ulaznoj težini umjesto prema ulaznom stupnju:

```
# In[ ]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

U nekim će slučajevima ova dva rezultata biti ista,npr. ako nijednog od ovih korisnika nije više puta spomenuo isti korisnik. Ovisno o vašem slučaju upotrebe, možete ili ne morate uzeti težine u obzir.

```
# ### Pokretači razgovora - Conversation drivers
```

Korisnik koji spominje mnoge druge u razgovoru možda "pokreće" razgovor i pokušava uključiti druge u dijalog. Takav korisnik može biti i spam. Da vidimo tko ovdje najviše spominje - ovdje gledamo vrijednost out degree:

```
# In[ ]:
```

```
sorted(D.out_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# ### Povezanost
```

```
# Možemo pitati predstavljaju li tweetovi dobiveni pretraživanjem jedan veliki razgovor ili mnogo malih razgovora; općenito govoreći, svaka slabo povezana komponenta predstavlja razgovor.
```

```
# In[ ]:
```

```
nx.is_weakly_connected(D)
```

```
# In[ ]:
```

```
nx.number_weakly_connected_components(D)
```

```
# ### Crtanje grafa
```

```
# In[ ]:
```

```
node_sizes= [D.in_degree(n, weight='weight') * 20 for n in D.nodes]
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
nx.draw(D, node_size=node_sizes)
```

Kao i u prethodnom primjeru, u ovom pojednostavljenom crtežu čvorovi s nultom vanjskom težinom nisu nacrtani na grafu jer je njihova veličina 0. To nam odgovara; ovdje su izvučeni samo korisnici koji su bili nekad spomenuti od strane drugih korisnika.

2.2. PAGE RANK ALGORITAM

PageRank je algoritam za izračunavanje mjere centralnosti koja ima za cilj uhvatiti važnost svakog čvora. Obično se koristi u usmjerenim grafovima (mrežama). Kada se primijeni na webu, algoritam svakoj stranici dodjeljuje PageRank vrijednost. Algoritam za rangiranje tražilice tada može koristiti ovu vrijednost, u kombinaciji s mnogim drugim čimbenicima — kao što je podudaranje između upita i teksta stranice — za sortiranje rezultata upita. Stranica s visokim PageRank-om smatra se važnom, a algoritam za rangiranje joj daje prednost: ako su ostale stvari iste, stranice s većim PageRank-om rangirane su više.

```
# Učitati ćemo novi dataset kao DiGraf: math Wikipedia dataset:
```

```
# In[ ]:
```

```
D = nx.read_graphml('enwiki_math.graphml')
```

```
# In[ ]:
```

```
len(D) # broj čvorova -> isto kao da pise len(D.nodes)
```

```
# In[ ]:
```

```
sorted(D.degree, key=lambda x: x[1], reverse=True)[:5]
```

```
# Nad učitanim datasetom pokrenut ćemo PageRank algoritam i izračunati PageRank za svaki od članaka:
```

```
# In[ ]:
```

```
pr = nx.pagerank(D, alpha=0.85)
```

```
# Zanima nas kojih je top 10 članaka po izračunatom PageRank-u:
```

```
# In[ ]:
```

```
sorted(pr, key=itemgetter(1), reverse=True)[:10]
```

```
# Želimo usporediti top 10 članaka po PageRanku s top 10 članaka po in degree-u. Hoće li to biti isti članci?
```

```
# In[ ]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:10]
```

Distribucija PageRanka je prilično slična distribuciji in-degree-a na webu. Zašto onda jednostavno ne upotrijebite in-degree za rangiranje? Moramo uzeti u obzir da nisu sve staze jednake. Putevi sa stranica koje se često posjećuju daju veći doprinos. Drugim riječima, na važnost stranice utječe važnost stranica koje povezuju na nju.

```
# In[ ]:
```

Auditorne 3

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# # 2.1. USMJERENE MREŽE SOCIJALNIH INTERAKCIJA
```

U drugom djelu ove pokazne vježbe koristimo znanstvene Twitter podatke za stvaranje i istraživanje usmjerenih mreža društvenih interakcija.

```
# ## DATASET:
```

```
# In[69]:
```

```
import json  
search_tweets = json.load(open('science_tweets.json'))
```

```
# Svaki tweet zapravo je jedna instanca Tweet objekta (https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet)
```

```
# In[70]:
```

```
search_tweets[:2]
```

```
# # Twitter retweetanje
```

```
# Temeljna interakcija u ekosustavu Twittera je "retweet" -- ponovno emitiranje tweeta drugog korisnika  
vašim pratiteljima.
```

```
# ## Filtriranje retweetova
```

```
# U našem skupu podataka nalaze se retweetovi. Objekt tweeta koji se nalazi u našem datasetu je  
retweet ako uključuje 'retweeted_status'. Napraviti ćemo novi skup podataka koji će se sastojati samo od  
retweetova:
```

```
# In[71]:
```

```
retweets = []
```

```
for tweet in search_tweets:  
    if 'retweeted_status' in tweet:  
        retweets.append(tweet)  
len(retweets)
```

```
# ## Izrada DiGrafa
```

```
# Prikazat ćemo tweetove na ovom popisu retweetova u smjeru protoka informacija: od korisnika koji je retweetao do retweetara, korisnika čija je objava retweetana. Budući da korisnik može retweetati objave drugog korisnika više puta, želimo da ovaj graf bude težinski, s brojem retweeta kao težinom - brojimo koliko je puta neki korisnik retweetao objave nekog drugog korisnika.
```

```
# In[72]:
```

```
import networkx as nx
```

```
D = nx.DiGraph() #inicijalizacija usmjerjenog grafa
```

```
for retweet in retweets:
```

```
   retweeted_status = retweet['retweeted_status']
```

```
   retweeted_sn = retweeted_status['user']['screen_name'] #ime korisnika koji je retweetao
```

```
   retweeter_sn = retweet['user']['screen_name'] #ime ciji je tweet retweetan
```

```
if D.has_edge(retweeted_sn, retweeter_sn):
```

```
    D.edges[retweeted_sn, retweeter_sn]['weight'] += 1
```

```
else:
```

```
    D.add_edge(retweeted_sn, retweeter_sn, weight=1)
```

```
# In[73]:
```

```
D.edges
```

```
# Logika dodavanja bridova je povećati težinu brida za 1 ako brid postoji ili stvoriti brid s težinom 1 ako ne postoji.  
#  
# Kada pišete kod kao što je ovaj koji se više puta referira na isti usmjereni brid, pazite da budete u skladu sa smjerom brida.
```

```
# ## Analiza grafa
```

```
# #### Najviše retweetani korisnik
```

```
# Budući da su bridovi u smjeru protoka informacija, out-degree nam daje broj korisnika koji retweetaju određenog korisnika. Možemo dobiti korisnika s najvišim stupnjem izlaza pomoću ugrađene max funkcije (korisnika čije se objave najviše retweetaju):
```

```
# In[74]:
```

```
max(D.nodes, key=D.out_degree)
```

```
# ali možemo dobiti i više informacija za "najboljih N" korisnika:
```

```
# In[75]:
```

```
from operator import itemgetter
```

```
sorted(D.out_degree(), key=itemgetter(1), reverse=True)[:5]
```

```
# U ovom dijelu koda koristimo činjenicu da D.out_degree() vraća niz (ime, stupanj) tuplova;  
key=itemgetter(1) govori sortiranoj funkciji da sortira ove tuplove prema njihovoj vrijednosti na indeksu  
1. Postavljanje reverse=True govori sortiranoj funkciji da to želimo u silaznom redoslijedu, a [:5] daje nam  
prvih 5 stavki s rezultirajuće liste.
```

```
#
```

```
# Međutim, ovo je težinski graf! Prema zadanim postavkama, out_degree() zanemaruje težine rubova.  
Možemo dobiti izlaznu težinu tako da kažemo funkciji out_degree() da uzme u obzir težinu bridova:
```

```
# In[76]:
```

```
sorted(D.out_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# U nekim će slučajevima ova dva rezultata biti ista, npr. ako niti jednog od ovih korisnika nije više puta  
retweetao isti korisnik. Ovisno o vašem slučaju upotrebe, možete ili ne morate uzeti težine u obzir.
```

```
# #### Detekcija anomalija
```

```
# Jedna vrsta manipulacije društvenih medija uključuje račune koji stvaraju vrlo malo originalnog  
sadržaja, umjesto toga "spammaju" retweetove svih sadržaja u određenom razgovoru. To su potencijalno  
korisnici koji puno više retweetaju od ostalih. Možemo li otkriti da neki korisnici znatno više retweetaju  
od ostalih? Pogledajmo N korisnika koji najčešće retweetaju:
```

```
# In[77]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# ### Povezanost
```

```
# Možemo se pitati predstavljaju li tweetovi jedan veliki razgovor ili mnogo malih razgovora; općenito  
govoreći, svaka slabo povezana komponenta predstavlja razgovor.
```

```
# In[78]:
```

```
nx.is_weakly_connected(D)
```

```
# Tweetovi definitivno ne predstavljaju jedan veliki razgovor, no ono što možemo očekivati je da postoji  
velik broj malih razgovora. Pogledajmo koliko:
```

```
# In[79]:
```

```
nx.number_weakly_connected_components(D)
```

```
# ### Crtanje grafa
```

```
# Možemo pokušati nacrtati ovaj graf s čvorovima veličine prema njihovoј izlaznoј snazi:
```

```
# In[80]:
```

```
node_sizes = [D.out_degree(n, weight='weight') * 50 for n in D.nodes] # množimo s 50 da bi nam čvorovi  
na slici izgledali veće
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
nx.draw(D, node_size=node_sizes)
```

Imajte na umu da u ovom pojednostavljenom crtežu čvorovi s nultom vanjskom težinom nisu nacrtani na grafu jer je njihova veličina 0. To nam odgovara; ovdje su izvučeni samo korisnici koji su retweetani, ne i oni čije objave nitko nikad nije retweetao.

```
# # Twitter spominjanja
```

Druga Twitter interakcija između korisnika događa se kada jedan korisnik spomene drugog u tweetu pod svojim @screen_name. Kao primjer, razmotrite sljedeći hipotetski tweet od @osome_iu:

```
# "Check out the new @IUSICE and @USC_ISI research https://..."
```

```
#
```

Od ovog tweeta stvorili bismo dva brida:

```
#
```

```
# ('osome_iu', 'IUSICE')
```

```
# ('osome_iu', 'USC_ISI')
```

```
#
```

Na nama je u kojem ćemo smjeru povući te rubove, ali moramo biti dosljedni. U ovom primjeru nacrtat ćemo rubove u smjeru toka pozornosti: @osome_iu posvećuje pozornost @IUSICE i @USC_ISI.

```
# ## Izrada DiGrafa
```

Kao što smo na početku spomenuli, svaki tweet predstavljen je značajkom Tweet Object i svaki tweet ima Entitete (<https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/entities#entitiesobject>) koji uvijek sadržavaju popis 'user_mentions' pa čak i kad je taj popis prazan. Zbog toga nije potrebno filtrirati tweetove koji sadrže spominjanja.

```
# In[81]:
```

```
import networkx as nx

D = nx.DiGraph()

for tweet in search_tweets:
    tweet_sn = tweet['user']['screen_name']
    for user_mention in tweet['entities']['user_mentions']:
        mentioned_sn = user_mention['screen_name']

        edge = (tweet_sn, mentioned_sn)
        if D.has_edge(*edge):
            D.edges[edge]['weight'] += 1
        else:
            D.add_edge(*edge, weight=1)

D.edges

# ## Analiza grafa

# ### Najpopularniji korisnici

# Budući da su ti bridovi u smjeru protoka pažnje, in-degree nam daje broj drugih korisnika koji spominju određenog korisnika. Možemo dobiti korisnika s najvišim stupnjem pomoću ugrađene max funkcije - korisnika koji se najčešće spominje od strane drugih:

# In[82]:
```

```
max(D.nodes, key=D.in_degree)
```

ali opet možemo dobiti i više informacija za "najboljih N" korisnika - korisnika koji se najčešće spominju:

```
# In[83]:
```

```
from operator import itemgetter
```

```
sorted(D.in_degree(), key=itemgetter(1), reverse=True)[:5]
```

Korištenjem weight='weight' možemo dobiti prvih 5 korisnika prema ulaznoj težini umjesto prema ulaznom stupnju:

```
# In[84]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

U nekim će slučajevima ova dva rezultata biti ista,npr. ako nijednog od ovih korisnika nije više puta spomenuo isti korisnik. Ovisno o vašem slučaju upotrebe, možete ili ne morate uzeti težine u obzir.

```
# ### Pokretači razgovora - Conversation drivers
```

```
# Korisnik koji spominje mnoge druge u razgovoru možda "pokreće" razgovor i pokušava uključiti druge u dijalog. Takav korisnik može biti i spam. Da vidimo tko ovdje najviše spominje - ovdje gledamo vrijednost out degree:
```

```
# In[85]:
```

```
sorted(D.out_degree(weight='weight'), key=itemgetter(1), reverse=True)[:5]
```

```
# ### Povezanost
```

```
# Možemo pitati predstavljaju li tweetovi dobiveni pretraživanjem jedan veliki razgovor ili mnogo malih razgovora; općenito govoreći, svaka slabo povezana komponenta predstavlja razgovor.
```

```
# In[86]:
```

```
nx.is_weakly_connected(D)
```

```
# In[87]:
```

```
nx.number_weakly_connected_components(D)
```

```
# ### Crtanje grafa
```

```
# In[88]:
```

```
node_sizes= [D.in_degree(n, weight='weight') * 20 for n in D.nodes]
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
nx.draw(D, node_size=node_sizes)
```

Kao i u prethodnom primjeru, u ovom pojednostavljenom crtežu čvorovi s nultom vanjskom težinom nisu nacrtani na grafu jer je njihova veličina 0. To nam odgovara; ovdje su izvučeni samo korisnici koji su bili nekad spomenuti od strane drugih korisnika.

2.2. PAGE RANK ALGORITAM

PageRank je algoritam za izračunavanje mjere centralnosti koja ima za cilj uhvatiti važnost svakog čvora. Obično se koristi u usmjerenim grafovima (mrežama). Kada se primijeni na webu, algoritam svakoj stranici dodjeljuje PageRank vrijednost. Algoritam za rangiranje tražilice tada može koristiti ovu vrijednost, u kombinaciji s mnogim drugim čimbenicima — kao što je podudaranje između upita i teksta stranice — za sortiranje rezultata upita. Stranica s visokim PageRank-om smatra se važnom, a algoritam za rangiranje joj daje prednost: ako su ostale stvari iste, stranice s većim PageRank-om rangirane su više.

```
# Učitat ćemo novi dataset kao DiGraf: math Wikipedia dataset:
```

```
# In[89]:
```

```
D = nx.read_graphml('enwiki_math.graphml')
```

```
# In[90]:
```

```
len(D) # broj čvorova -> isto kao da pise len(D.nodes)
```

```
# In[91]:
```

```
sorted(D.degree, key=lambda x: x[1], reverse=True)[:5]
```

```
# Nad učitanim datasetom pokrenut ćemo PageRank algoritam i izračunati PageRank za svaki od članaka:
```

```
# In[92]:
```

```
pr = nx.pagerank(D, alpha=0.85)
```

```
# Zanima nas kojih je top 10 članaka po izračunatom PageRank-u:
```

```
# In[93]:
```

```
sorted(pr, key=itemgetter(1), reverse=True)[:10]
```

```
# Želimo usporediti top 10 članaka po PageRanku s top 10 članaka po in degree-u. Hoće li to biti isti članci?
```

```
# In[94]:
```

```
sorted(D.in_degree(weight='weight'), key=itemgetter(1), reverse=True)[:10]
```

Distribucija PageRanka je prilično slična distribuciji in-degree-a na webu. Zašto onda jednostavno ne upotrijebite in-degree za rangiranje? Moramo uzeti u obzir da nisu sve staze jednake. Putevi sa stranica koje se često posjećuju daju veći doprinos. Drugim riječima, na važnost stranice utječe važnost stranica koje povezuju na nju.

```
# In[94]:
```

Auditorne 4

```
#!/usr/bin/env python  
# coding: utf-8  
  
## LAB 4 - complex networks FER3 (community edition)
```

```
# In[43]:
```

```
import networkx as nx  
import random  
  
get_ipython().run_line_magic('matplotlib', 'inline')  
  
## 3.1 Synthetic example analysis (with modularity)
```

```
# In[44]:
```

```
G = nx.barbell_graph(5, 0)
```

```
# In[45]:
```

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

```
# In[46]:
```

```
test_graph = nx.Graph()
```

```
# In[47]:
```

```
nx.add_cycle(test_graph, [0, 1, 2, 3, 4, 5])
nx.add_cycle(test_graph, [6, 7, 8, 9])
nx.add_cycle(test_graph, [10, 11, 12, 13])
nx.add_cycle(test_graph, [14, 15, 16, 17])
```

```
# In[48]:
```

```
test_graph.add_edge(0, 7)
test_graph.add_edge(0, 11)
test_graph.add_edge(0, 14)
```

In[49]:

```
test_graph.add_edge(13, 14)
```

In[50]:

```
nx.draw(test_graph, with_labels=True)
```

In[51]:

```
partition = [{0, 1, 2, 3, 4, 5}, {6, 7, 8, 9}, {10, 11, 12, 13}, {14, 15, 16, 17}]
```

In[52]:

```
nx.community.is_partition(test_graph, partition)
```

```
# In[53]:
```

```
partition_map = {}  
for idx, cluster_nodes in enumerate(partition):  
    for node in cluster_nodes:  
        partition_map[node] = idx  
  
partition_map
```

```
# In[54]:
```

```
partition_map[0] == partition_map[15]
```

```
# In[55]:
```

```
partition_map[0] == partition_map[1]
```

```
# In[56]:
```

```
node_colors = [partition_map[n] for n in test_graph.nodes]
```

```
# In[57]:
```

```
nx.draw(test_graph, node_color=node_colors, with_labels=True)
```

```
# goal: finding a partition that achieves good separation between the groups of nodes
```

```
#
```

```
# in general: get a partition on some objective function
```

```
#
```

```
# more in general: get some information using the connections/weights between nodes (distance) -  
adjacency matrix
```

```
#
```

```
#
```

```
link:https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.quality.modularity.html
```

```
#
```

```
# Modularity is the fraction of the edges that fall within the given groups minus the expected fraction if  
edges were distributed at random. The value of the modularity for unweighted and undirected graphs  
lies in the range. It is positive if the number of edges within groups exceeds the number expected on the  
basis of chance.
```

```
# In[58]:
```

```
partition_example = [{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {10, 11, 12, 13, 14, 15, 16, 17}]
```

```
# In[59]:
```

```
nx.community.quality.modularity(test_graph, partition_example)
```

```
# In[60]:
```

```
partition_example2 = [{0, 1, 2}, {3, 4, 5}, {6, 7, 8, 9}, {10, 11, 12, 13}, {14, 15, 16, 17}]
```

```
# In[61]:
```

```
nx.community.quality.modularity(test_graph, partition_example2)
```

```
# In[62]:
```

```
nx.community.quality.modularity(test_graph, partition)
```

```
# a random network does not have community structure - modularity concept introduces a test to analyse fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random.
```

```
#
```

```
# Analysed at the total network level goal is to maximize modularity score
```

```
# In[63]:
```

```
random_alocation = random.sample(test_graph.nodes, 9)  
random_alocation
```

```
# In[ ]:
```

```
partition_rnd = [set(random_alocation), set(test_graph.nodes) - set(random_alocation)]  
partition_rnd
```

```
# how dose the score look on a random alocation vector - should be close to 0
```

```
# In[ ]:
```

```
nx.community.quality.modularity(test_graph, partition_rnd)
```

```
# In[ ]:
```

```
#visualize  
random_node_color_map = ['red' if n in random_alocation else 'lightblue' for n in test_graph.nodes]  
nx.draw(test_graph, node_color=random_node_color_map)
```

```
# ## 3.2 Karate Club example we can not avoid

# paper "An Information Flow Model for Conflict and Fission in Small Groups" by Wayne W. Zachary

#
#

# wiki: A social network of a karate club was studied by Wayne W. Zachary for a period of three years
# from 1970 to 1972.[2] The network captures 34 members of a karate club, documenting links between
# pairs of members who interacted outside the club.

#
# During the study a conflict arose between the administrator "John A" and instructor "Mr. Hi"
# (pseudonyms), which led to the split of the club into two. Half of the members formed a new club
# around Mr. Hi; members from the other part found a new instructor or gave up karate. Based on
# collected data Zachary correctly assigned all but one member of the club to the groups they actually
# joined after the split.

# In[ ]:

karate_graph = nx.karate_club_graph()

# In[ ]:

nx.draw(karate_graph, with_labels=True)

# In[ ]:

# there are properties
```

```
karate_graph.nodes[0]
```

```
# In[ ]:
```

```
karate_graph.nodes[1]
```

```
# In[ ]:
```

```
karate_graph.nodes[2]
```

```
# In[ ]:
```

```
karate_graph.nodes[20]
```

```
# In[ ]:
```

```
#fancy collors
```

```
club_color = {'Mr. Hi': 'gray', 'Officer': 'red'}
```

```
node_colors = [club_color[karate_graph.nodes[n]['club']] for n in karate_graph.nodes]
```

```
nx.draw(karate_graph, node_color=node_colors, with_labels=True)
```

```
# In[ ]:
```

```
groups = { 'Mr. Hi': set(), 'Officer': set()}
```

```
for n in karate_graph.nodes:
```

```
    club = karate_graph.nodes[n]['club']
```

```
    groups[club].add(n)
```

```
data_partition = list(groups.values())
```

```
data_partition
```

```
# In[ ]:
```

```
nx.community.is_partition(karate_graph, data_partition)
```

```
# In[ ]:
```

```
nx.community.quality.modularity(karate_graph, data_partition)
```

```
# can we automate the process? Find communities in G using greedy modularity maximization.
```

```
#
```

```
#  
  
# paper: Clauset, A., Newman, M. E., & Moore, C. "Finding community structure in very large networks."  
Physical Review E 70(6), 2004.  
  
#  
  
# Greedy modularity maximization begins with each node in its own community and repeatedly joins the  
pair of communities that lead to the largest modularity until no further increase in modularity is possible  
(a maximum).
```

```
# In[ ]:
```

```
from networkx.algorithms.community import greedy_modularity_communities  
  
network_set = greedy_modularity_communities(karate_graph)  
  
network_set
```

```
# In[ ]:
```

```
y = [list(x) for x in network_set]  
  
y
```

```
# In[ ]:
```

```
nx.community.quality.modularity(karate_graph, y)
```

```
# In[ ]:
```

```

partition_map = {}

for idx, cluster_nodes in enumerate(y):
    for node in cluster_nodes:
        partition_map[node] = idx

partition_map

# In[ ]:

node_colors = [partition_map[n] for n in karate_graph.nodes]
nx.draw(karate_graph, node_color=node_colors, with_labels=True)

# so the results find a subcommunity in mr. Hi that does not mingle with the 'traitors'. this is the 3rd group

# ## 3.3. k-clique communities add-on

# A k-clique community is the union of all cliques of size k that can be reached through adjacent (sharing k-1 nodes) k-cliques.

#
# paper: Gergely Palla, Imre Derényi, Illés Farkas1, and Tamás Vicsek, Uncovering the overlapping community structure of complex networks in nature and society Nature 435, 814-818, 2005,
doi:10.1038/nature03607

#
# Clique - subsets of vertices, all adjacent to each other, also called complete subgraphs

```

```
# In[ ]:

sum(1 for c in nx.find_cliques(karate_graph)) # The number of maximal cliques in Karate graph


# In[ ]:

max(nx.find_cliques(karate_graph), key=len) # The largest maximal clique in Karate graph


# In[ ]:

from networkx.algorithms.community import k_clique_communities
c = list(k_clique_communities(karate_graph, 4))
c


# what do we get?


## 3.4 Game of Thrones example (with Girvan-Newman clustering)
#
# These networks were created by connecting two characters whenever their names (or nicknames)
# appeared within 15 words of one another in one of the books in "A Song of Ice and Fire." The edge
# weight corresponds to the number of interactions.
#
```

```
# data at: https://github.com/mathbeveridge/asoiaf

# In[ ]:

import pandas as pd

GOT_books = pd.read_csv('C:/Users/demij/asoiaf-all-edges.csv')

# In[ ]:

GOT_books.head()

# In[ ]:

GOT_graph = nx.Graph()

# In[ ]:

for row in GOT_books.iterrows(): GOT_graph.add_edge(row[1]['Source'], row[1]['Target'],
weight=row[1]['weight'])

# In[ ]:
```

```
list(GOT_graph.edges(data=True))[0]
```

```
# In[ ]:
```

```
list(GOT_graph.edges(data=True))[100]
```

```
# ### Node characteristics
```

```
# In[ ]:
```

```
GOT_graph.number_of_nodes()
```

```
# In[ ]:
```

```
# get that degree stat - fraction of nodes it is connected to
```

```
degree_stats = nx.degree_centrality(GOT_graph)
```

```
# In[ ]:
```

```
degree_stats
```

```
# In[ ]:
```

```
# get that degree stat - those most importaint plot characters
```

```
sorted(degree_stats.items(), key=lambda x:x[1], reverse=True)[0:10]
```

```
# In[ ]:
```

```
## betweenness centrality
```

```
betweenness_centrality_stats = nx.betweenness_centrality(GOT_graph)
```

```
# In[ ]:
```

```
# get that betweenness_centrality - those most importaint plot twisters characters - close to the action
```

```
sorted(betweenness_centrality_stats.items(), key=lambda x:x[1], reverse=True)[0:10]
```

```
# In[ ]:
```

```
degree_value = GOT_graph.degree()
```

```
# In[ ]:
```

```
import numpy as np  
np.mean([d for _, d in GOT_graph.degree()])
```

```
# In[ ]:
```

```
#lets remove some marginal nodes to go easy on ourselves  
remove_val = [node for node,degree in dict(GOT_graph.degree()).items() if degree < 2*7]  
remove_val
```

```
# In[ ]:
```

```
GOT_graph.remove_nodes_from(remove_val)
```

```
# In[ ]:
```

```
GOT_graph.number_of_nodes()
```

```
# In[ ]:
```

```
degree_stats2 = nx.degree_centrality(GOT_graph)
```

```
degree_stats2
```

```
# In[ ]:
```

```
betweenness_centrality_stats2 = nx.betweenness_centrality(GOT_graph)
```

```
betweenness_centrality_stats2
```

```
# The Girvan-Newman algorithm for the detection and analysis of community structure relies on the iterative elimination of edges that have the highest number of shortest paths between nodes passing through them. By removing edges from the graph one-by-one, the network breaks down into smaller pieces, so-called communities. The algorithm was introduced by Michelle Girvan and Mark Newman.
```

```
#
```

```
# https://networkx.guide/algorithms/community-detection/girvan-newman/
```

```
# In[ ]:
```

```
nx.draw(GOT_graph, with_labels=True)
```

```
# so if we remove the connection with the highest betweenness centrality it would be the central one leaving us with the 2 groups
```

```
# note that this is an ideal case: groups are fully connected
```

```
#
```

```
# The Girvan-Newman algorithm can be divided into four main steps:  
#  
# 1. For every edge in a graph, calculate the edge betweenness centrality.  
# 2. Remove the edge with the highest betweenness centrality.  
# 3. Calculate the betweenness centrality for every remaining edge.  
# 4. Repeat steps 2-4 until there are no more edges left.  
#  
# https://networkx.guide/algorithms/community-detection/girvan-newman/  
#  
# At the end - Evaluate each partition in the sequence and choose the one with the highest modularity
```

```
# In[ ]:
```

```
import matplotlib.pyplot as plt
```

```
# In[ ]:
```

```
# draw the subgraph using the spring layout  
pos = nx.spring_layout(GOT_graph)  
  
# color the nodes based on degree centrality  
node_colors = [degree_stats2[node] for node in GOT_graph]  
nx.draw(GOT_graph, pos, node_color=node_colors, cmap=plt.cm.Reds)
```

```
# show the plot
```

```
plt.savefig("ref_plot.png",dpi=500)
```

```
plt.show()

# In[ ]:

from networkx.algorithms.community.centrality import girvan_newman

partition_girvan_newman = girvan_newman(GOT_graph)
list(partition_girvan_newman)

# conclusion: central to plots and interactions in those plots

# In[ ]:

# k-clique with a high k
from networkx.algorithms.community import k_clique_communities
c_got = list(k_clique_communities(GOT_graph, 10))
c_got

# conclusion - main plot (across all 5 books) and a big side plot (Night's Watch)
```

Auditorne 5

```
#!/usr/bin/env python
# coding: utf-8
```

```
# # LAB 5 - Karate club, Bala Goyal 98 learning, DeGroot and clustering all in one
```

```
# In[1]:
```

```
import networkx as nx
```

```
import random
```

```
import numpy as np
```

```
random.seed(123)
```

```
p_val = 0.55
```

```
### Bala Goyal 98 LEARNIING - repeated action, observe one another
```

```
#-----
```

```
# In[2]:
```

```
# Create an empty graph
```

```
G = nx.Graph()
```

```
# Add 20 nodes to the graph
```

```
for i in range(1, 21):
```

```
    G.add_node(i)
```

```
# Add random edges to the graph
```

```
for i in range(1, 21):
    for j in range(i+1, 21):
        if random.random() < 0.5:
            G.add_edge(i, j)

import matplotlib.pyplot as plt
nx.draw(G, with_labels = True)
plt.show()

# In[3]:


# Add random state "A" or "B" to each node
for node in G.nodes():
    if random.random() < 0.5:
        G.nodes[node]["state"] = "A"
        G.nodes[node]["value"] = 1
    else:
        G.nodes[node]["state"] = "B"
        G.nodes[node]["value"] = 2 if random.random() < p_val else 0

# In[4]:


#printing the node state and value
for node in G.nodes():
    print(f'{node},{G.nodes[node]["state"]},{G.nodes[node]["value"]}'")
```

```
# In[5]:
```

```
# Create a dictionary to map each state to a color
state_colors = {"A": "blue", "B": "red"}

# Draw the graph, coloring each node by its state
pos = nx.spring_layout(G)
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]]] for node in G.nodes(), with_labels = True)
plt.show()
```

```
# In[6]:
```

```
# Create a dictionary to map each value to a color
value_colors = {1: "blue", 2: "green", 0: "red"}

# Draw the graph, coloring each node by its value
pos = nx.spring_layout(G)
nx.draw(G, pos, node_color=[value_colors[G.nodes[node]["value"]]] for node in G.nodes(), with_labels = True)
plt.show()
```

```
# In[7]:
```

```

#20 iterations for loop

for i in range(20):

    new_states = {}

    new_values = {}

    for node in G.nodes():

        # calculate average value of state A

        A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]

        if len(A_neighbors) == 0:

            A_average = 0

        else:

            A_average = sum(A_neighbors) / len(A_neighbors)

        # calculate average value of state B

        B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]

        if len(B_neighbors) == 0:

            B_average = 0

        else:

            B_average = sum(B_neighbors) / len(B_neighbors)

        # change state of node to the one of higher average value

        if A_average > B_average:

            new_states[node] = "A"

            new_values[node] = 1

        elif B_average > A_average:

            new_states[node] = "B"

```

```
new_values[node] = 2 if random.random() < p_val else 0  
for node in new_states:  
    G.nodes[node]["state"] = new_states[node]  
    G.nodes[node]["value"] = new_values[node]
```

In[8]:

```
#printing the node state and value  
for node in G.nodes():  
    print(f'{node},{G.nodes[node]["state"]},{G.nodes[node]["value"]}')  
#-----
```

In[9]:

```
# Create a dictionary to map each state to a color  
state_colors = {"A": "blue", "B": "red"}  
  
# Draw the graph, coloring each node by its state  
pos = nx.spring_layout(G)  
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)  
plt.show()
```

In[10]:

```
# Create a dictionary to map each value to a color
value_colors = {1: "blue", 2: "green", 0: "red"}

# Draw the graph, coloring each node by its value
pos = nx.spring_layout(G)

nx.draw(G, pos, node_color=[value_colors[G.nodes[node]["value"]] for node in G.nodes()], with_labels = True)

plt.show()
```

```
# In[11]:
```

```
##### what is the issue - random network
```

```
G = nx.karate_club_graph()
```

```
random.seed(12345)
```

```
# Add random state "A" or "B" to each node
```

```
for node in G.nodes():
```

```
    if random.random() < 0.75:
```

```
        G.nodes[node]["state"] = "A"
```

```
        G.nodes[node]["value"] = 1
```

```
    else:
```

```
        G.nodes[node]["state"] = "B"
```

```
        G.nodes[node]["value"] = 2 if random.random() < p_val else 0
```

```
# In[12]:
```

```
import matplotlib.pyplot as plt  
nx.draw(G, with_labels = True)  
plt.show()
```

```
# In[13]:
```

```
# Create a dictionary to map each state to a color  
state_colors = {"A": "blue", "B": "red"}  
  
# Draw the graph, coloring each node by its state  
pos = nx.spring_layout(G)  
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)  
plt.show()
```

```
# In[14]:
```

```
# Create a dictionary to map each value to a color  
value_colors = {1: "blue", 2: "green", 0: "red"}  
  
# Draw the graph, coloring each node by its value
```

```
nx.draw(G, pos, node_color=[value_colors[G.nodes[node]["value"]]] for node in G.nodes(), with_labels = True)
plt.show()
```

In[15]:

```
average_values = []
```

In[16]:

```
new_states = {}
new_values = {}
for node in G.nodes():
    # calculate average value of state A
    A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]
    if len(A_neighbors) == 0:
        A_average = 0
    else:
        A_average = sum(A_neighbors) / len(A_neighbors)

    # calculate average value of state B
    B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]
    if len(B_neighbors) == 0:
        B_average = 0
    else:
```

```

else:
    B_average = sum(B_neighbors) / len(B_neighbors)

# change state of node to the one of higher average value

if A_average > B_average:
    new_states[node] = "A"
    new_values[node] = 1

elif B_average > A_average:
    new_states[node] = "B"
    new_values[node] = 2 if random.random() < 0.5 else 0

for node in new_states:
    G.nodes[node]["state"] = new_states[node]
    G.nodes[node]["value"] = new_values[node]

average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes()))

# Create a dictionary to map each state to a color

state_colors = {"A": "blue", "B": "red"}


# Draw the graph, coloring each node by its state

nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)

plt.show()

```

In[17]:

```

new_states = {}

new_values = {}

```

```

for node in G.nodes():

    # calculate average value of state A

    A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]

    if len(A_neighbors) == 0:

        A_average = 0

    else:

        A_average = sum(A_neighbors) / len(A_neighbors)

    # calculate average value of state B

    B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]

    if len(B_neighbors) == 0:

        B_average = 0

    else:

        B_average = sum(B_neighbors) / len(B_neighbors)

    # change state of node to the one of higher average value

    if A_average > B_average:

        new_states[node] = "A"

        new_values[node] = 1

    elif B_average > A_average:

        new_states[node] = "B"

        new_values[node] = 2 if random.random() < 0.5 else 0

    for node in new_states:

        G.nodes[node]["state"] = new_states[node]

        G.nodes[node]["value"] = new_values[node]

average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes()))

```

```
# Create a dictionary to map each state to a color
state_colors = {"A": "blue", "B": "red"}

# Draw the graph, coloring each node by its state
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)

plt.show()
```

In[18]:

```
new_states = {}
new_values = {}

for node in G.nodes():

    # calculate average value of state A

    A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]

    if len(A_neighbors) == 0:
        A_average = 0
    else:
        A_average = sum(A_neighbors) / len(A_neighbors)

    # calculate average value of state B

    B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]

    if len(B_neighbors) == 0:
        B_average = 0
    else:
        B_average = sum(B_neighbors) / len(B_neighbors)
```

```

# change state of node to the one of higher average value

if A_average > B_average:
    new_states[node] = "A"
    new_values[node] = 1

elif B_average > A_average:
    new_states[node] = "B"
    new_values[node] = 2 if random.random() < 0.5 else 0

for node in new_states:
    G.nodes[node]["state"] = new_states[node]
    G.nodes[node]["value"] = new_values[node]

average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes()))

# Create a dictionary to map each state to a color
state_colors = {"A": "blue", "B": "red"}

# Draw the graph, coloring each node by its state
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)

plt.show()

# In[19]:


new_states = {}
new_values = {}

for node in G.nodes():

    # calculate average value of state A

```

```

A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]

if len(A_neighbors) == 0:
    A_average = 0
else:
    A_average = sum(A_neighbors) / len(A_neighbors)

# calculate average value of state B

B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]

if len(B_neighbors) == 0:
    B_average = 0
else:
    B_average = sum(B_neighbors) / len(B_neighbors)

# change state of node to the one of higher average value

if A_average > B_average:
    new_states[node] = "A"
    new_values[node] = 1
elif B_average > A_average:
    new_states[node] = "B"
    new_values[node] = 2 if random.random() < 0.5 else 0

for node in new_states:
    G.nodes[node]["state"] = new_states[node]
    G.nodes[node]["value"] = new_values[node]

average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes()))

# Create a dictionary to map each state to a color

state_colors = {"A": "blue", "B": "red"}

```

```
# Draw the graph, coloring each node by its state  
  
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)  
  
plt.show()
```

```
# In[20]:
```

```
new_states = {}  
new_values = {}  
  
for node in G.nodes():  
  
    # calculate average value of state A  
  
    A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if  
    G.nodes[neighbor]["state"] == "A"]  
  
    if len(A_neighbors) == 0:  
  
        A_average = 0  
  
    else:  
  
        A_average = sum(A_neighbors) / len(A_neighbors)  
  
    # calculate average value of state B  
  
    B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if  
    G.nodes[neighbor]["state"] == "B"]  
  
    if len(B_neighbors) == 0:  
  
        B_average = 0  
  
    else:  
  
        B_average = sum(B_neighbors) / len(B_neighbors)  
  
    # change state of node to the one of higher average value
```

```

if A_average > B_average:
    new_states[node] = "A"
    new_values[node] = 1

elif B_average > A_average:
    new_states[node] = "B"
    new_values[node] = 2 if random.random() < 0.5 else 0

for node in new_states:
    G.nodes[node]["state"] = new_states[node]
    G.nodes[node]["value"] = new_values[node]

average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes())))

# Create a dictionary to map each state to a color
state_colors = {"A": "blue", "B": "red"}


# Draw the graph, coloring each node by its state
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)
plt.show()

# In[21]:

```

```

new_states = {}
new_values = {}

for node in G.nodes():

    # calculate average value of state A
    A_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "A"]

```

```

if len(A_neighbors) == 0:
    A_average = 0
else:
    A_average = sum(A_neighbors) / len(A_neighbors)

# calculate average value of state B

B_neighbors = [G.nodes[neighbor]["value"] for neighbor in G.neighbors(node) if
G.nodes[neighbor]["state"] == "B"]

if len(B_neighbors) == 0:
    B_average = 0
else:
    B_average = sum(B_neighbors) / len(B_neighbors)

# change state of node to the one of higher average value

if A_average > B_average:
    new_states[node] = "A"
    new_values[node] = 1
elif B_average > A_average:
    new_states[node] = "B"
    new_values[node] = 2 if random.random() < 0.5 else 0
for node in new_states:
    G.nodes[node]["state"] = new_states[node]
    G.nodes[node]["value"] = new_values[node]
average_values.append(sum([G.nodes[node]["value"] for node in G.nodes()])/len(G.nodes()))

# Create a dictionary to map each state to a color

state_colors = {"A": "blue", "B": "red"}


# Draw the graph, coloring each node by its state

```

```
nx.draw(G, pos, node_color=[state_colors[G.nodes[node]["state"]] for node in G.nodes()], with_labels = True)  
plt.show()
```

In[22]:

```
from sklearn.cluster import AgglomerativeClustering
```

In[]:

```
A = nx.floyd_marshall_numpy(G) # distance mx - instance[i, j] is the distance along a shortest path from i  
to j
```

In[]:

A

In[]:

```
# Create an AgglomerativeClustering object  
agg_clustering = AgglomerativeClustering(n_clusters=2, affinity = "precomputed", linkage = 'complete')
```

```
# Fit the model to the data
agg_clustering.fit(A)

# Retrieve the labels for each node
cluster_labels = agg_clustering.labels_

# In[ ]:

# Define colors for the nodes
colors = ["gray" if cluster_labels[i] == 0 else "red" for i in range(len(cluster_labels))]

# Plot the graph with the colors
nx.draw(G, pos, node_color=colors, with_labels=True)
plt.show()

# In[ ]:

# Retrieve the labels for each node
# Get the original community labels
original_labels = nx.get_node_attributes(G,'club')

# Plot the graph with the colors
nx.draw(G, pos, node_color=colors, with_labels=False)
```

```
# Add the original labels to the graph  
nx.draw_networkx_labels(G, pos, original_labels)  
plt.show()
```

```
# In[ ]:
```

```
propagating_state = nx.get_node_attributes(G,'state')  
nx.draw(G, pos, node_color=colors, with_labels=False)  
nx.draw_networkx_labels(G, pos, propagating_state)  
plt.show()
```

```
# In[ ]:
```

```
#### DeGroot learning
```

```
import networkx as nx  
import numpy as np  
  
# Import the Karate Club graph  
G = nx.karate_club_graph()  
  
np.random.seed(123)  
  
# Initialize the opinion vector with random values - probability of event X occurring  
# Get the club labels of each node  
club_labels = nx.get_node_attributes(G, "club")
```

```
# In[ ]:
```

```
club_labels
```

```
# In[ ]:
```

```
# Create a dictionary to map each node to its club label
```

```
node_club = {}
```

```
for node, label in club_labels.items():
```

```
    node_club[node] = label
```

```
# In[ ]:
```

```
# Initialize the opinion vector with random values
```

```
opinions = np.random.rand(G.number_of_nodes())
```

```
# Make the opinions 50% less for one group
```

```
for i in range(G.number_of_nodes()):
```

```
    if node_club[i] == 'Officer':
```

```
        opinions[i] = opinions[i]*0.5
```

```
# In[ ]:
```

```
opinions
```

```
# In[ ]:
```

```
# Draw the graph
```

```
pos = nx.spring_layout(G)
```

```
# Plot the graph with the colors
```

```
nx.draw(G, pos, node_color=colors, with_labels=False)
```

```
# Add the opinions to each node
```

```
for i in range(G.number_of_nodes()):
```

```
    plt.annotate(round(opinions[i], 2), xy=pos[i], fontsize=8)
```

```
plt.show()
```

```
# In[ ]:
```

```
# Initialize the dictionary to store the average opinion for each group
```

```
avg_opinions = {'Mr. Hi': 0, 'Officer': 0}
```

```
count = {'Mr. Hi': 0, 'Officer': 0}
```

```
# Calculate the average opinion for each group
```

```
for i in range(G.number_of_nodes()):
```

```
group = node_club[i]
avg_opinions[group] += opinions[i]
count[group] +=1
```

```
# In[ ]:
```

```
# Divide the total opinion by the number of nodes in each group
for group in avg_opinions.keys():
    avg_opinions[group] = avg_opinions[group]/count[group]
```

```
# Print the average opinion for each group
print(avg_opinions)
```

```
# In[ ]:
```

```
# Set the number of iterations
num_iters = 20
```

```
# Set the constant alpha also known as Tii - how much we are stubborn (confirmed by experiments)
alpha = 0.7
```

```
# Simulate the DeGroot model - talking to neigbours gettinng average of their opinion
# and then update my belief in the event X
# note the unweighted graph vs full weights as in example - so this is different
for i in range(num_iters):
```

```

new_opinions = np.zeros(G.number_of_nodes())

for j in range(G.number_of_nodes()):
    neighbors = list(G.neighbors(j))
    neighbors_opinions = opinions[neighbors]
    new_opinions[j] = alpha * opinions[j] + (1 - alpha) * np.mean(neighbors_opinions)
opinions = new_opinions

# Print the final opinions
print(opinions)
####

# In[ ]:

# Get the club labels of each node
club_labels = nx.get_node_attributes(G, "club")

# Create a list of colors for each club label
colors = ["gray" if club_labels[i] == 'Officer' else "red" for i in range(len(club_labels))]

# Draw the graph
pos = nx.spring_layout(G)

# Plot the graph with the colors
nx.draw(G, pos, node_color=colors, with_labels=False)

# Add the opinions to each node
for i in range(G.number_of_nodes()):
    plt.annotate(round(opinions[i], 2), xy=pos[i], fontsize=8)

```

```
plt.show()
```

```
# In[ ]:
```

```
# Initialize the dictionary to store the average opinion for each group
```

```
avg_opinions = {'Mr. Hi': 0, 'Officer': 0}
```

```
count = {'Mr. Hi': 0, 'Officer': 0}
```

```
# In[ ]:
```

```
# Calculate the average opinion for each group
```

```
for i in range(G.number_of_nodes()):
```

```
    group = node_club[i]
```

```
    avg_opinions[group] += opinions[i]
```

```
    count[group] +=1
```

```
# Divide the total opinion by the number of nodes in each group
```

```
for group in avg_opinions.keys():
```

```
    avg_opinions[group] = avg_opinions[group]/count[group]
```

```
# Print the average opinion for each group
```

```
print(avg_opinions)
```

```
# In[ ]:
```

```

# Set the number of iterations
num_iters = 100

# Set the constant alpha also known as Tii - how much we are stubborn (confirmed by experiments)
alpha = 0.7

# Simulate the DeGroot model - talking to neighbours getting average of their opinion
# and then update my belief in the event X
# note the unweighted graph vs full weights as in example - so this is different

for i in range(num_iters):
    new_opinions = np.zeros(G.number_of_nodes())
    for j in range(G.number_of_nodes()):
        neighbors = list(G.neighbors(j))
        neighbors_opinions = opinions[neighbors]
        new_opinions[j] = alpha * opinions[j] + (1 - alpha) * np.mean(neighbors_opinions)
    opinions = new_opinions

# Print the final opinions
print(opinions)

###-----
# In[ ]:

# Initialize the dictionary to store the average opinion for each group
avg_opinions = {'Mr. Hi': 0, 'Officer': 0}

```

```
count = {'Mr. Hi': 0, 'Officer': 0}
```

```
# In[ ]:
```

```
# Calculate the average opinion for each group
```

```
for i in range(G.number_of_nodes()):  
    group = node_club[i]  
    avg_opinions[group] += opinions[i]  
    count[group] += 1
```

```
# Divide the total opinion by the number of nodes in each group
```

```
for group in avg_opinions.keys():  
    avg_opinions[group] = avg_opinions[group]/count[group]
```

```
# Print the average opinion for each group
```

```
print(avg_opinions)
```

```
# In[ ]:
```

```
# the society is wise precisely when even the most influential individual's influence vanishes in the large  
society limit
```

```
# if the society grows without bound, over time they will have a common and accurate belief on the  
uncertain subject
```

```
#
```

#

Kompleksne mreže

1. predavanje

1997 / 1998

O predmetu

- 13 predavanja (2 sata)
- 6 auditornih (2 sata)
- Zavodi:
 - Zavod za elektroničke sustave i obradbu informacija, D-zgrada, 1. kat
 - Zavod za telekomunikacije, C-zgrada, 8. kat

Pregled predavanja:

Rbr.	Datum	Tema	Auditorne
	Srijeda		
1.	4.10.	Uvod u predmet. Mrežni elementi	
2.	11.10.	Mreže malog svijeta, socijalna udaljenost	
3.	18.10.	Centralnosti (hubovi, robustnost)	Aud 1
4.	25.10.	Usmjerenost i težine	
5.	1.11.	Praznik	
6.	8.11.	Mrežni modeli (slučajne mreže, preferencijalni modeli)	Aud 2
7.	15.11	Dinamika (širenje ideja, informacija i utjecaj)	Aud 3
MEĐUISPIT (ne održava se)			
8.	6.12.	Učenje u mrežama (Bayesian & DeGroot model, mudrost masa)	
9.	13.12.	Igre u mrežama (peer utjecaj, odnos mrežne strukture i ponašanja)	Aud 4
10.	20.12.	Zajednice u mrežama (koncepti zajednica, detekcija zajednica)	Aud 5
11.	10.1.	Društvene mreže (definicija, razvoj, vrste)	
12.	17.1.	Analiza društvenih mreža	Aud 6
13.	24.1.	Izabrane teme u području kompleksnih mreža	
ZAVRŠNI ISPIT (termin prema akademskom kalendaru)			

Auditorne vježbe

- 6 pokaznih vježbi po 2 sata
- Python
- NetworkX

Predavači

Prof. Mile Šikić

FER, Genome Institute of Singapore,
Oraclum Ltd, ORCA hedge fond



Prof. Vedran Podobnik

FER i Hewlett Packard Enterprise



Asistenti

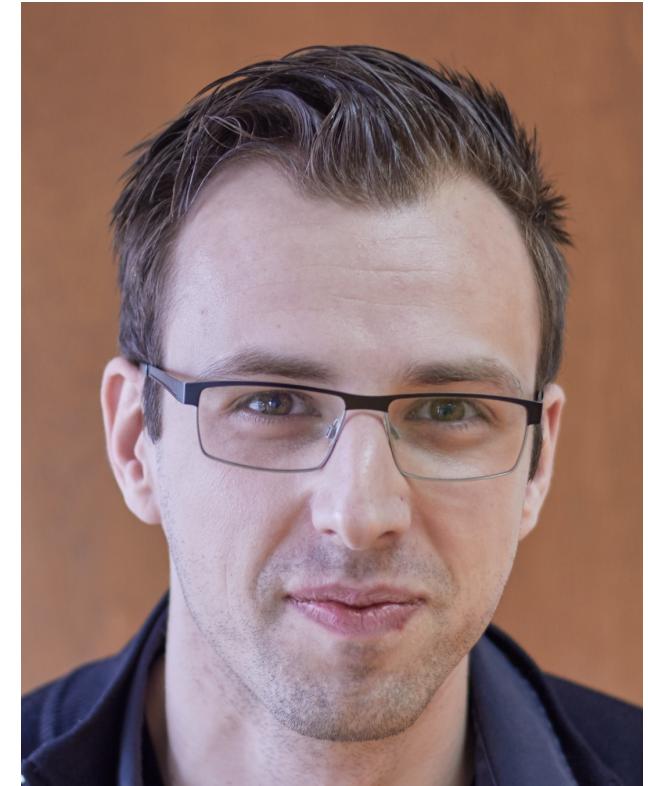
Filip Tomas, mag.ing.



Josipa Lipovac, mag.ing.



Demijan Grgić, mag.ing.



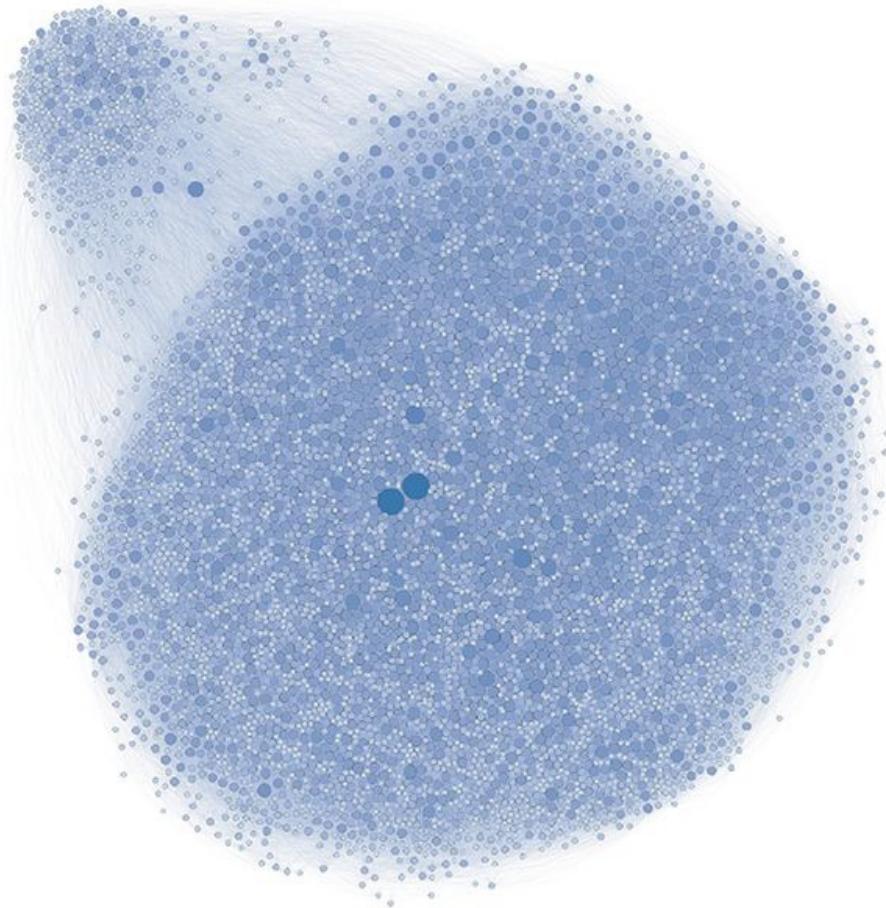
Bodovanje

- Prisustvo nastavi – 10 bodova
- Prisustvo auditornim vježbama – 10 bodova
- Međuispit – Nema
- Završni ispit (pismeni i na računalu – 90 minuta) – 80 bodova
- Ispitni rok (pismeni i na računalu – 90 minuta) – 80 bodova

Ocjenvivanje (pragovi)

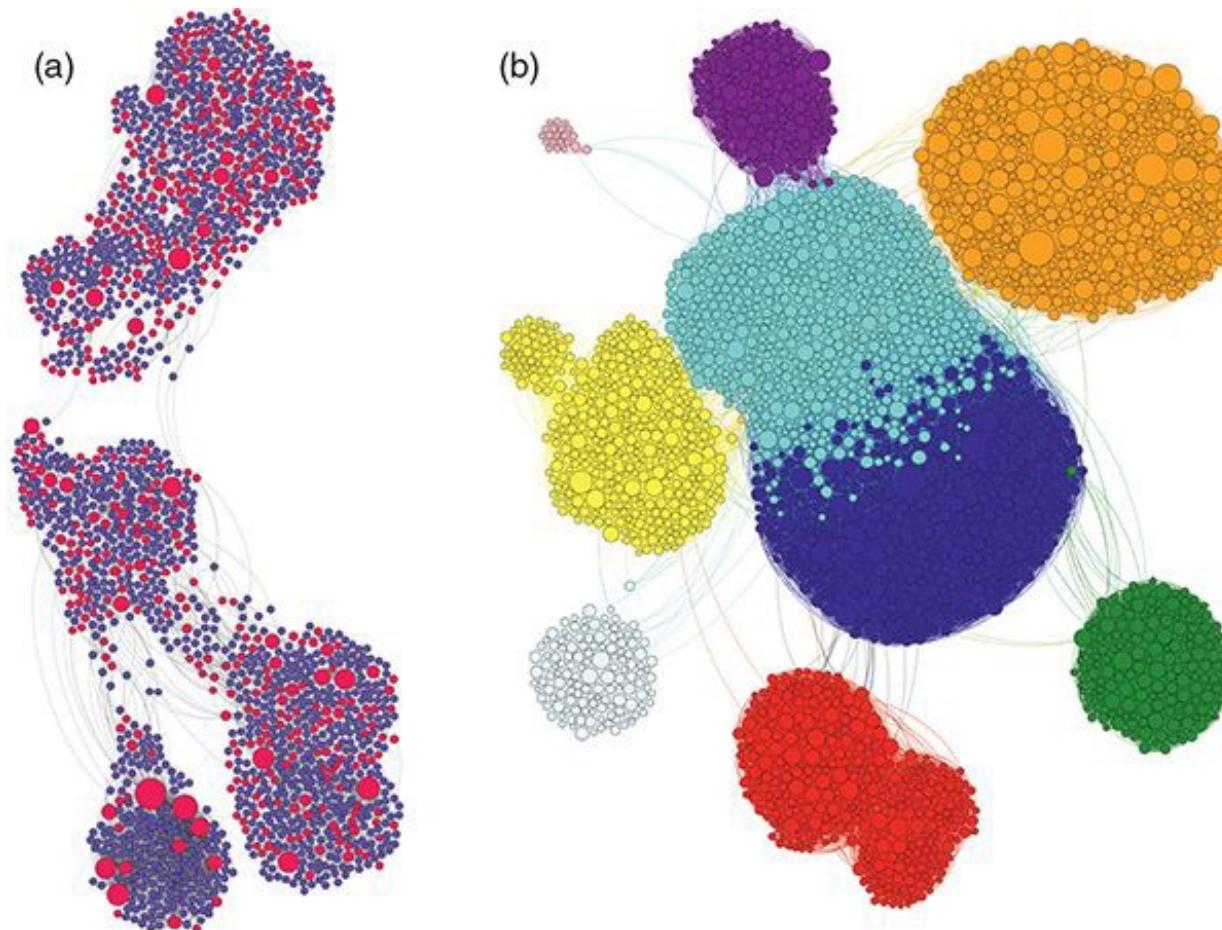
- 2 – 50 bodova
- 3 – 62 boda
- 4 – 74 boda
- 5 – 86 bodova

Kompleksne mreže - Facebook



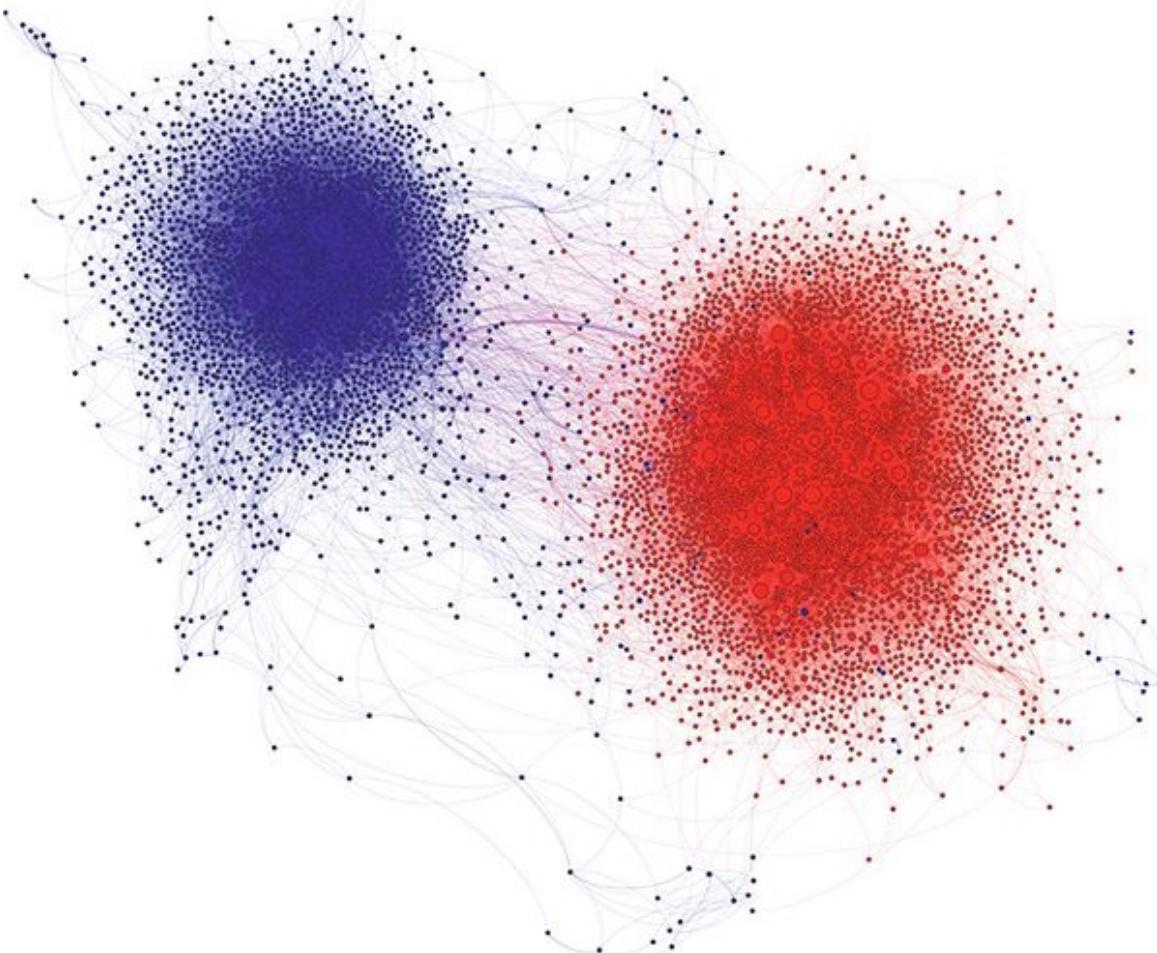
Mreža Facebook korisnika na Northwestern University

Kompleksne mreže – mreža glumaca



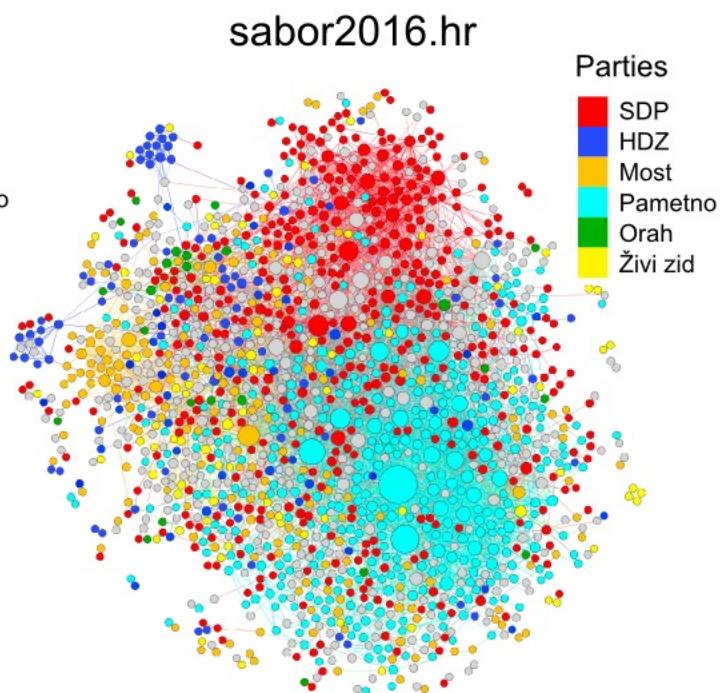
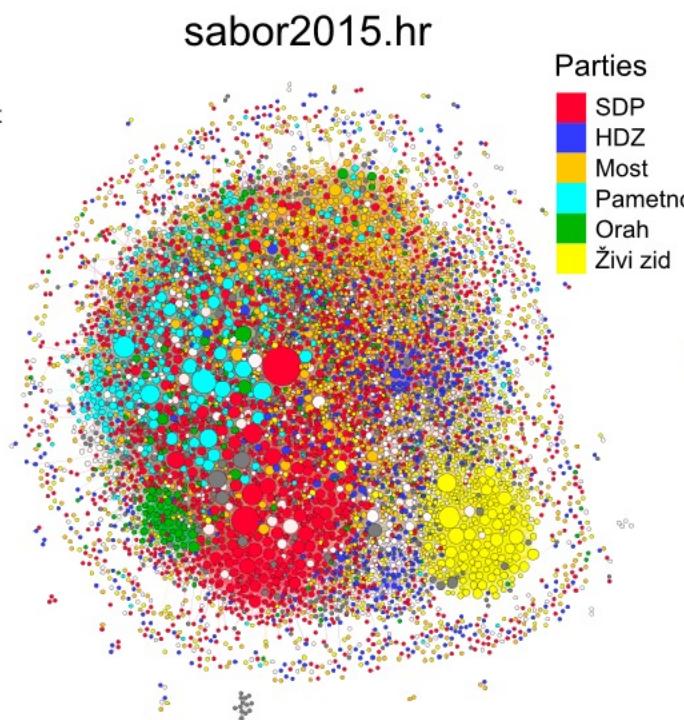
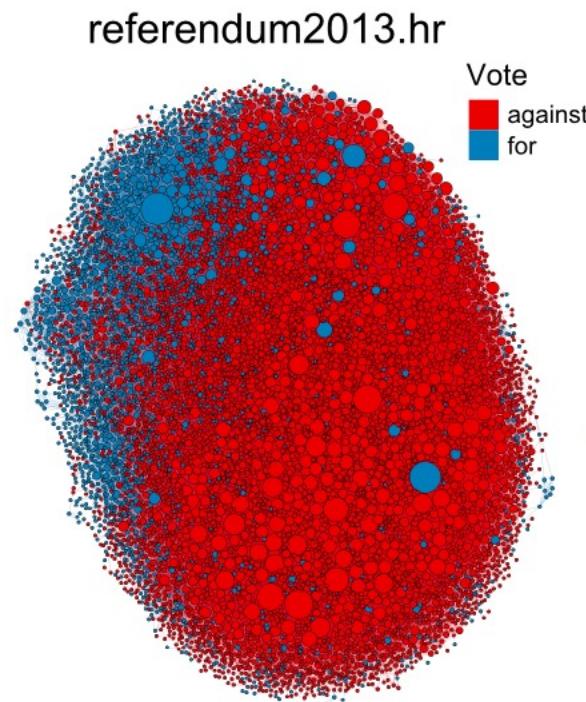
- (a) Film – zvijezda mreža temeljena na manjem uzorku filmova, glumaca iz IMDB. Čvorovi su filmovi (plavo) ili glumci (crveno)
- (b) Mreža glumaca koji su glumili istim filmovima iz IMDB. Boje predstavljaju žanrove ili jezike/zemlje (Hollywood –plavi, vesterni – tirkizna, meksički – ljubičasta, kineski – žuta, filipinski – narančasta, turski i istočno europski – zeleno, indijski – crvena, grčki – bijela, za odrasle – ružičasta

Kompleksne mreže – Twitter (US politika 2010)

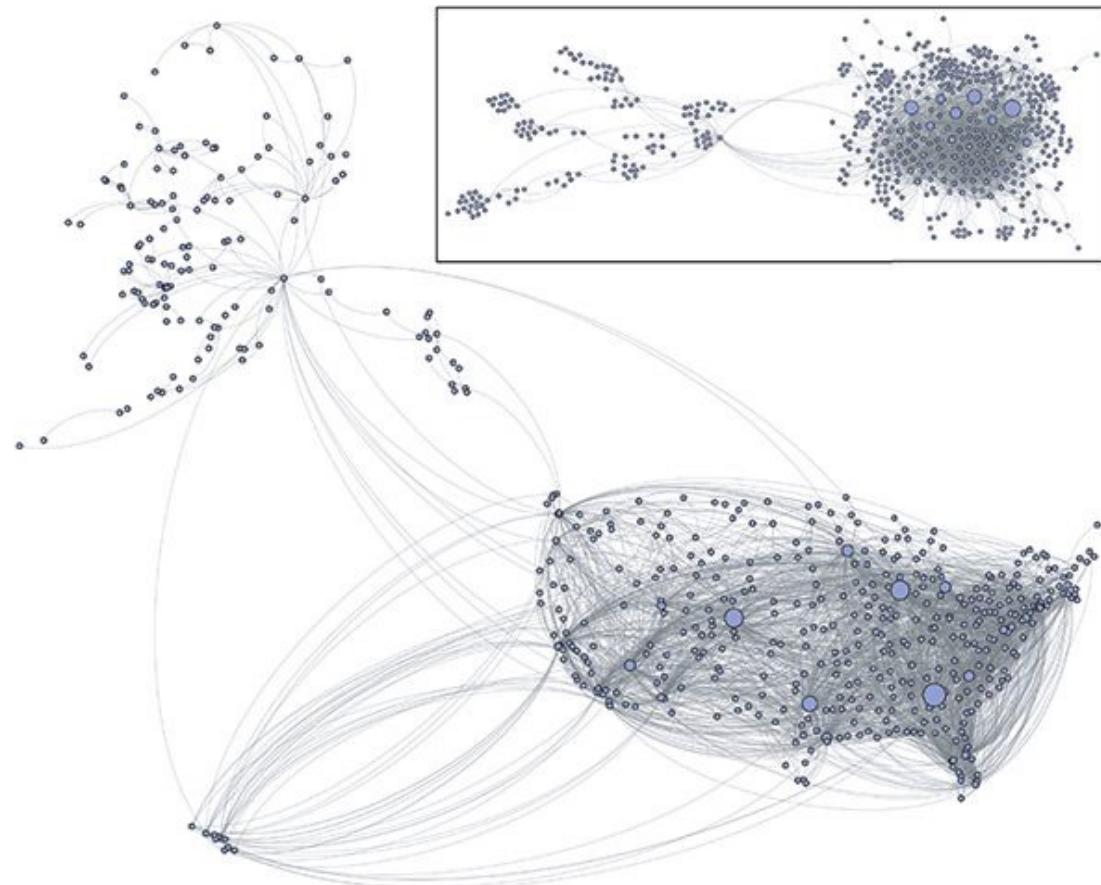


Mreža retweeta vezana uz američke izbore za kongres i senat 2010 . Crveno su konzervativci, a plavo progresivni

Kompleksne mreže - Facebook - Hrvatska



Kompleksne mreže – mreža aerodroma



Mreža US avionskog prometa (podaci o prometu sa OpenFlights.org). Čvorovi su pozicionirani u skladu sa svojim geografskim koordinatama. Bitni aerodromi (Atlanta, Chicago, Denver) su uočljivi. Manja mapa prikazuje drugačije nacrtanu mrežu.

Kompleksne mreže – biološke mreže



Tri biološke mreže. Lijevo: mreža proteinskih interakcija kvasca. Sredina: neuralna mreža crva. Desno: mreža hranidbenog lanca u Everglades nacionalnom parku (Florida). Three biological networks. Left: Protein interaction network of yeast. Direktne veze od plijena prema predotoru. Veliki plavi čvorovi su na vrhu hranidbenog lanca, a mali crveni na dnu.

Osnovni pojmovi

Matematika

- Graf
- Vrh
- Brid

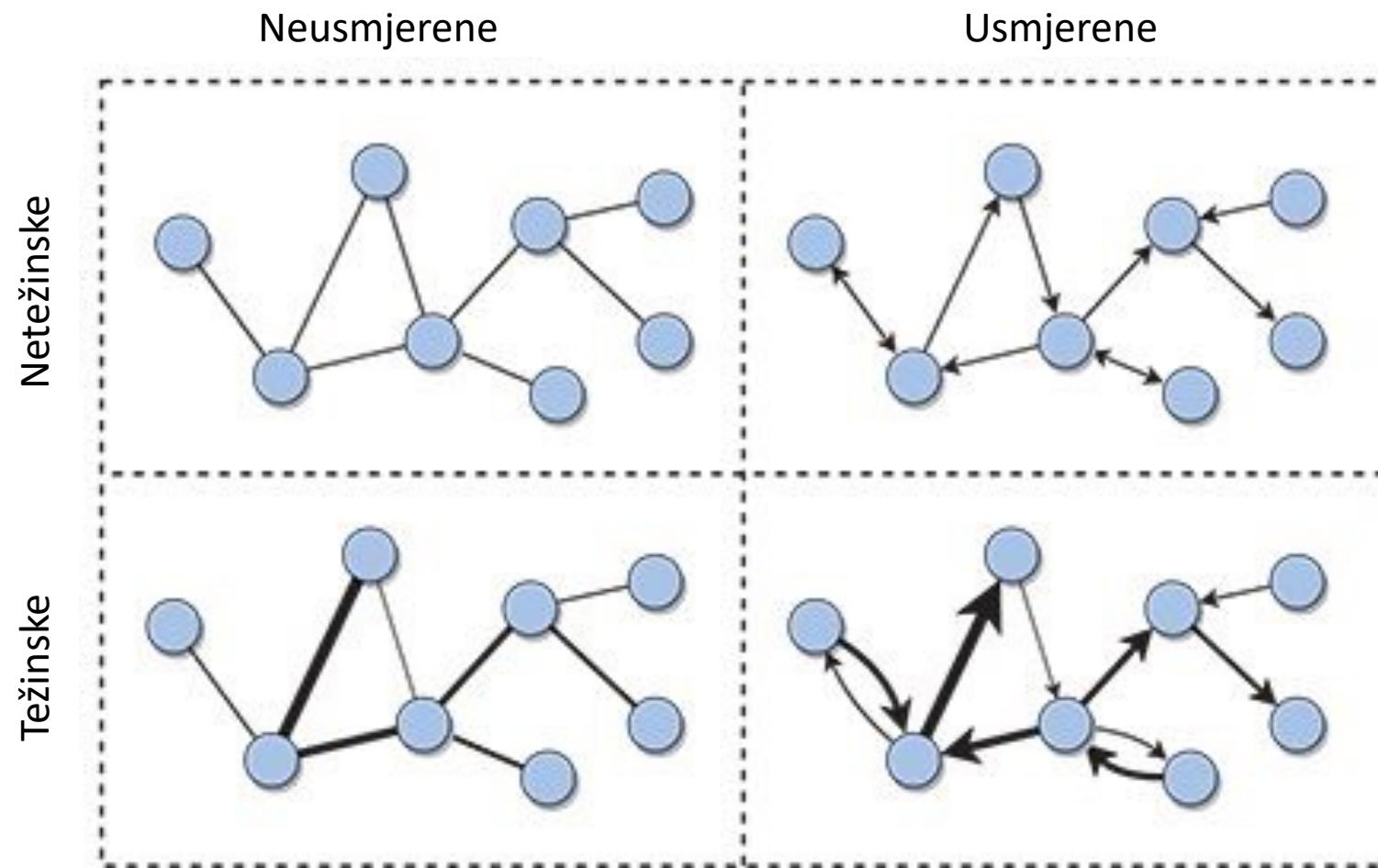
Fizika

- Mreža
- Čvor
- Veza

Osnovni pojmovi

- Mreža G se sastoji od skupa od N elemenata (čvorovi) i skupa od L parova čvorova (veze)
- Usmjerenost
 - Direktna (digraf) – veza (i,j) ide od izvorišnog čvora i do odredišnog čvora j
 - Neusmjerena (veze dvosmjerne)
- Težina
 - Težinske (i,j,w) w - težina
 - Netežinske

Osnovni pojmovi



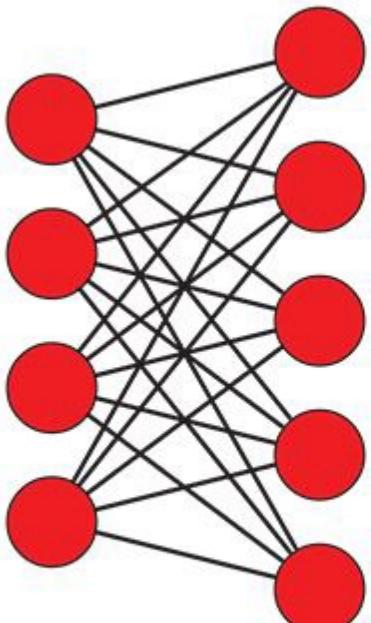
NetworkX

- <https://networkx.org/>

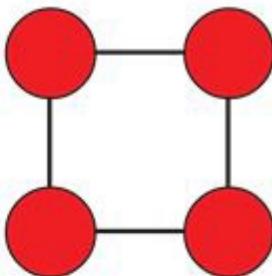


Jednostavne mreže

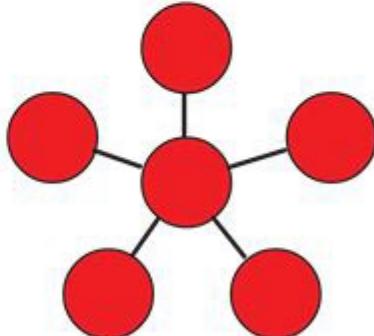
B = nx.complete_bipartite_graph(4,5)



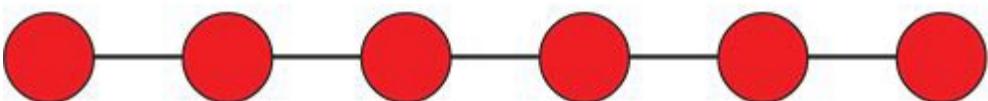
C = nx.cycle_graph(4)



S = nx.star_graph(6)



P = nx.path_graph(5)

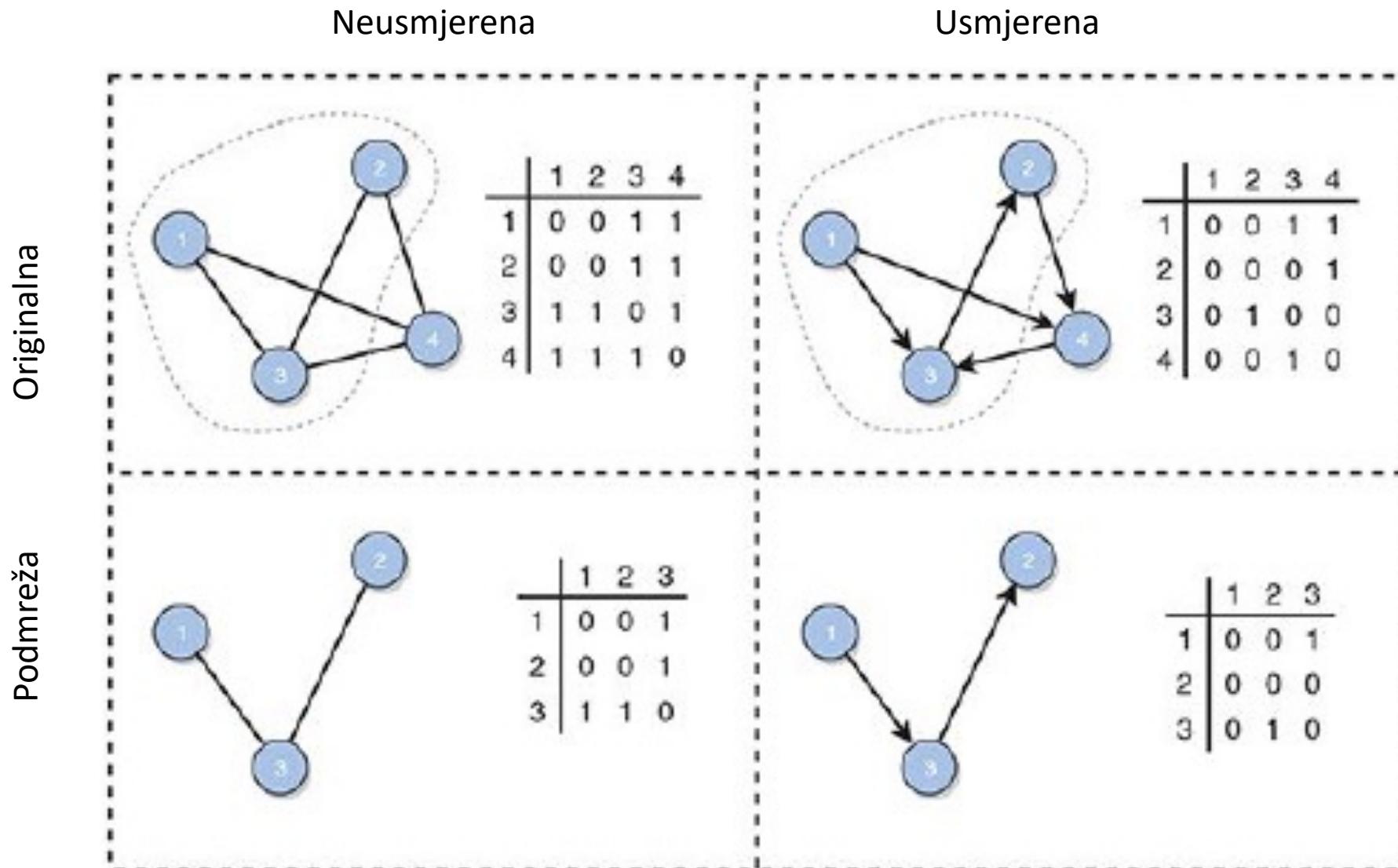


- Bipartitni graf
- Zvijezda
- Ciklički graf
- Lanac

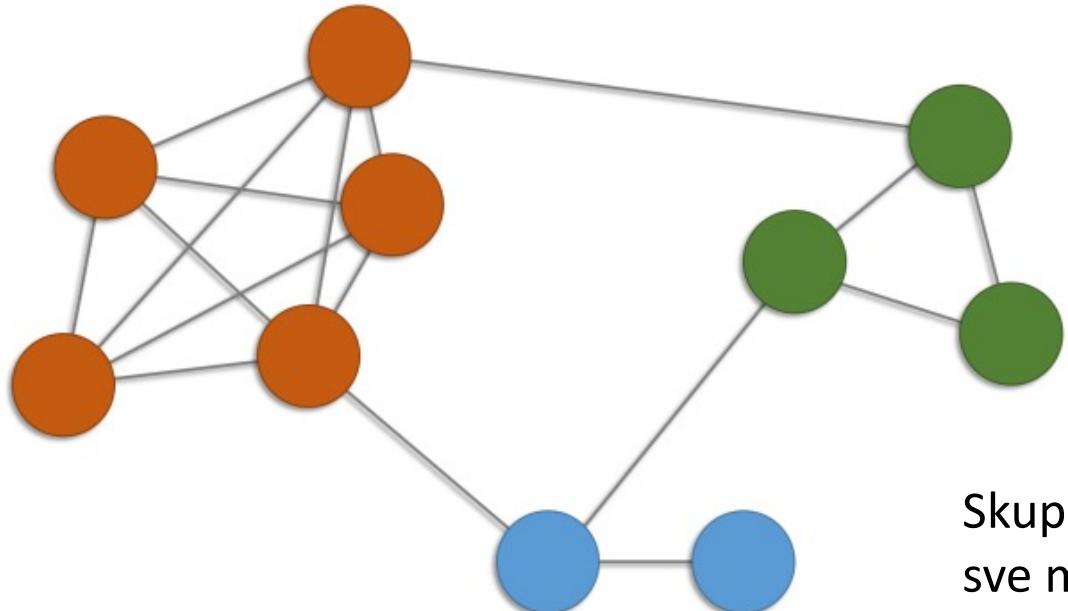
Gustoća i rijetkost

- Maksimalna gustoća
 - Neusmjereni $L_{max} = \binom{N}{2} = \frac{N(N-1)}{2}$
 - Usmjereni $L_{max} = N(N - 1)$
- Gustoća
 - Neusmjereni $d = \frac{L}{L_{max}} = \frac{2L}{N(N-1)}$
 - Usmjereni $d = \frac{L}{L_{max}} = \frac{L}{N(N-1)}$
 - Rijetka mreža $L \sim N$
 - Gusta mreža $L \sim N^2$

Podmreža

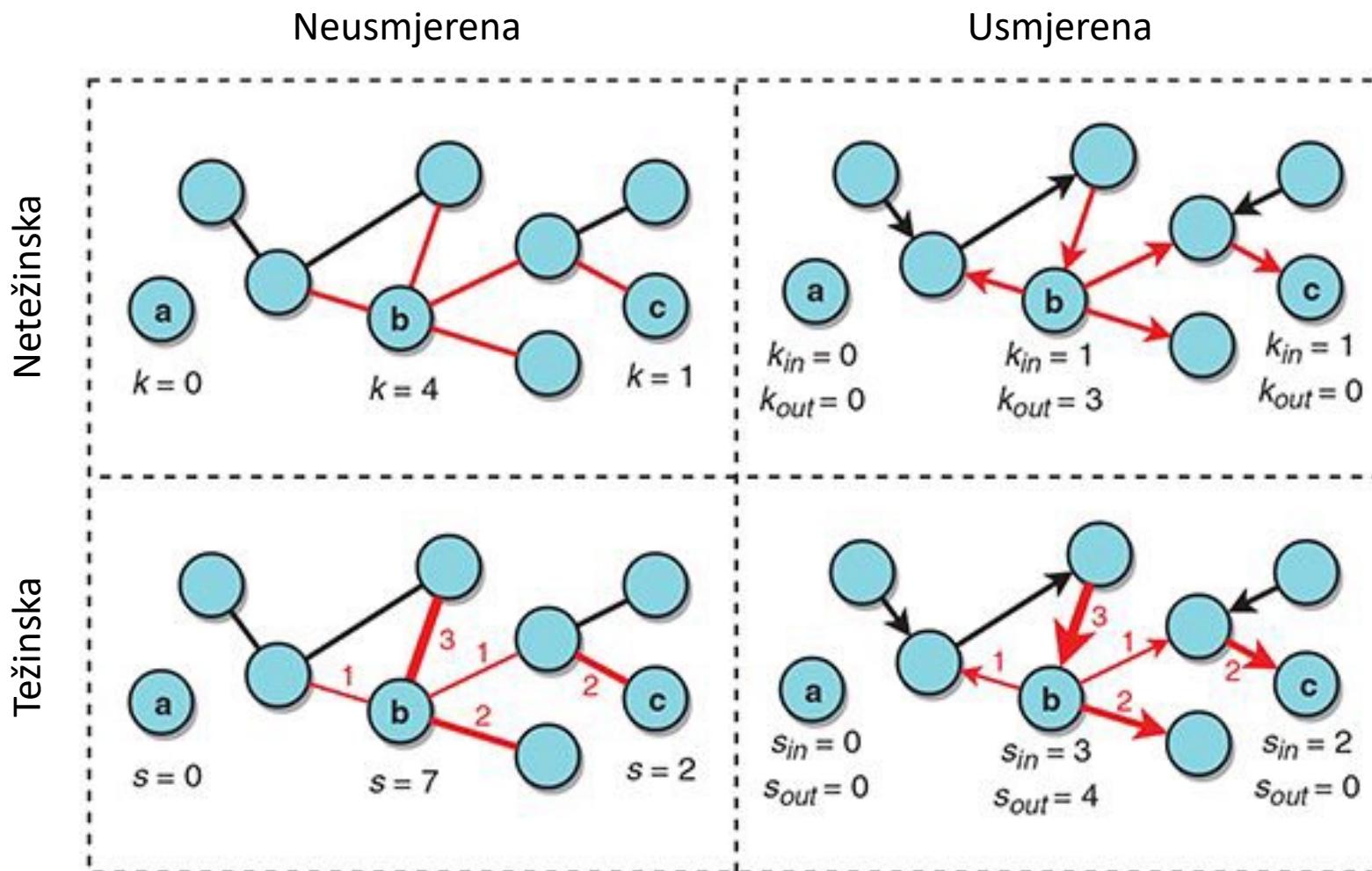


Klika



Skup čvorova tvori kliku (potpuna podmreža) ako postoji
sve moguće veze između čvorova.

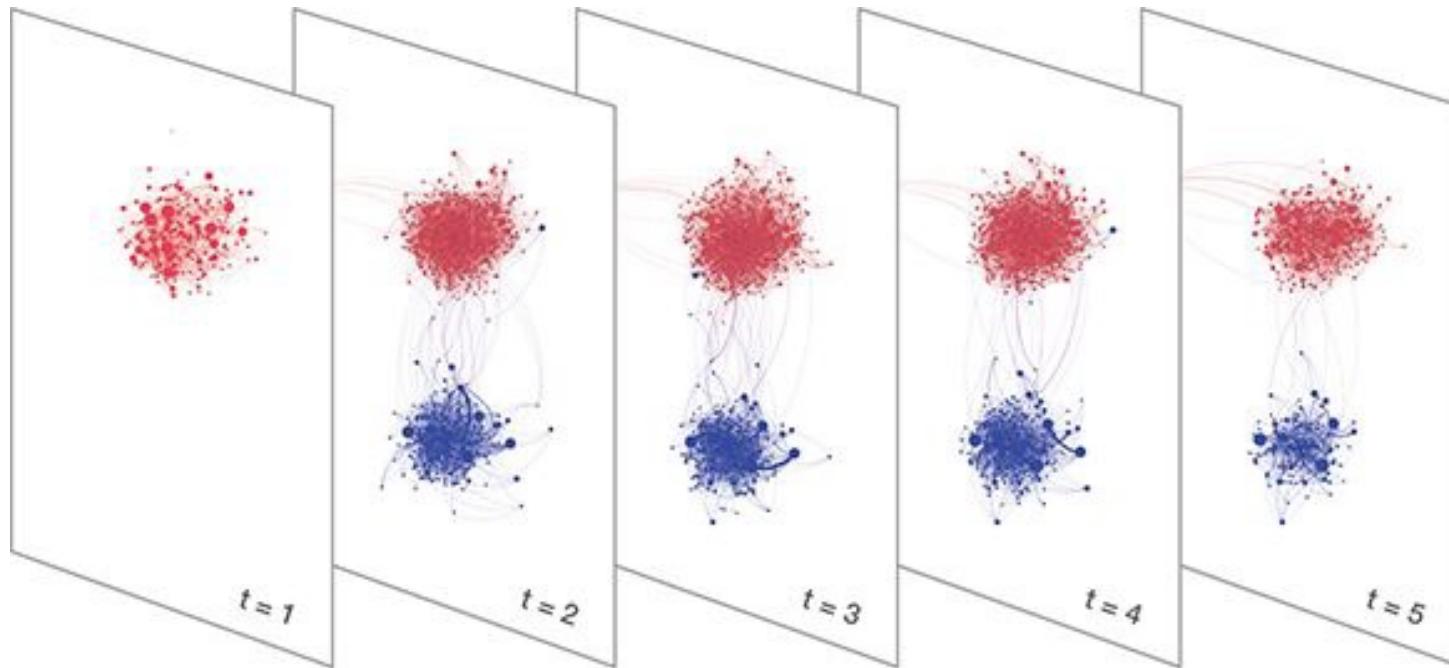
Stupanj čvora



Prosječan stupanj mreže

- $d = \frac{2L}{N(N-1)}$
- $\langle k \rangle = \frac{\sum_i k_i}{N}$
- $\langle k \rangle = \frac{2L}{N} = \frac{dN(N-1)}{N} = d(N - 1)$
- $d = \frac{\langle k \rangle}{N-1}$ - omjer prosječnog i maksimalnog stupnja
- $k_{max} = N - 1$
- k_{in} i k_{out} - ulazni i izlazni stupanj čvora

Temporalne i višeslojne mreže



Temporalna mreža političkih retweetova – svaki sloj predstavlja isti skup čvorova i takva mreža se naziva multipleks

Višeslojna mreža – svaki sloj predstavlja isti skup čvorova i takva mreža se naziva multipleks

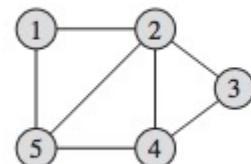
Temporalna mreža – specijalan slučaj multipleksa kod koje se u svakom vremenskom odsječku mijenjaju čvorovi i veze

Multipleks mreže

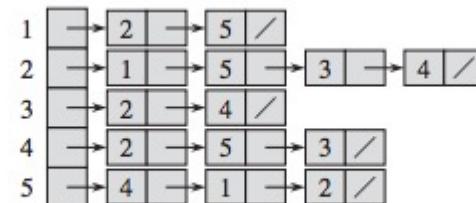
- Unutarslojne veze
- Izvanslojne veze
- Analiza multipleks mreža – agregiranje podataka iz različitih slojeva
- Svaki sloj može biti karakteriziran svojim skupom čvorova i veza – **mreža mreža**
- Primjer: energetska mreža uparena s lokalnom računalnom mrežom
- Kaskadna pogreška

Mrežna reprezentacija

- Graf (a)
- Lista susjedstva (b)
- Matrica susjedstva (c)
- Lista veza



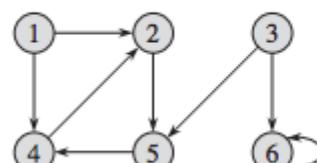
(a)



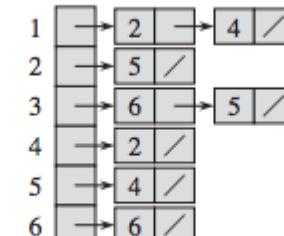
(b)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)



(a)

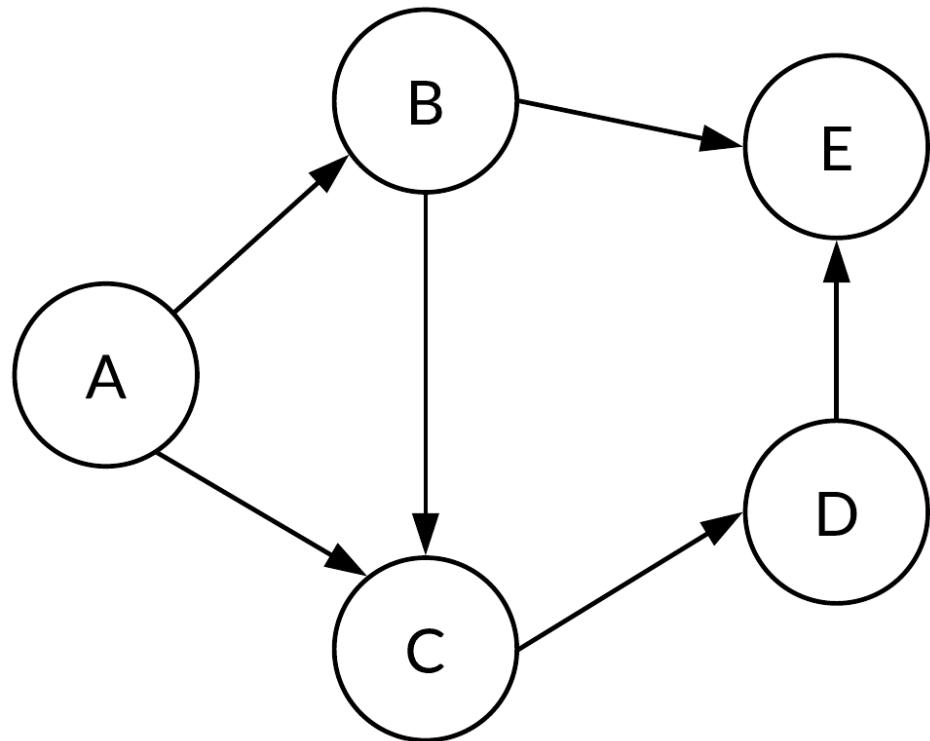


(b)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

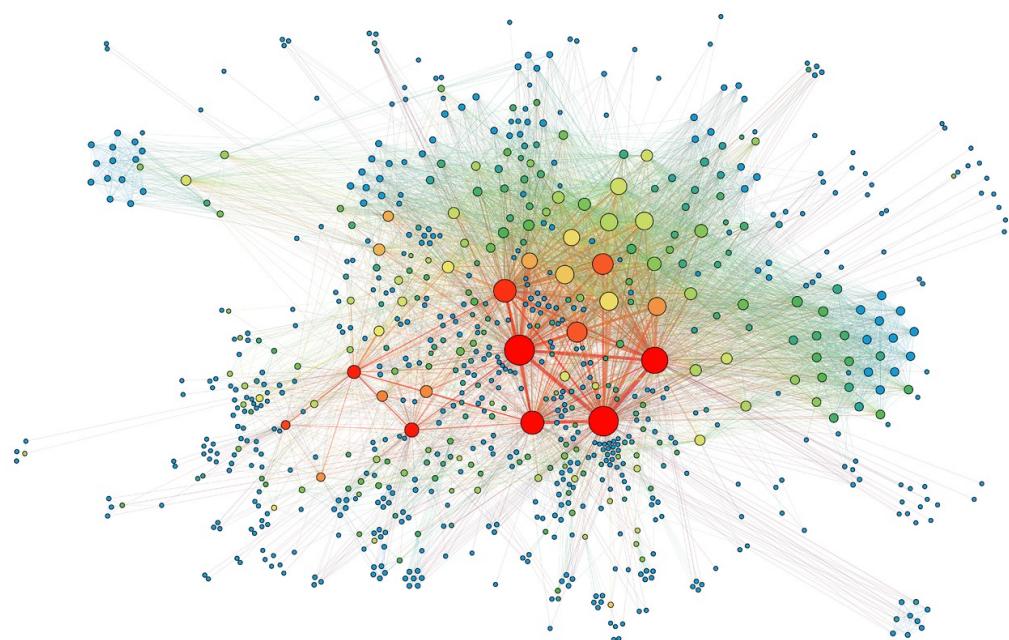
Lista veza



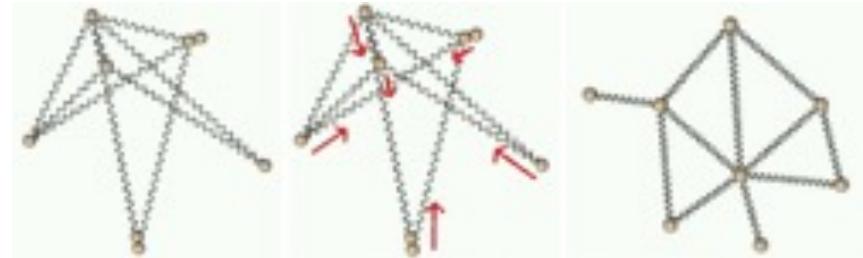
Edge list:

```
graph = [('A', 'B'),  
        ('A', 'C'),  
        ('B', 'C'),  
        ('B', 'E'),  
        ('C', 'D'),  
        ('D', 'E')  
]
```

Crtanje grafa



Force directed algoritam



Kompleksne mreže

2. predavanje

Svojstva mreža

- Sličnost među susjedima
- Najkraći put koji povezuje čvorove
- Trokuti koji povezuju zajedničke susjede

Socijalne mreže

- Svojstva čvorova:
 - Godine
 - Spol
 - Nacionalnost
 - Lokacija
 - Seksualne preferencije
 - Teme interesa



Cambridge Analytica

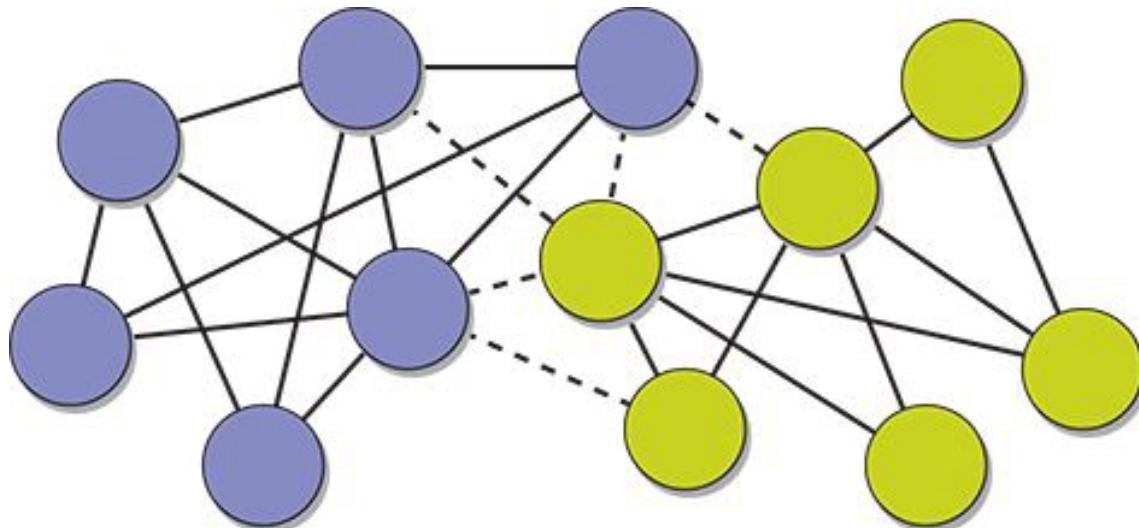
- Ocean model:
 - Openness to experience, conscientiousness, extraversion, agreeableness and neuroticism.
 - Inicijalni kviz
 - Informacije o susjedima
 - 11 US država



Asortativnost

- Povezani čvorovi imaju slična svojstva - asortativnost
- Primjeri:
 - Rođaci mogu živjeti blizu jedni drugih
 - Prijatelji mogu dijeliti iste interese
- Predviđanje osobnih kvaliteta poznajući susjedstvo:
 - Facebook – seksualna orientacija
 - Twitter/X – političke preferencije

Asortativnost



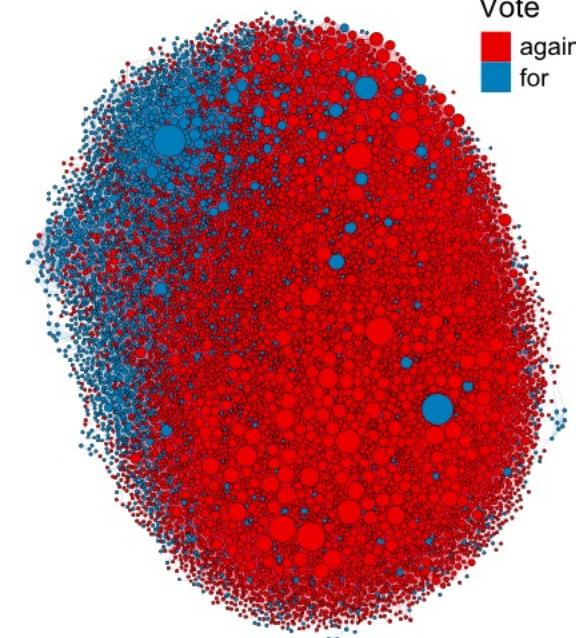
Veća vjerojatnost povezivanje čvorova iste boje

Razlozi za pojavu asortativnosti

- Povezivanje sa sličnima – homofilija
- S vremenom postajemo sličniji (socijalni utjecaj)
- Ljudi su skloni imitiranju drugih
- Echo komore – zatvaranje u krugove sličnih
- Sklonost potvrđivanju (engl. *confirmation bias*)

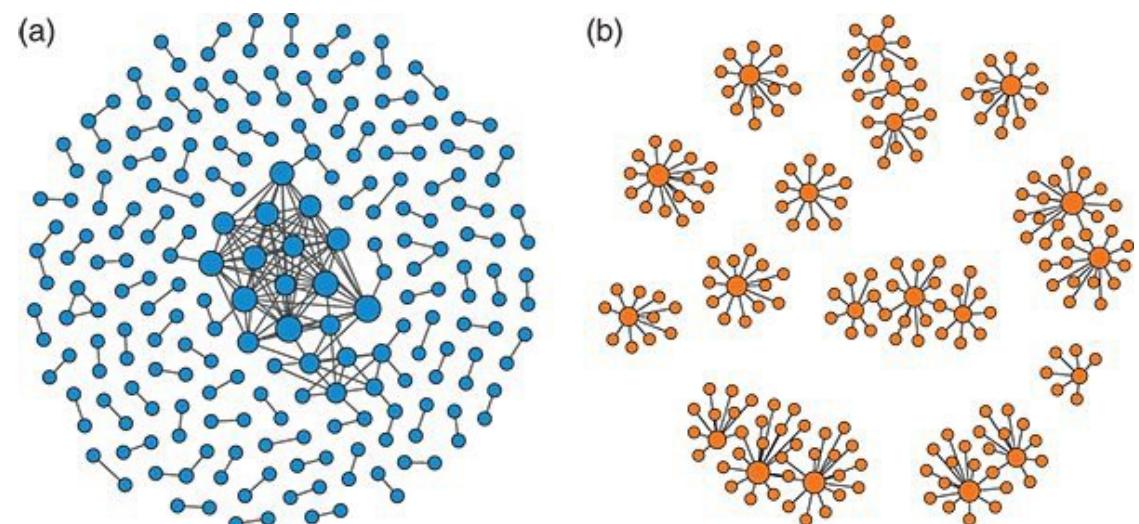
referendum2013.hr

Vote
against
for



Asortativnost stupnja

- Asortativna mreža
 - Čvorovi s velikim stupnjem su povezani sa sličima.
 - Čvorovi s niskim stupnjem su povezani sa sličnim
- Čvorovi s velikim stupnjem (hubovi)
- Jezgra – periferija strategija
- Disortativne mreže (Internet, biološke mreže)



Mjere asortativnosti

- Koeficijent asortativnosti – Pearsonova korelacija između stupnjeva para povezanih čvorova
- Srednji stupanj susjeda čvora i

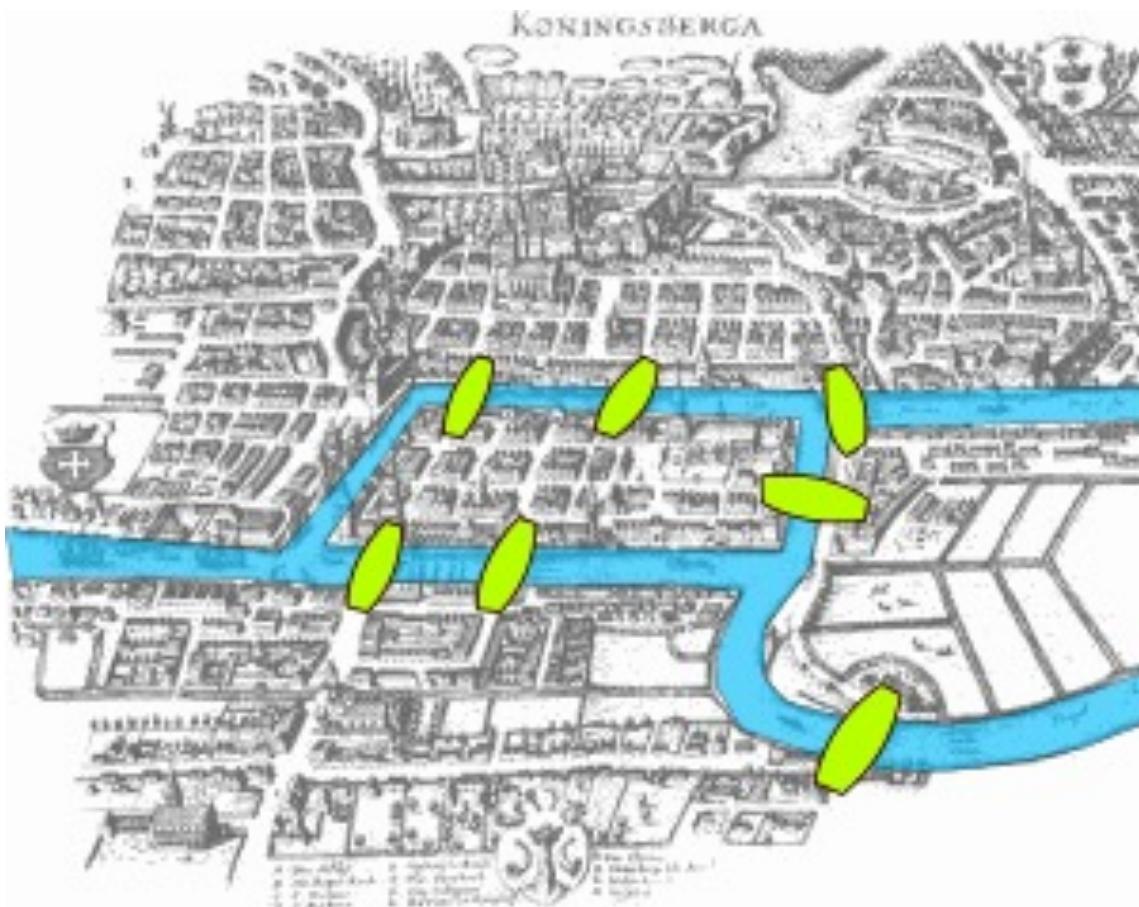
$$k_{nn}(i) = \frac{1}{k_i} \sum_j a_{ij} k_j$$

- $a_{ij} = 1$, ako su i i j susjedi, inače 0
- $\langle k_{nn}(k) \rangle$ k – najbližih susjeda za stupanj k
- Ako $\langle k_{nn}(k) \rangle$ raste s k – asortativna mreža

Šetnja, staza, put, ciklus

- **Šetnja** - konačan ili beskonačan slijed veza koja povezuje slijed čvorova
- **Staza** – šetnja u kojoj svakom vezom prolazimo jedanput
- **Put** – staza u kojoj svakim čvorom prolazimo jedanput
- **Ciklus** – put koji počinje i završava istim čvorom

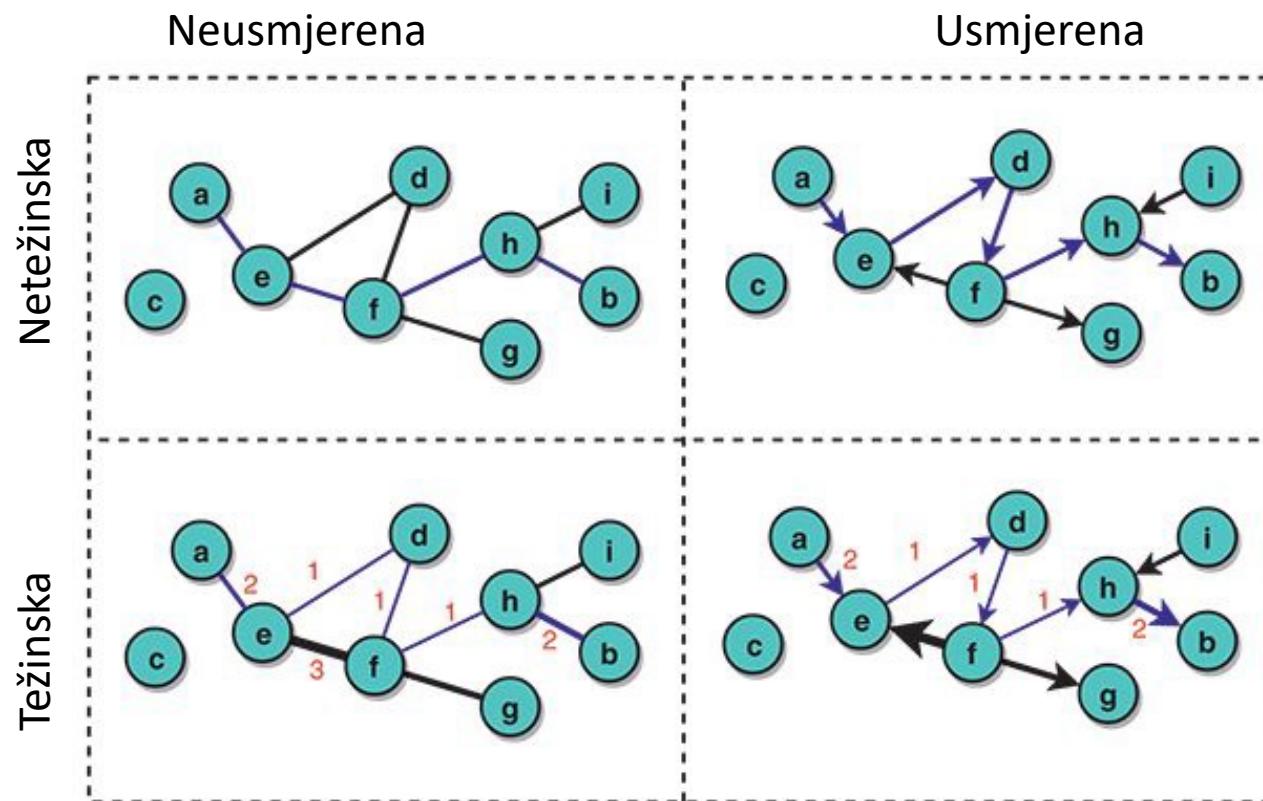
Eulerov put/staza



- Zadatak - Proći svih sedam mostova u Konigsbergu točno jedanput
- Eulerova staza postoji ako:
 - Svi čvorovi osim početnog i krajnjeg imaju paran stupanj

Udaljenost između čvorova u mreži

- Najkraći put
 - Minimalan broj veza koje treba proći u putu između dva čvora

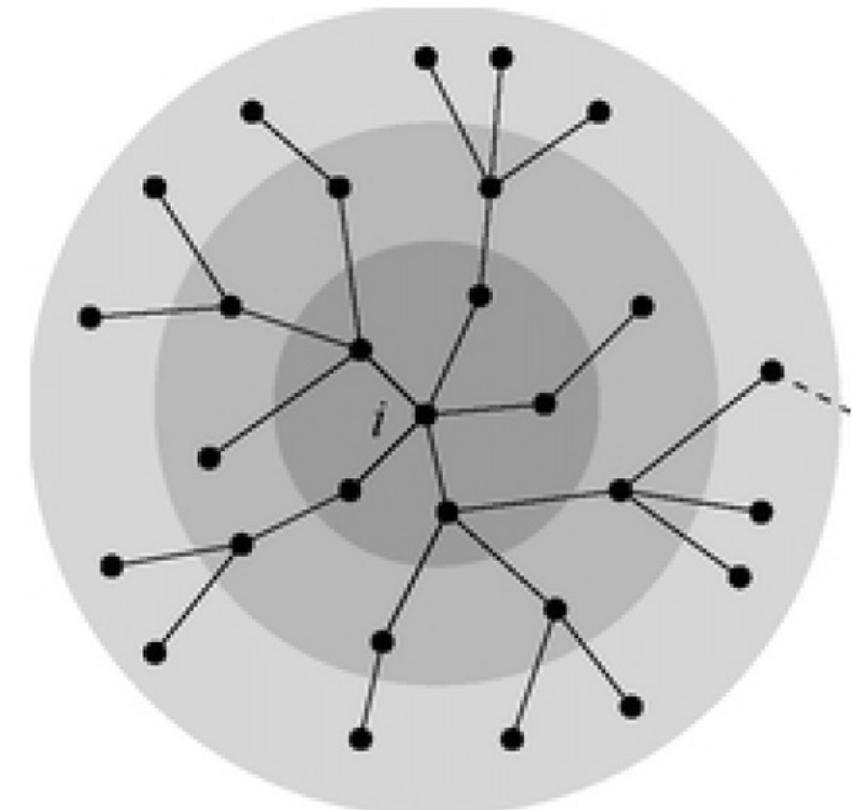


Prosječan najkraći put

- Neusmjereni, netežinska mreža
 - $\langle l \rangle = \frac{\sum_{i,j} l_{ij}}{\binom{N}{2}} = \frac{2 \sum_{i,j} l_{ij}}{N(N-1)}$, gdje je l_{ij} najkraći put između čvorova i i j , a N broj čvorova
- Usmjerena, netežinska mreža
 - $\langle l \rangle = \frac{\sum_{i,j} l_{ij}}{N(N-1)}$
- Neusmjereni mreži u kojoj ne postoji neki putovi
 - $\langle l \rangle = \left(\frac{\sum_{i,j} \frac{1}{l_{ij}}}{\binom{N}{2}} \right)^{-1}$

Diametar mreže

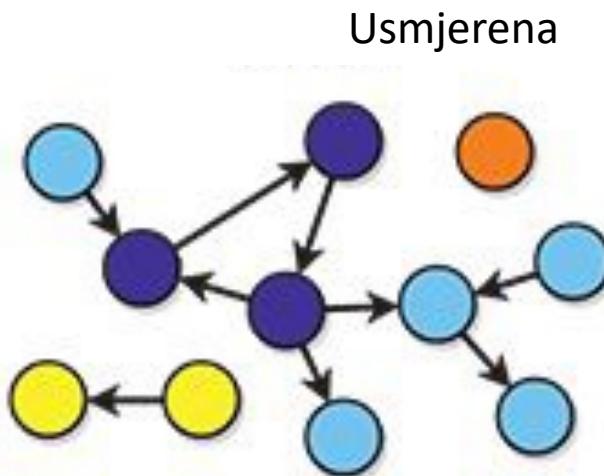
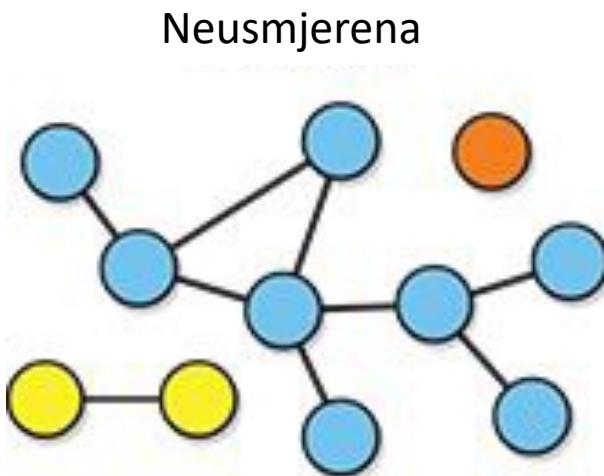
- Maksimalan najkraći put među svim parovima čvorova
- $l_{max} = \max_{i,j} l_{ij}$
- Više komponenti mreže
 - Maksimalna vrijednost diametara pojedinih komponenti



Povezanost mreže i komponente

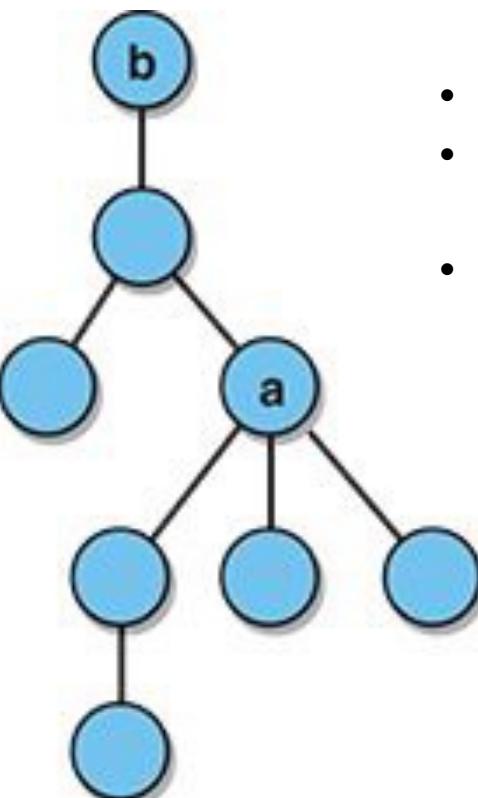
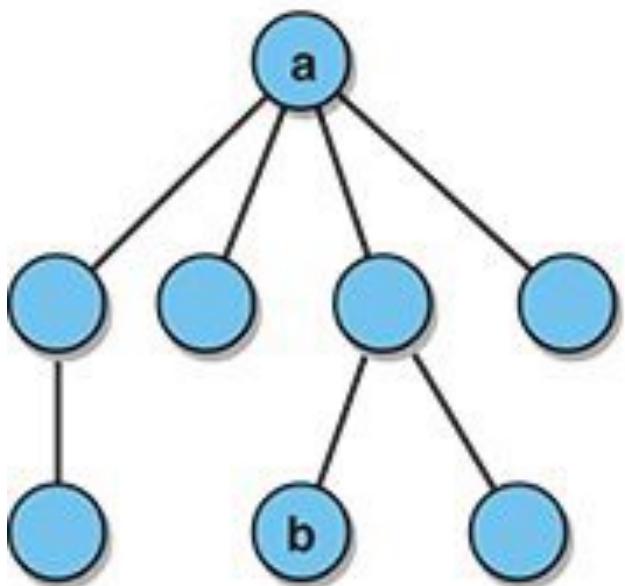
- Povezana mreža
 - Mogućnost dosizanja svakog čvora iz bilo kojeg čvora
- Nepovezana mreža
 - Više od jedne komponente
 - Komponenta – podmreža od jednog ili više čvorova gdje postoji put između bilo kojeg para čvorova
 - Gigantska komponenta – najveća povezana komponenta u mreži

Komponente



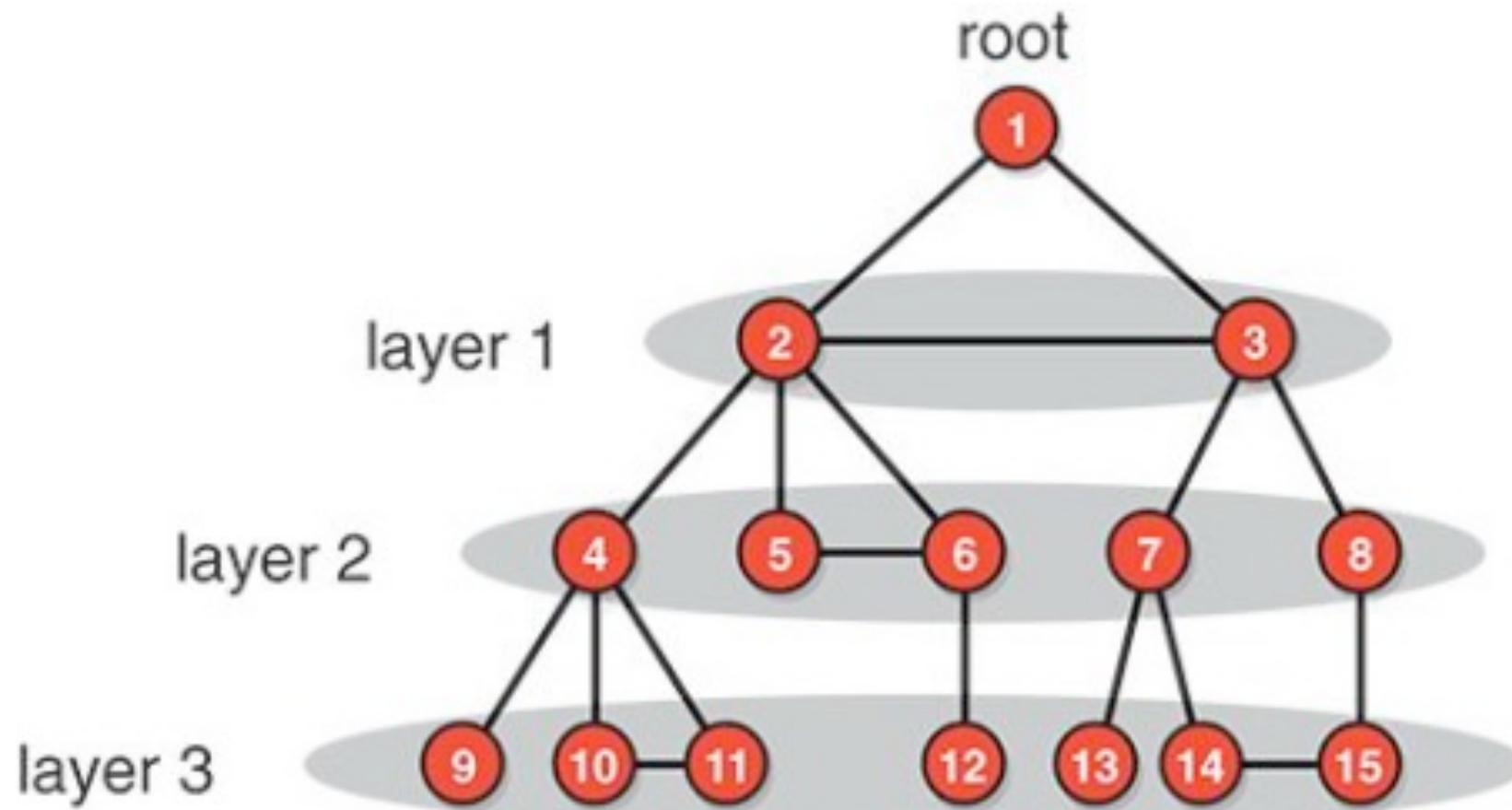
- Bojama označene komponente
- Usmjerena mreža:
 - Slabo povezana komponenta – kada ignoriramo usmjerenost (gigantska – različite nijanse plave)
 - Strogo povezana komponenta (najveća – tamna plava)

Stabla



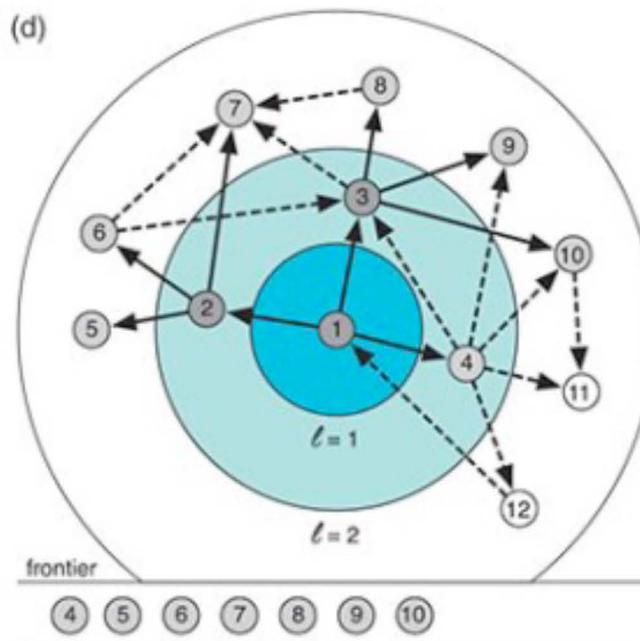
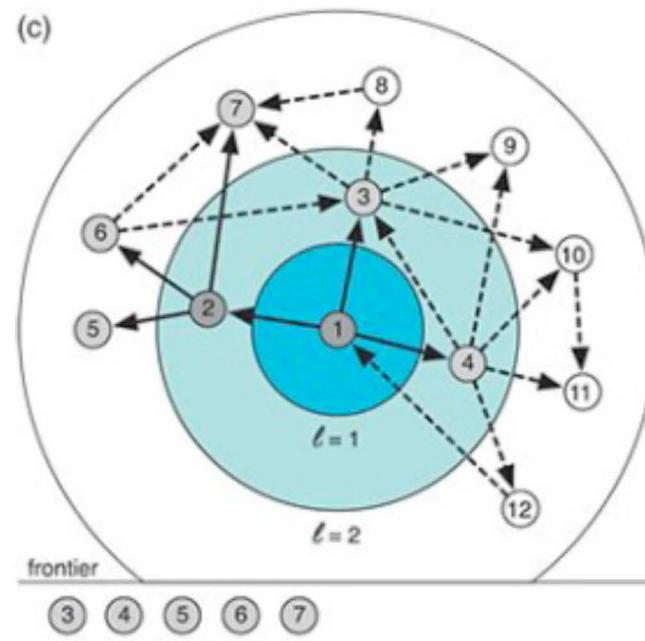
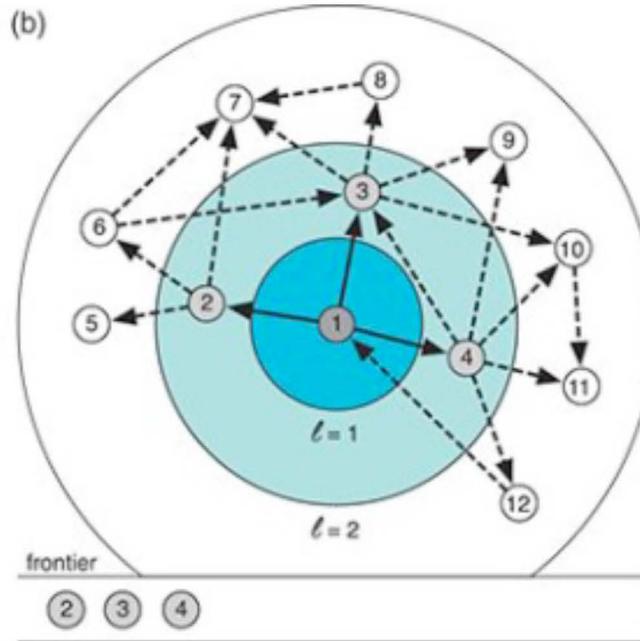
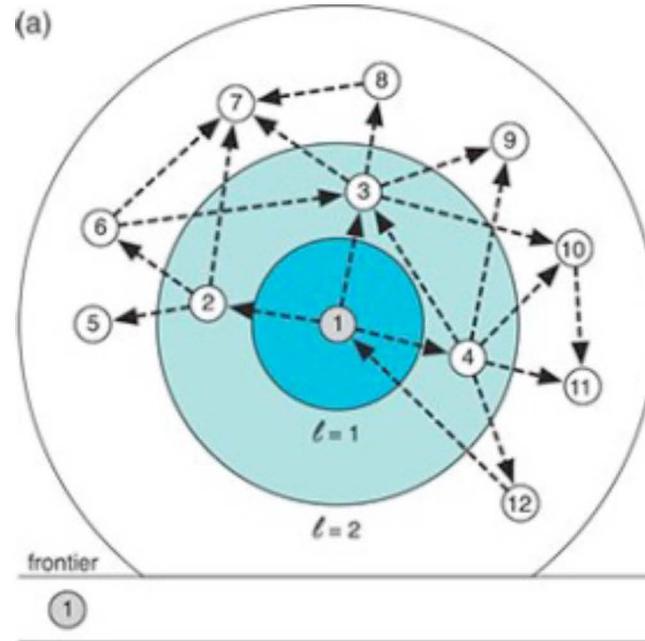
- Povezane mreže koje bi brisanje bilo koje veze bi razdvojilo mrežu na dvije komponente
- $L = N - 1$, nema ciklusa
- Na slici dva prikaza stabla sa čvorovima a i b kao korijenima stabala
- Stabla imaju hijerarhiju

Pronalazak najkraćeg puta - BFS



Breadth first search

- Nerekurzivni algoritam za BFS grafa G krenuvši od vrha v :
 - procedura $\text{BFS}(G, i)$ označi i kao obiđen
 - ispiši i
 - stavi i u red Q
 - dok ima elemenata u redu Q
 - uzmi i iz reda Q
 - za sve neobiđene vrhove j susjedne vrhu i
 - ako j nije obiđen
 - označi j kao obiđen
 - ispiši j
 - stavi j u red Q
- Složenost $\Theta(|E|+|V|)$



BFS – najkraća udaljenost

Krenemo s početnim čvorom s i stavimo ga u red Q .

Njegova udaljenost je $I(s,s)=0$, a za sve ostale čvorove postavimo -1.

Inicijaliziramo stablo najkraćeg puta sa čvorovima bez veza.

U svakoj iteraciji posjetimo sljedeći čvor u redu.

Za svakog neobiđenog susjeda j od i ponavljamo ova tri koraka

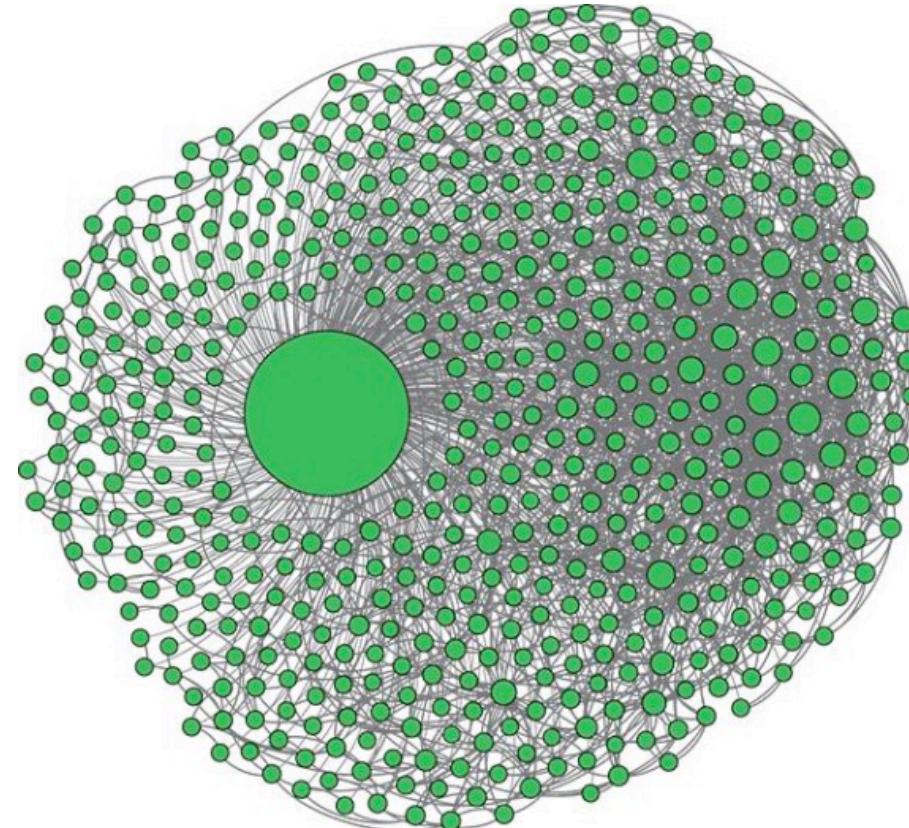
1. Stavimo j u red Q .
2. Postavimo udaljenost j od s na $I(s,j) = I(s,i) + 1$.
3. Dodamo direktnu vezu $(i \rightarrow j)$ u stablo najkraćeg puta

Put pretražimo unatrag od odredišnog do izvorišnog čvora (stablo)

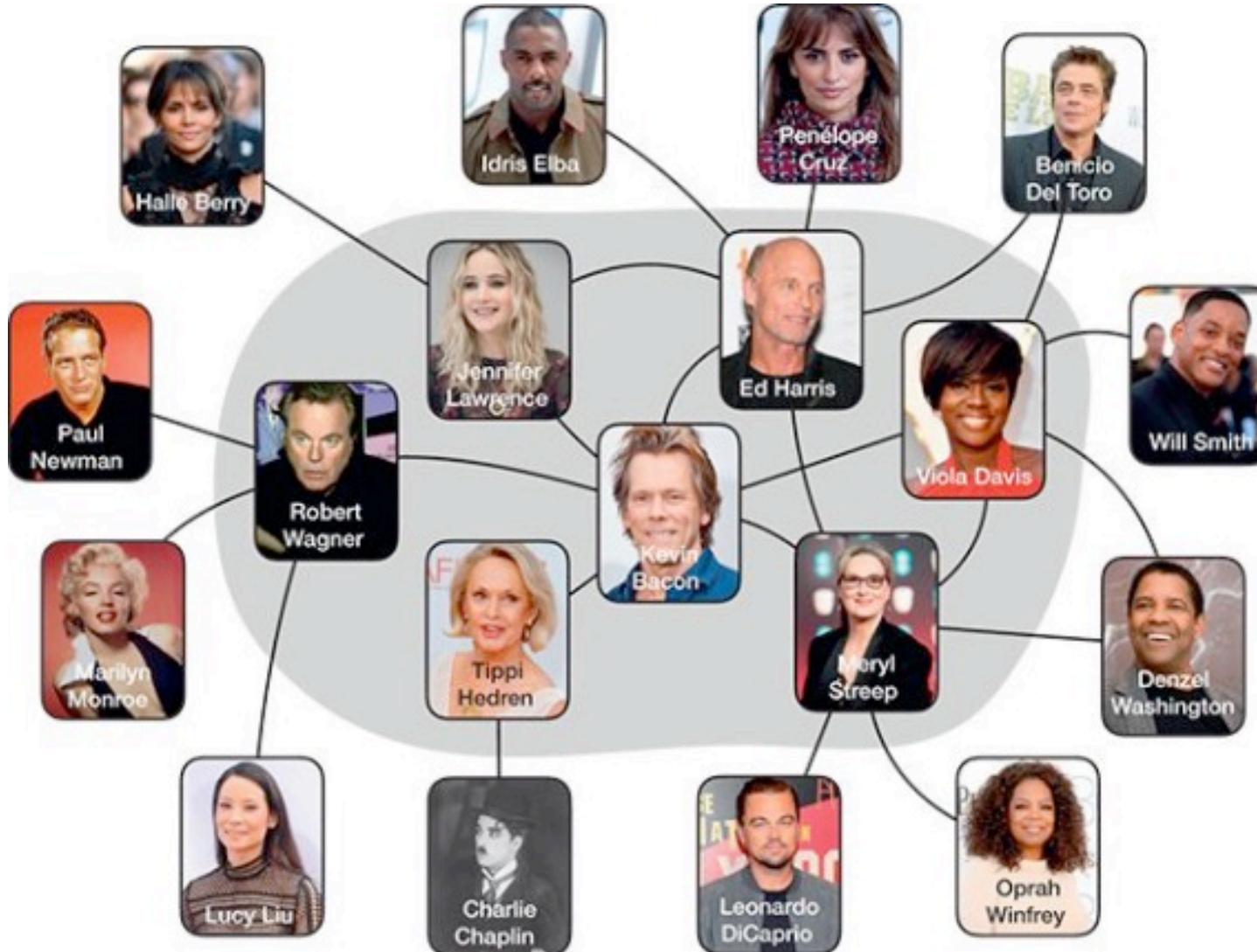
Network	Nodes (N)	Links (L)	Average path length ($\langle l \rangle$)	Clustering coefficient (C)
Facebook Northwestern Univ.	10,567	488,337	2.7	0.24
IMDB movies and stars	563,443	921,160	12.1	0
IMDB co-stars	252,999	1,015,187	6.8	0.67
Twitter US politics	18,470	48,365	5.6	0.03
Enron email	87,273	321,918	3.6	0.12
Wikipedia math	15,220	194,103	3.9	0.31
Internet routers	190,914	607,610	7.0	0.16
US air transportation	546	2,781	3.2	0.49
World air transportation	3,179	18,617	4.0	0.49
Yeast protein interactions	1,870	2,277	6.8	0.07
<i>C. elegans</i> brain	297	2,345	4.0	0.29
Everglades ecological food web	69	916	2.2	0.55

Socijalna udaljenost

- Očekivanje koliko su dva čvora u mreži udaljena
- U mreži cesta i energentskog napajanja mogu biti jako daleko
- Mreža koautorstva – Erdosov broj



Six degrees of Kevin Bacon



Mali svijet

- Otkrivanje zajedničkih prijatelja
- „Svijet je mali!!”
- Socijalne mreže imaju mali dijаметар i još manji prosječni najkraći put
- 6 stupnjeva razdvojenosti

Stanley Milgram

- 160 učesnika iz Nebraska i Kanzasa su slali pismo do ciljane osobe u Massachusettsu
- Pismo je slano nekom koga znamo i tko bi mogao znati ciljanu osobu
- 42 pisma (26 %) došla su do cilja
- Prosječni najkraći put je bio 6
- Facebook 2011 – 4.74
- Kratak put $\langle l \rangle = \log N$
- N – veličina mreže

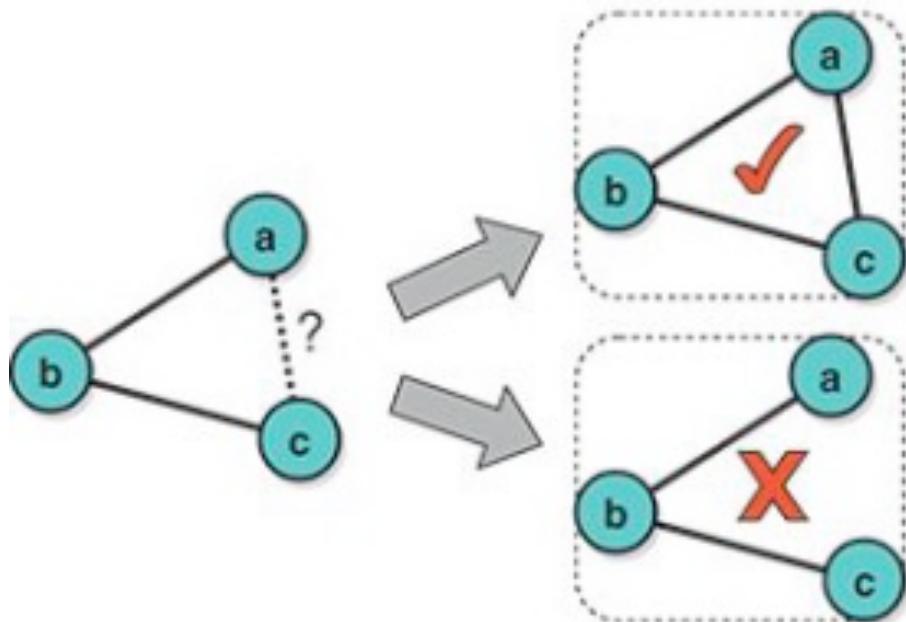


Prijateljev prijatelj

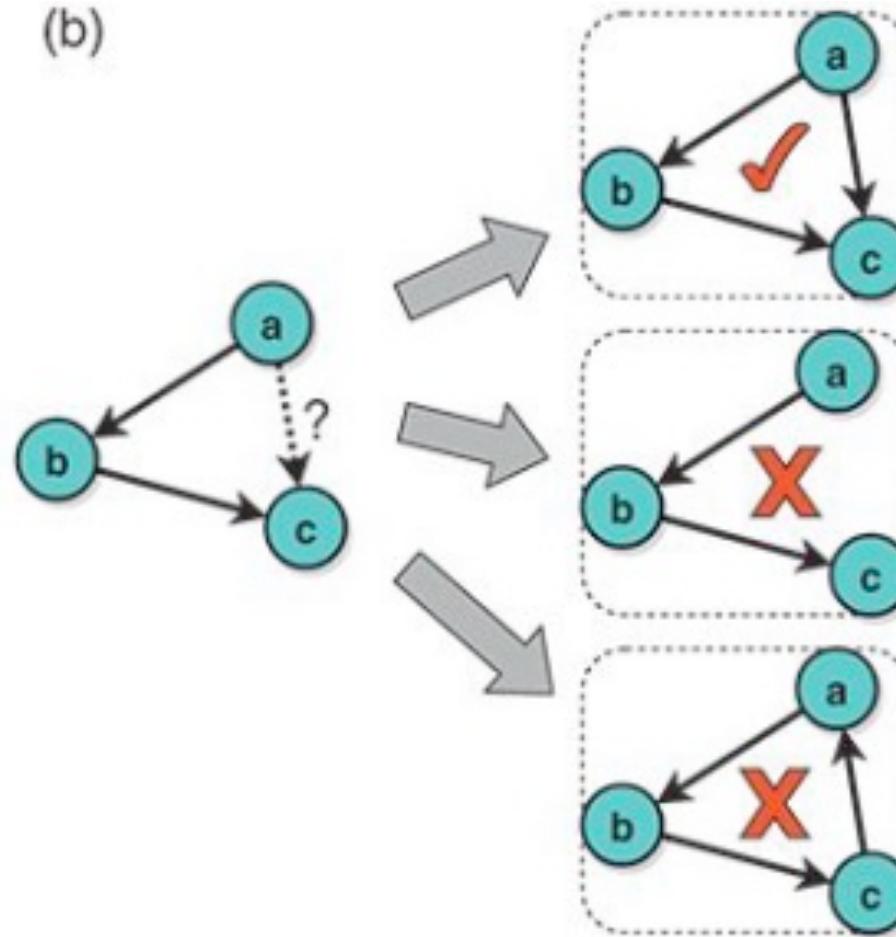
- Primjer - ako su Andrej i Zoran prijatelji s Gordanom, vjerojatno su i sami prijatelji
- „Prijatelj moga prijatelja je moj prijatelj”
- Tri prijatelj tvore trokut
- Triada - skup od tri čvora – kada su svi povezani -> trokut

Trokuti i triade

(a)



(b)



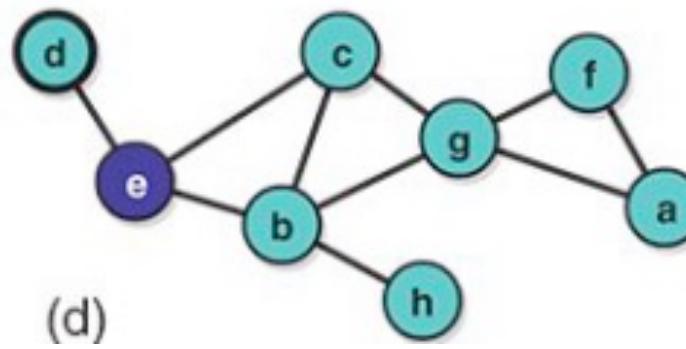
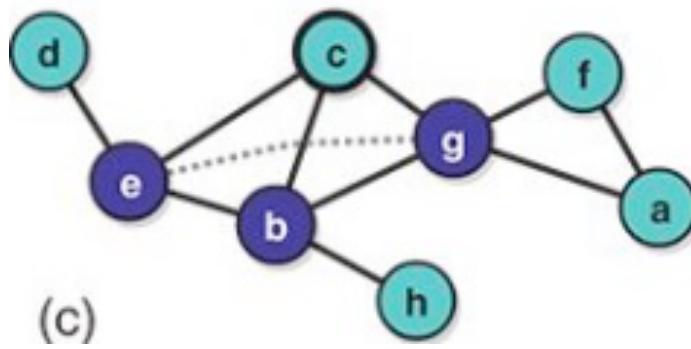
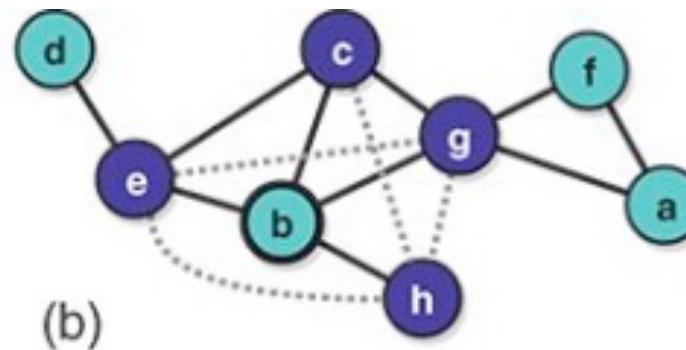
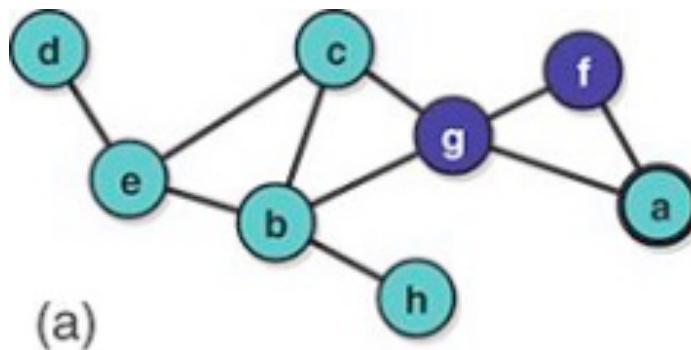
Koeficijent klasteriranja čvora

- Postotak parova susjeda promatranog čvora koji su međusobno povezani
- Omjer broja trokuta koji uključuju čvor i maksimalan broj trokuta u kojima može sudjelovati
- $C(i) = \frac{\tau(i)}{\tau_{max}(i)} = \frac{\tau(i)}{\binom{k_i}{2}} = \frac{2\tau(i)}{k_i(k_i - 1)},$
- $\tau(i)$ – broj trokuta koji uključuje i
- $\tau_{max}(i)$ – maksimalan broj trokuta koji uključuje i i njegove susjede k_i
- $k_i > 1$

Koeficijent klasteriranja mreže

- $C = \frac{\sum_{i:k_i>1} C(i)}{N_{k>1}}$
- Čvorovi za koje je $k < 2$ ne ulaze u računanje prosječno koeficijenta klasteriranja

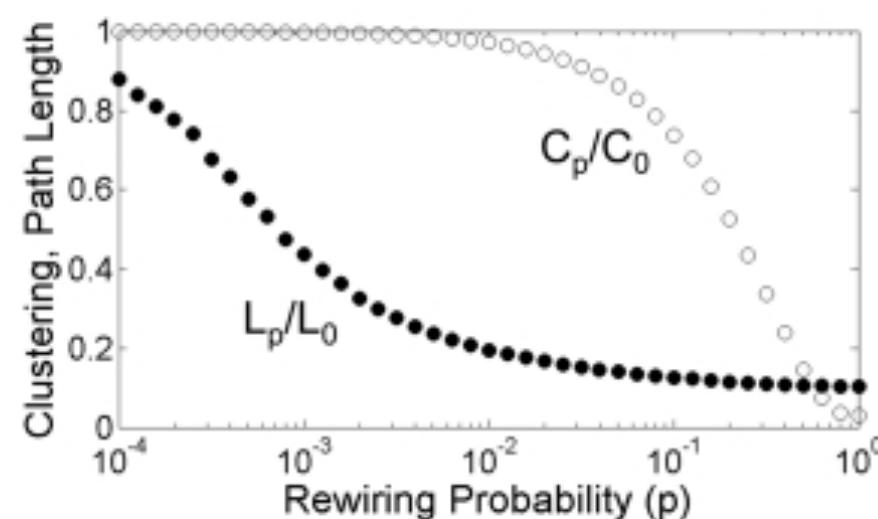
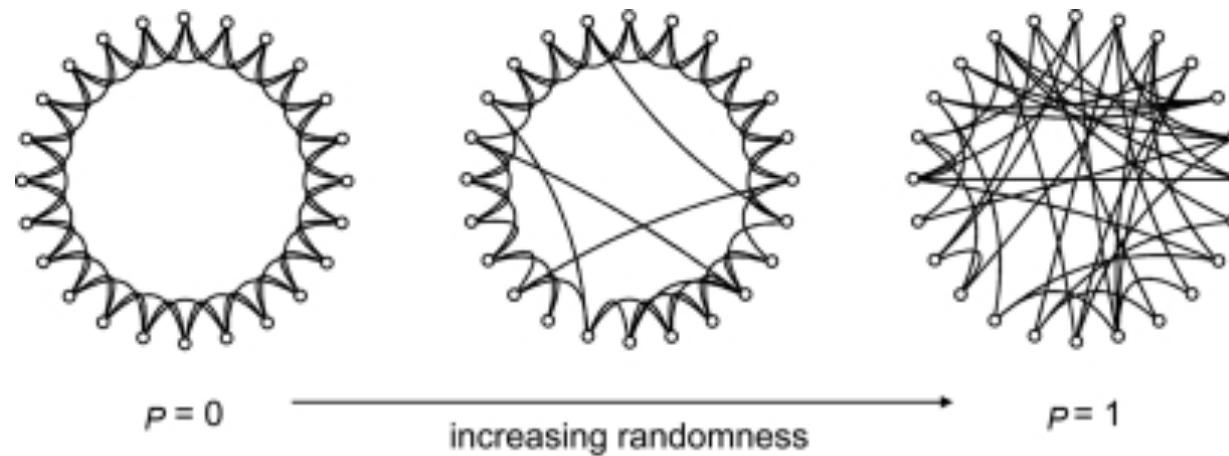
Primjer koeficijenta klasteriranja



Koeficijent klasteriranja

- Definicija se odnosi samo na neusmjerenе мreže
- Za usmjerenе потребно је definirati tipove trokuta
- Socijalne mreže obično imaju velik koeficijent klasteriranja (preporuke)

Watts-Strogatz (1998) – Mali svijet



Kompleksne mreže

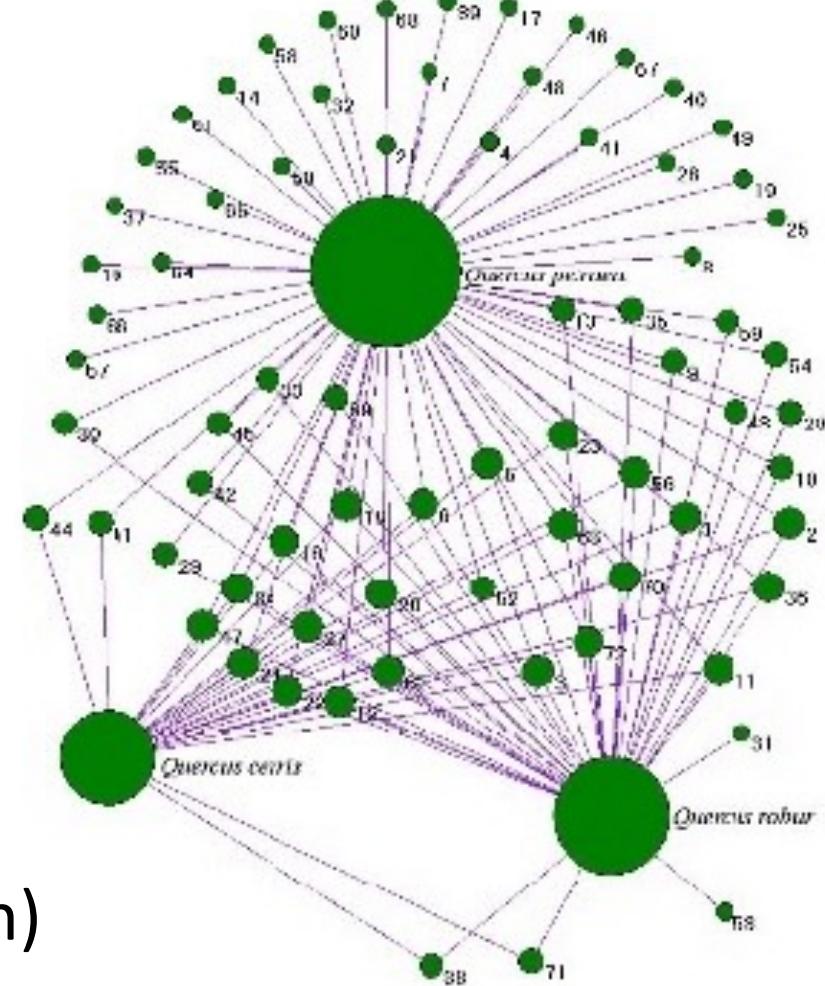
3. predavanje

Što im je zajedničko ?

- Elon Mask
- Khabby Lame
- Christiano Ronaldo
- Blac Chyna
- Bill Gates
- James Stephen Donaldson

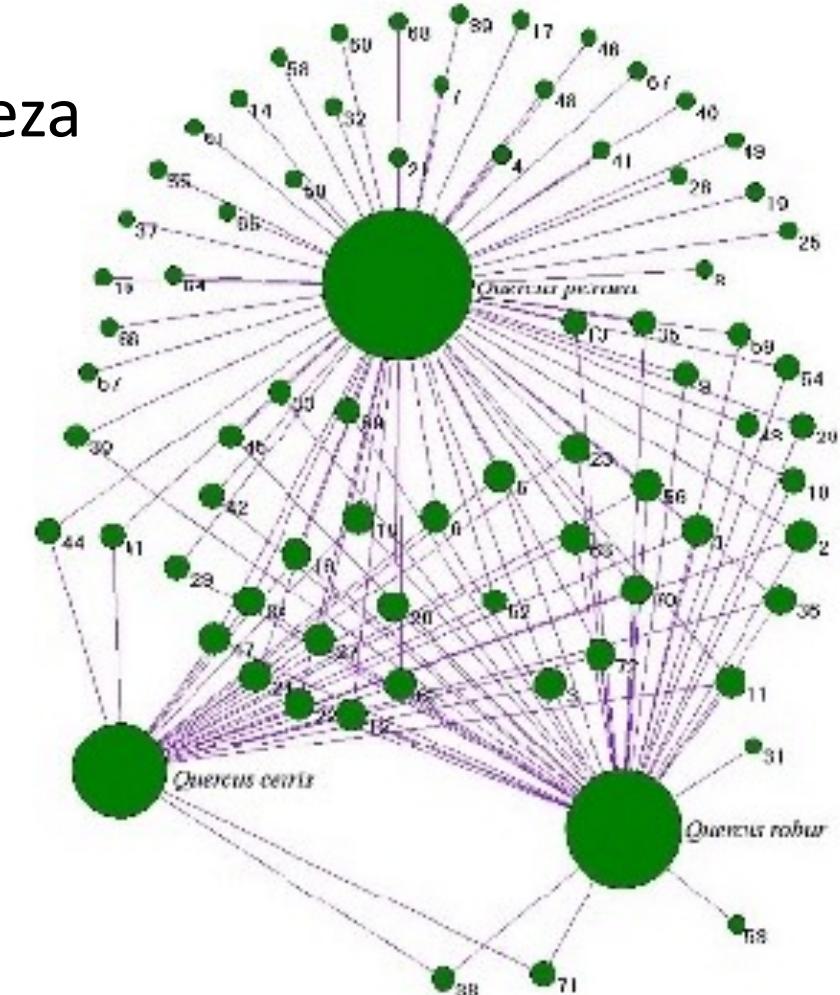
Hubovi

- Primjeri
 - Aerodromi
 - Amsterdam
 - Doha
 - Singapore
 - Istanbul
- Popularne stranice
- Influenceri (X, Instagram, TikTok, LinkedIn)



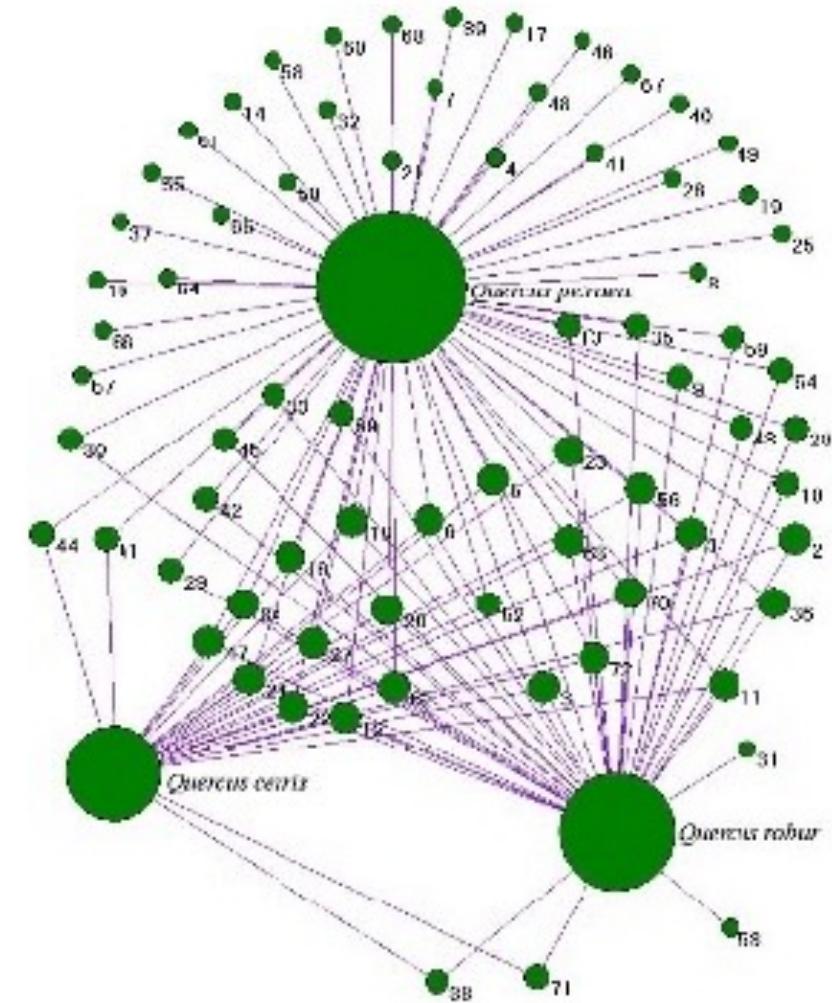
Heterogenost mreža

- Raznolikost u svojstvima i ulogama čvorova i veza
- Izvor heterogenosti – stupanj čvora
- Mjerenje centralnosti
- Hubovi – čvorovi visokog stupnja



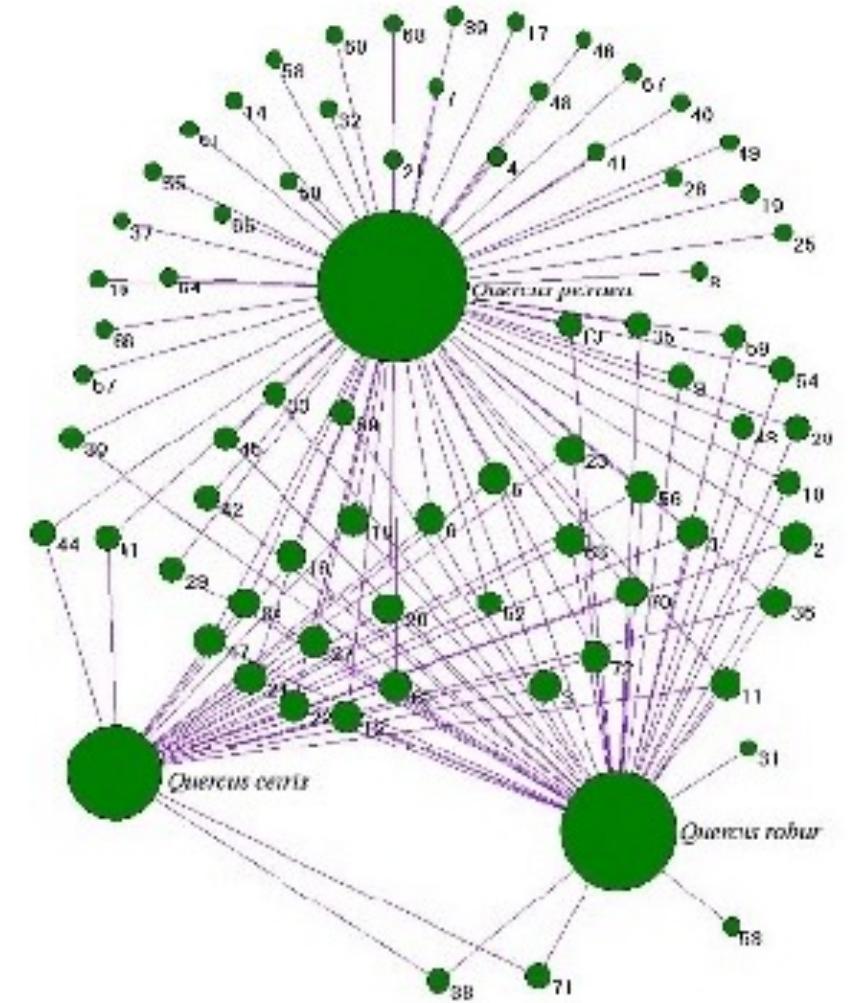
Osnovne mjere centralnosti

- Stupanj (*degree*)
- Bliskost (*closeness*)
- Međupoloženost (*betweenness*)



Stupanj

- Primjeri:
 - Broj direktno dostupnih aerodroma od polaznog
 - Broj socijalnih veza
- Prirodna mjera centralnosti
- Prosječan stupanj – ne odražava heterogenost



Bliskost

- Koliko je pojedini čvor blizak ostalima
- $g_i = \frac{1}{\sum_{j \neq i} l_{ij}}$, gdje je l_{ij} udaljenost između i i j
- $\tilde{g}_i = (N - 1)g_i = \frac{N-1}{\sum_{j \neq i} l_{ij}} = \frac{1}{\sum_{j \neq i} l_{ij}/(N-1)}$ alternativna formulacija
 - Mjera postaje usporediva među različitim mrežama
 - $\sum_{j \neq i} l_{ij}/(N - 1)$ je zapravo prosječna udaljenost čvora i od ostatka mreže
 - Inverzija prosječne udaljenosti čvora

Međupoloženost

- Difuzni procesi u mreži
 - Prenošenje informacije kroz socijalne mreže
 - Promet dobara kroz luku
 - Širenje epidemije u mreži fizičkih kontakata
- Uključenost čvora u te procese – međupoloženost
 - Računamo koliko najkraćih puteva prolazi čvorom

Međupoloženost

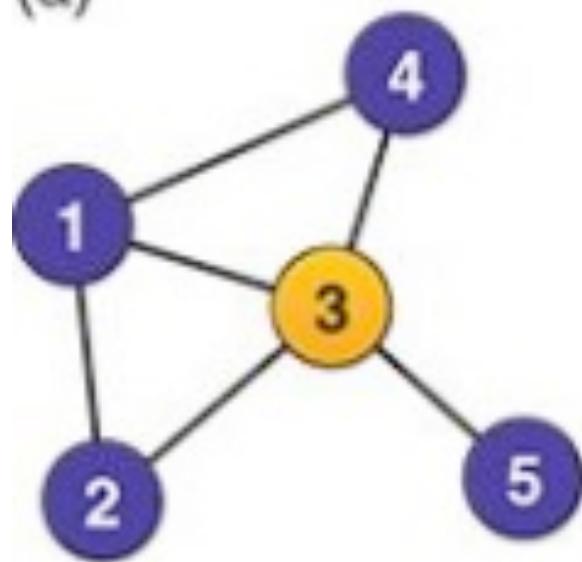
- Može postojati više najkraćih puteva između dva čvora u mreži iste duljine
- σ_{hj} ukupan broj najkraćih puteva između čvorova h i j
- $\sigma_{hj}(i)$ ukupan broj najkraćih puteva koji prolaze čvorom i
- međupoloženost je definirana kao

$$b_i = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

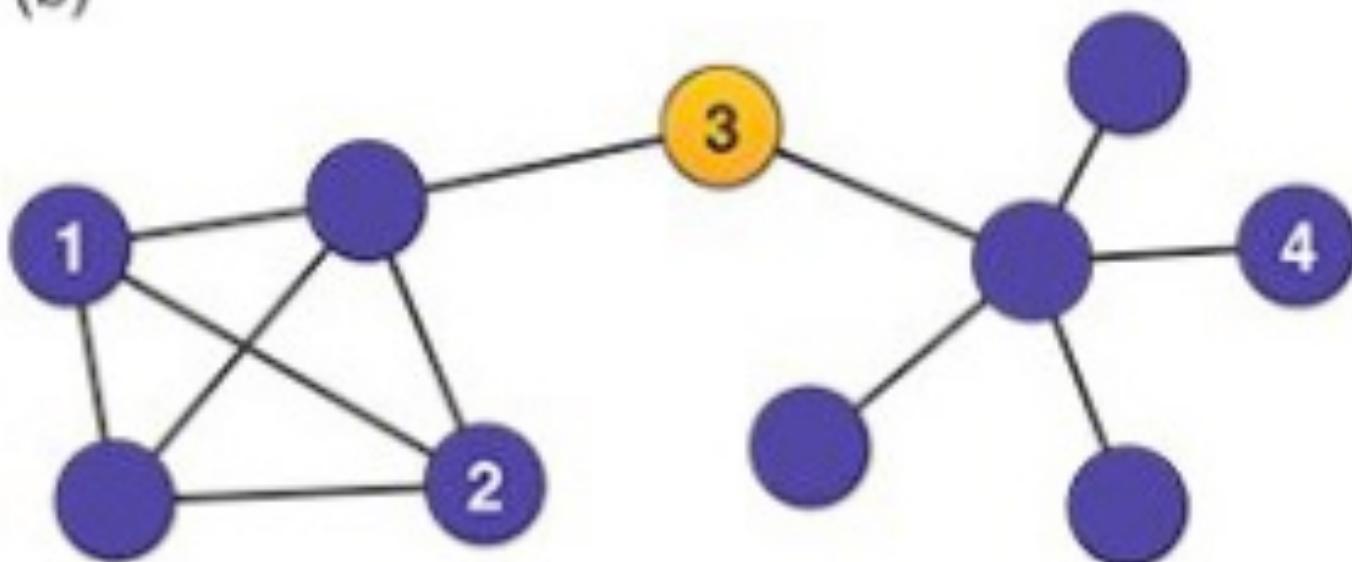
- Normalizacija s $\binom{N-1}{2}$ - ukupan broj parova čvorova bez i

Međupoloženost

(a)



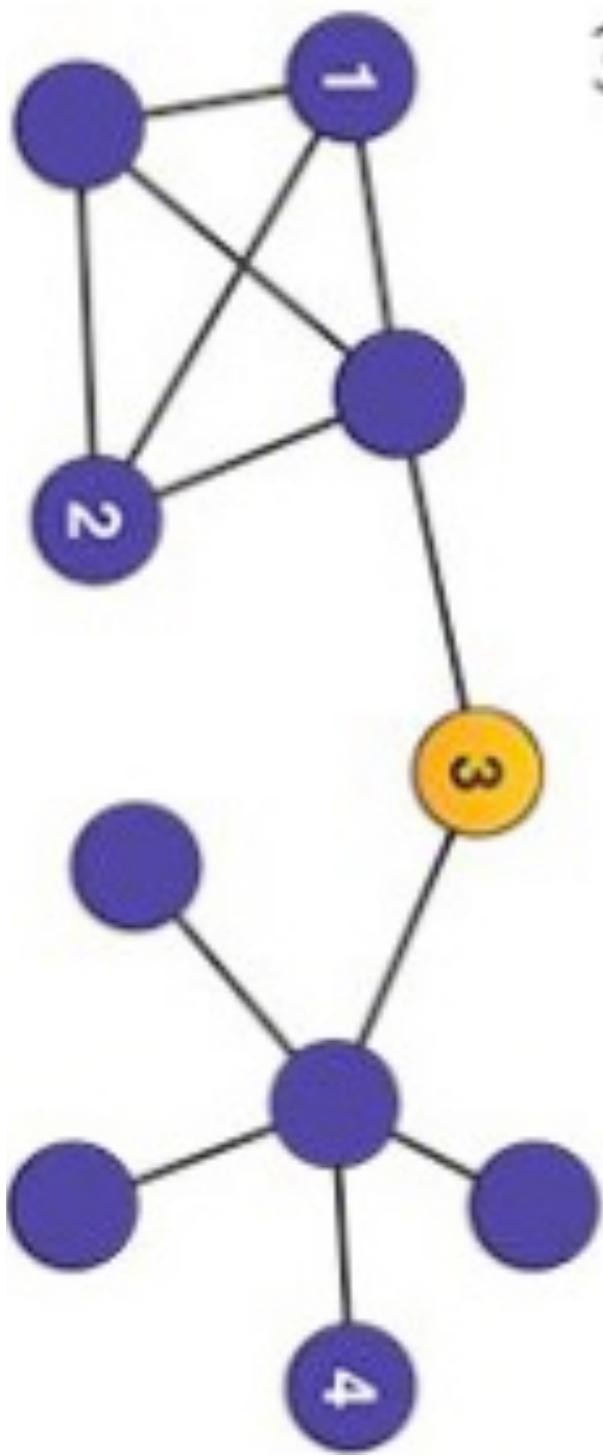
(b)



Međupoloženost

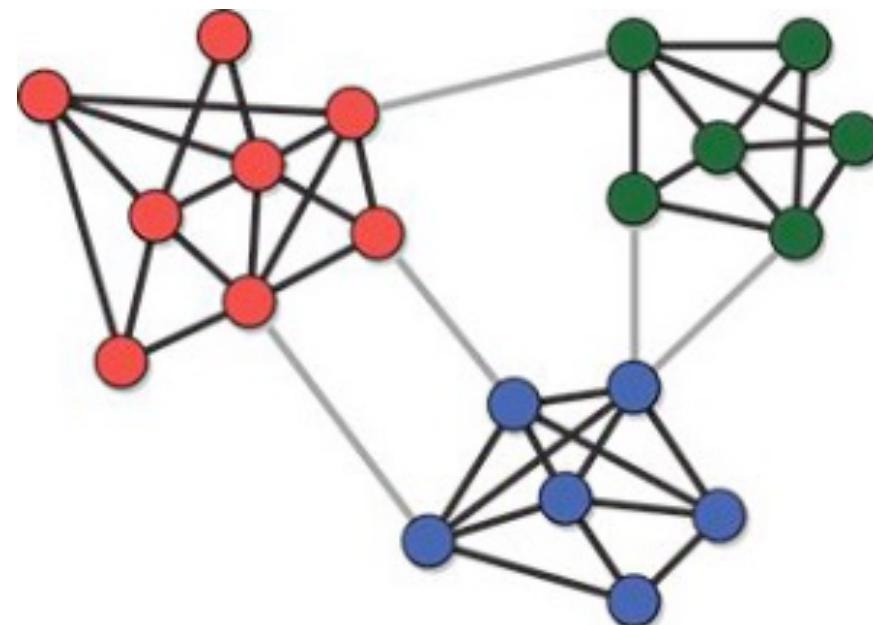
- Čvor s velikim iznosom međupoloženosti ne mora imati veliki stupanj
- Čvorovi koji premošćuju regije mreže imaju visoku međupoloženost

(b)



Međupoloženost veza

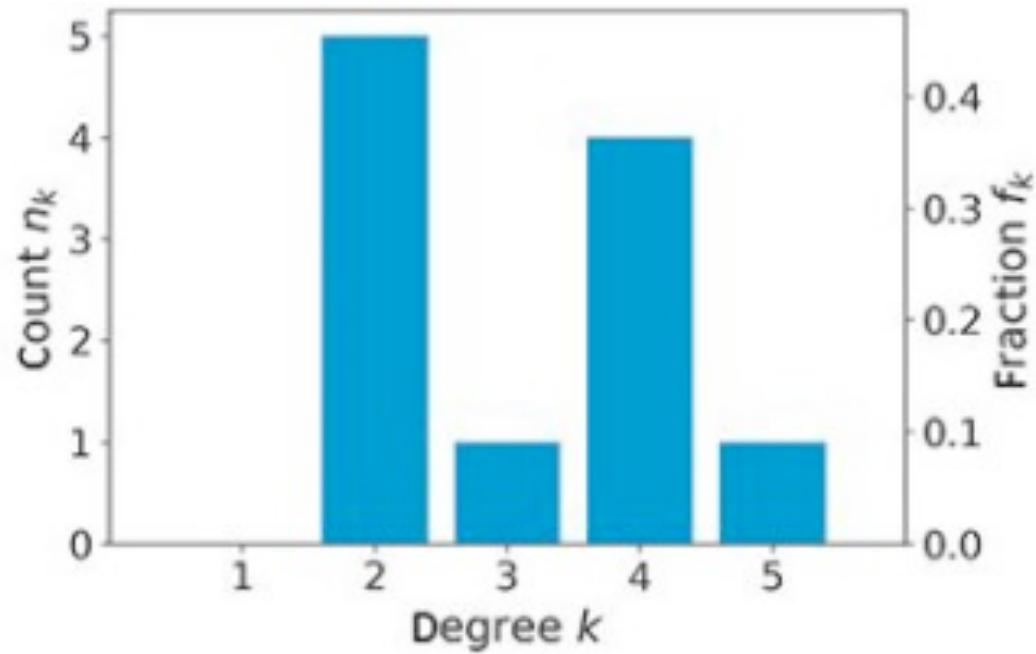
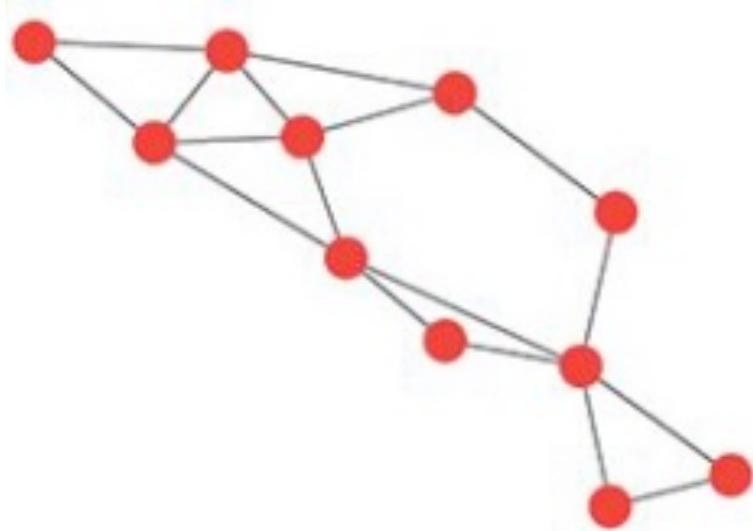
- Dio svih najkraćih putova svih čvorova koji prolaze tom vezom
- Veze s velikom međupoloženošću često povezuju povezane regije – zajednice
- Uklanjanjem veza s visokom međupoloženošću omogućava određivanje zajednica



Distribucije centralnosti

- Veliki grafovi – svatko, bez obzira koliko popularan, vezan je uz manji dio mreže
- Statistički pristup:
 - fokus na klase čvorova i veza koje imaju slična svojstva
 - ne pojedini čvor ili veza
- Statistička distribucija – koliko elemenata (čvorova ili veza) ima istu vrijednost
- Utvrđivanje klasa elemenata iz distribucije

Distribucija stupnja za manju mrežu

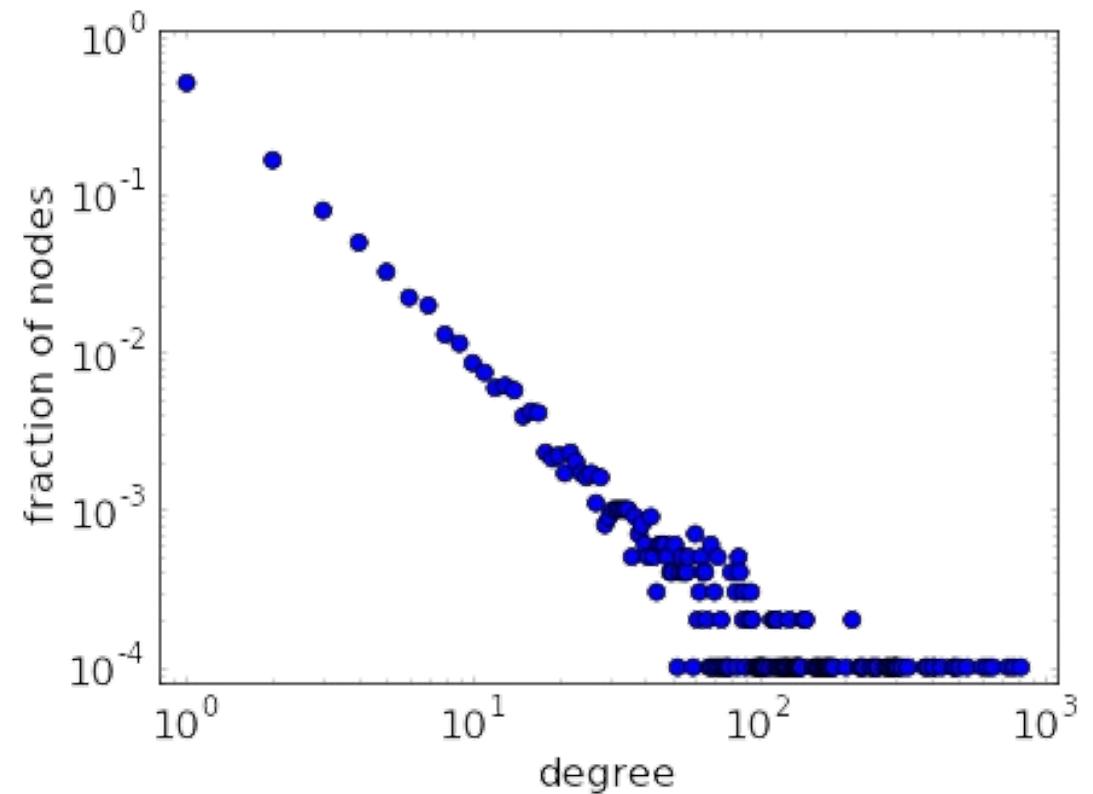


Komplementarna kumulativna distribucija

- $P(x)$ - vjerojatnost da događaj ima veću vjerojatnost od x
- $P(x) = \sum_{v \geq x} f_v$
- Koristi se često kada je raspon varijabilnosti širok
 - Primjer: distribucija stupnja u kompleksnim mrežama
 - Visoke vrijednosti su rijetke -> šum u repu distribucije
 - Kumulativna distribucija to izgladi

Distribucija stupnja čvora

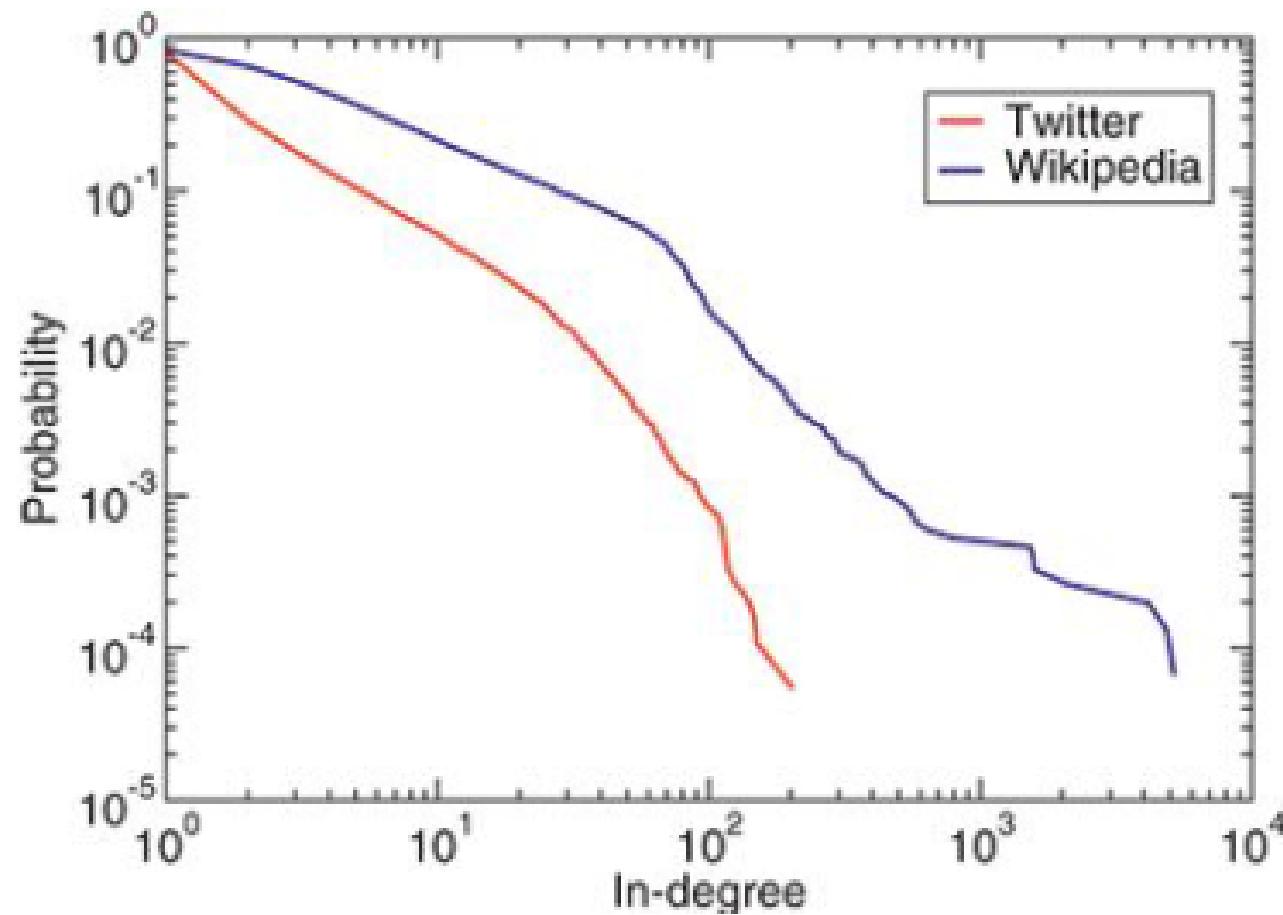
- Protežu se kroz nekoliko redova veličine
- Široke distribucije -> otežani rep -> kumulativna distribucija izgladi
- Distribucije s otežanim repom -> log log skala



Kumulativne distribucije

- Distribucije s otežanim repom pokazuju veliku heterogenost u vrijednostima stupnja čvora.
- Mnogo čvorovi samo nekoliko prijatelja, dio njih velik broj susjeda -> veća uloga u mreži -> takve čvorove nazivamo hubovima
- Mnoge prirodne, društvene, informacijske, ručno kreirane mreže -> otežani rep distribucije s visoko povezanim hubovima

Distribucija stupnja primjer



Heterogenost mreže

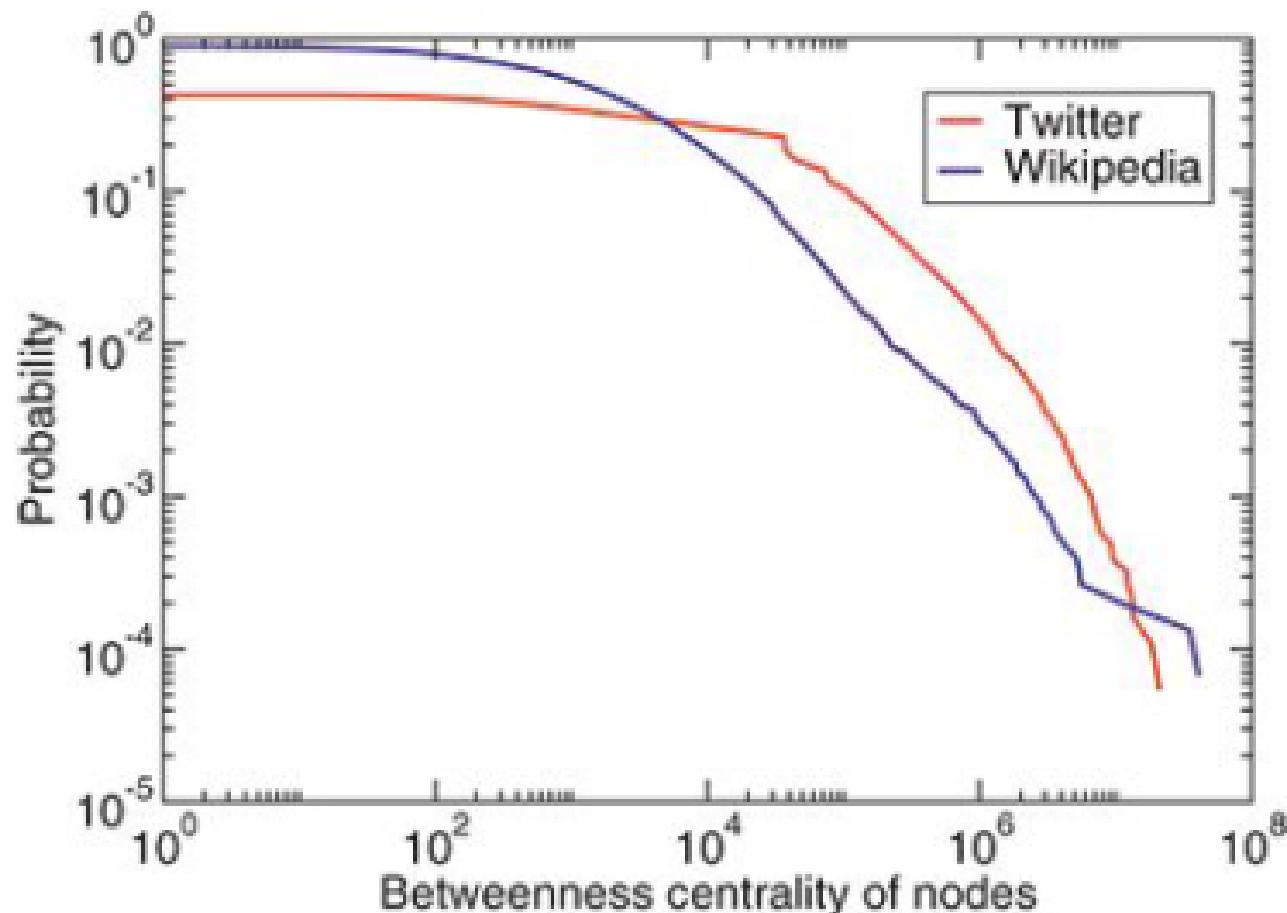
- Parametar heterogenosti $\kappa(kappa)$
- Heterogenost mrežnog stupnja distribucije
- Prosječan kvadratni stupanj $\langle k^2 \rangle = \frac{k_1^2 + k_2^2 + \dots + k_{N-1}^2 + k_N^2}{N} = \frac{\sum_i k_i^2}{N}$
- $\kappa = \frac{\langle k^2 \rangle}{\langle k \rangle^2}$ - parametar heterogenosti

Heterogenost - primjer

Network	Nodes (N)	Links (L)	Average degree ($\langle k \rangle$)	Maximum degree (k_{max})	Heterogeneity parameter (κ)
Facebook Northwestern Univ.	10,567	488,337	92.4	2,105	1.8
IMDB movies and stars	563,443	921,160	3.3	800	5.4
IMDB co-stars	252,999	1,015,187	8.0	456	4.6
Twitter US politics	18,470	48,365	2.6	204	8.3
Enron email	87,273	321,918	3.7	1,338	17.4
Wikipedia math	15,220	194,103	12.8	5,171	38.2
Internet routers	190,914	607,610	6.4	1,071	6.0
US air transportation	546	2,781	10.2	153	5.3
World air transportation	3,179	18,617	11.7	246	5.5
Yeast protein interactions	1,870	2,277	2.4	56	2.7
<i>C. elegans</i> brain	297	2,345	7.9	134	2.7
Everglades ecological food web	69	916	13.3	63	2.2

Heterogenost izračunata koristeći ulazni stupanj

Distribucija međupoštenosti - primjer

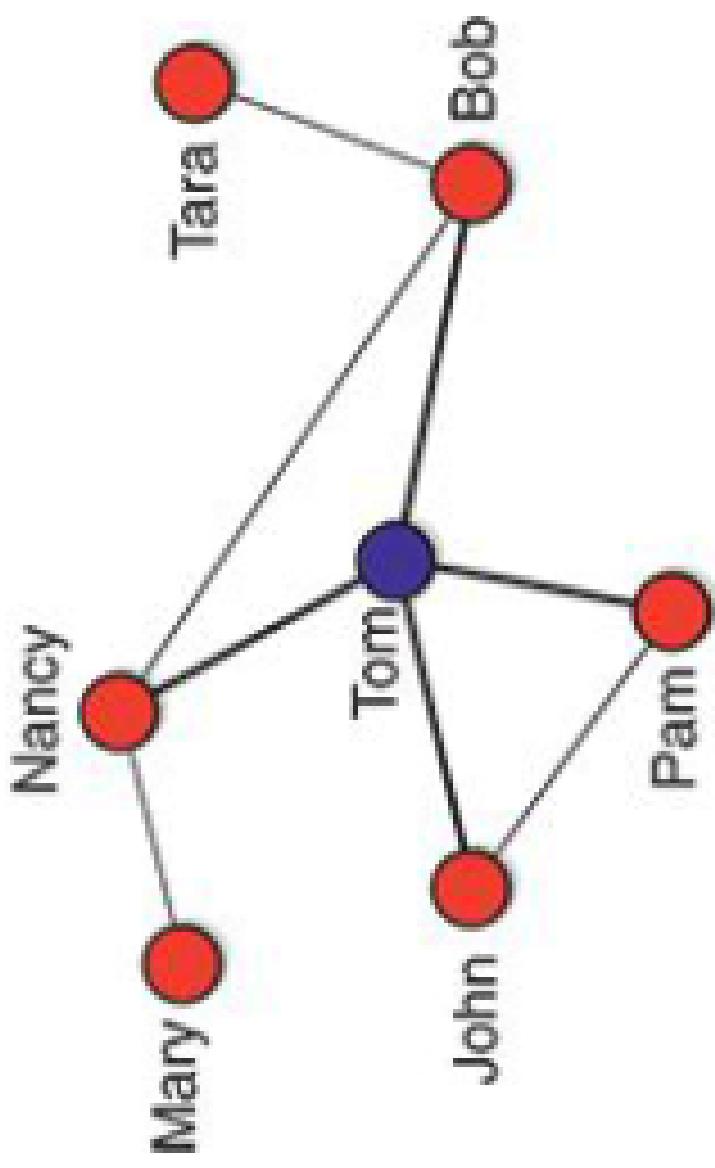


Izračunato samo za najveću komponentu Wikipedie

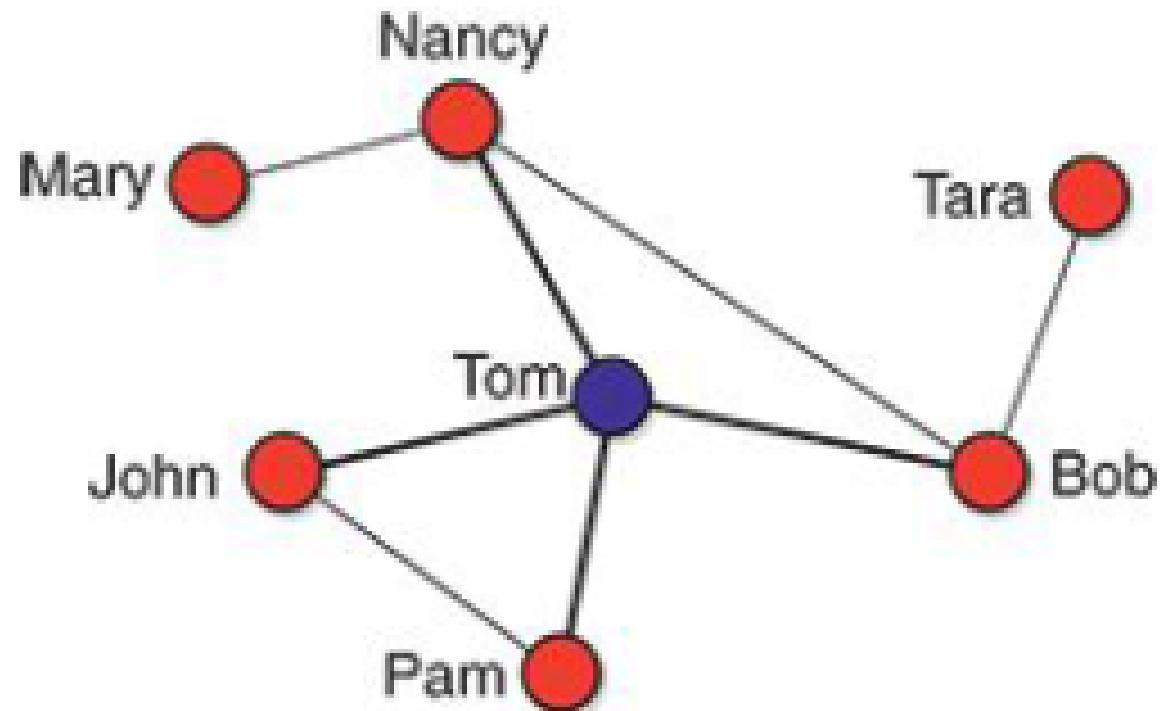
Paradoks prijateljstva

- Većina ljudi ima manje prijatelja nego što to imaju njihovi prijatelji u prosjeku 😊
- Tražimo osobu s najviše prijatelja u grupi od N ljudi
- Imamo samo njihove brojeve telefona i jedan poziv
- Ako slučajnim odabirom nazovemo jednu osobu, vjerojatnost je $1/N$
- Što ako nazovemo jednu osobu i pitamo ju neka navede jednog prijatelja?

Paradoks prijateljstva



Paradoks prijateljstva



Tom je osoba s najviše prijatelja. Isprobamo obje strategije

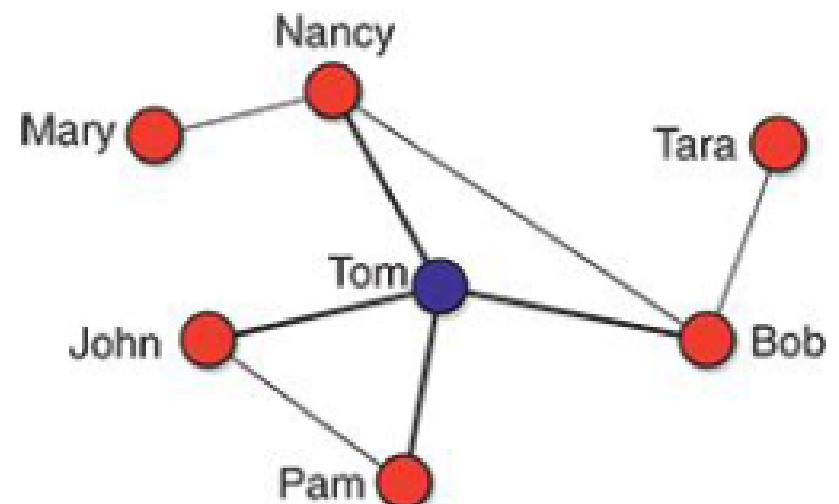
- Slučajno nazovemo jednu osobu – vjerojatnost $1/7 \approx 0.14$
- Nazovemo osobu i pitamo ju za jednog prijatelja $5/21 \approx 0.24$

Paradoks prijateljstva

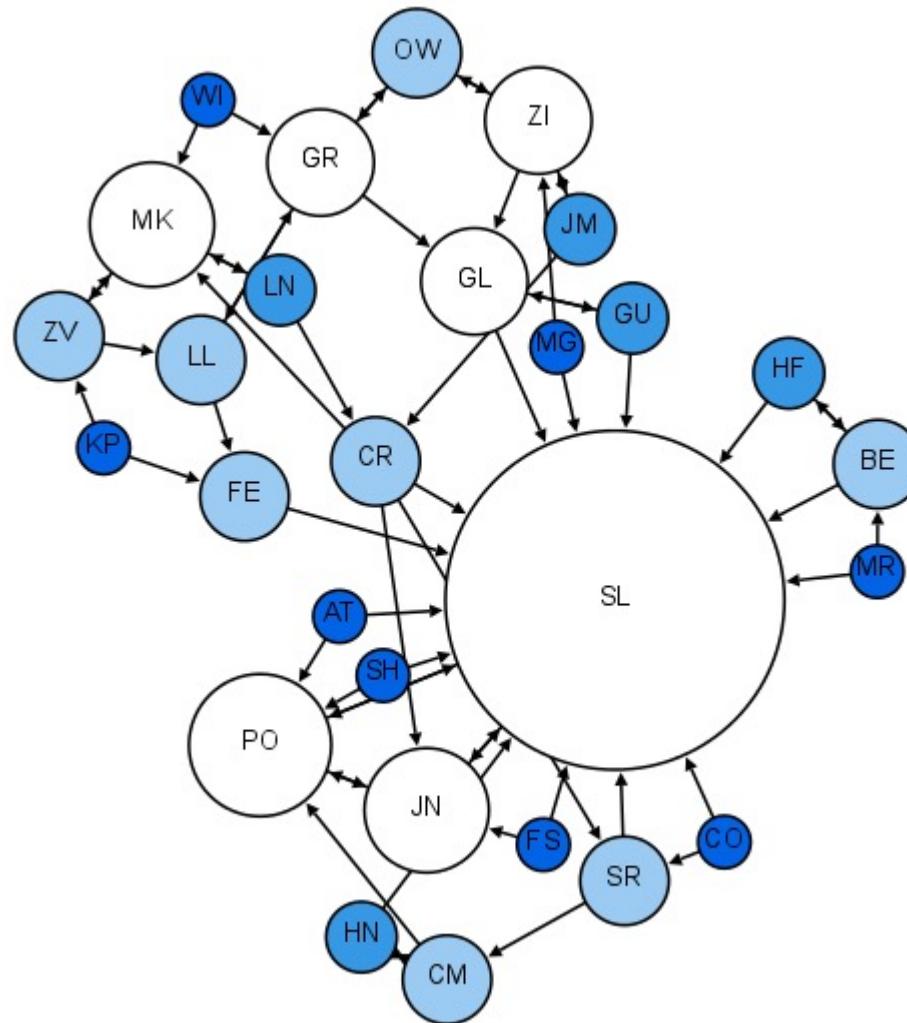
- Biramo veze umjesto čvorova
- Kada biramo čvorove svaki ima jednaku vjerojatnost biti izabran, neovisno o stupnju
- Kada biramo veze, veći broj susjeda -> veća vjerojatnost dohvata
- U našem slučaju – 4 moguća kanal prema Tomu
- Vjerojatnost pronađenja huba raste ako se prebacimo na drugo susjedstvo – raste broj veza

Paradoks prijateljstva

- Prosječan broj susjeda od susjeda čvora je $17/6 = 2.83$
- Prosječan stupanj mreže je $(1+3+3+1+4+2+2)/7=2.29$
- Prosječan stupanj susjeda je veći od prosječnog stupnja čvora
- Šira distribucija stupnja -> jači efekt
- Posebno jak u slučaju distribucije s otežanim repom



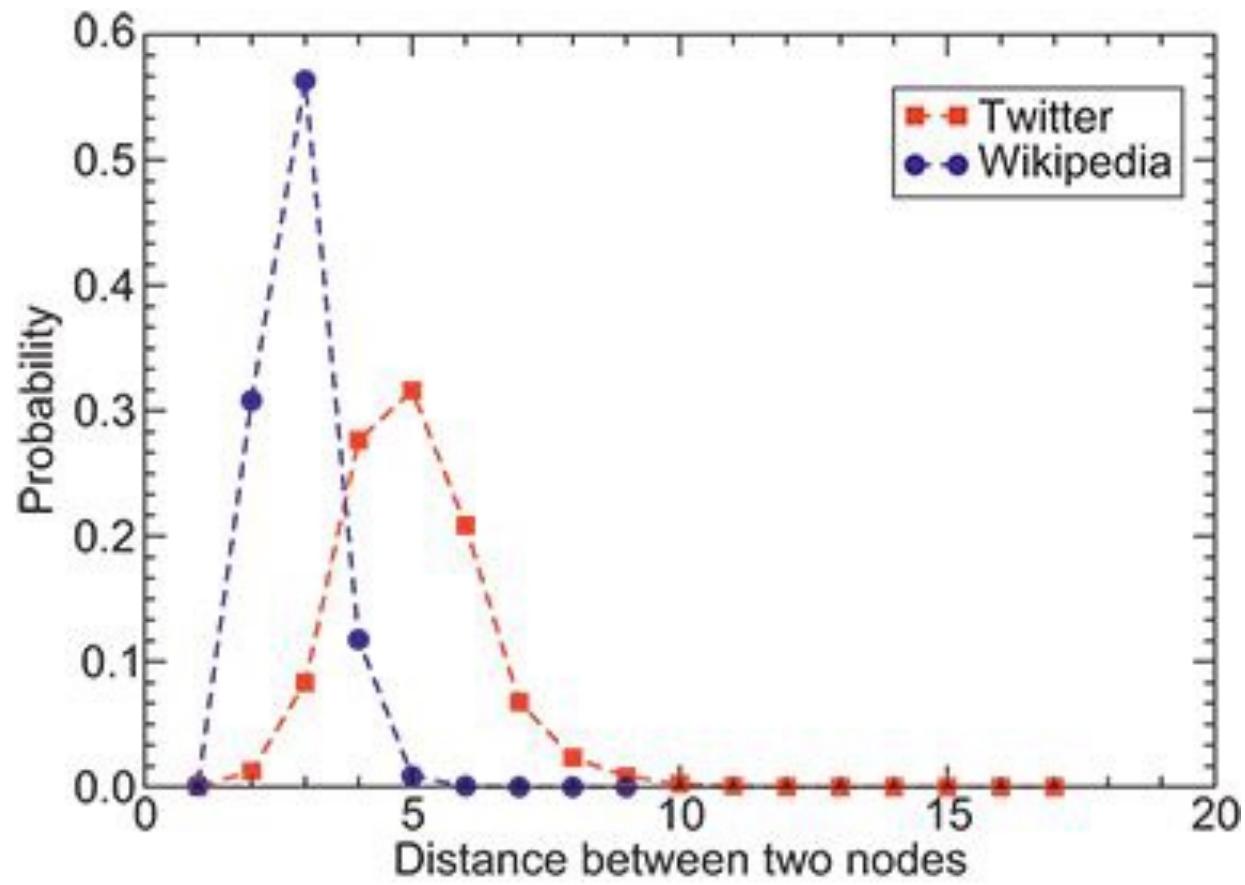
Prvi razred. Dvoje s kojima bi željeli sjediti



Ultra mali svijet

- Hubovi jednostavni za pronaći i postoji velika potražnja za njima
- Npr. Ako želimo letjeti iz A u B. Ako nema direktnog leta -> najbliži hub
- Mreže sa širokom distribucijom stupnja imaju svojstvo ultra malog svijeta
- Ultra-mali svijet – udaljenost između čvorova jako kratke

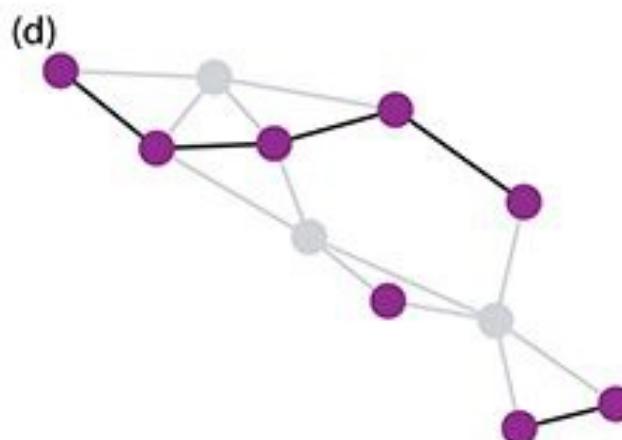
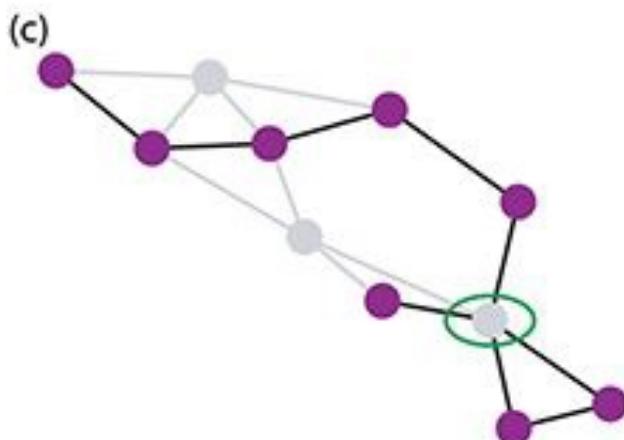
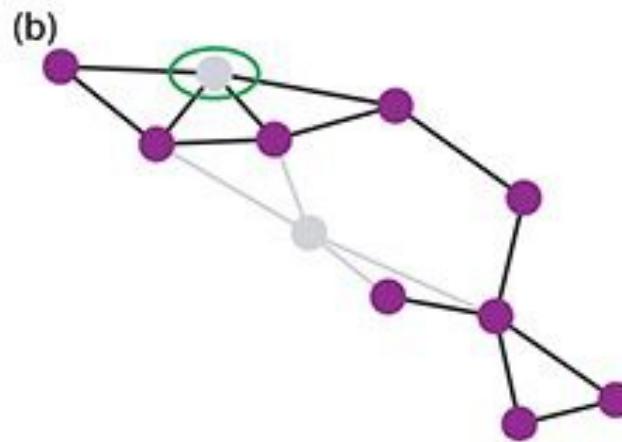
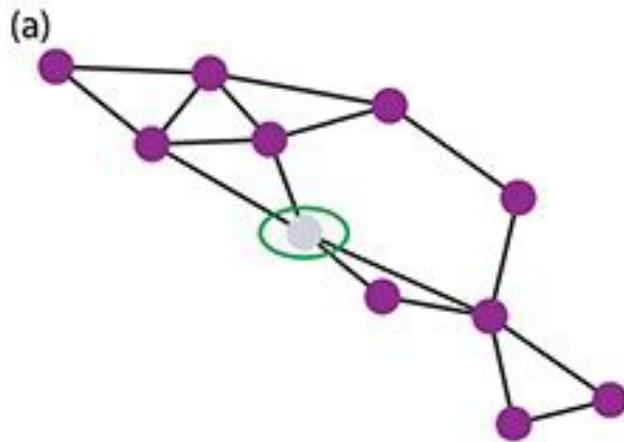
Ultra mali svijet



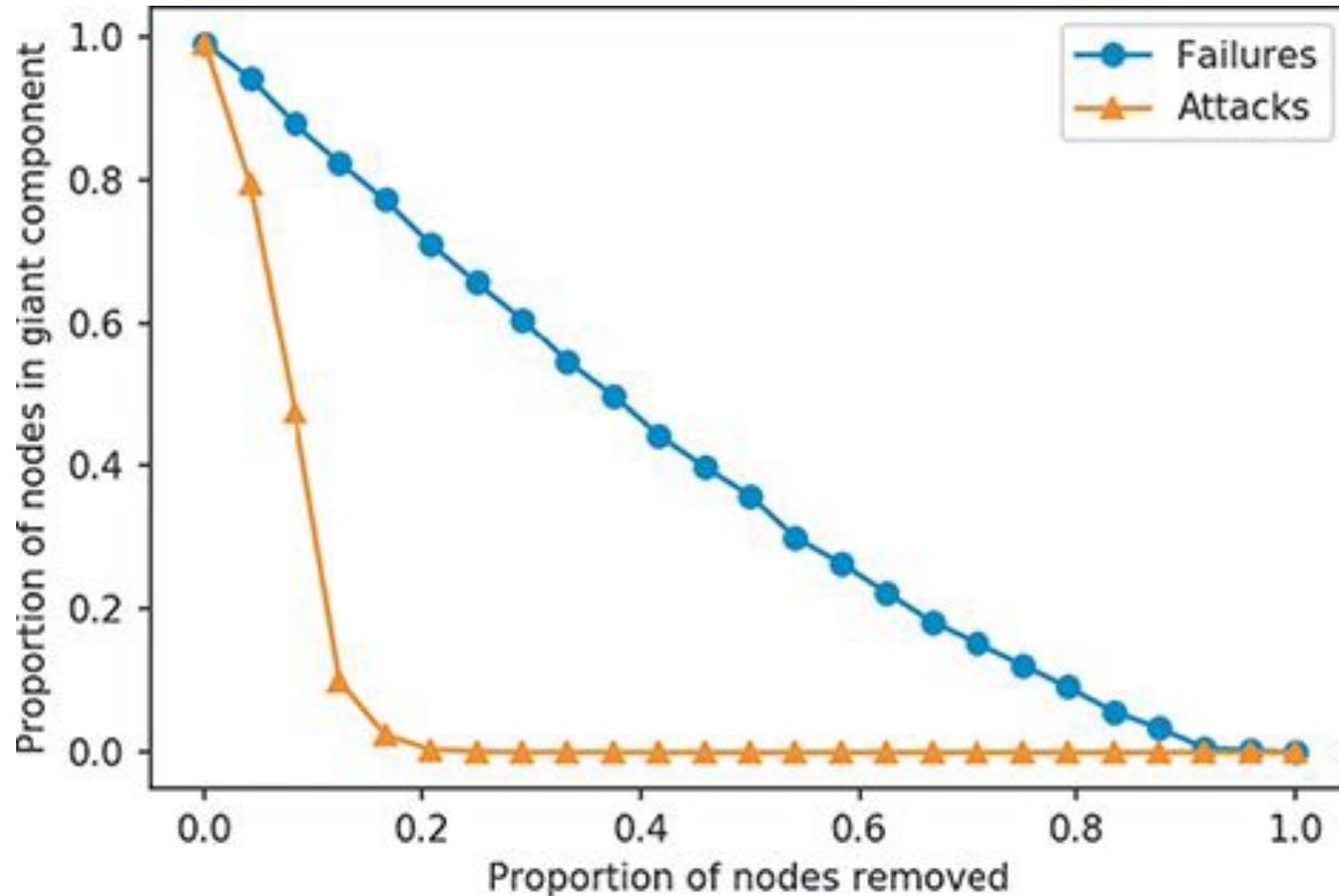
Robusnost

- Kvar na jednoj komponenti ne utječe na funkcionalnost (npr. Avionski motori)
- Definicija robusnosti mreže
 - Povezanost je važno svojstvo
 - Mjerimo utjecaj micanja čvora i njegovih veza na povezanost
 - Slom mreže u nepovezane dijelove signalizira štetu koja utječe na funkcionalnost

Robusnost



Robusnost



Mjerenje broj čvorova u najvećoj komponenti u odnosu na početnu mrežu

Kvar – slučajno uklanjanje
Napad – ciljano uklanjanje (prvo hubovi)

Otpornost na kvar i ranjivost na napad

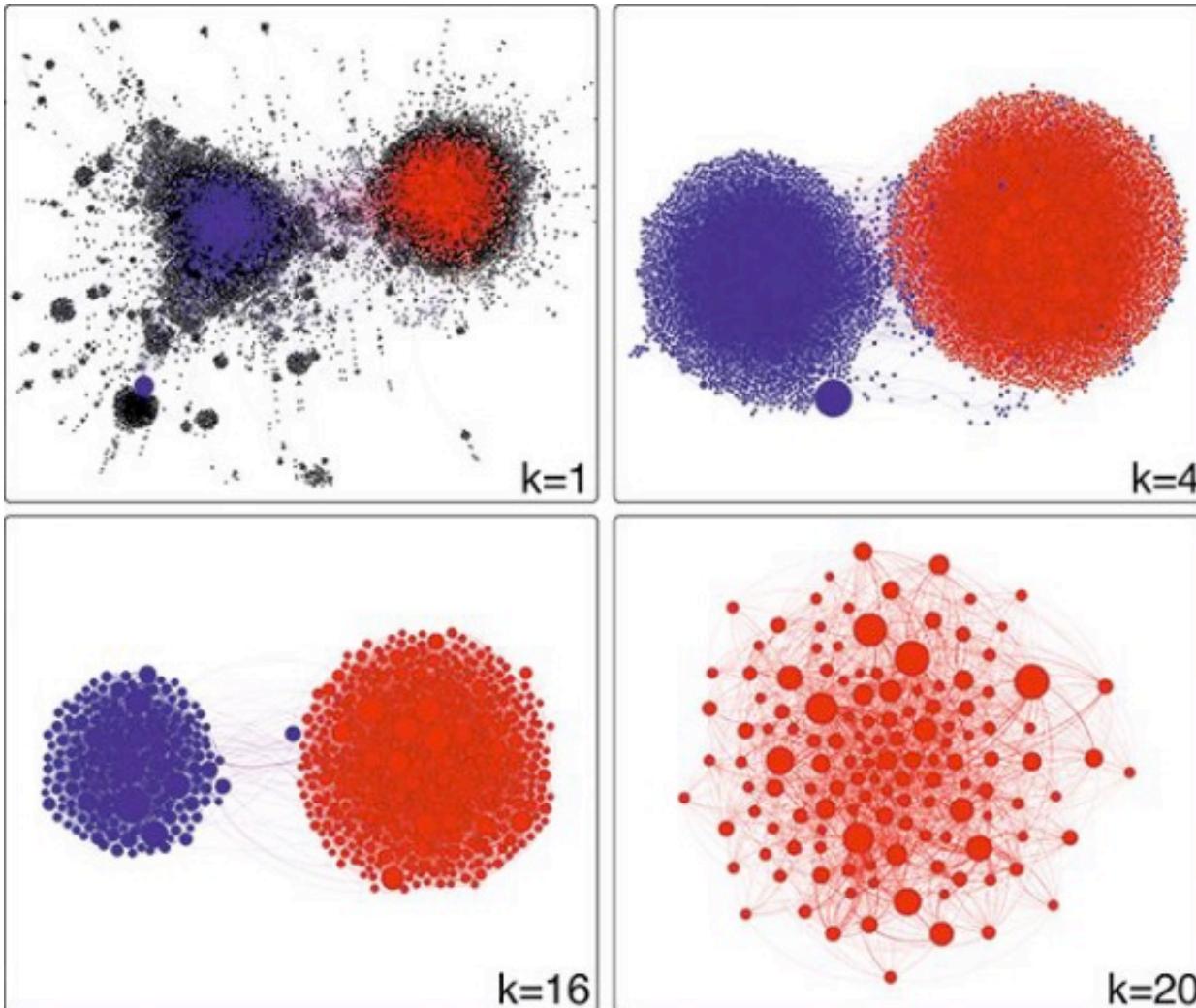
Dekompozicija jezgre mreže

- Jezgra i periferija mreže
- Razdvajanje mreže u međusobno isključive dijelove (ljeske)
 - Koristimo stupanj
 - Zavisno o poziciji u jezgra-periferija strukturi mreže
 - Vanjske ljeske niskog stupnja su periferija
 - Ljuštimo jednu po jednu – krećemo sa čvorovima sa stupnjem 0
 - Na kraju ostaje jezgra

K-jezgrena dekompozicija

- Počinjemo s $k=0$
- Iterativno:
 1. Rekurzivno uklanjanje čvorova stupnja k , dok niti jedan ne preostane
 2. Uklonjeni čvorovi su *k-ljuska*, preostali čine *k+1 jezgru*
 3. Ukoliko nema više čvorova, završiti; inače, inkrementalno povećati k

Dekompozicija jezgre



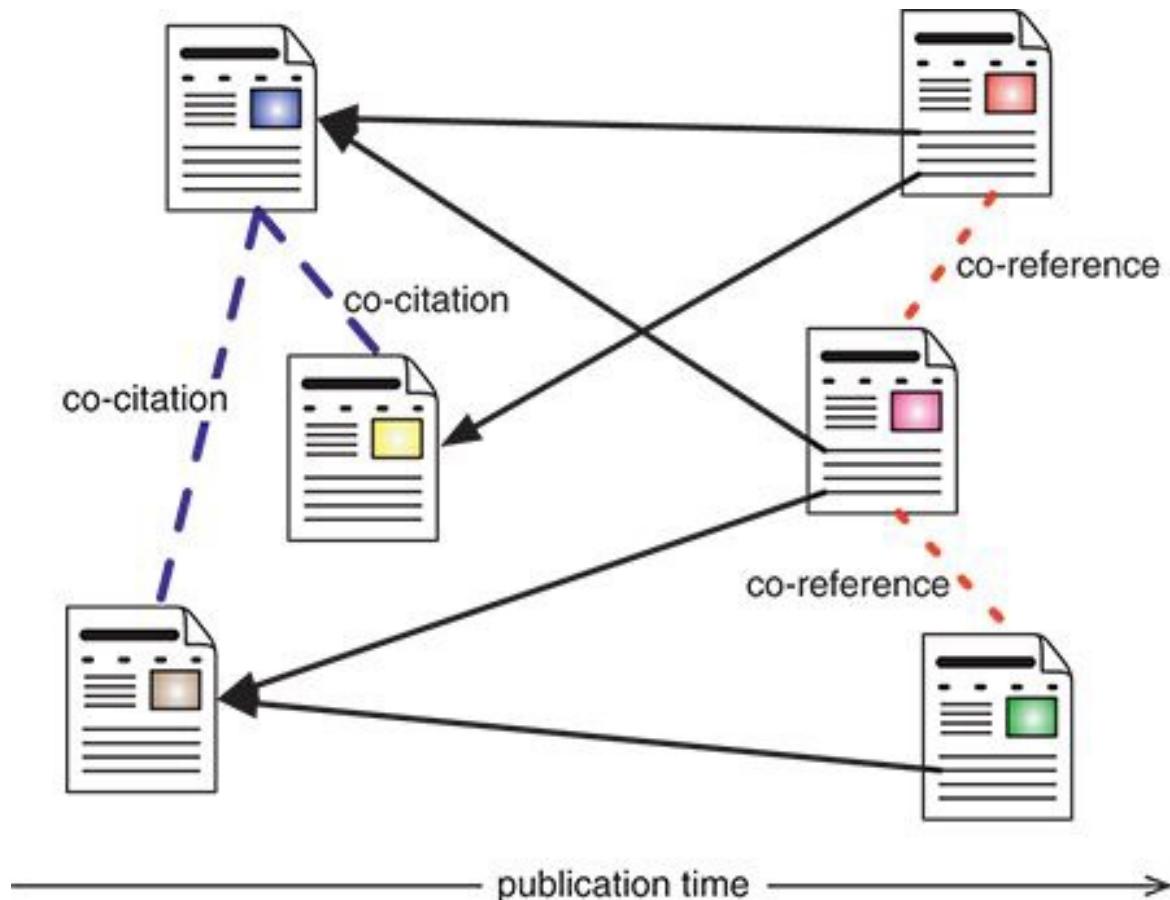
Kompleksne mreže

4. predavanje

Usmjerene mreže

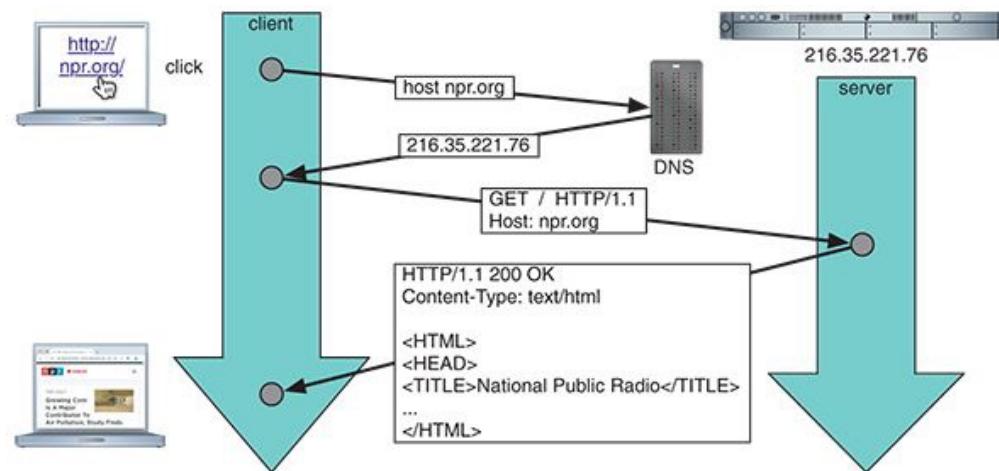
- Socijalne mreže – prijateljstvo simetrično (teoretski ☺)
- Promet (avionski, željeznički, cestovni) – dvosmjeran
- Usmjerene mreže:
 - Twitter
 - Komunikacijske i informacijske mreže
 - Email
 - Wikipedia

Mreža citata



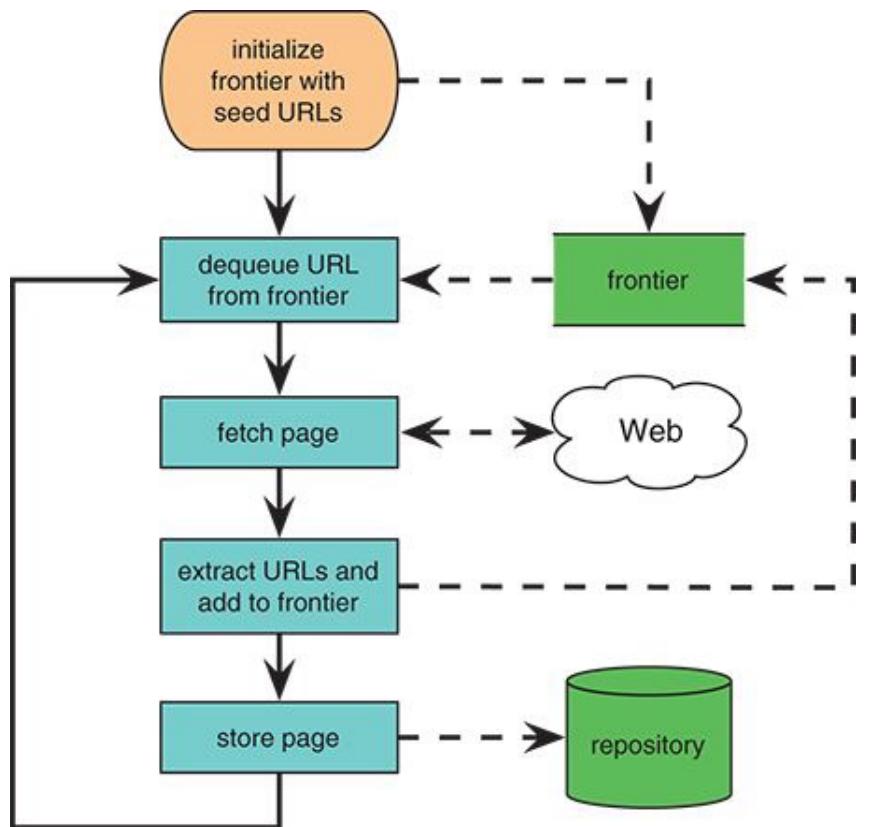
- Citat – usmjereni veza jednog rada na drugi
- Dodatne neusmjereni veze
 - Citirani zajedno (co-citation)
 - Citiraju isti rad (co-reference)

HTTP (Hyper Text Transfer Protocol)



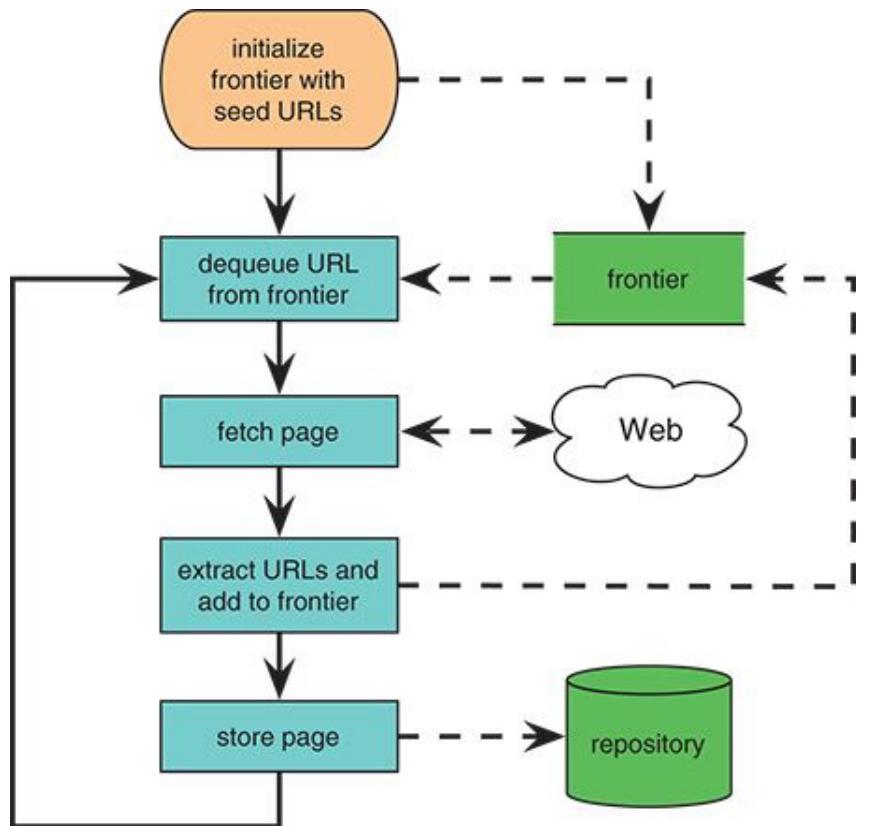
- Zahtjevi
 - Zaglavje, prazna linija, tijelo poruke
 - GET
 - Zahtjev za stranicom
 - Samo zaglavje i prazna linija
- POST
 - Tijelo poruke sadrži dodatne parametre sadržaja
 - Npr. slanje unosa u formu
- Obavezan *hostname* (jedan server više webova)
- Odgovor
 - Zaglavje, prazna linija, tijelo poruke
 - Zaglavje
 - Tip servera, vrijeme, broj vraćenih okteta...
 - Kod odgovora (200-uspjeh, 404 nije pronađen)
 - Tijelo – HTML stranica

Web crawler



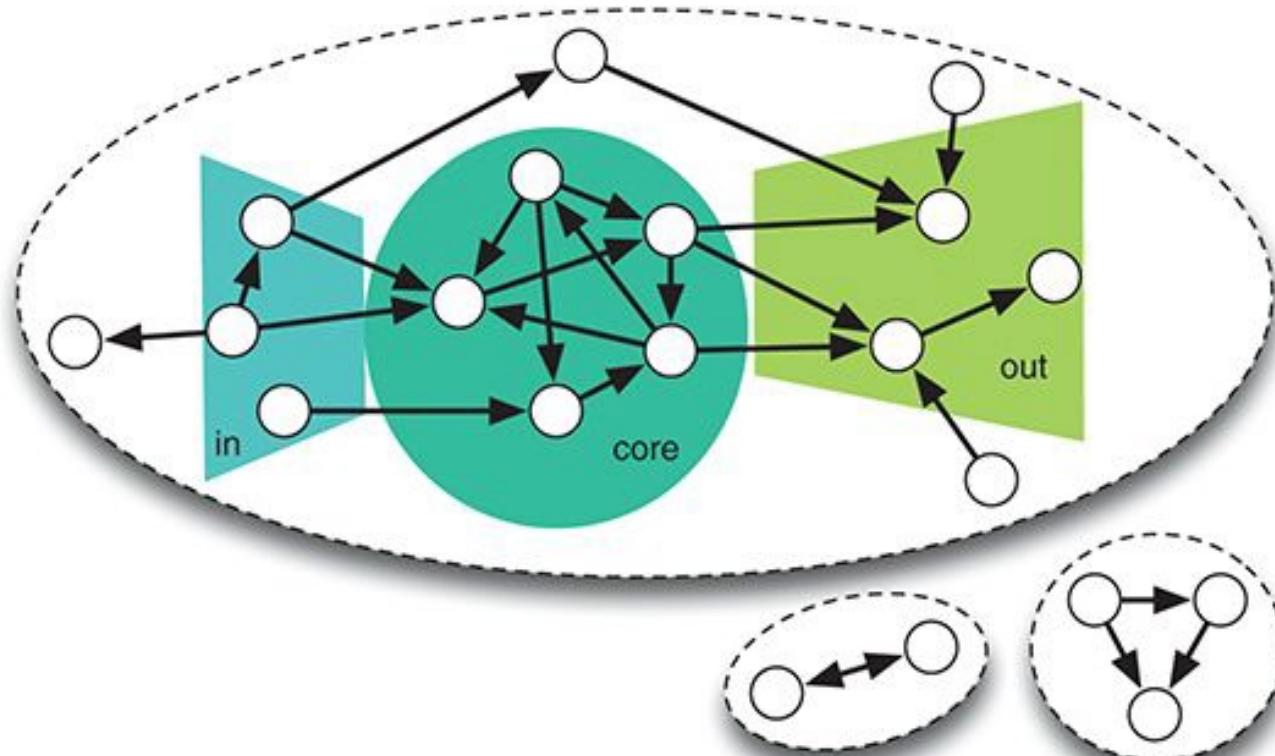
- Programi za automatsko skidanje web stranica
- Najčešća primjena – pretraživači (npr. Google)
 - Indeksiranje sadržaja – mapiranje sadržaja (ključnih riječi i fraza te stranica na kojima se nalaze)
 - Rangiranje
 - Naj sofisticiraniji i najosjetljiviji dio pretraživača (Brin i Page)
- Druge primjene:
 - Poslovna inteligencija
 - Digitalne biblioteke
 - Webometric alati – utjecaj institucija
 - Maliciozno korištenje (skupljanje email adresa za slanje spam poruka)
 - Znanstvene svrhe (npr. Struktura Weba)

Web crawler



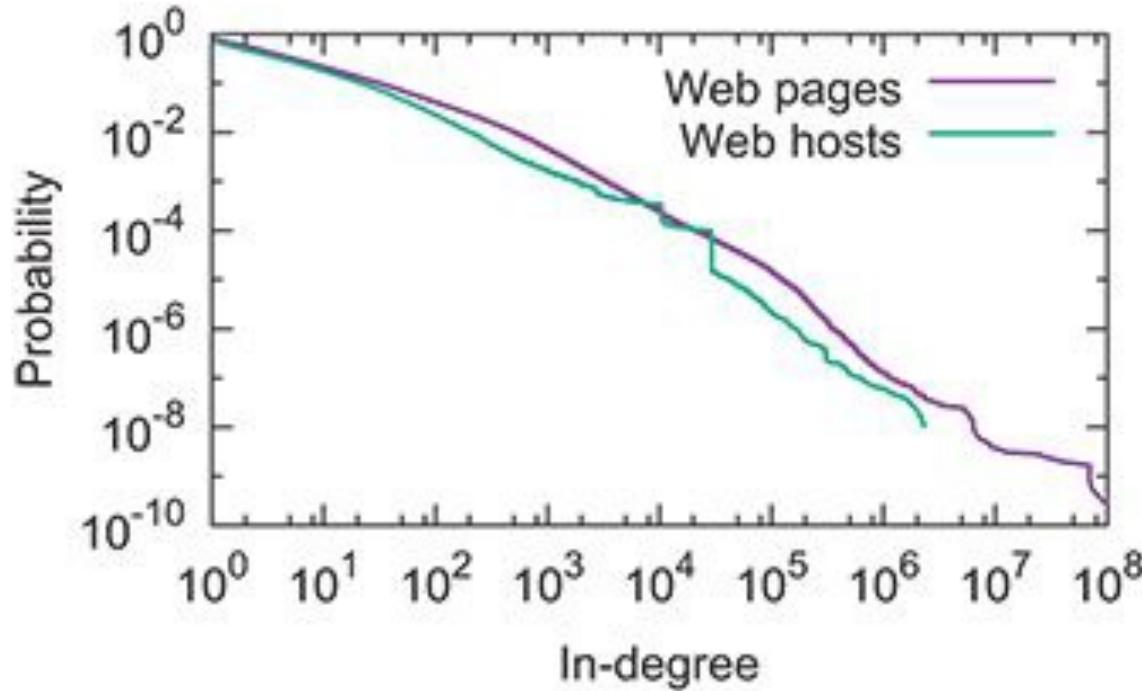
- BFS pristup
- Heuristike za prioritiziranje veza
- Što idemo dublje manja šansa za pronalazak dobrih stranica
- Optimizacija i paralelizacija

Struktura Weba – Web graf



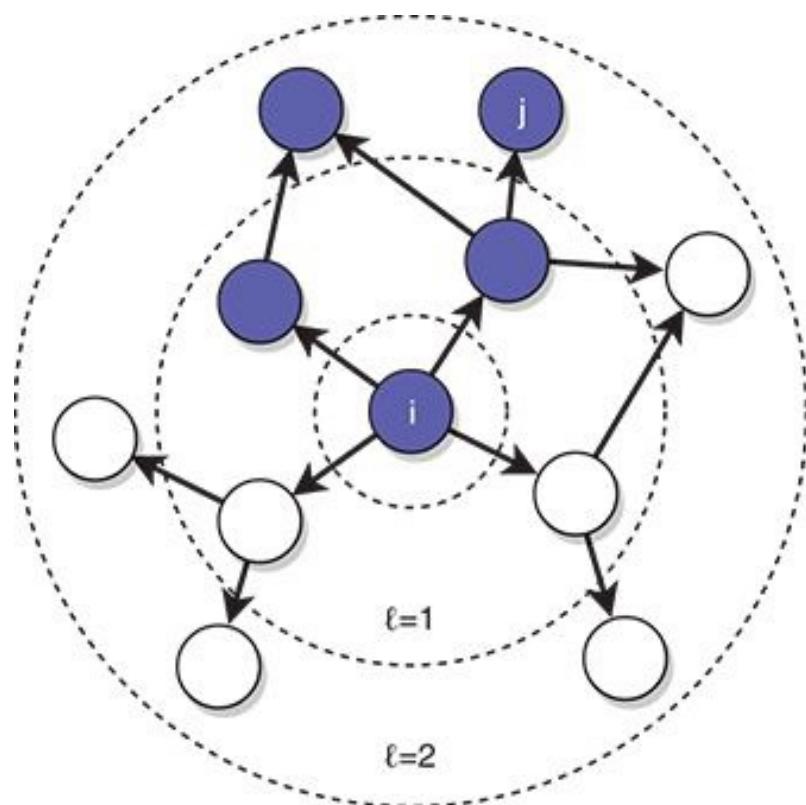
- Dobiven *crawlerima*
- Postoje različite slabo povezane komponente
- Najveća komponenta 90% svih stranica
- Unutar nje imamo snažno povezanu jezgru te *in* i *out* komponente
- “Leptir kravata” struktura

Komplementarna kumulativna distribucija ulaznog stupnja (2012)



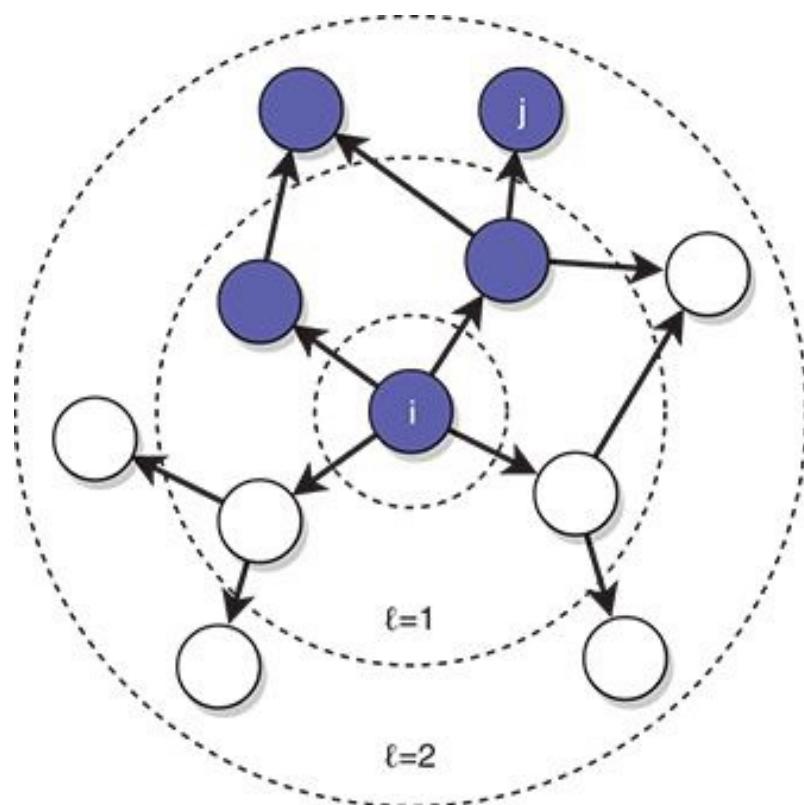
- Prosječan ulazni stupanj – 10-30
- Standardna devijacija red veličine veća -> velik faktor heterogenosti κ
- Oblik distribucije se ne mijenja od vremena kada je Web bio samo par godina star
- Distribucija izlaznog stupnja
 - Mnogo teža za analizu
 - Mnogo uža
 - Puno veza na druge čvorove (spam)

Tematska lokalnost



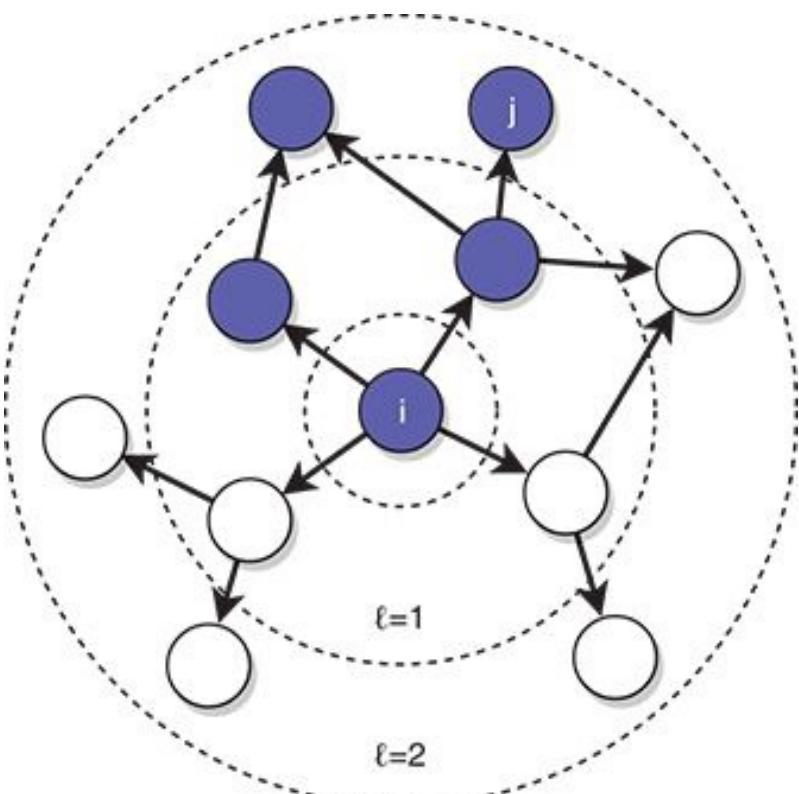
- **Mjerenje**
 - Izglednost da odredišna stranica u danoj udaljenosti od izvořne ima istu temu
 - Usporedimo očekivanjem da stranica slučajno bude iste teme (ovisi o temi)
- Stranice unutar 1 – 2 veze od odredišne imaju red veličine veću izglednost u usporedbi sa slučajnom

Tematska lokalnost



- Homofilija – povezivanje sličnih čvorova
- Lokalnost po temi
 - Stranice s povezanim temama povezane ili imaju kratku udaljenost
 - Nove stranice -> povezivanjem s informacijama relevantnim za temu

Tematska lokalnost



- Mjerenje sličnosti teksta
- Više zajedničkih ključnih riječi -> jači dokaz o sličnosti
- Kosinusna sličnost
- Postupak:
 - BFS *crawl* od jedne ili više početnih
 - Mjerenje sličnosti obiđene i početne stranice, usrednjujući za sve početne i obiđene stranice
 - Prikaz promjene sličnosti s udaljenošću
- Tematski *drift* – manja izglednost da naletimo na sličnu stranicu s većom udaljenošću

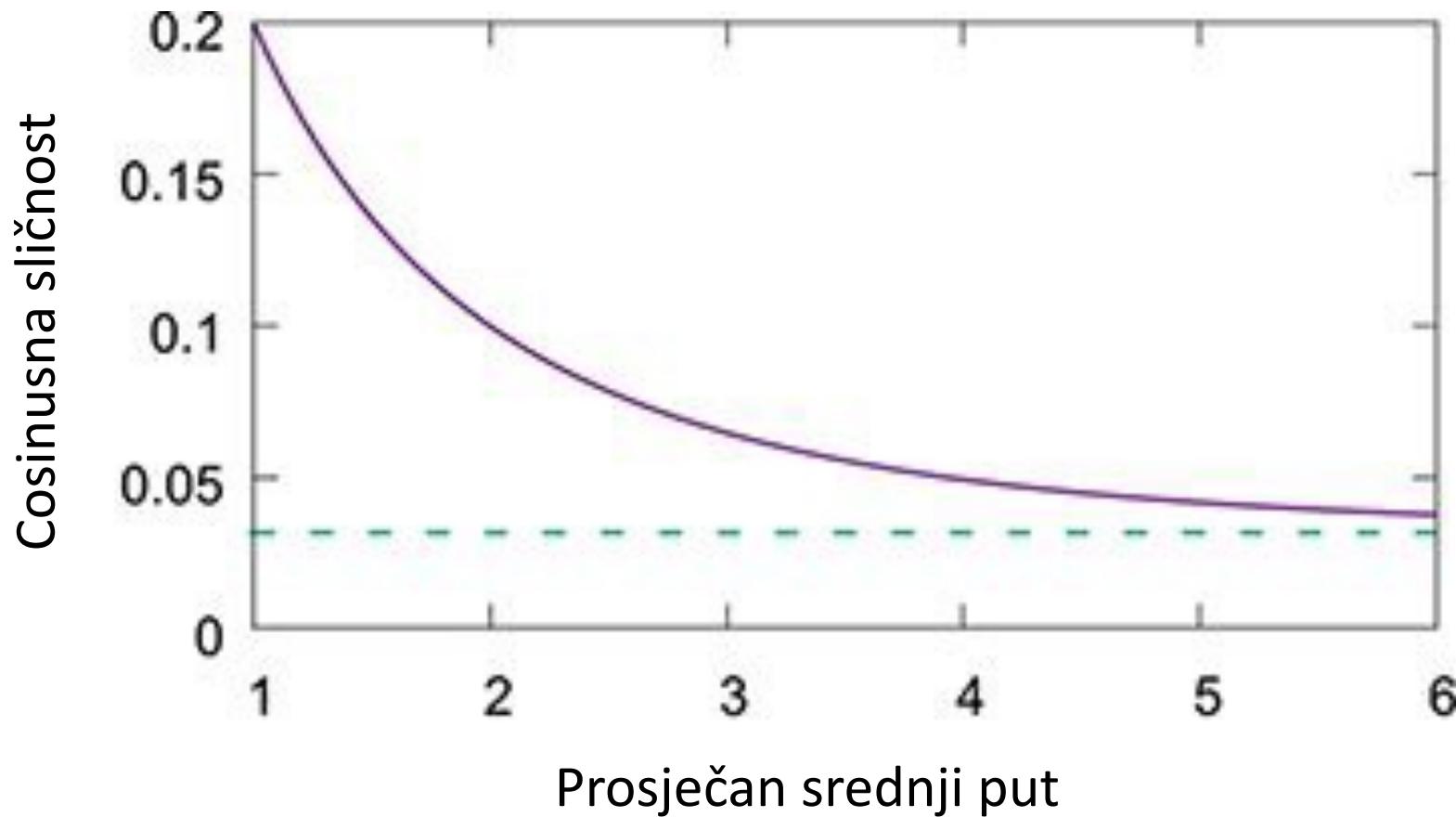
Kosinusna sličnost

- Mjerenje sličnosti između dva teksta (dokument, web stranica, ...)
- Reprezentacija dokumenta vektorom $\vec{d} = \{w_{d,1}, \dots, w_{d,n_t}\}$
 - $w_{d,t}$ - težina izraza t u d
 - n_t - ukupan broj izraza
 - Suvremeni NLP imaju slične vektore, no dimenzije su latentne reprezentacije umjesto riječi
- Težine
 - Obično proporcionalne frekvenciji pojavljivanja izraza
 - Uklanjanje stop riječi (veznici, članovi, ...)
 - Umanjenje težine izrazima koji se često pojavljuju u raznim dokumentima

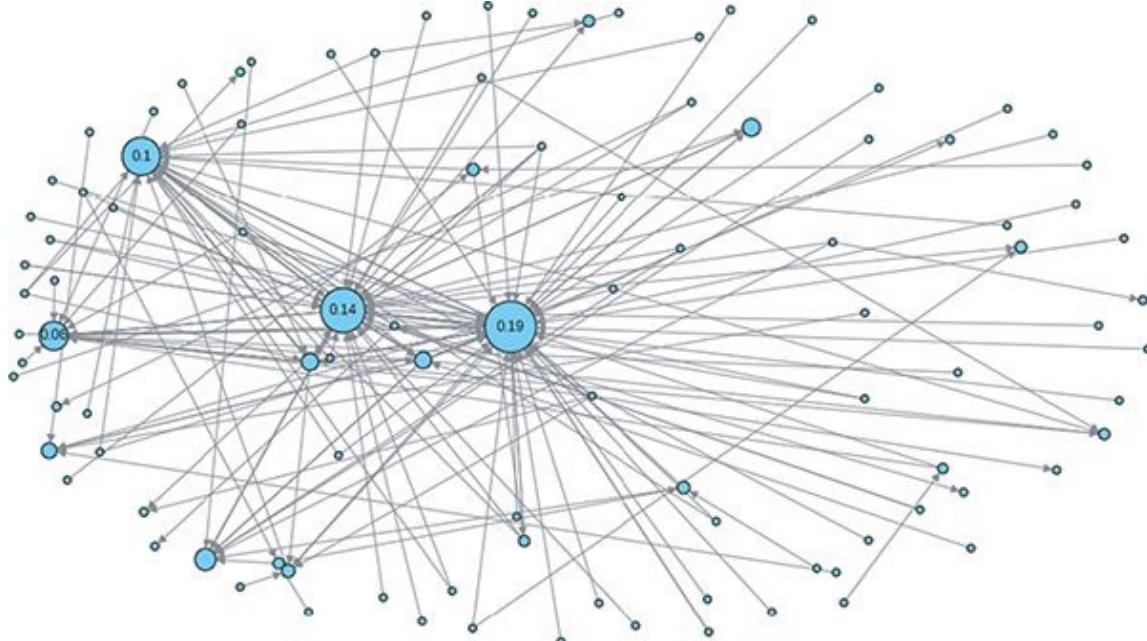
Kosinusna sličnost

- $\cos(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1}{\|\vec{d}_1\|} \cdot \frac{\vec{d}_2}{\|\vec{d}_2\|} = \frac{\sum_t w_{d_1,t} w_{d_2,t}}{\sqrt{\sum_t w_{d_1,t}^2} \sqrt{\sum_t w_{d_2,t}^2}}$
- Ako su izrazi u \vec{d}_1 prisutni i u \vec{d}_2 cosinus će biti blizu 1
- Ako dva dokumenta ne dijele niti jedan izraz cosinus će biti 0
- $\|\vec{d}\| = \sqrt{\sum_t w_{d,t}^2}$ - normizacija normom vektora
- Normizacija - dugačkim dokumentima će norma smanjiti sličnost

Primjer tematske lokalnosti weba

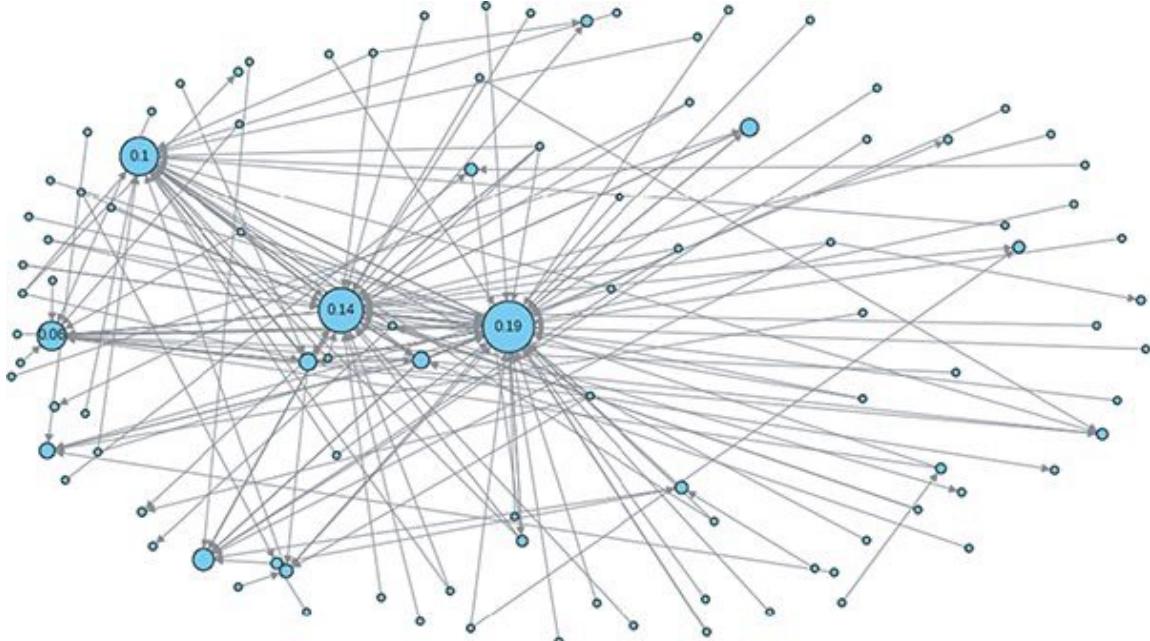


PageRank



- Algoritmi rangiranja
- Mrežna centralnost 1998
(PageRank - Google)
- Stranice s većim PageRankom su bolje rangirane

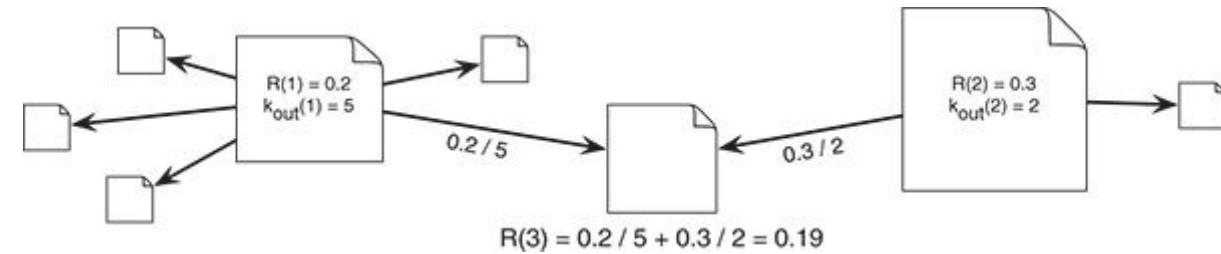
PageRank



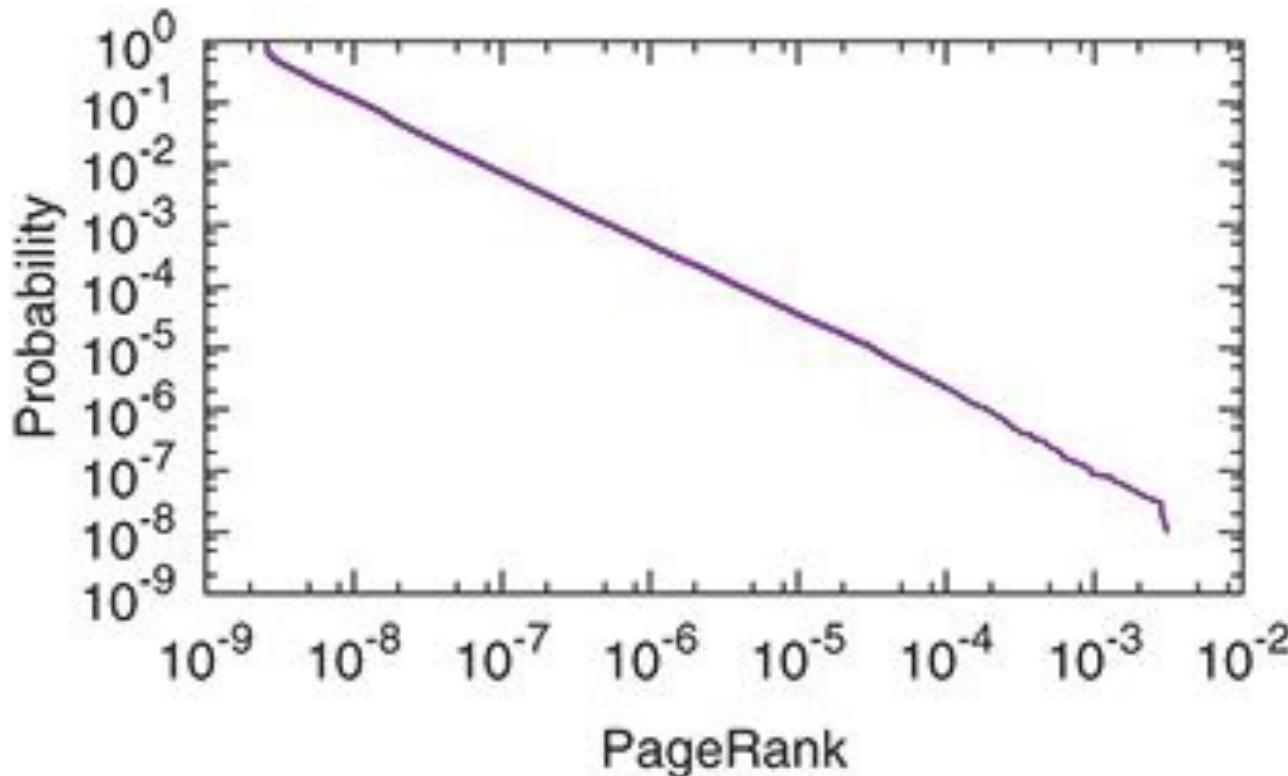
- Slučajna šetnja – svaki susjedni čvor ima jednaku vjerovatnost da bude posjećen
- Slučajni skokovi. Korisnik kreće u sasvim novu potragu (teleportacija)
- PageRank dio vremena koje provedemo stranici

PageRank računanje

- Iterativna metoda
- Suma svih vrijednosti je 1
- Inicijalna vrijednost za svaki čvor je $1/N$
- Slučajni skokovi s događaju s parametrom α , obično $\alpha \approx 0.15$
- U svakom koraku vjerojatnost skoka je α , a nastavka pretrage je $1 - \alpha$
- $R_t(i) = \frac{\alpha}{N} + (1 - \alpha) \sum_{j \in pred(i)} \frac{R_{t-1}(j)}{k_{out}(j)}$
 - Prvi izraz je teleportacija do i
 - Drugi izraz je slučajan hod i uključuje vjerojatnosti dolaska u i iz svih njegovih prethodnika
 - Kada je $\alpha > 0$ konvergira u manje od 100 koraka



Komplementarna kumulativna distribucija PageRanka



- Distribucija slična distribuciji čvorova – zašto ne uzeti onda istu ?
- Nisu svi putevi isti – na važnost stranice utječu stranice koje povezuju na nju
- Između dvije stranice s istim ulaznim stupnjem, ona povezana stranicama s većim PageRankom pobjeđuje

Optimizacija za pretraživač

- Poboljšanje ranga
 - Prilagođenje opisa stranice i mogućnosti navigacije
 - Neetičke prilagodbe
 - Kreiranje velikog broja fiktivnih stranica koje povezuju jedna drugu i ciljnu stranicu
 - Kada suvremeni algoritmi najdu na takve zloupotrebe – micanje iz indeksa pretraživanja

Težinske mreže

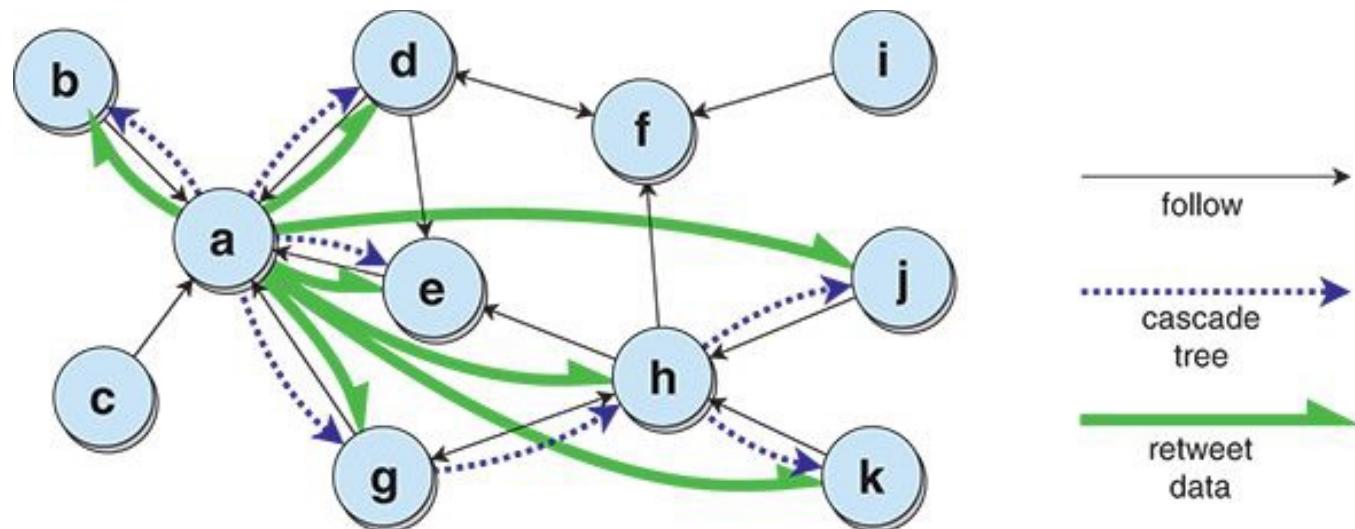
- Primjeri:
 - Dva korisnika mogu retweetati jedan drugoga proizvoljan broj puta
 - Mreža emailova
 - Mreže mozga – sinapse – različite razine signala
 - Internet – brzina prijenosa paketa između usmjerivača
- Mreže za koje mislimo da su netežinske
 - Facebook – nisu nam svi prijatelji isti
 - Mreža glumaca koji su glumili zajedno u filmovima
- Informacijske i transportne mreže
- Jačina veza, jačina ulaznih i izlaznih veza

Informacija i dezinformacija

- Mreže koje difuziraju informaciju
 - Čvorovi – ljudi
 - Veze – dijelovi informacije (ideje, koncepti, novosti, ponašanja) koji se prenose s osobe na osobu
 - Prijenosna jedinica informacije – *meme*
 - Korištenje # kod Twittera #FER

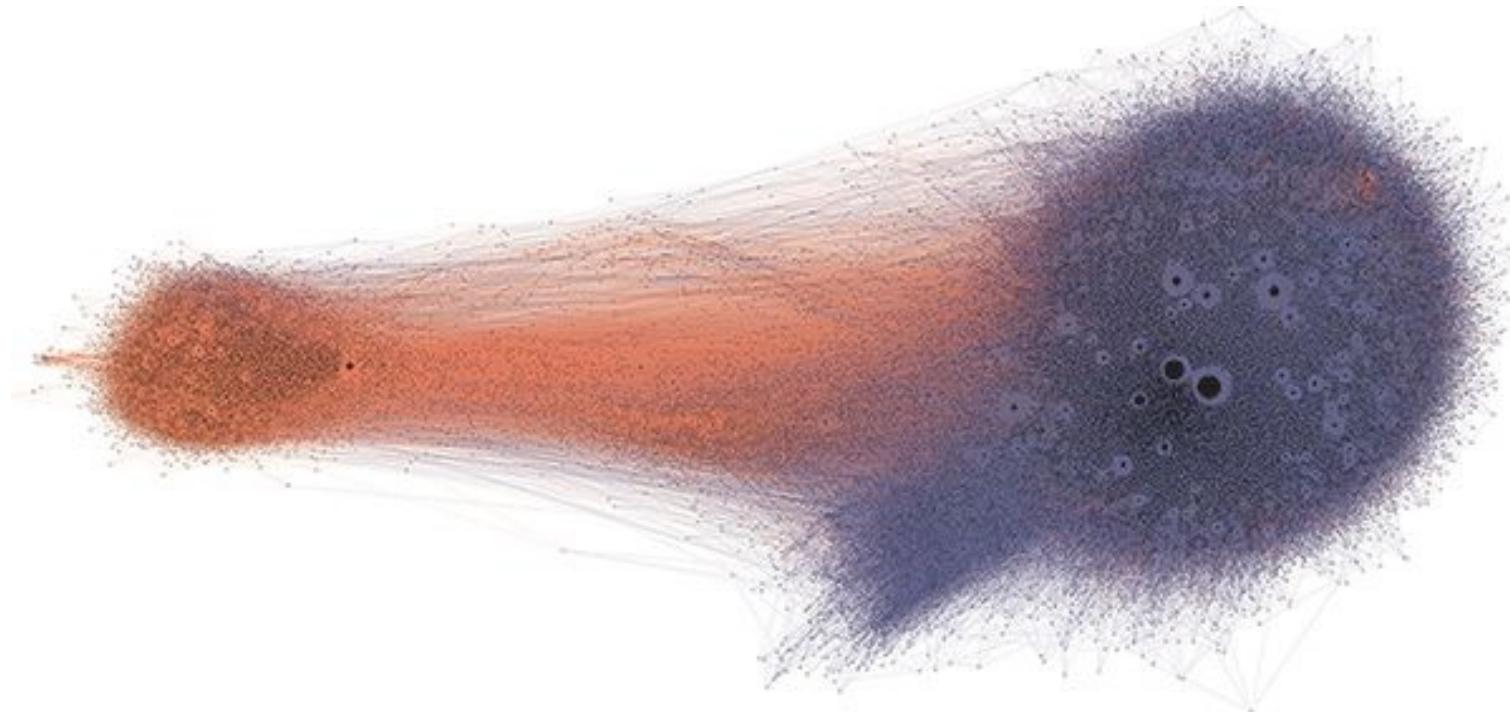


Twitter/X

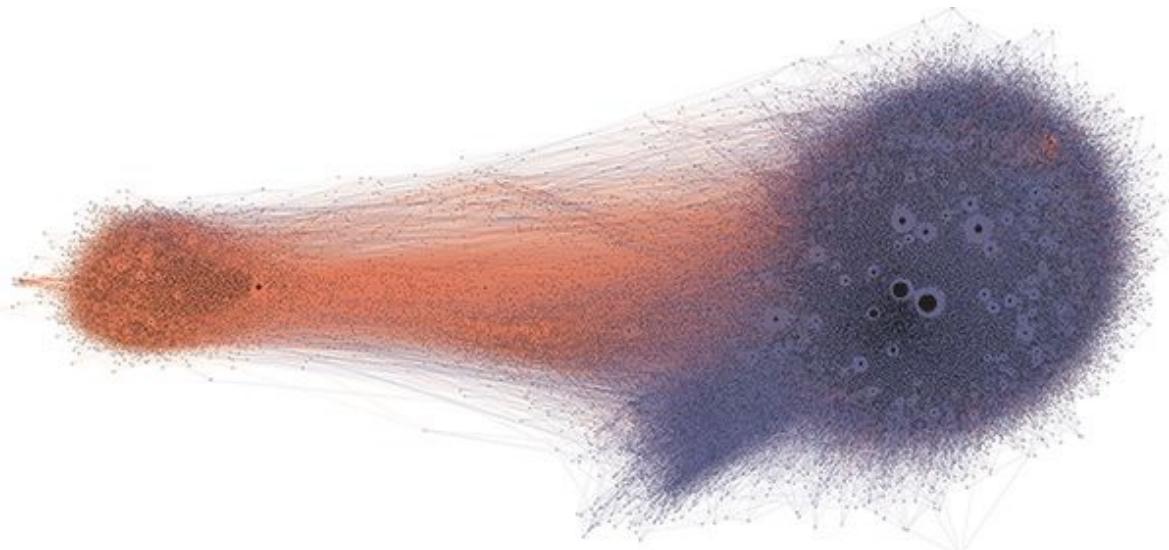


- Difuzija u Twitter mreži:
 - Retweet, quoted retweet, spominjanje, odgovori
 - Retweet kaskada – usmjereno stablo tijekom širenja meme od kreatora do svih izloženih korisnika
 - Jedan meme može generirati više kaskadnih stabala
 - Agregacija stabala – šuma
 - Kaskadnu šumu možemo promatrati kao sloj u višeslojnoj mreži

Primjer difuzne mreže – 2016 izbori u US

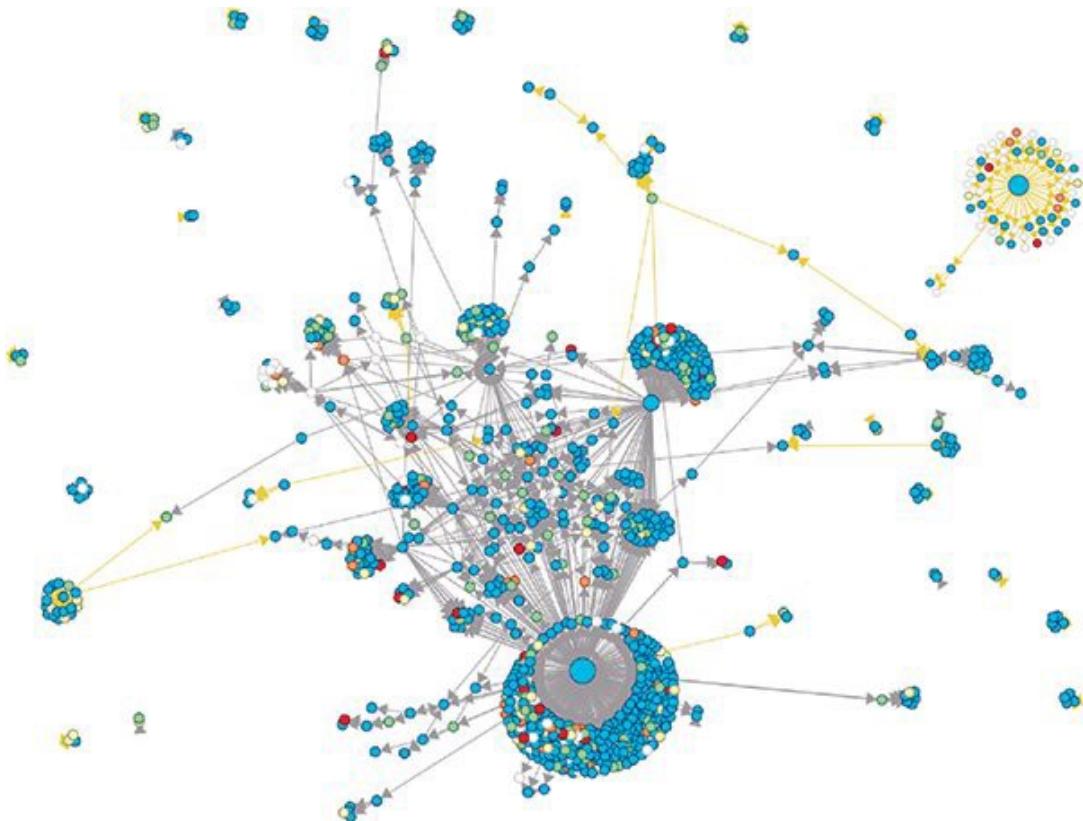


Primjer difuzne mreže – 2016 izbori u US



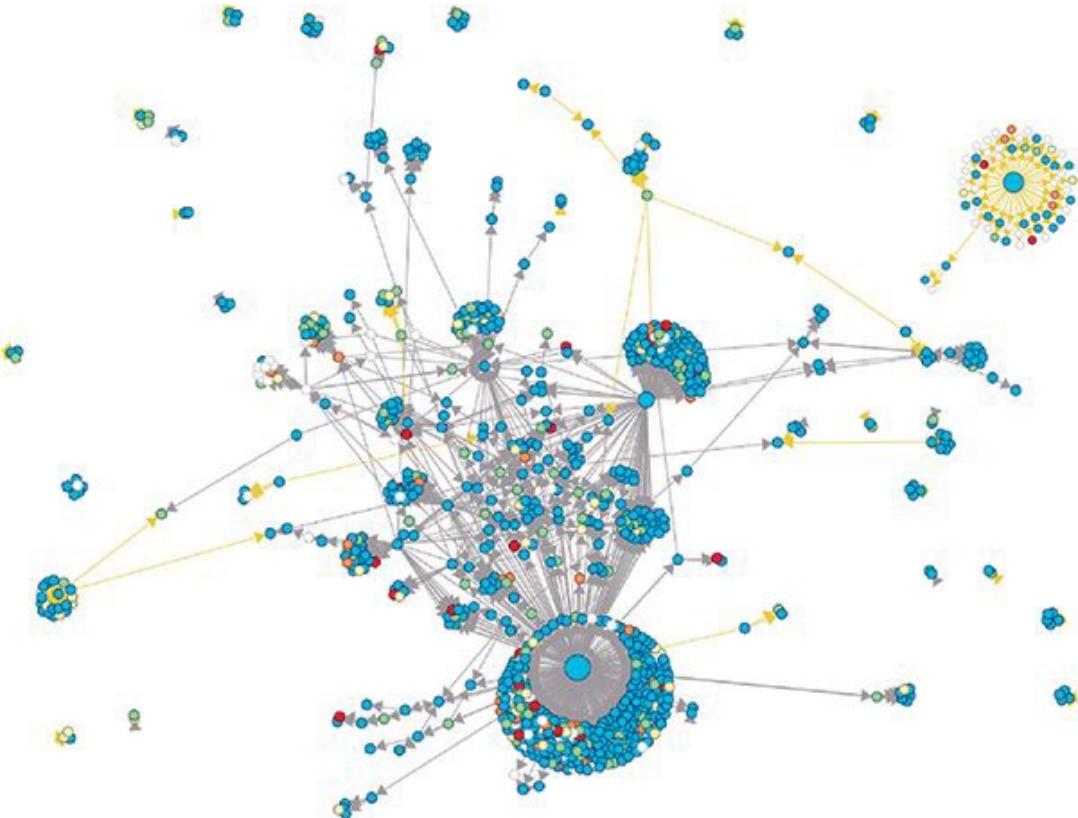
- Podmreža retweetova članaka vezanih uz izbore 2016
- Čvorovi – Twitter računi,
- Veze - retweet linka na članak (crveni – provjera informacije, ljubičasta niska vjerodostojnost)
- $k = 5$ jezgra pune mreže
- Korisnici koji šire dezinformacije dijele jako mala članaka iz izvora koji provjeravaju istinitost

Zaraznost memea



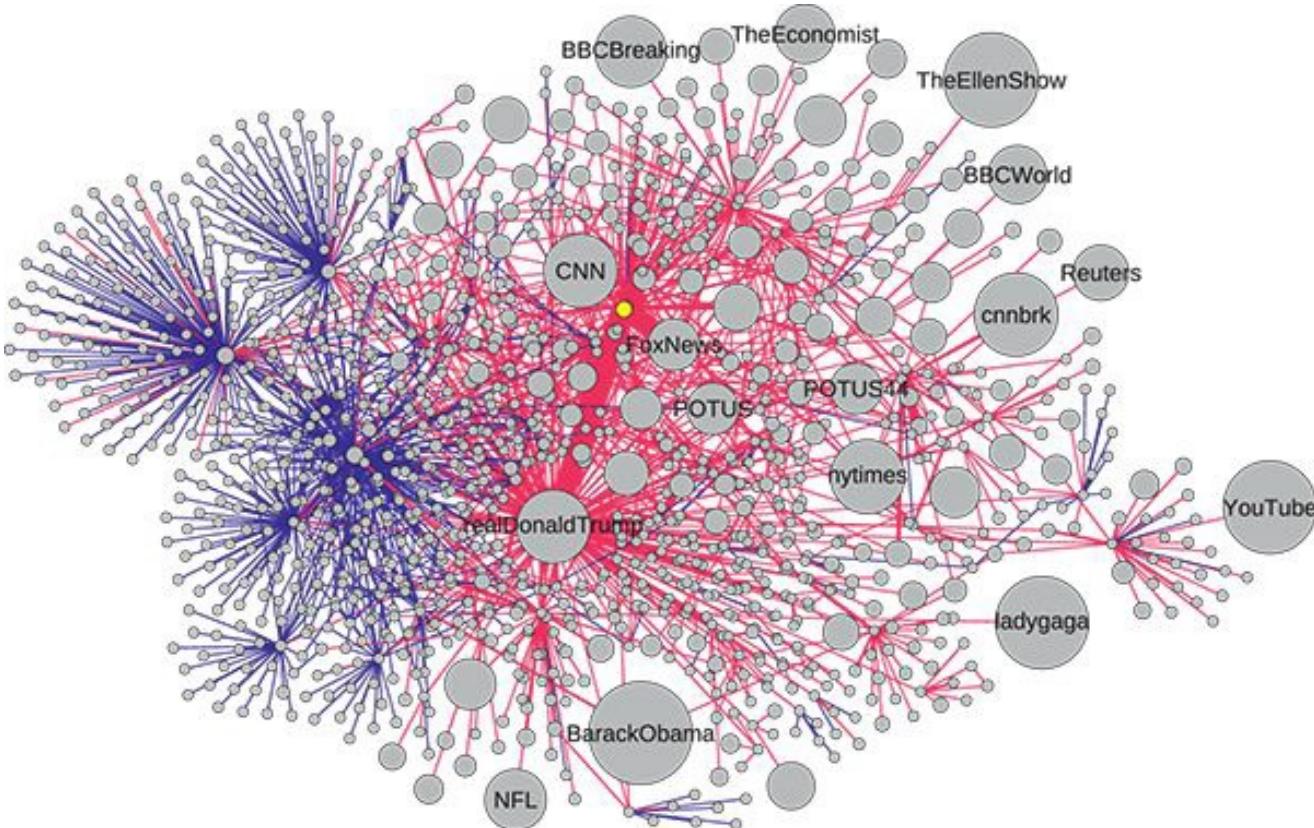
- Mjerenje zaraznosti:
 - Broj izloženih korisnika
 - Struktura mreže. *Meme* koji šire popularne osobe može biti više vezan uz to osobu nego *meme*
 - Duboka mreža *retweetova* može govoriti o široj privlačnosti poruke

Retweet mreža dva članka Bijelih Kaciga u Siriji



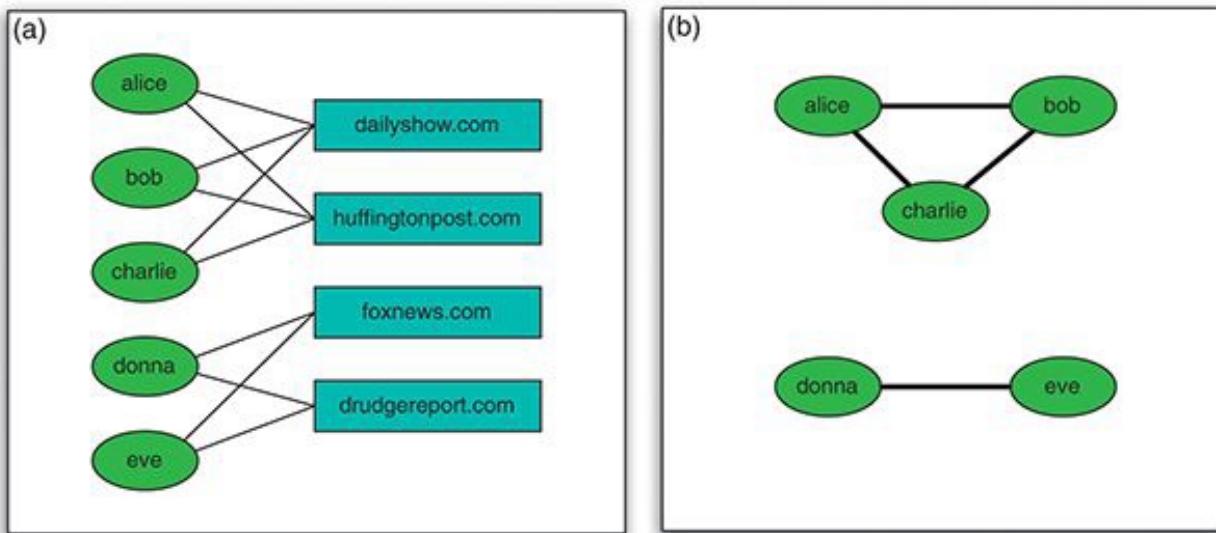
- Kampanja dezinformacijama
 - Lažna povezanost s teroristima
 - Druge teorije zavjere
 - Sive veze – dezinformacija
 - Žute veze – provjerena tvrdnja
 - Plavi čvorovi – ljudi
 - Crveni čvorovi – botovi
- DeepFake

Mreža lažnih vijesti o izbornoj prevari izbora 2016 od strane ilegalnih stranaca



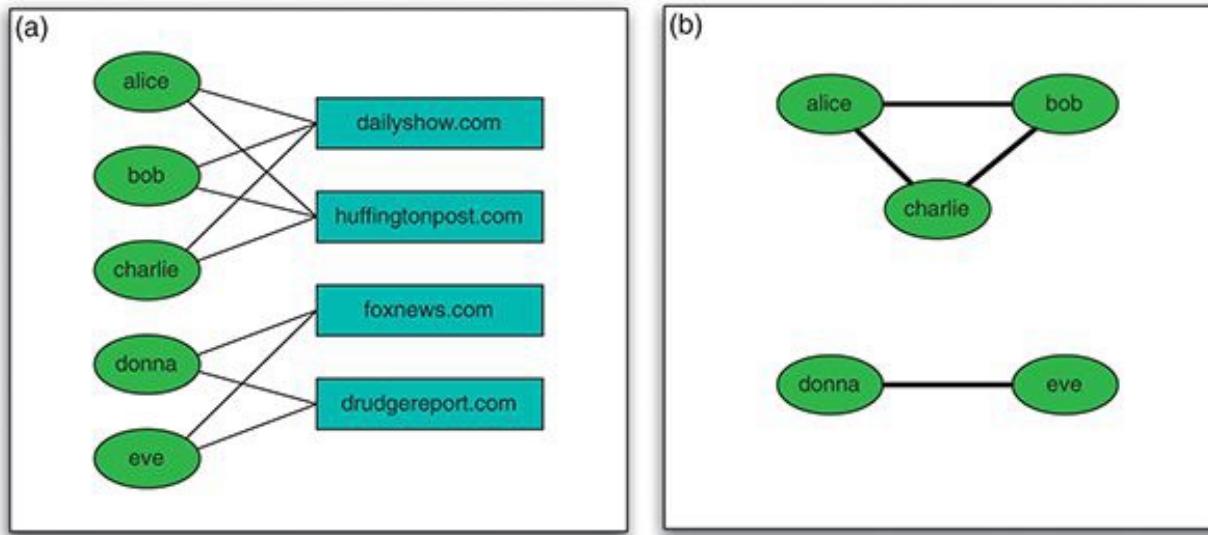
- Plavo - Citirani tweetovi i retweetovi
- Crveno – spominjanje i odgovori
- Širina linka – težina
- Mali žuti čvor – *bot* koji je sistematski širio dezinformacije u odgovorima na informacije koje su spominjale US predsjednika

Mreže zajedničkog pojavljivanja



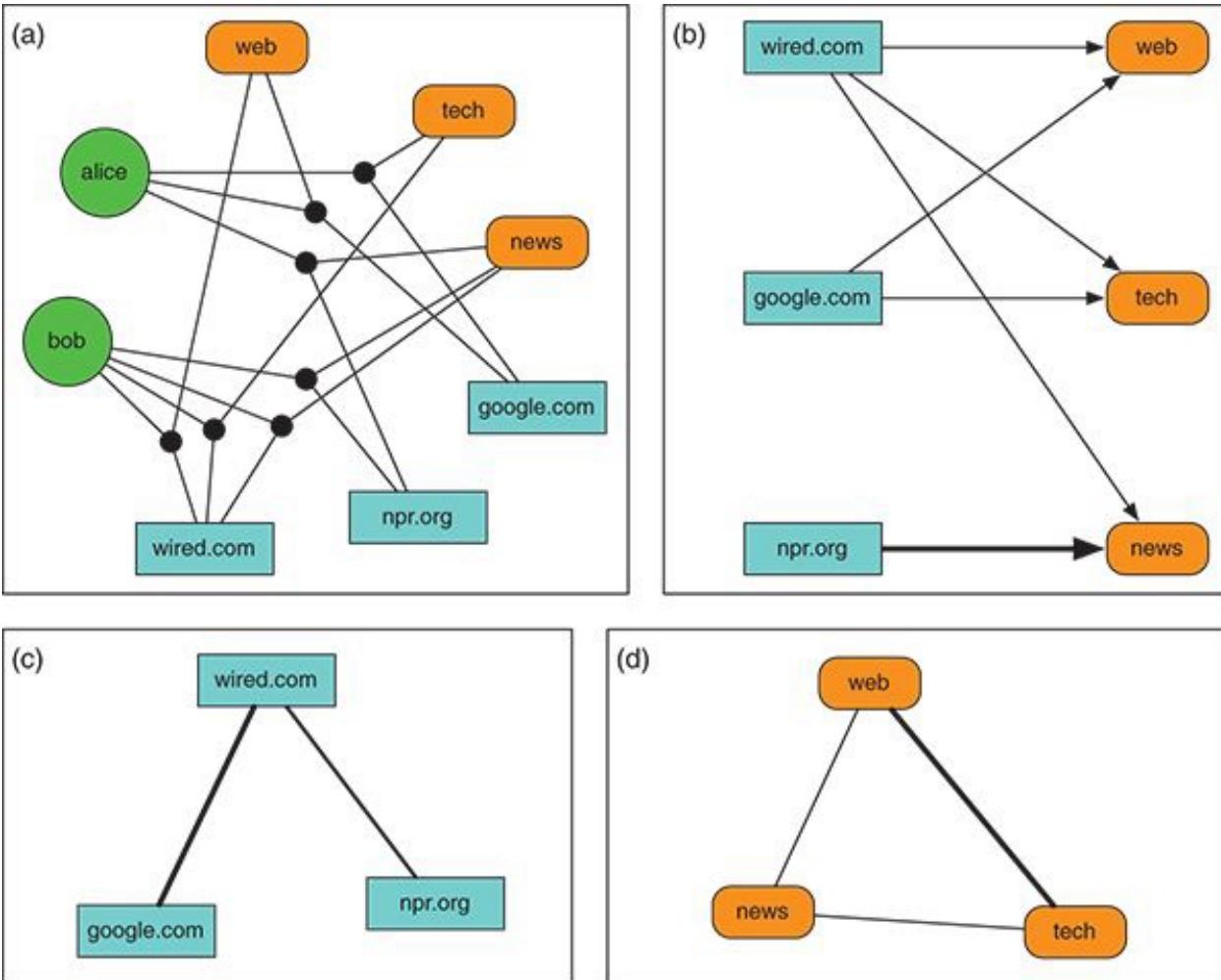
- Veze između različitih vrsta entiteta
- Usmjerena mreža
 - Svi izvođeni čvorovi na jednu stranu
 - Svi odredišni na drugu
 - Isti čvor može biti na obje strane (difikat)
 - Mreža citata (rad koji citira i citirani rad) - > nova mreža u svakoj od grupe
 - Radovi mogu biti ko-citirani i biti ko-referencirani (nekoliko radova citira jedan ili više radova)
- Bipartitne mreže – način reprezentacije ovakvih mreža

Bipartitna mreža -> Mreža zajedničkog pojavljivanja



- Veze između dva čvora različitog tipa
- Mreža ko-zvijezda u filmovima
 - Kreiramo težinsku mrežu iz bipartitne (projekcija) – mreža zajedničkog pojavljivanja
- Bipartitna mreža „likeanja“ (a)
- Mreža zajedničkog pojavljivanja (b)
- Koriste se za preporuke i ciljano oglašavanje

Folksonomy

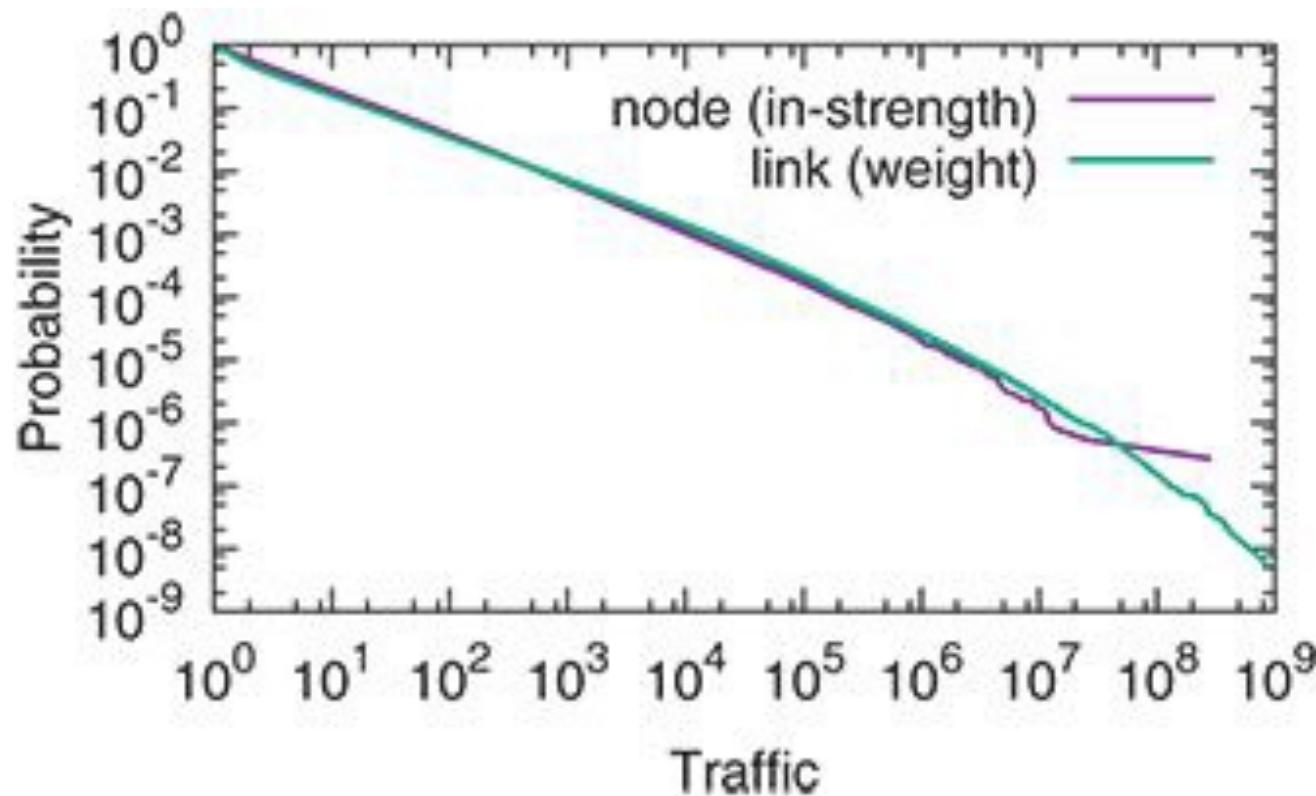


- Bipartitne mreže isto mogu imati težine.
- Sustavi ocjenjivanja – koliko nam se npr. sviđa knjiga ili film
- Označavanje – označavanje izvora (identificiranog URL-om) s jednom ili više oznaka (npr. YouTube)
- Osnovni element je trojka (u, r, t) gdje korisnik u , označava resurs r s oznakom (*hashtag* kod Twittera)
- Agregirano po puno korisnika dobijemo skup koji nazivamo *folksonomy*

Heterogenost težina

- Težina veze može nositi informaciju o procesu ili odnosima modeliranim mrežom
- Različite težine mogu predstavljati različito udruživanje
- Mreže prometa
 - Putnici ili letovi između aerodroma
 - Auti između križanja
 - Internet (paketi između usmjerivača)
 - Wikipedia (klikovi između članaka)

Mreža Web prometa



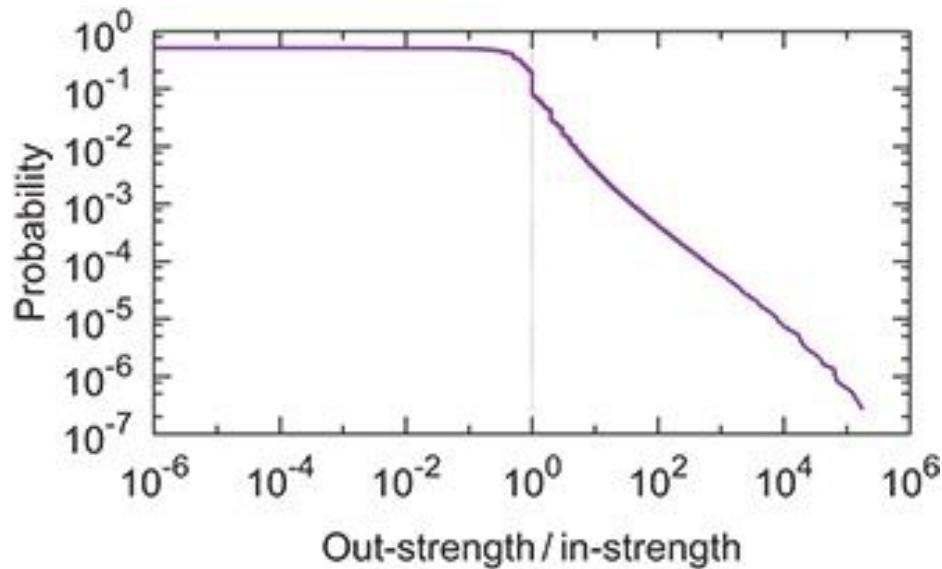
- Ulazna snaga čvora – ukupan broj klikova na stranicu
- Težina veze – ukupan broj klikova na hiperlink
- Obje distribucije heterogene (otežani rep)

Postoji li korelacija između
PageRanka i ulazne snage ??

Usporedba PageRanka i ulazne snage

- Može li PageRank predvidjeti promet?
- NE, usprkos sličnoj distribuciji
- Korelacija je zapravo slaba
- Pretpostavka PageRank modela – model slučajnog surfera
- Inicijalne pretpostavke PageRank modela su narušene

Koje pretpostavke PageRanka su najmanje realistične?



- Omjer ulazne i izlazne snage čvorova
 - Osim za početne i krajnje čvorove jednog pretraživanja omjer treba biti 1
- Teleportiranje ne favorizira niti jedan čvor
 - Jednaka vjerojatnost za svaki čvor da će biti izvor ili odredište skoka (i s ovime omjer je blizu 1)
- Očekujemo usku distribuciju oko 1
- U stvarnosti krećemo od manjeg skupa stranica
- Većina čvorova nije zanimljiva i na njima završavamo pretragu i skačemo dalje
- **Slučajna teleportacija je nerealistična**

Kompleksne mreže

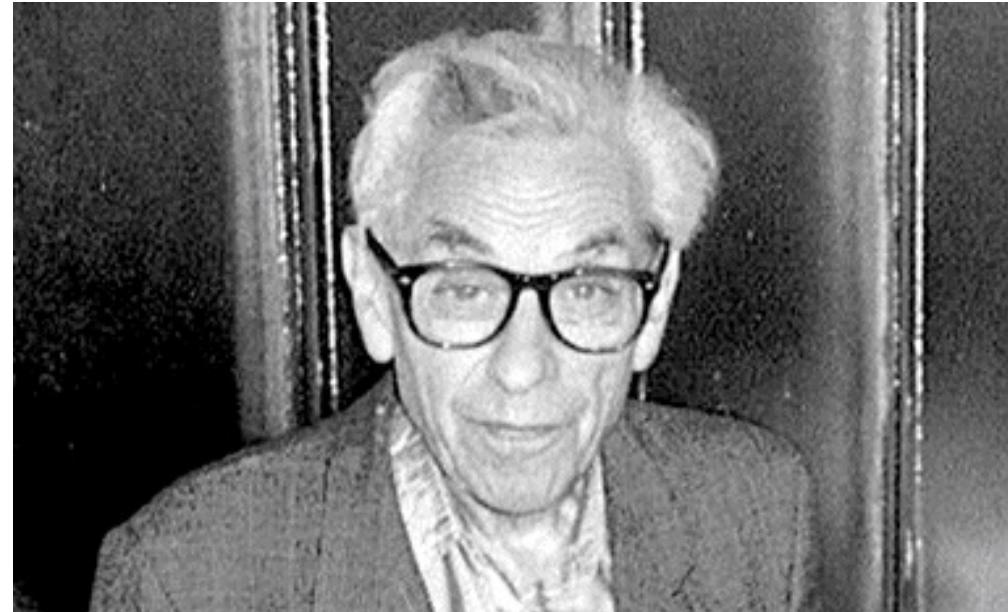
5. Predavanje

Realne mreže

- Kratak najkraći put
- Puno trokuta, što rezultira visokim koeficijentom klasteriranja
- Heterogene distribucije varijabli čvora i veza poput stupnja i težina
- Od kuda nam ta svojstva - **MODEL** za izgradnju mreža

Slučajne mreže

- Krenemo sa skupinom nepovezanih čvorova
- Dodamo veze između slučajno odabralih parova čvorova
- Slučajna ili Erdos-Renyi mreža



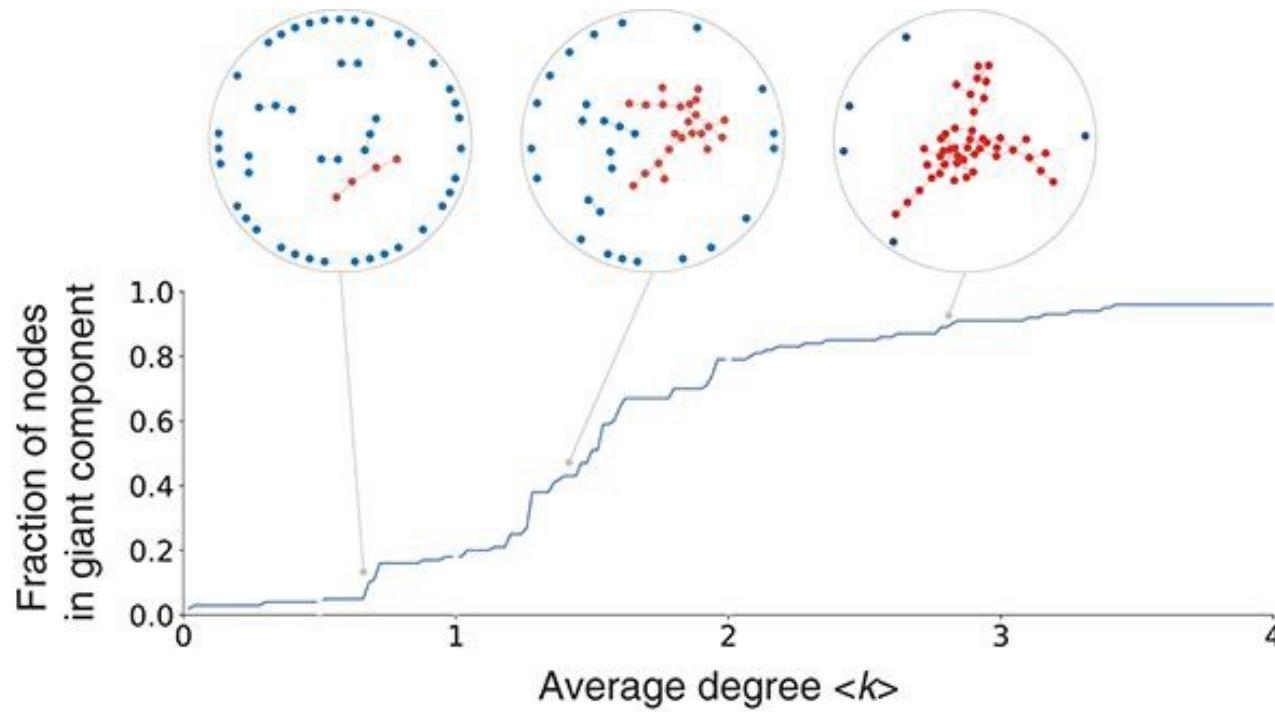
Slučajne mreže

- Gilbertov model nasuprot Erdos-Reny model
 - Erdos-Reny – broj čvorova i veza fiskan
 - Gilbert model – broj čvorova fiksan, broj veza varira
- Gilbertov model:
 - Zadan je broj čvorova N i vjerojatnost veze p
 - 1. Odaberemo par čvorova i i j
 - 2. Generiramo slučajan broj r između 0 i 1. Ako je $r < p$, dodamo vezu između i i j
 - 3. Ponavljamo (1) i (2) za sve parove čvorova

Rast mreže

- Zamislimo velik broj čvorova, bez veza
- Sustav je razlomljen u singletone (izolirane čvorove)
- Dodajemo veze, jednu u svakom vremenskom trenutku
- Sve više čvorova parova čvorova se povezuje
- Nastanak povezanih podmreža
- Mreža postaje povezana
- Tranzicija od puno malih povezanih komponenti u jednu veliku (većina grafa)
- Suprotno očekivanju, tranzicija je nagla – za $\langle k \rangle = 1$

Evolucija Erdos-Renyi grafa za različiti srednji stupanj



Gustoća

- Očekivani broj veza u slučajnoj mreži – proporcionalan vjerovatnosti veze i broju parova čvorova
- Broj mogućih parova čvorova $\binom{N}{2} = \frac{N(N-1)}{2}$
- Očekivani broj veza u slučajnom grafu $\langle L \rangle = p \binom{N}{2} = \frac{pN(N-1)}{2}$
- Očekivani prosječni stupanj $\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N - 1)$
- Gustoća $d = \frac{\langle k \rangle}{N-1} \rightarrow$ očekivana gustoća $\langle d \rangle = p$
- Realne mreže – rijetke (mali $\langle k \rangle$ u odnosu na ukupan broj čvorova i mala gustoća) $\rightarrow p$ treba biti mali

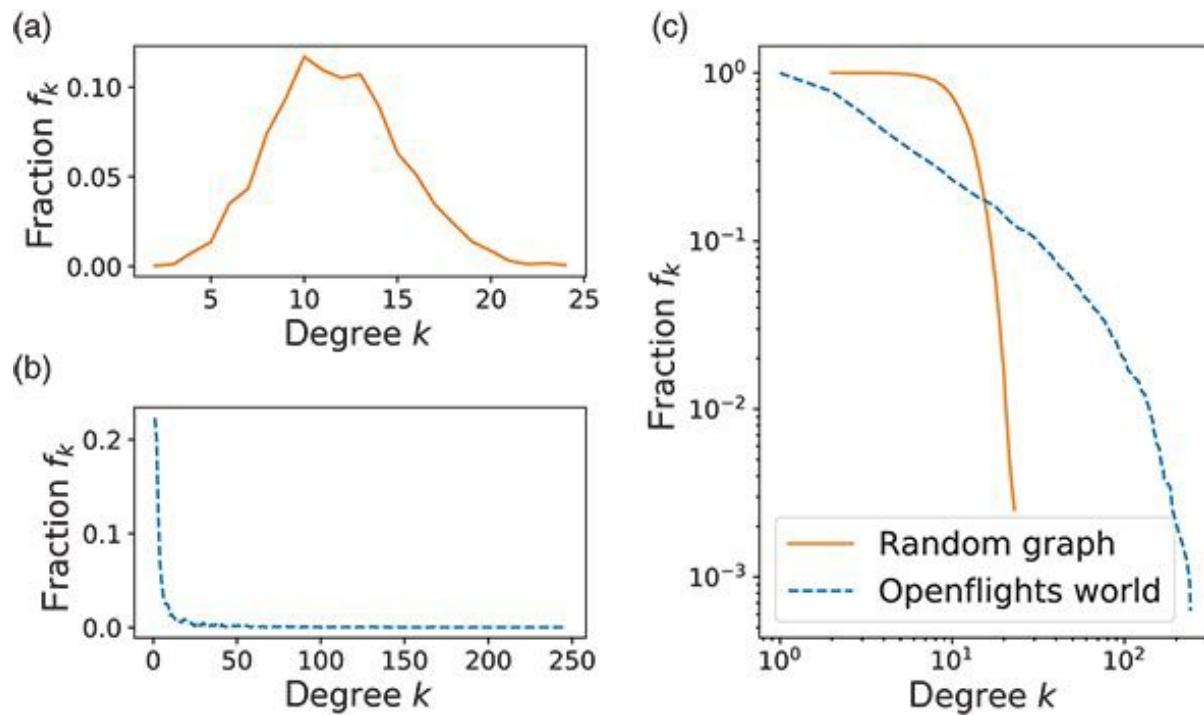
Distribucija stupnja

- Distribucija stupnja slučajne mreže - vjerojatnost da čvor ima k susjeda
- Niti jedan čvor nema posebnu ulogu u ovom modelu
- Kolika je vjerojatnost da čvor i ima nula, jednog, dva ili više susjeda
- Svaki od preostalih $N-1$ čvorova može biti susjed od i
- Svaki par koji uključuje i ima vjerojatnost p da bude spojen, neovisno o ostatku mreže

Distribucija stupnja

- Ekvivalentan problem – broj glava u $N-1$ bacanja (vjerojatnost glave p)
- Binomna distribucija $P(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}$
- Za veliki N , $pN \approx \langle k \rangle$ pri čemu binomna distribucija se dobro aproksimira Poissonovom sa srednjom vrijednošću i varijancom $\langle k \rangle$

Distribucija stupnja



- Usporedba Erdos-Renyi slučajnoj grafa s mrežom avionskih letova istoga broja čvorova i veza

Kratki putevi

- Pretpostavimo da je mreža povezana i svi čvorovi imaju stupanj k
- Unutar jednog koraka $l = 1$ dosežemo k čvorova
- Svaki od njih ima $k-1$ susjeda -> nakon dva koraka smo dosegli $k(k-1)$ čvorova
- Unutar tri koraka dosižemo $k(k - 1)^2$ čvorova
- Možemo zaključiti da na udaljenosti l od korijena možemo naći $k(k - 1)^{l-1}$ čvorova
- Za veliki k u l koraka dosižemo k^l čvorova

Kratki putevi

- $k^{l_{max}} = N$
- $l_{max} = \log_k N = \frac{\log N}{\log k}$
- Aproksimacija diametra mreže ako uzmemo u obzir preklapanja i fluktuacije u stupnju oko $\langle k \rangle$
- udaljenosti su male čak i kada je mreža jako velika

Koeficijent klasteriranja

- Udio trokuta kojima je središte u promatranom čvoru
- Slučajna mreža – vjerojatnost da je par susjeda čvora povezan je p
- Koeficijent klasteriranja pojedinog čvora može razlikovati od p , no srednja vrijednost kroz sve čvorove se može aproksimirati s p
- Zahtjev za realne mreže je mali $p \rightarrow$ mali očekivani koeficijent klasteriranja

Usporedba realne i slučajne mreže

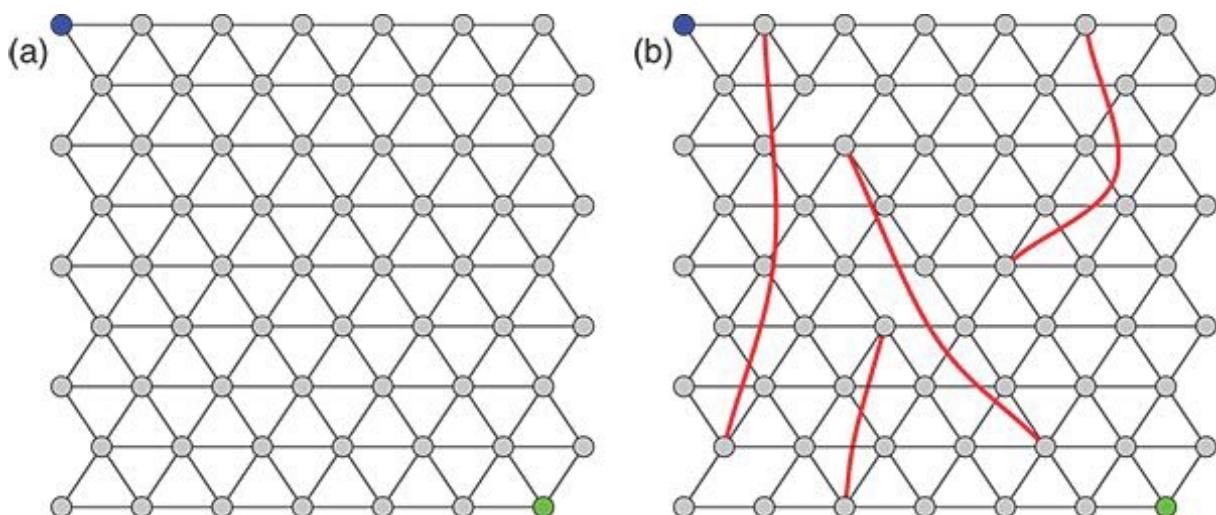
Svojstvo	Realna	Slučajna
Rijetka mreža	DA	DA, ako je p mali
Distribucija stupnja	Opadajuća s otežanim repom	Zvonolika
Najkraći srednji put	Kratak	Kratak
Koeficijent grupiranja	Velik	Malen za mali p

Mali svijet

- Slučajne mreže različite od realnih
- Watts-Strogatz model (1998)
- Kratak najkraći put i visok koeficijent klasteriranja

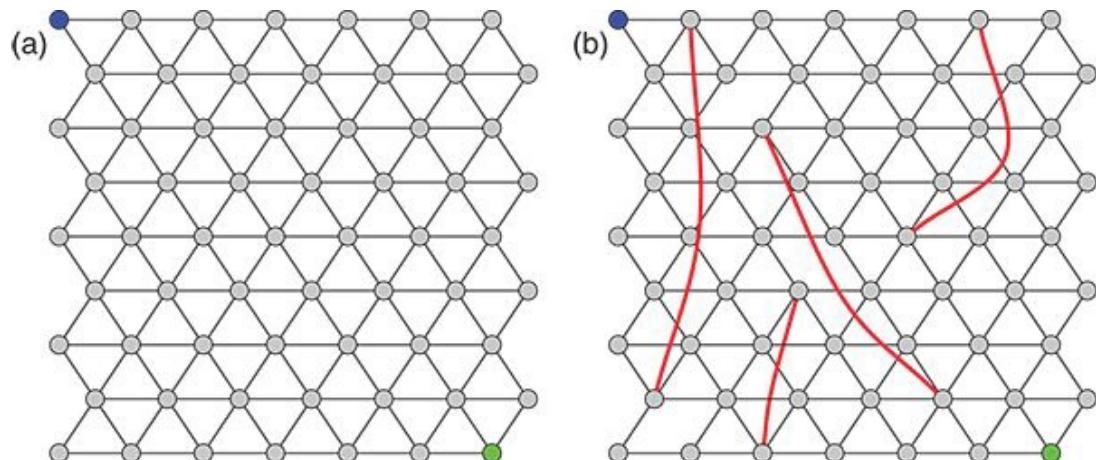
Model malog svijeta

- Krenemo od rešetke (svaki čvor ima isti broj susjeda)
- Visok stupanj klasteriranja – svaki par uzastopnih susjeda povezan
- $C = 6/{6 \choose 2} = \frac{2}{5}$
- za rubne čvorove klastering i veći
- Dugačak najkraći put



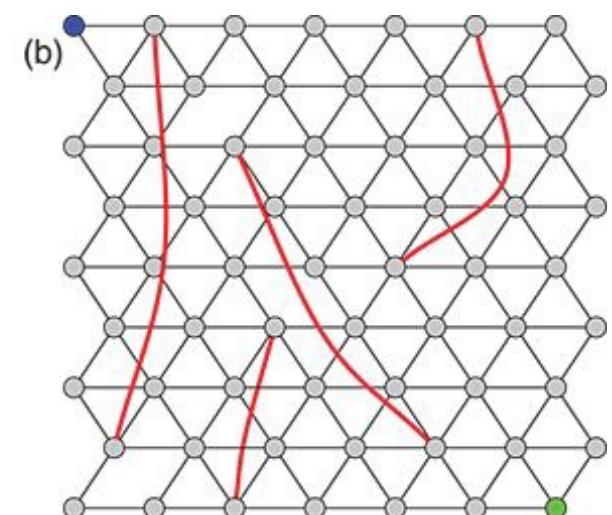
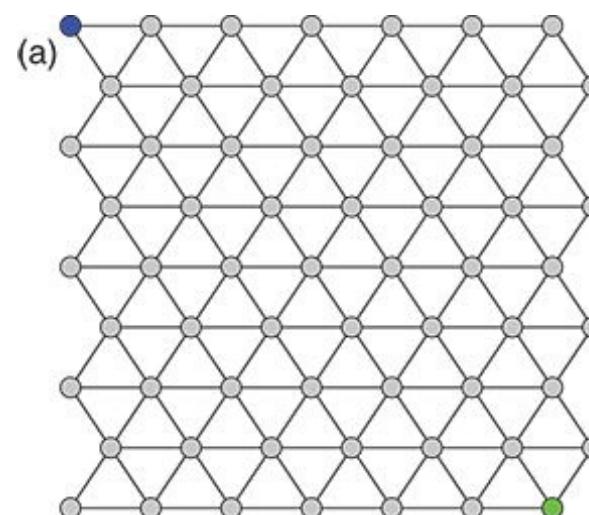
Model malog svijeta

- Smanjenje udaljenosti između čvorova kreiranje prečica među čvorovima
- Odabir početnih veza na slučajan način, sačuvamo jedan čvor, a drugi zamjenom sa slučajno odabranim
- Vjerojatnost premošćivanja pojedine veze p
- Broj premoštenih veza proporcionalan vjerojatnosti premošćivanja

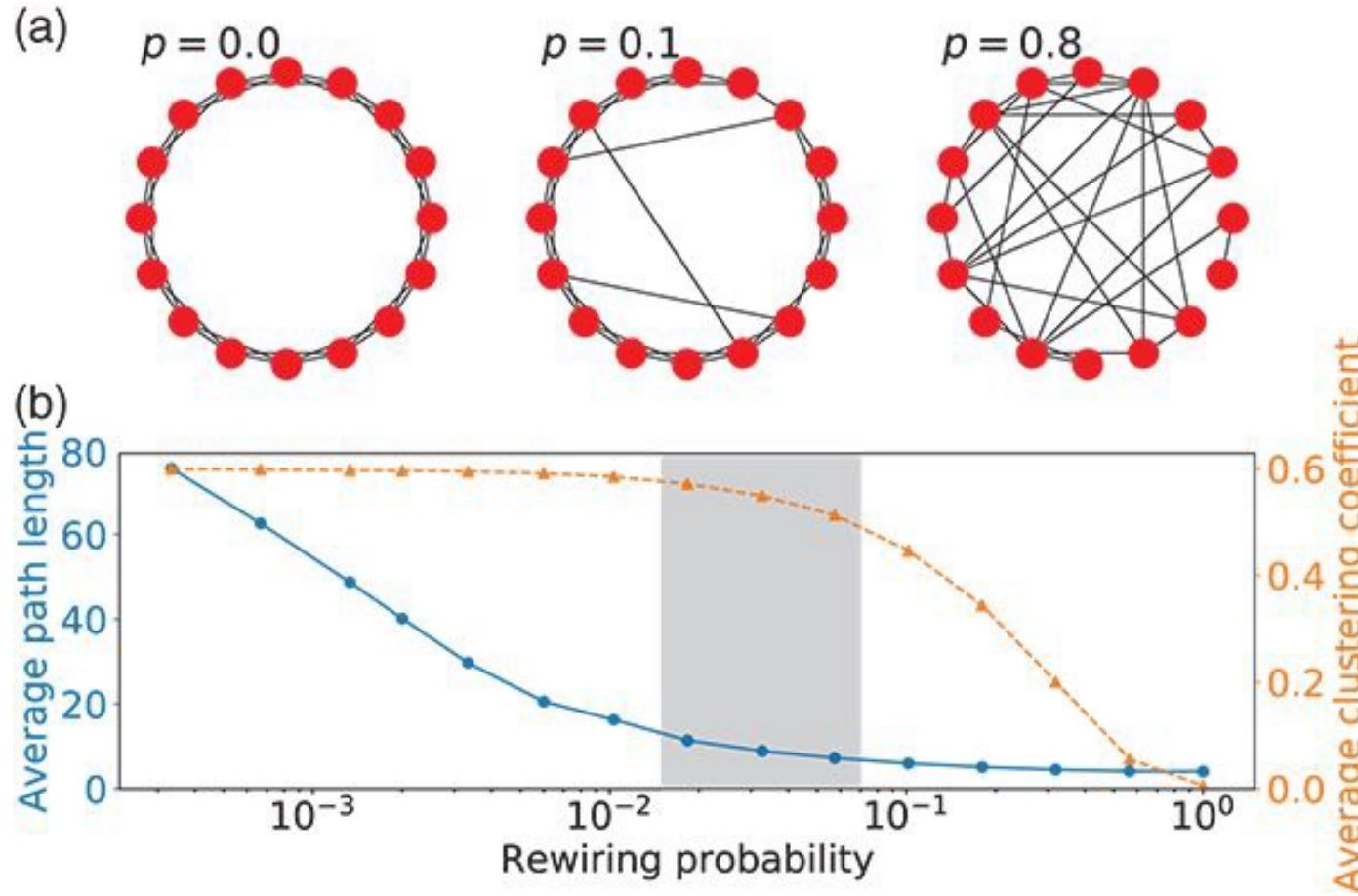


Model malog svijeta

- Premošćivanje rijetko – malo se toga dogodi
- Premošćivanje često – slučajna mreža
- Ako je p niti velik ni mali -> moguće postići mali najkraći put sa sačuvanim visokim koeficijentom klasteriranja



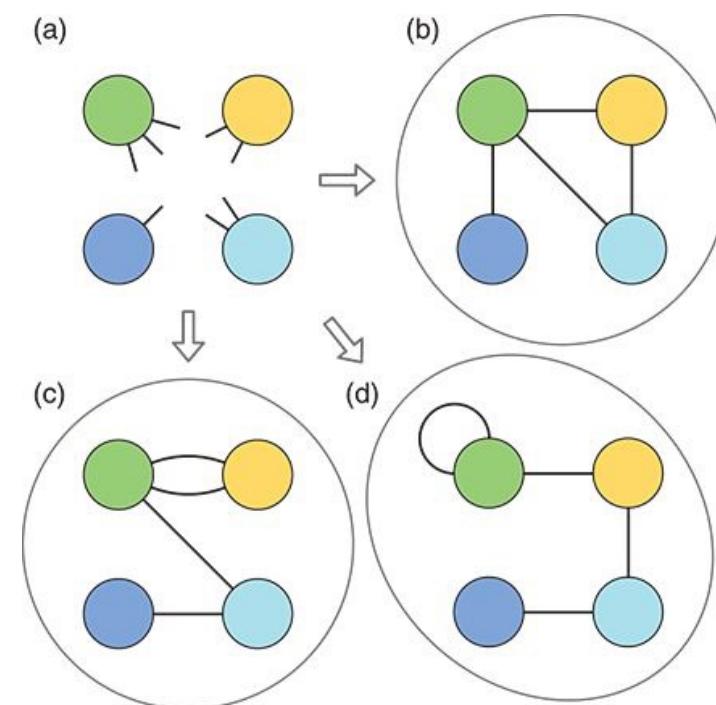
Originalan rad Wattsa i Strogatza



- Kratki putevi i veliki koeficijent klasteriranja
- **Model ne može proizvesti hubove**
- Za bilo koju vjerojatnost premošćivanja broj čvorova i veza ostaje isti
- Distribucija stupnja zvonolika

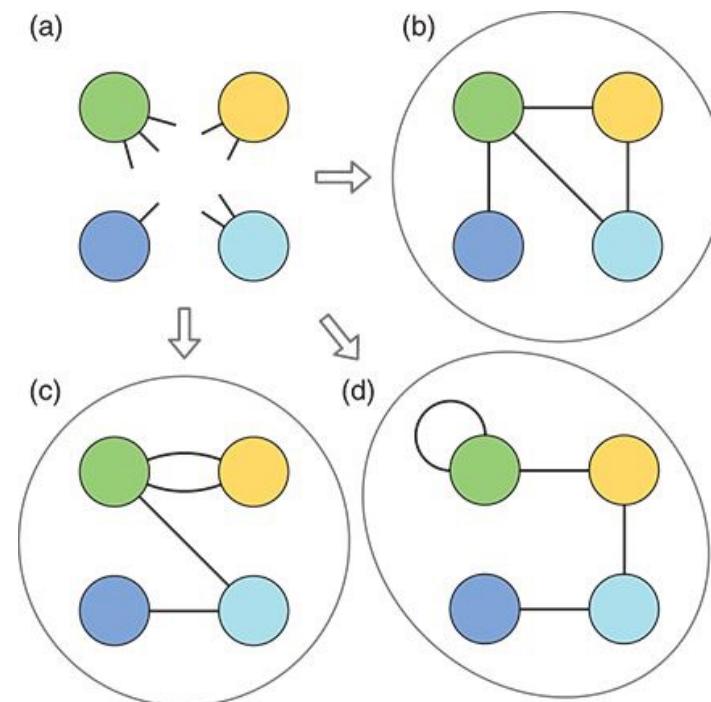
Konfiguracijski model

- Cilj – proizvesti mrežu čiji čvorovi imaju proizvoljan slijed stupnjeva (npr. prvi čvor ima stupanj k_1 , drugi čvor ima stupanj k_2 i tako redom)
- Slijed stupnjeva
 - Željena distribucija
 - Iz realne mreže



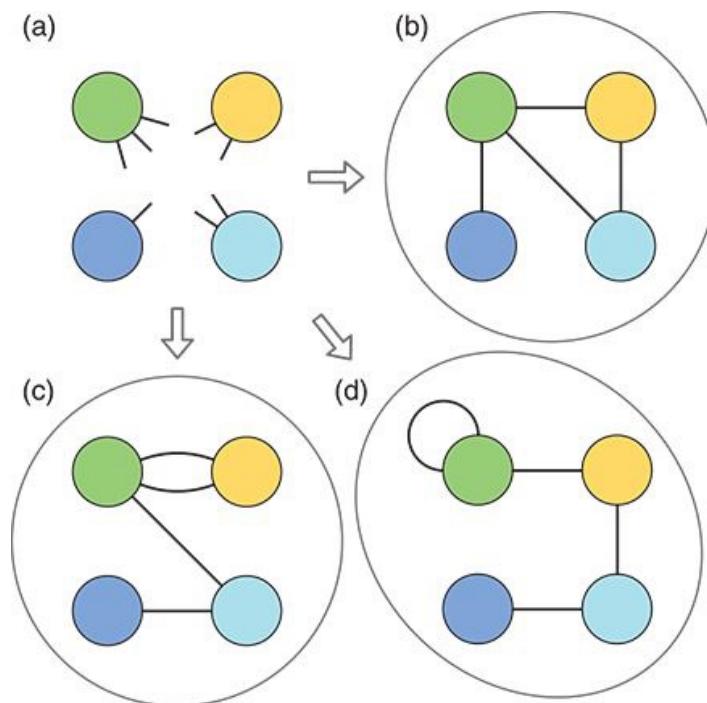
Konfiguracijski model

- Dodjela svakom čvoru izdanka s kojim može kreirati vezu
- Suma izdanaka mora biti paran broj
- Različite mreže kreirane – ovisno o broju kombinacija parova izdanaka
- Pojedini rezultati su neželjeni (npr. Višestruke veze između dva čvora)



Konfiguracijski model

- Za danu mrežu, možemo istražiti je li specifično svojstvo definirano distribucijom stupnja
- Kreiramo po volji nove mreže s istom distribucijom
- Provjerimo jesu li nove mreže zadržale promatrano svojstvo
- Ako ne, postoji drugi faktori u pozadini
- Primjer – koeficijent klasteriranja
- Eksponencijalni slučajni grafovi



Eksponencijalni slučajni modeli

- Proučavanje slučajno proizvedenih mreža koje dijele zajednička kvantitativna svojstva, s razlikom u detaljnoj strukturi
- Potencijalna alternativa specifičnim mrežnim konfiguracijama koje srećom u realnom svijetu
- Omogućeno istraživanje međudjelovanja različitih strukturnih svojstava
- Primjer – koje vrijednosti koeficijenta klasteriranja su kompatibilne specifičnoj vrijednosti gustoće

Eksponencijalni slučajni grafovi

- Klasa slučajnih mreža s ograničenjima
- Definiramo klasu mreža na osnovu skupa M mrežnih mjera x_m , $m=1,\dots,M$
- Unosimo ograničenje za svaku mjeru x_m : srednja vrijednost kroz sve mreže unutar klase mora biti specificirana vrijednost $\langle x_m \rangle = x_m^*$
- Eksponencijalni slučajni grafovi – mreže koje zadovoljavaju ograničenje dok maksimiziraju nasumičnost

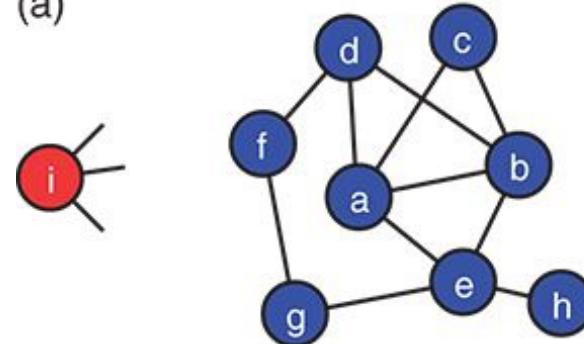
Preferencijalno pridruživanje

- Modeli koje smo do sada istraživali
 - Statični
 - Nema hubova ili znamo veličinu od početka – ne znamo zašto nastaju
- Broj čvorova poznat od početka, dodajemo veze
- Realne mreže su obično dinamičke
- Popularne mreže: Internet, Twitter, LinkedIn
- Veličina stvarnih mreža raste
- Čvorovi mogu nestajati
- Veća vjerojatnost pojave novih čvorova u mreži
- Dinamički modeli rasta

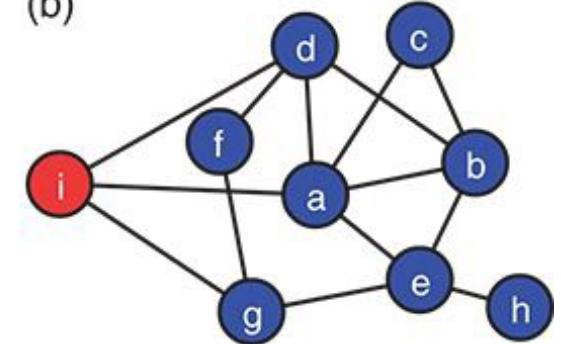
Preferencijalno pridruživanje

- Krećemo od inicijalne konfiguracije (npr. jako mala klika)
- Dodajemo čvor po čvor
- Novi čvor dodajemo na neki broj postojećih čvorova na osnovu pravila – karakteristika modela

(a)



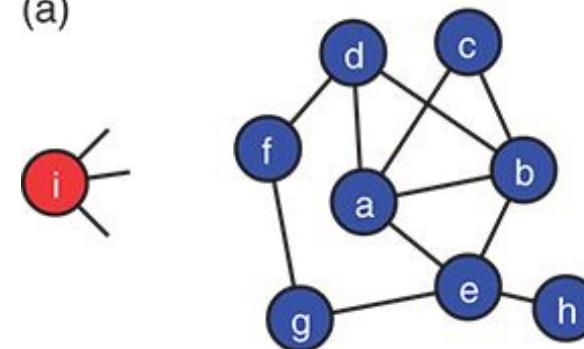
(b)



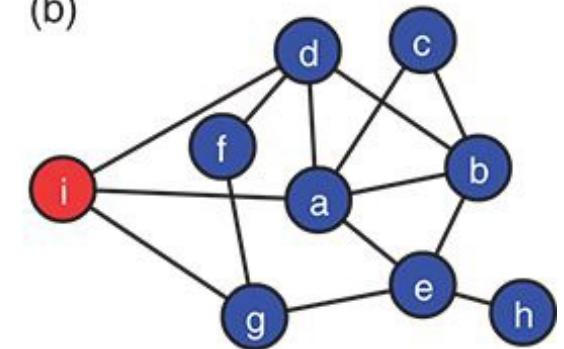
Preferencijalno pridruživanje

- Slučajne mreže i mreže malog svijeta bez hubova
- Jednakost – čvorovi biraju susjede totalno slučajno
- Niti jedan čvor nema prednost pred ostalima
- Mehanizam koji favorizira pojedine čvorove – **preferencijalno pridruživanje**
- Veći stupanj - > više novih veza

(a)

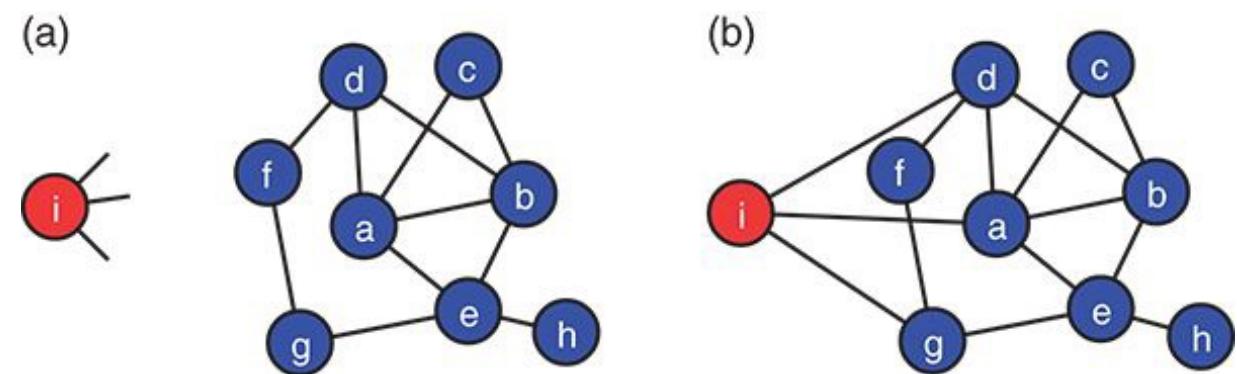


(b)



Preferencijalno pridruživanje

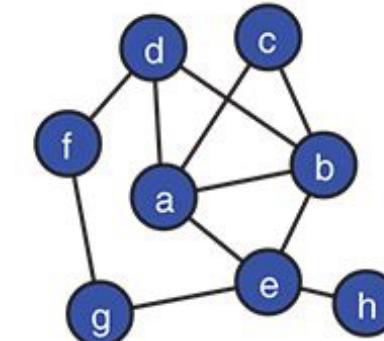
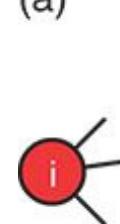
- **Primjer Weba**
- Biljni stranica
- Većina stranica kojih smo svjesni – popularne s velikim brojem veza
- Nova stranica – favoriziramo povezivanje na popularne, visoko povezane stranice
- Sličan primjer sa citatima – citiramo one citirane od strane drugih autora
- Čvorovi s visokim stupnjem - veća vjerojatnost povezivanja



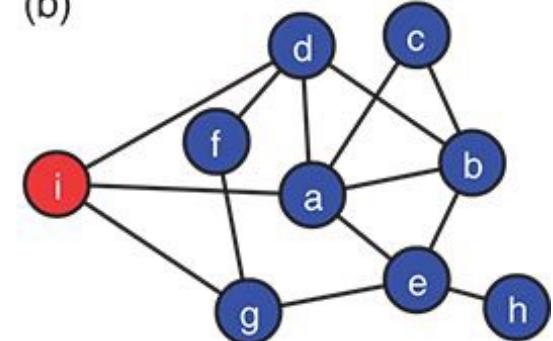
Albert – Barabasi model

- Model preferencijalnog povezivanja
- Dodavanje novog čvora na postojeći je proporcionalno stupnju postojećeg
- Evanđelje po Mateju (Matthew):
“Tko ima, dat će mu se još pa će obilovati, a onome tko nema oduzet će se i ono što ima”

(a)



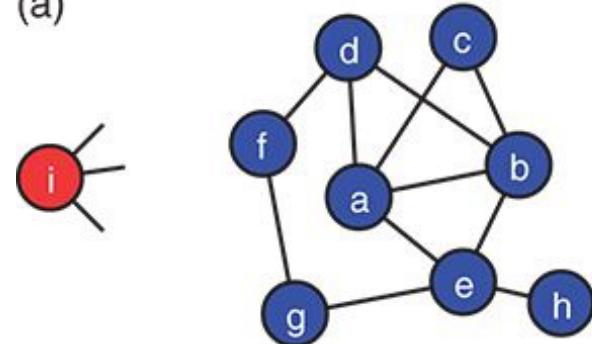
(b)



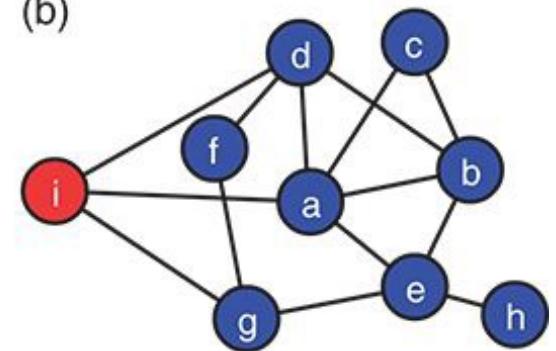
Preferencijalno pridruživanje

- Bogati postaju još bogatiji
- Siromašniji postaju siromašniji
- Dodatni nazivi
 - Matthew efekt
 - Kumulativna prednost

(a)



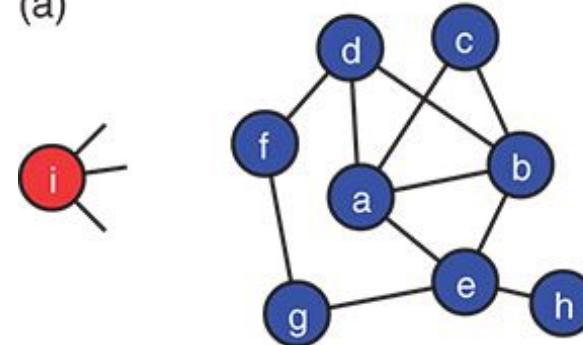
(b)



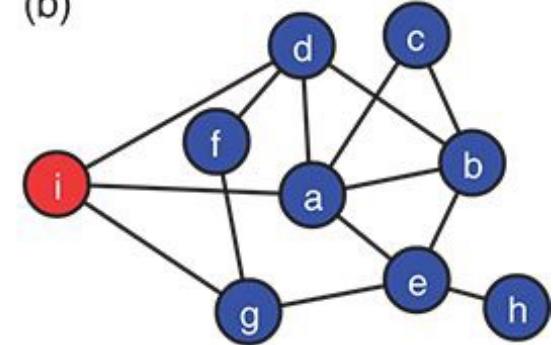
Preferencijalno pridruživanje

- Preferencijalno pridruživanje se koristi za objašnjenje otežanog repa distribucije
 - Broj vrsta biljaka u pojedinom genuusu
 - Broj riječi u tekstu
 - Populacija gradova
 - Individualno bogatstvo
 - Znanstvena produkcija
 - Statistika citiranosti

(a)



(b)



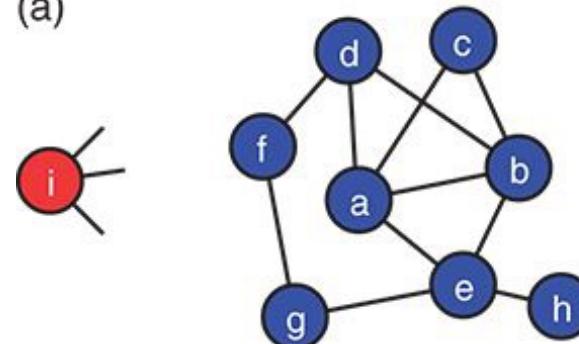
Preferencijalno pridruživanje

- Krećemo s potpunim grafom s m_0 čvorova. Svaka iteracija se sastoji od dva koraka:

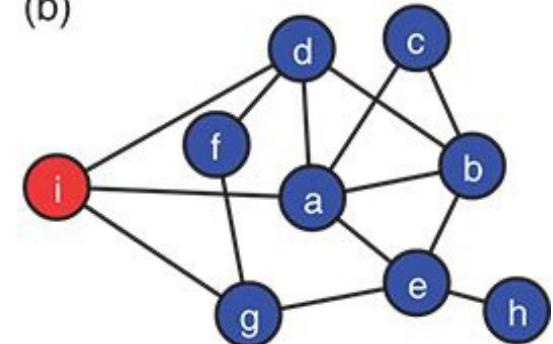
1. Novi čvor se dodaje u mrežu s $m \leq m_0$ novih veza povezanih na njega. Parametar m – prosječan stupanj
2. Svaka nova veza s povezuje s postojećim čvorom j s vjerojatnošću $\Pi(i \leftrightarrow j) = \frac{k_j}{\sum_l k_l}$

- Ponavljamo dok ne dođemo do željenog broja čvorova

(a)



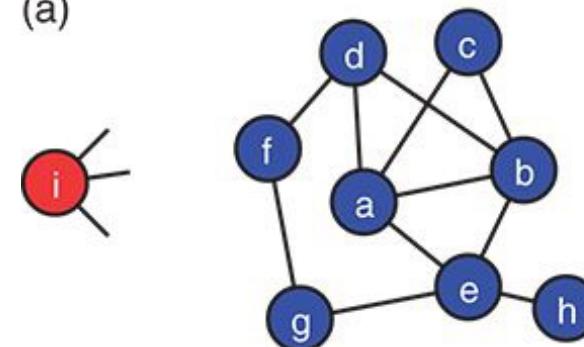
(b)



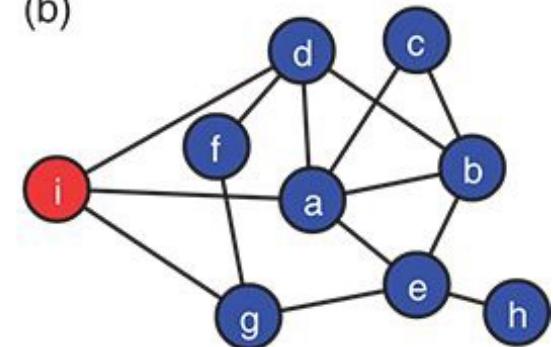
Preferencijalno pridruživanje

- Na početku svi čvorovi imaju isti stupanj
- Dodavanjem novih čvorova i veza raste stupanj čvorova
- Najstariji čvorovi mogu dobiti nove veze u bilo koje trenutku – prednost pred onim koji su kasnije dodani
- Stupanj starijih prelazi onaj novih -> raste vjerojatnost povezivanja
- Bogati postaju bogatiji
- Najstariji čvorovi postaju hubovi

(a)

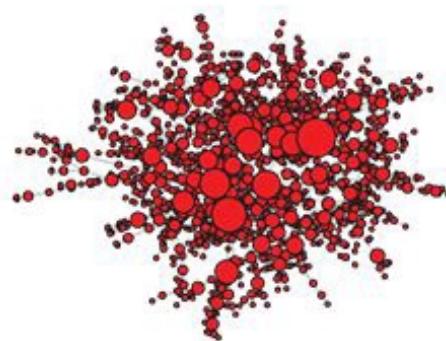


(b)



Preferencijalno pridruživanje

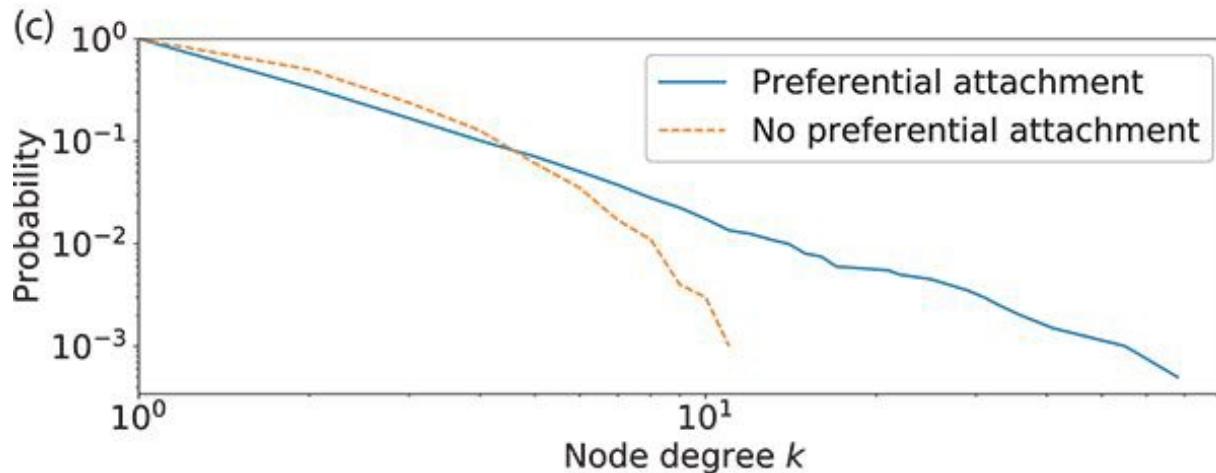
(a)



(b)



(c)



- a) Mreža generirana Albert-Barabasi modelom
- b) sličan model rasta sa slučajnim odabirom umjesto preferencijalnog
- c) kumulativne distribucije



Drugi preferencijalni modeli

- Albert-Barabasi model – linearno pridruživanje
- Pridruživanje s potencijom stupnja – nelinearno pridruživanje
 - $\Pi_\alpha(i \leftrightarrow j) = \frac{k_j^\alpha}{\sum_l k_l^\alpha}$
 - Za $\alpha = 1$ –Albert-Barabasi (AB) model. Za $\alpha \neq 1$, imamo dva slučaja:
 1. $\alpha < 1$, vjerojatnost veze raste sporije nego kod AB modela -> manja razlika u veličini čvorova, nema otežanog repa i nestanak hubova
 2. $\alpha > 1$, čvorovi visokog stupnja puno brže akumuliraju nove veze nego oni niskoga -> jedan od čvorova će biti povezan s velikim udjelom ostalih. Kada $\alpha > 2$, **pobjednik uzima sve** – jedan čvor povezan sa svima ostalima koji imaju sličan mali stupanj
 - Nelinearno pridruživanje ne stvara hubove koje vidimo u stvarnim mrežama

Ograničenja AB modela

- Mora biti striktno linearno pridruživanje
- Nagib krivulje kumulativne distribucije neovisan o izboru parametara modela – stvarne distribucije mogu opadati brže ili sporije
- Hubovi su najstariji čvorovi, novi ne mogu postići veći stupanj
- Ne proizvodi trokute -> koeficijent klasteriranja manji nego u stvarnim mrežama
- Čvorovi i veze mogu se samo dodati, u stvarnim mrežama mogu nestati
- Jedna povezana komponenta. Stvarna mreža može imati više komponenti

Model privlačnosti

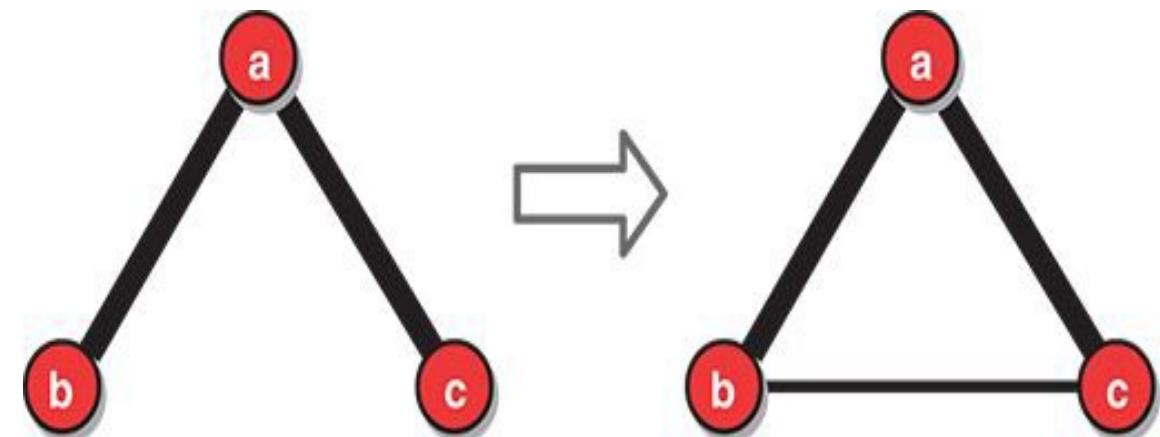
- Preferencijalno pridruživanje – što ako čvor nema susjeda ? -> vjerojatnost dodjele veze je nula!
- Usmjerene mreže, vjerojatnost veze ovisi samo o ulaznom stupnju -> problem su novododani čvorovi koji svi imaju ulazni stupanj nula
- Rješenje
 - Intristična privlačnost
 - Vjerojatnost veze proporcionalna zbroju stupnja i konstantne privlačnosti
 - $\Pi(i \leftrightarrow j) = \frac{A+k_j}{\sum_l(A+k_l)}$
 - Za $A=0$ -> AB model

Model sposobnosti

- Primjer
 - Google (1998)
 - Prije njega milijuni stranica
 - Usprkos tome najpopularniji
 - Slično s novim znanstvenima radovima (citati)
- Individualno svojstvo svakog čvora (sposobnost)
- Svakom čvoru i dodijelimo sposobnost $\eta_i > 0$ iz distribucije $\rho(\eta)$
- $\Pi(i \leftrightarrow j) = \frac{\eta_j k_j}{\sum_l \eta_j k_l}$, ako su sve sposobnosti iste dolazimo do AB
 - $\rho(\eta)$ po volji velika vrijednosti -> pobjednik uzima sve efekt
 - $\rho(\eta)$ ograničen -> distribucija ima otežani rep
- Pojava više hubova
- Velika sposobnost omogućava novim čvorovima takmičenje bez obzira na starost i status

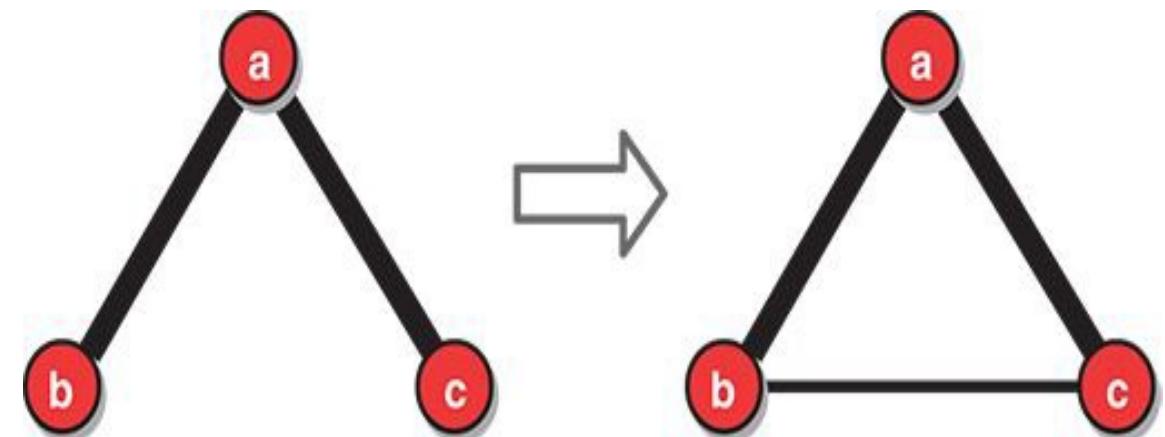
Model slučajne šetnje

- Mreže dobivene AB modelom imaju niski koeficijent klasteriranja
- U AB modelu vjerojatnost da čvor dobije vezu je proporcionalna stupnju bez obzira je li novi par susjeda ima zajedničkog susjeda
- Trokuti se rijetko formiraju
- Potrebno dodati mehanizam koji favorizira kreiranje veza između čvorova sa zajedničkim susjedima



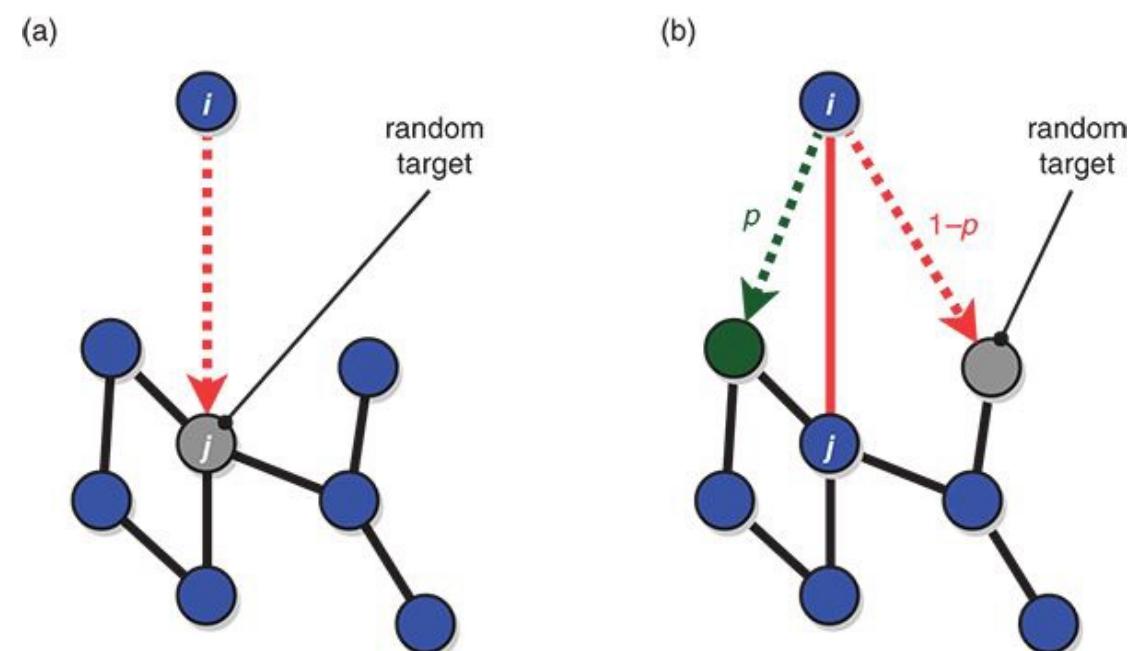
Model slučajne šetnje

- Formiranje trokuta dodavanjem veze se naziva zatvaranje triade
- U praksi mnogo ljudi su upoznati preko zajedničkog poznanika



Model slučajne šetnje

- $m > 1$ novih veza
- Novi čvor i se dodaje na slučajno odabran čvor j
- Svaka dodatna veza na i se dodaje na susjeda od j s vjerojatnošću p što vodi kreiranju trokuta
- Inače je povezan sa slučajno odabranim čvorom



Princip jakog zatvaranja triada

- Osoba a ima jaku vezu s b i $c \rightarrow b$ i c su prijatelji ili će to postati
- Ako b i c provode puno vremena s a , vrlo vjerojatno da će sresti preko a
- S obzirom da je a dobar prijatelj s oba, b i c teže da vjeruju jedan drugome
- Ako b i c ignoriraju jedan drugog to može biti izvor stresa za grupu
- Princip jakog zatvaranja triada propisuje da mora biti veza između b i c
- *"The strength of weak ties"* Mark S. Granovetter (1973). Bliska veza između trokuta, težina veza i zajednica

Društvene zajednice

- Veze s velikom težinom signaliziraju čvrste veze - unutar zajednice
- Slabe veze - između zajednica
- Slabe veze su kritične za strukturu socijalnih mreža, zbog povezivanja zajednica i omogućuju prijenos informacija kroz mrežu

Kompleksne mreže

6. predavanje

<https://www.youtube.com/shorts/nmHzvQr3kYE>

<https://www.youtube.com/shorts/kWePYEdVbhc?feature=share>

Fake news spreads faster than true news on Twitter—thanks to people, not bots

Tweets containing falsehoods were 70% more likely to be retweeted than truthful tweets

8 MAR 2018 • BY KATIE LANGIN



Tweets containing false news (depicted in orange in this data visualization) spread to more people through Twitter than tweets containing true news (teal). CREDITS: (IMAGE) PETER BESHAI; (DATA) SOROUSH VOSOUGH, DEB ROY, AND SINAN ARAL

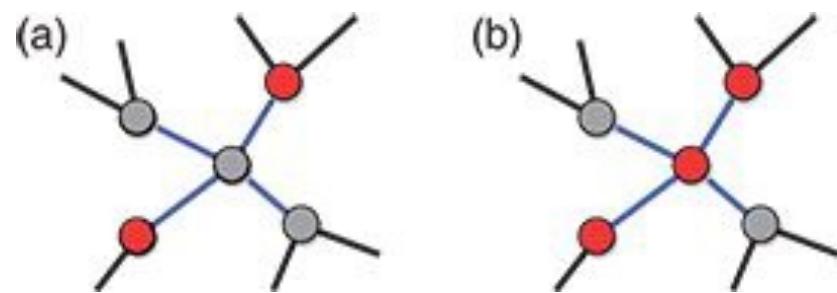
Ideja, informacija, utjecaj

- Središnja uloga mreža u širenju ideja i informacija u socijalnim zajednicama
- Izloženost novim stvarima preko prijatelja – npr. novi model mobitela, odjeća, novosti
- Socijalni utjecaj
 - Prilagođenje ponašanja
 - Donošenje odluka
 - Prihvatanje inovacija
 - Izoštrevanje naših kulturnih, političkih ili vjerskih pogleda

Ideja, informacija, utjecaj

- Modeliranje kako utjecaj, ideja i informacija se šire u socijalnim mrežama – ključna primjena mrežne znanosti
- Proces širenje se naziva i **socijalna zaraznost** – sličnost širenju zaraze kontaktima

Difuzija socijalnog utjecaja u mreži



- Određen broj čvorova (influencera) inicialno aktiviran - ovisno o pravilima
- Aktivacija neaktivnih ovisno o prisustvu aktivnih susjeda ili drugih okolnosti ili parametara
- Slika – aktivacija središnjeg čvora
- Rezultat procesa – kreiranje kaskade utjecaja (aktivacija u slijedu podskupa čvorova)
- Kaskada – od par čvorova do **globalne kaskade**

Modeli praga

- Čvor je aktiviran jedino ako njegov utjecaj preko njegovih aktivnih susjeda prijeđe vrijednost praga
- Linearan model – suma težina veza koje povezuju s aktivnim susjedima
- Prijeđen prag -> čvor postaje aktivan (usvaja ideju, informaciju ili ponašanje)
- $I(i) = \sum_{j:aktivan} w_{ji}$
- Aktivacija $I(i) \geq \theta_i$ - prag
- Netežinski graf $n_i^{on} \geq \theta_i$ - svodi se na broj aktivnih susjeda

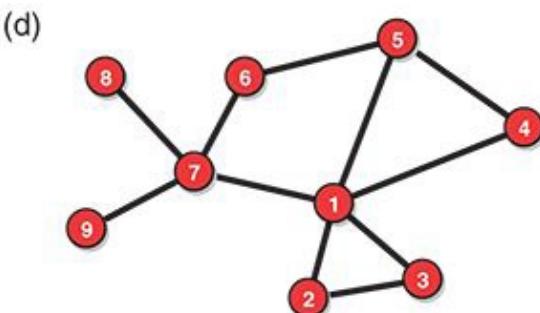
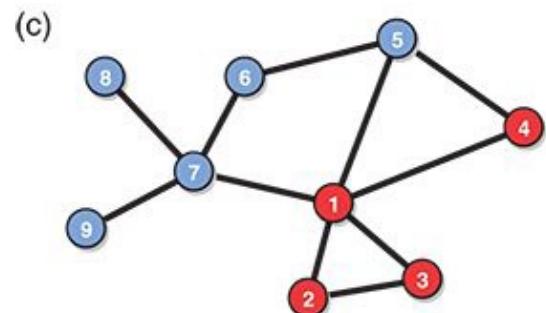
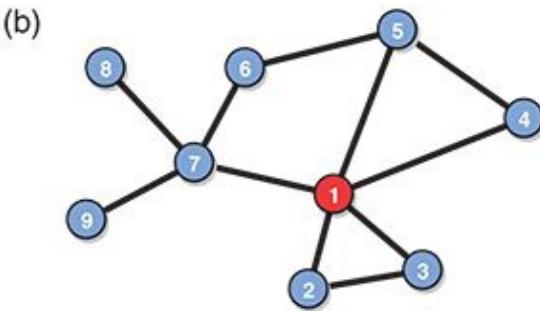
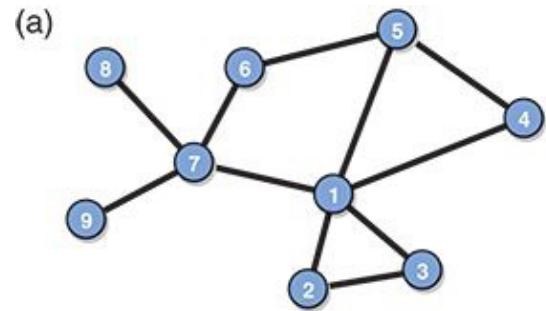
Model praga

- Odabratи mrežu (pretpostavimo da je netežinska)
- Pridružiti prag svim čvorovima
- Zadani broj čvorova aktivan – odaberemo slučajno
- Iterativni koraci
 - Svi aktivni čvorovi ostaju aktivni
 - Svaki neaktivni čvor je aktiviran ako broj aktivnih čvorova je iznad praga
 - Ponavljamo sve dok ne postoje čvorovi koje možemo aktivirati
- Poredak u kojem promatramo čvorove ne smije utjecati na realizaciju u modelima mrežne dinamike

Poredak u kojem promatramo čvorove

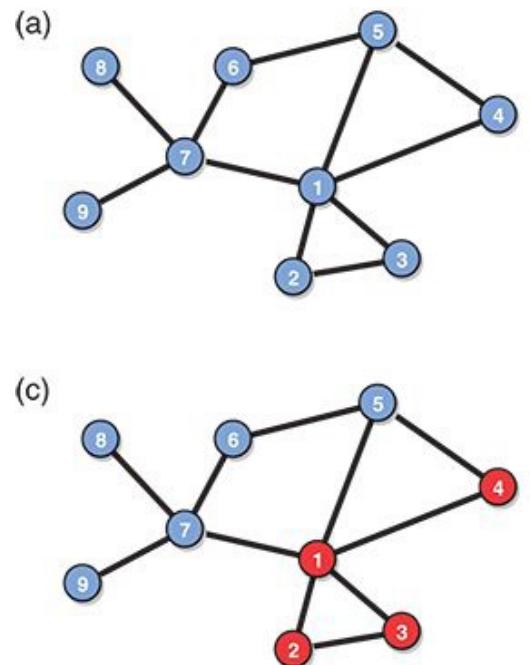
- Asinkrona implementacija
 - Čvorovi su evaluirani u različitoj slučajnoj sekvenci u svakoj iteraciji
 - Izbjegavamo pristranosti koje mogu nastati ako svaki put biramo isti slijed
- Sinkrona implementacija
 - Novo aktivirano stanje svakog čvora u svakoj iteraciji je određeno koristeći aktivacijske vrijednosti drugih čvorova u prethodnoj iteraciji
 - U ovom slučaju poredak je nebitan

Varijante modela praga



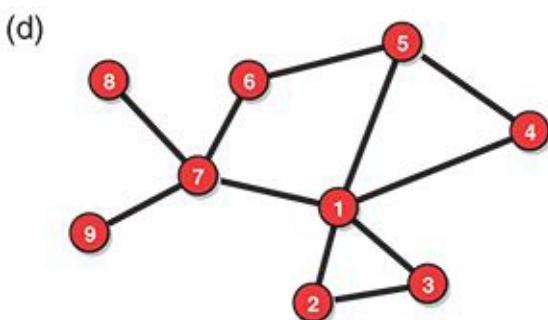
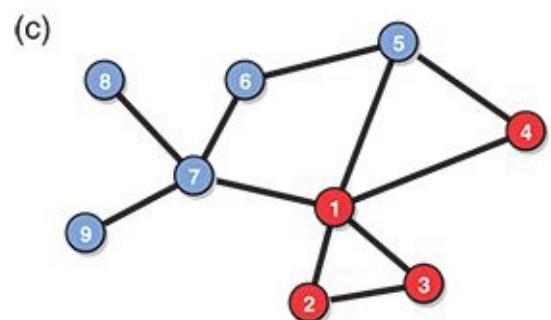
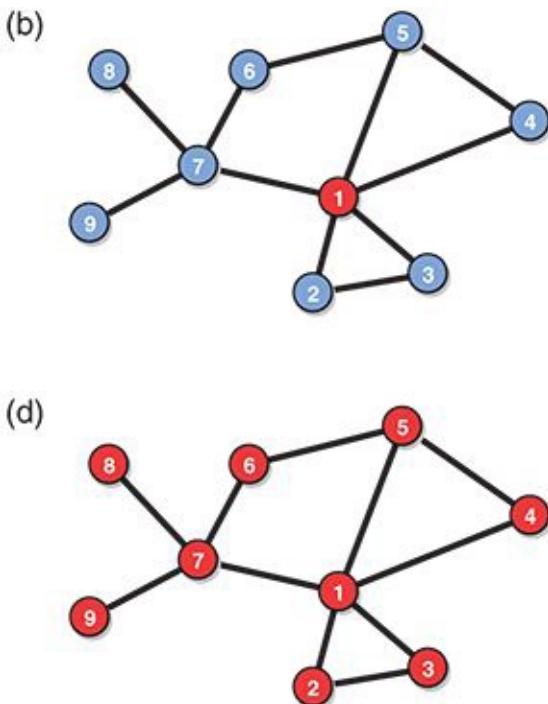
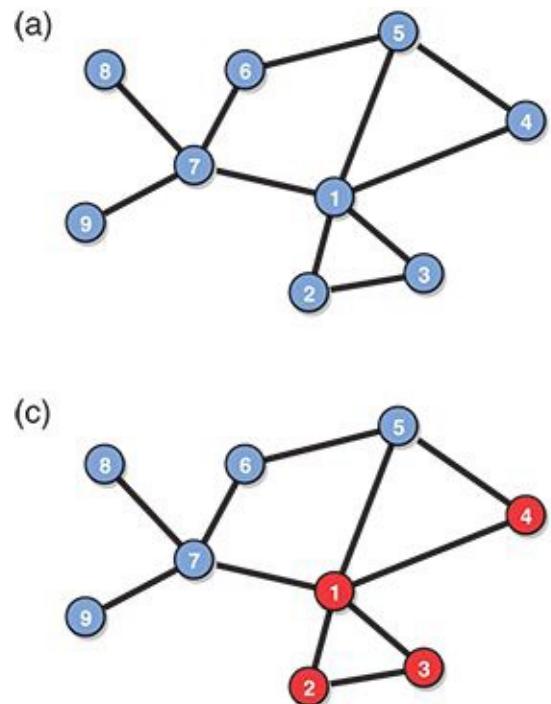
- Model udjela praga – udio aktivnih susjeda umjesto broja
- Primjer 1/2 – barem pola čvorova aktivno
- $\frac{n_i^{on}}{k_i} \geq \theta_i$

Kaskade



- Ako je mreža rijetka pojava globalne kaskada ovisi o strukturi
- **Ranjivi čvorovi**
 - mogu biti aktivirani jednim aktivnim susjedom
 - Ranjiv čvor ako $k_i \leq 1/\theta_i$
- Globalna kaskada – broj ranjivih čvorova velik

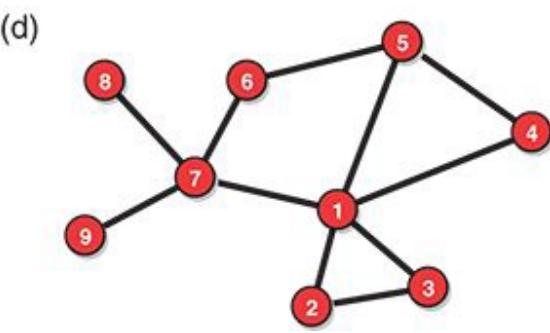
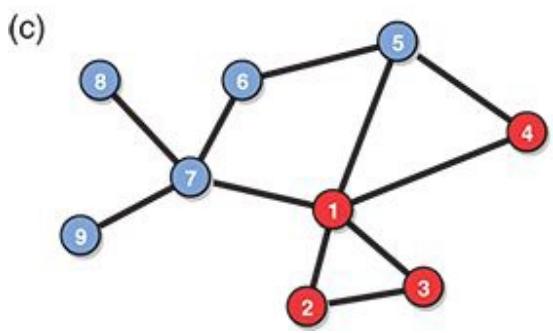
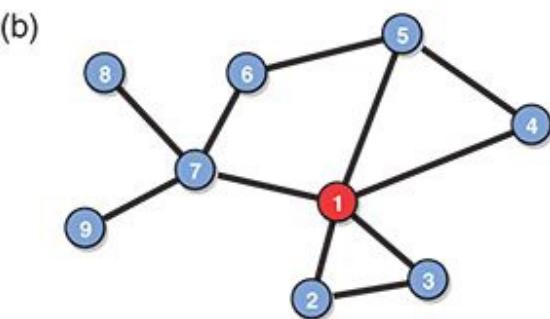
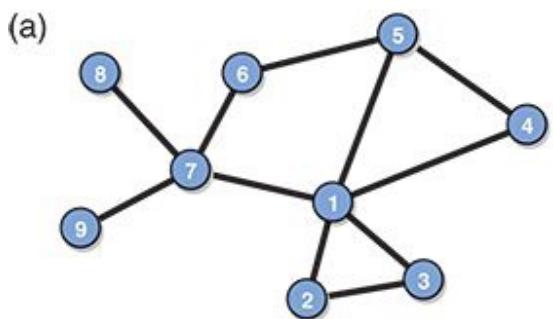
Kaskade



- Hubovi

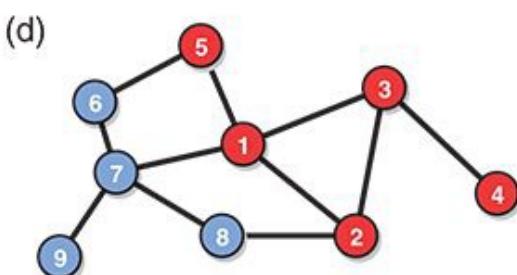
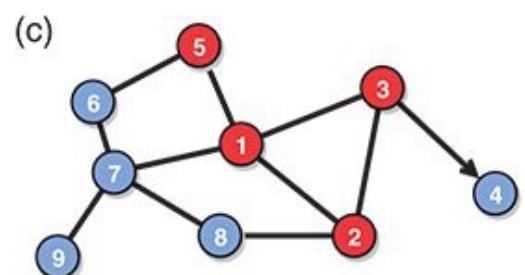
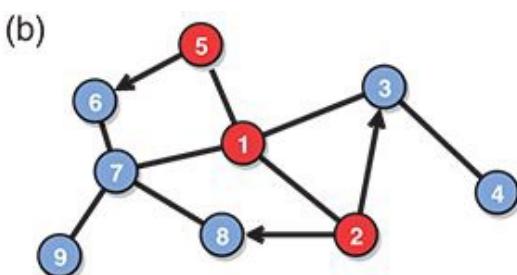
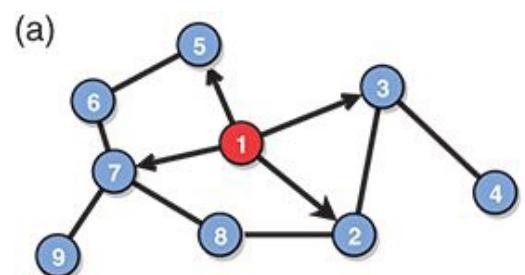
- Obično efektivni influenci
- Velik broj susjeda – veća vjerojatnost da neki ima dovoljno mali stupanj da postane ranjiv
- Ponekad nije dovoljno biti hub
- Pozicija u mreži bitna – kaskada na periferiji teško može utjecati na jezgru

Kaskade



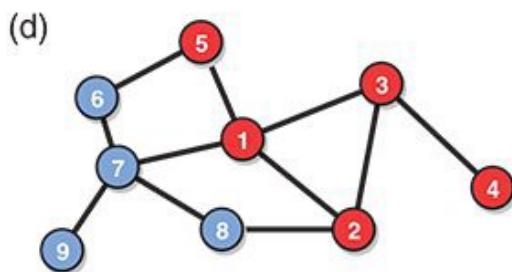
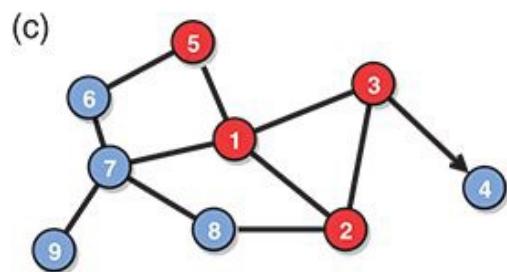
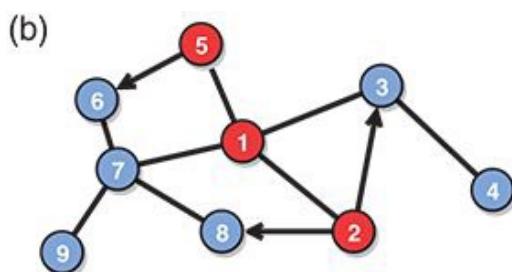
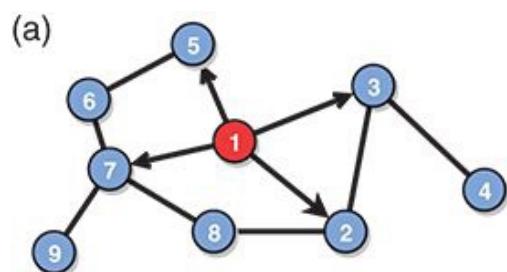
- Gustoća i odvojenost zajednica
 - Širenje potpomognuto u gustim zajednicama, ometano među zajednicama
- Poznavanje strukture - bitno
- Primjer čvor 7 je influencer

Nezavisni kaskadni modeli



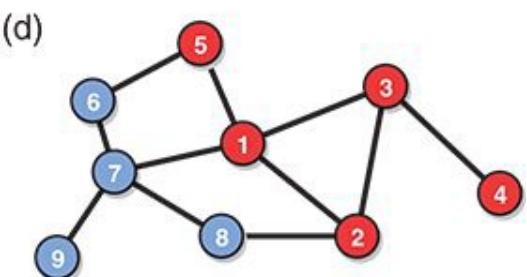
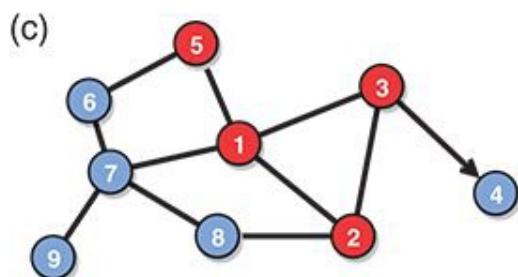
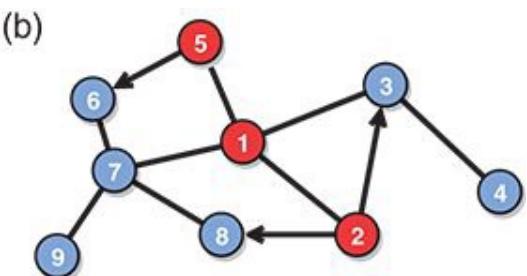
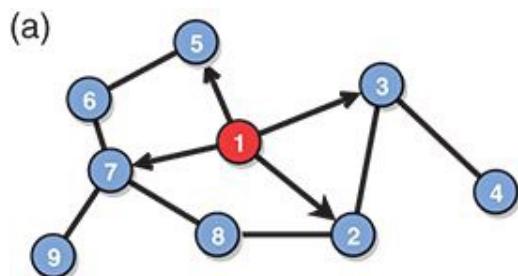
- Modeli praga
 - pritisak istovrsnih (*engl. peer pressure*)
 - više kontakata dijele ideju ili proizvod
-> veća vjerojatnost da ćemo usvojiti
 - Kao da naši aktivni susjedi zajedno utječu na nas da prihvativimo
- Nezavisni kaskadni modeli
 - Svaki od čvorova ima svoj utjecaj

Nezavisni kaskadni modeli



- Postavke iste kao i kod modela praga
- Aktivni čvor ima šansu “nagovoriti” svakog od neaktivnih susjeda
- Svaki susjed se aktivira s nekom vjerojatnošću utjecaja
- Ako čvor ne uspije aktivirati susjeda, ne može ponoviti – može biti nagovoren od ostalih aktivnih
- Primjer – vjerojatnost utjecaja 1/2

Nezavisni kaskadni modeli



- Aktivni čvor i ima vjerojatnost p_{ij} da nagovori neaktivnog susjeda j
- Nema utjecaja ostalih čvorova
- Asinkrona implementacija
 - j ima nekoliko aktivnih susjeda
 - Pokušaji aktivacije u proizvoljnem redu za izbjegavanje pristranosti
- p_{ij} i p_{ji} se mogu razlikovati (Koliko smo uspješni u nagovaranju i koliko nas lako nagovoriti)

Usporedba modela praga i nezavisnih kaskada

- Veći broj aktivnih susjeda – veća vjerojatnost aktivacije
- Razlike između modela
 - Modeli praga – naglasak na ciljanom čvoru, deterministički (broj ili udio susjeda, uvijek ista realizacija)
 - Modeli nezavisnih kaskada - naglasak na influenceru, vjerojatnosni (dinamika ovisi o "sreći") – teško predvidjeti progres kaskade

Usporedba modela praga i nezavisnih kaskada

- Sofisticiranije verzije modela
 - Vjerojatnosna verzija modela praga (šansa za aktivaciju raste s brojem čvorova)
 - Slično kao kod nezavisnih kaskada, no aktivni susjedi nisu nezavisi
 - Kompleksni procesi širenja – svaka nova osoba koja nas izlaže produktu ili ideji ima veći utjecaj nego prethodni da uspije

Question If you were asked to vote today on the question of the United States entering the war against Germany and Italy, how would you vote—to go into the war, or to stay out of war?

1941

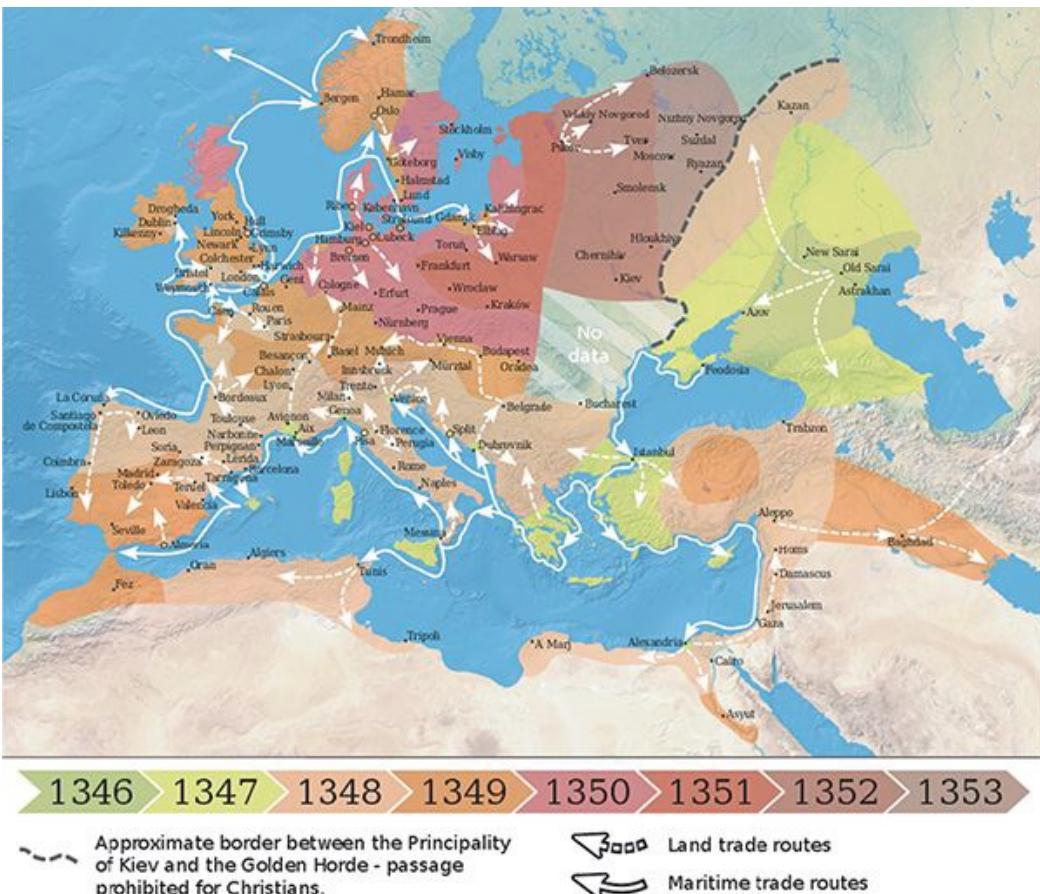
Would vote to go to war



19%

Source: Gallup Poll, March 1941. Margin of error ± 3%.

Širenje epidemija

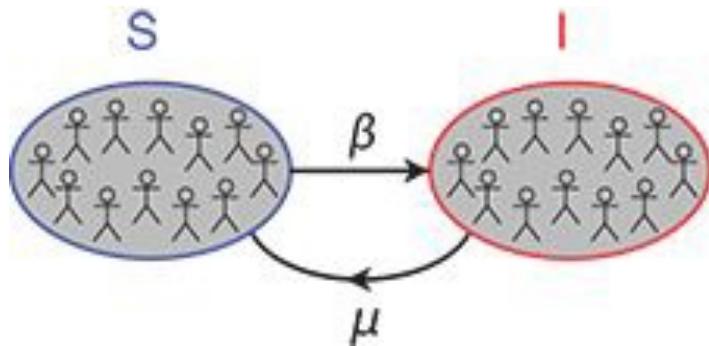


- Kuga (crna smrt)
- Prijenosnici – buhe koje žive na crnim štakorima koji su putovali na trgovackim brodovima
- Vjerojatno krenula u središnjoj Aziji
- Europa 1346 – 1353
- Umrlo 30-60 % europske populacije
- Danas
 - Efikasnije metode za izbjegavanje uništavajućih efekata zaraznih bolesti
 - Zbog povezanosti svijeta brzina širenja - povećana

Novi oblici epidemija

- Računalni virusi
- Virusi na mobilnim telefonima (širenje npr. Bluetooth)
- Širenje glasina, hoaxa, lažnih novosti, zavjera, pseudoznanosti
- Procesi širenja informacija – slični epidemijama zaraznih bolesti

Osnovni modeli širenja – SIS model

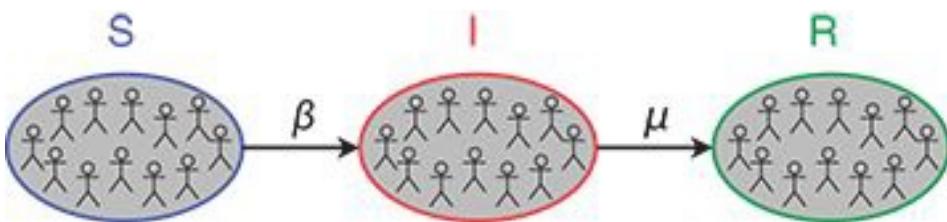


- Podjela populacije u odjeljke
 - Podložni - *Susceptible* (S)
 - Zaraženi - *Infected* (I)
- SIS model
 - Ne postoji dugotrajna imunost (npr. Prehlada, Covid, ...)

SIS model

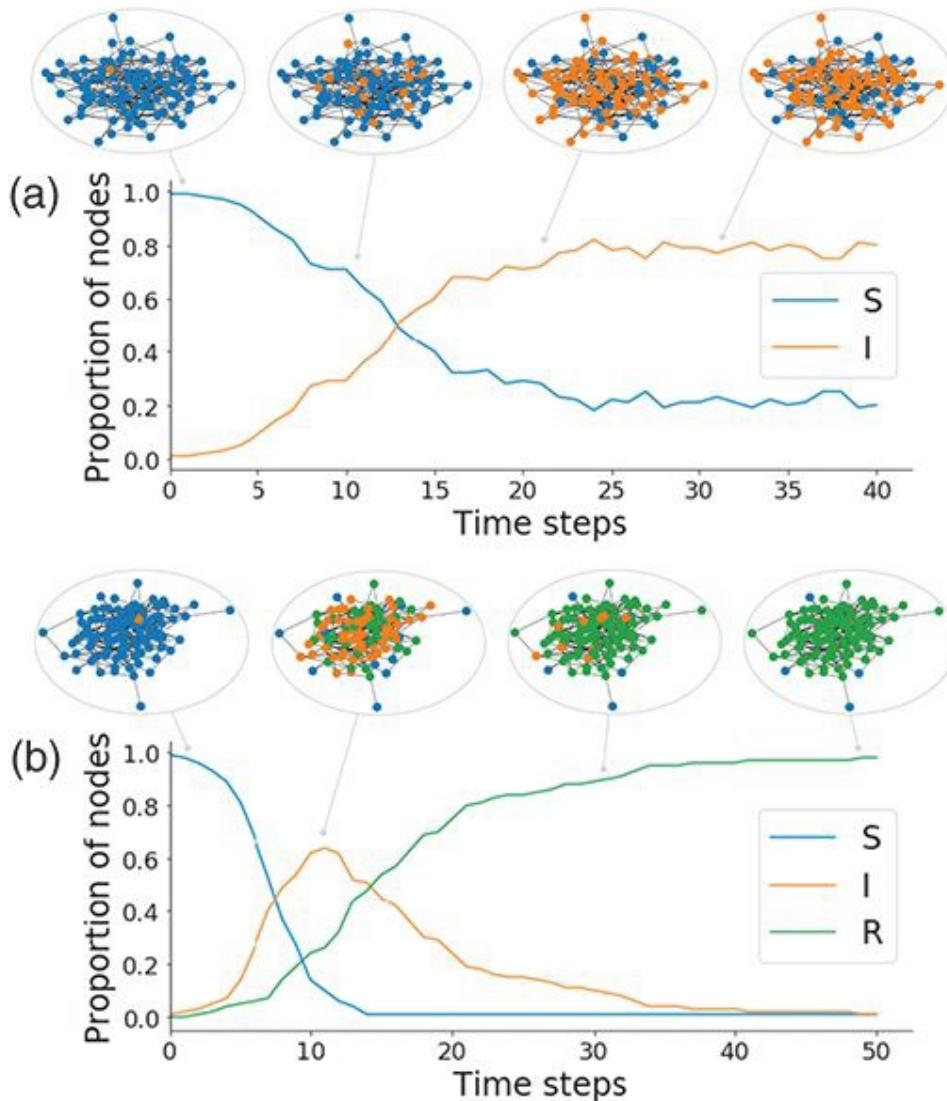
- Mreža kontakata od empirijskih podatka ili umjetna mreža, neki od čvorova zaraženih u skladu s kriterijem (npr. Slučajno)
- Svi ostali su podložni
- Iteracija, posjetimo svaki čvor. Za svaki čvor i
 - Ako je i podložan, prođemo sve njegove susjede. Za svakog zaraženog susjeda i postaje zaražen s vjerojatnošću β
 - Ako je i zaražen, postaje podložan s vjerojatnošću μ
- Čvorove obilazimo asinkrono u slučajnom poretku ili sinkrono
- Parametri β i μ ključni za model

SIR model



- Oporavljen – *Recovered* (R)
- Bolesti s dugotrajnim imunitetom (ospice, kozice, rubeola, mums)
- Umro – specijalan slučaj oporavljenog
- Širenje prestaje kada nema više zaraženih pojedinaca

Evolucija SIS i SIR modela



- Na početku par ljudi zaraženo
- Nakon toga eksponencijalni porast
- Stacionarno stanje
 - Endemična – djeluje na stabilan udjel populacije
 - Iskorijenjena

Klasični epidemiološki modeli

- Homogeno miješanje
 - pretpostavka da svaki pojedinac može biti u kontaktu s bilo kojim drugim.
 - Svi pojedinci u istom odjeljku imaju identično ponašanje
 - Pretpostavka punog grafa – svi povezani sa svima
 - Ima smisla za male populacije – npr. manje selo, razred
- U svakoj iteraciji novozaraženi (sekundarna zaraza) i oporavljeni
- Da bi se epidemija širila mora biti više novozaraženih od oporavljenih
- Kod homogenih mreža ovo vodi u efekt praga

Osnovni reproduksijski broj

- prosječan broj novozaraženih generiran od strane zaraženog pojedinca
 - Ovisi o stopi zaravnosti, stopi oporavka, prosječnom stupnju
 - Ako je veći od praga – epidemija može zahvatiti značajan udio populacije
 - Inače – zamire brzo bez značajnih utjecaja
- $\langle k \rangle$ - prosječan stupanj (aproksimacija)
- β – vjerojatnost zaraze susjeda
- μ – vjerojatnost oporavka
- $\beta\langle k \rangle$ - prosječan broj zaraženih od strane jedne osobe u jednoj iteraciji

Osnovni reproduksijski broj

- $I_{sec} = \beta\langle k \rangle I$ (sekundarna infekcija)
- $I_{rec} = \mu I$ (novooporavljeni)
- Za širenje epidemije $\beta\langle k \rangle I > \mu I \Rightarrow R_0 = \frac{\beta}{\mu}\langle k \rangle > 1$
- R_0 - osnovni reproduksijski broj
- $R_0 < 1$ – epidemija umire u kratkom vremenu
- $R_0 > 1$ – epidemija nastavlja širenje
- Razni faktori imaju dodatan utjecaj i spriječiti epidemiju– karantena, mrežna struktura zajednica
- Ospice $R_0 = 10$, ebola $R_0 = 2$

Realna mreža kontakata

- Nije homogena
- Prisustvo hubova značajno mijenja scenarij
- Ukoliko postoje čvorovi s vrlo visokim stupnjem -> **efektivno kao da nema praga**
- Čak i bolesti s niskom stopom zaravnosti ili/i visokom stopom oporavka mogu se proširiti na značajan udio populacije
- Čak i ako je vjerojatnost zaraze niska, jednostavno je zaraziti jedan ili više hubova koji su izloženi zbog velikog broja kontakata
- Jednom zaraženi hubovi su opasni širitelji

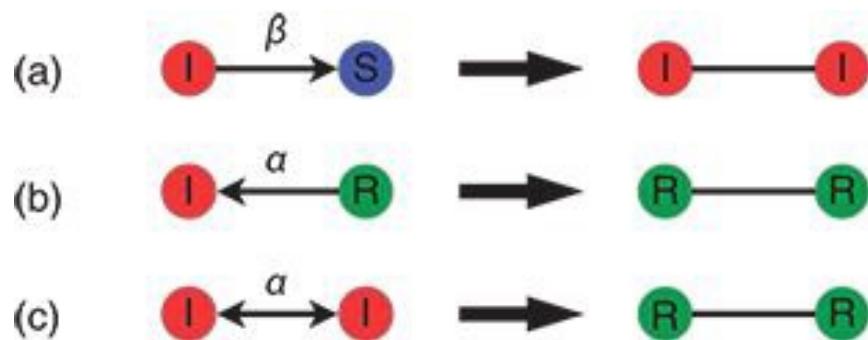
Uloga u hubova u epidemiji

- Izolirati i cijepiti populaciju s puno kontakata
- Seksualni radnici za spolno prenosive bolesti
- Često nije lako identificirati hubove
- Uzmemo slučajnu uzorak populacije i cijepimo njihove susjede (paradoks prijateljstva)
- Cijepljenje roditelja male djece

Model širenje glasina

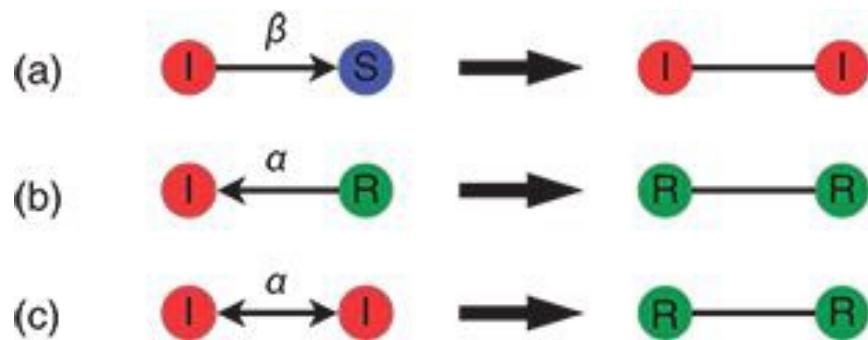
- Neupućen – Ignorant (S)
- Širitelj – Spreader (I)
- Zadržatelj – Stiffler (R)
- Ljudi su angažirani u širenje ideja sve dok postoje ljudi koji su ih nesvjesni

Model širenja glasina



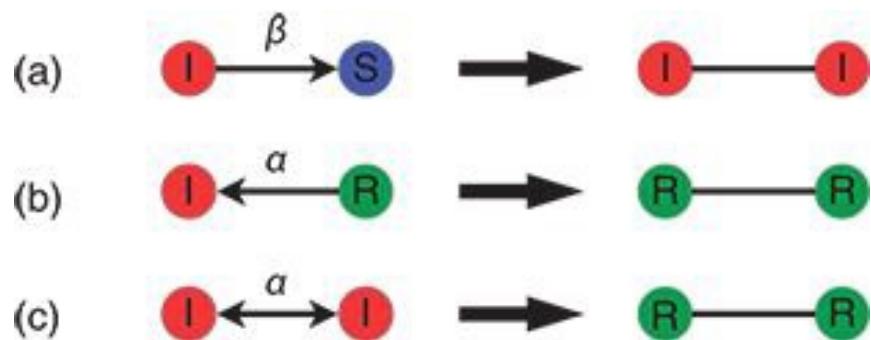
- Na početku su svi neupućeni osim malo broja odabralih za širenje glasina
- Kada širitelj sretne neupućenog, glasina se širi s vjerojatnosti prijenosa β
- Kada širitelj glasina sretne zadržatelja ili širitelja postaje zadržatelj s vjerojatnosti zaustavljanja širenja α
- Ako neupućen sretne zadržatelja, ništa se ne dogodi

Model širenja glasina



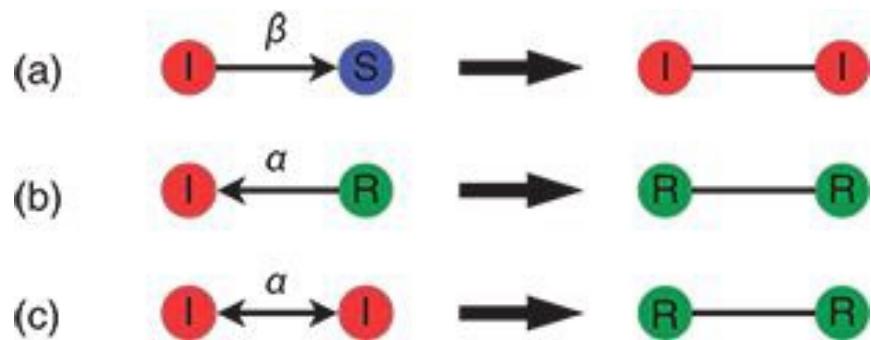
- U svakoj iteraciji, svi čvorovi se obilaze sinkrono ili asinkrono u slučajnom poretku. Za svaki čvor i :
 1. Ako je i neupućen, obići sve susjede. Za svakog susjeda širitelja, i postaje širitelj s vjerojatnošću β
 2. Ako je i širitelj, obići susjede:
 - i. Za svakog susjeda zadržatelja, i postaje zadržatelj s vjerojatnošću α
 - ii. Za svakog susjeda širitelja, susjed postaje zadržatelj s vjerojatnošću α

Model širenja glasina



- Za razliku od SIR-a prelazak iz I u R nije spontan, nego ovisi o interakciji među pojedincima
- Slično SIR modelu, krećemo s nekoliko širitelja, a na kraju će svi pojedinci biti neupućeni ili zadržatelji
- Broj zadržatelja je jednak ukupnom broju svih koji su saznali za glasine

Model širenja glasina



- Model nema praga ni za homogene mreže
- Glasine mogu doseći velik broj ljudi iako je vjerojatnost prijenosa mala
- Konačni broj ljudi svjestan glasine je manji nego u homogenim mreža s jednakim brojem čvorova i veza
- Rano dosezanje hubova
- Kada hubovi postanu zadržatelji, difuzioni proces usporava

[Life](#)

Algorithm zeroes in on origins of disease outbreaks

24 June 2015



(Image: Da-kuk/Getty)

WE'RE zeroing in. In a [disease outbreak](#), "patient zero" is the first person to be infected, and finding them can help stop the outbreak. But incomplete data mean that person is usually hard to track down. Now an algorithm could help with that hunt.

Dinamika mišljenja

- Imamo mišljenja o svemu i svačemu
- Mišljenje vode naše ponašanje, utječu na naše odabire, utječu na naše planove
- Politike implementirane od strane vlade su diktirana mišljenjima o trgovini, konfliktima, imigraciji, pandemiji, okolišu
- Dinamika mišljenja – kako se mišljenja formiraju i šire u društvu
- Društvene mreže – moćni alati za kruženje i čak manipuliranje mišljenjem
- Modeli dinamike mišljenja – slični modelima širenja utjecaja, ali sa specifičnim svojstvima
- Modeli: diskretni i kontinuirani

Diskretna mišljenja

- Često imamo limitirani broj opcija (Android/iPhone, kupi/prodaj,...)
- Mišljenje predstavljeno s cjelobrojnim atributom – stanjem čvora
- Za početak pretpostavimo binarna mišljenja
- Model je karakteriziran skupom pravila koja određuju kako se mišljenje čvora mijenja zbog mišljenja susjeda

Dinamika diskretnog mišljenja

1. Inicialno, mišljenja su slučajno pridijeljena čvorovima u mreži.
Inicialno isti broj ljudi ima jedno od mišljenja (neslaganje)
2. Mišljenja čvorova se osvježavaju. Svaka iteracija sastoji se od prolaska svih čvorova. Tipično, čvorovi se osvježavaju asinkrono u slučajnom poretku radi olakšanja konvergencije

Dinamika diskretnog mišljenja

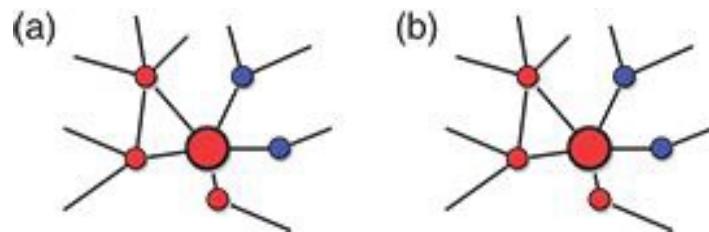
3. Postoje dvije moguće realizacije:

- i. Sustav dostiže stabilno stanje, gdje se mišljenja više ne mijenjaju. Dolazimo do konsenzusa (svi se slažu) ili polarizacije
- ii. Sustav ne dosiže stacionarno stanje, na način da neki čvorovi (ili svi) mijenjaju stanje u svakoj iteraciji. Ipak, neka svojstva, npr. srednja vrijednost svih varijabli se stabilizira na dulji period

Standardne varijable za praćenje modela

- Srednje mišljenje
 - Aritmetička sredina
 - Stacionarno stanje - konvergencija prema preciznoj vrijednosti
 - Konsenzus – jedan ili nula
- Izlazna vjerojatnost
 - Koliko je izgledno da mreža postigne konsenzus oko jednog mišljenje kao funkcija udjela čvorova s tim mišljenjem
 - Npr. da sve realizacije vode u konsenzus, a 30 % njih vodi u željeno mišljenje -> izlazna vjerojatnost je 0.3

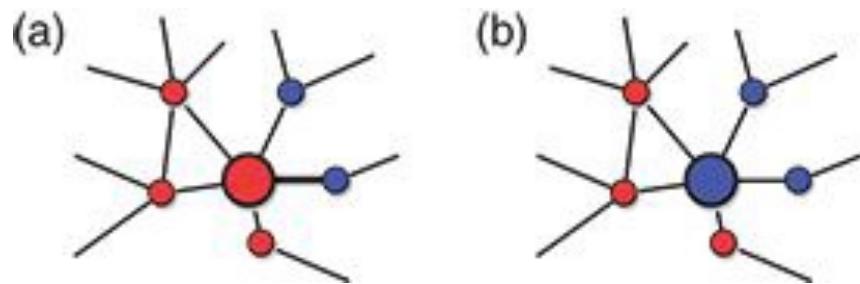
Model većine



- Model većine:

- Svaki čvor preuzima mišljenje većine svojih susjeda
- Ako je broj paran i jednak broj susjeda dijeli različita mišljenja, slučajno odabiremo jedno
- Stabilna stanja:
 - Konsenzus
 - Ako čvor ima mišljenje većine svojih susjeda to se mišljenje neće mijenjati
 - U većini realnih mreža nikada ne dolazi do konsenzusa

Model glasanja



- Svaki čvor preuzima mišljenje slučajno odabranog susjeda
- Konsenzus je jedino stabilno stanje ovoga modela
- Realizacija dinamike nije izvjesna
- Primjer:
 - 30 % čvorova ima opciju jedan inicijalne konfiguracije
 - Očekujemo da će 30 % od svih realizacija završiti u stanju jedan
 - Ne možemo reći hoće li pojedina realizacija završiti u konsenzusu jedan ili nula

Varijacije modela glasanja

- Prisustvo čvorova koji nikada ne mijenjaju mišljenje. Ako svi imaju isto mišljenje, favorizirat će to mišljenje, inače konsenzus neće biti postignut
- Razmatranje više od dva mišljenja. Interakcija može biti ograničena samo na čvorove koji imaju dovoljno bliska mišljenja.
- Mogućnost da čvorovi mijenjaju mišljenje spontano, npr. s izvjesnom vjerojatnošću u svakoj interakciji, dodatno na dinamiku glasanja

Kompleksne mreže

7. predavanje

Učenje u mrežama (Bayesian & DeGroot model, mudrost masa)

Dijeljenje informacija i stvaranje mišljenja

- Otvorena pitanja:
 - da li pojedinci u društvu imaju zajedničko uvjerenje ili ostaju podijeljeni u mišljenjima?
 - koji pojedinci imaju najveći utjecaj na uvjerenja u društvu?
 - koliko brzo pojedinci uče?
 - mogu li se u početku raznolike informacije razasute po društvu agregirati na točan način?

Koji pojedinci imaju najveći utjecaj na
uvjerenja u društvu u današnje vrijeme?





cristiano Follow Message +9 ...

3,593 posts 614M followers 577 following

Cristiano Ronaldo

Join my NFT journey on @Binance. Click the link below to get started.
ter.li/CR7ForeverZone

Followed by [mhdsdj](#), [foxsports](#), [roccolibroccoli](#) + 21 more



selenagomez Follow Message +9 ...

1,946 posts 429M followers 277 following

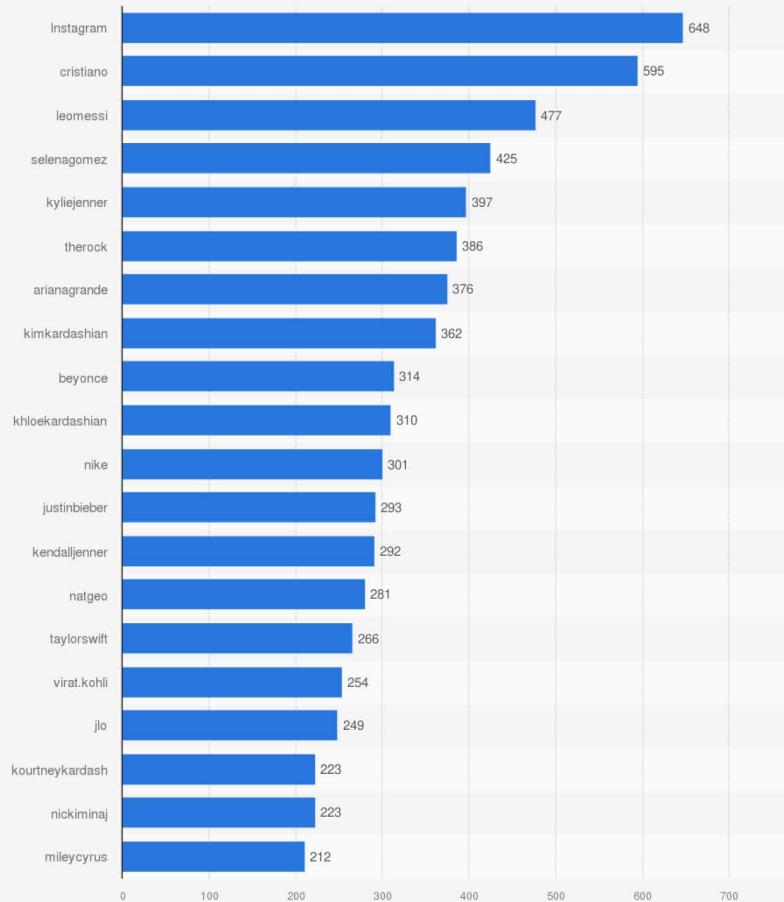
Selena Gomez

@[selenagomez](#)

Musician/Band
By grace, through faith.
Founder @rarebeauty
Founder/CIO @officialwondermind
linktree.ee/selenagomez

Followed by [kateupton](#), [rozgajelenofficial](#), [iamdaniela88](#) + 15 more

Instagram accounts with the most followers worldwide as of July 2023 (in millions)



Source
Social Blade
© Statista 2023

Additional Information:
Social Blade: July 2023; business / creator accounts



therock Follow Message +9 ...

7,493 posts 394M followers 822 following

Dwayne Johnson

founder of stuff
therock.komi.io

Followed by [mhdsdj](#), [foxsports](#), [hvoje3101](#) + 26 more



nike Follow Message +9 ...

1,339 posts 306M followers 164 following

Nike

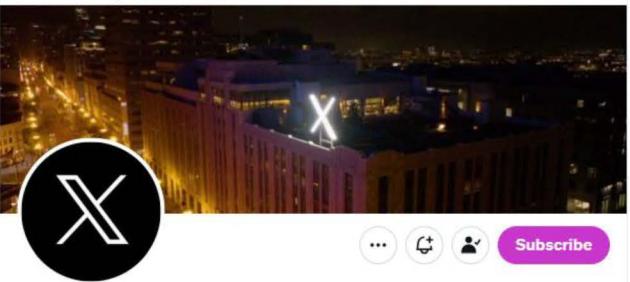
@[nike](#)

Spotlighting athlete* and 📸 stories
linkin.bio/nike + 1

Followed by [nenadjebiva](#), [sportsillustrated](#), [aannkkoo123](#) + 5 more

Elon Musk ✅

34.9K posts



Elon Musk ✅

@elonmusk

Joined June 2009

506 Following 166.2M Followers 136 Subscriptions

Followed by Angler, Vanja Smailovic, and 563 others you follow

Barack Obama ✅

16.9K posts



Barack Obama ✅

@BarackObama

Dad, husband, President, citizen.

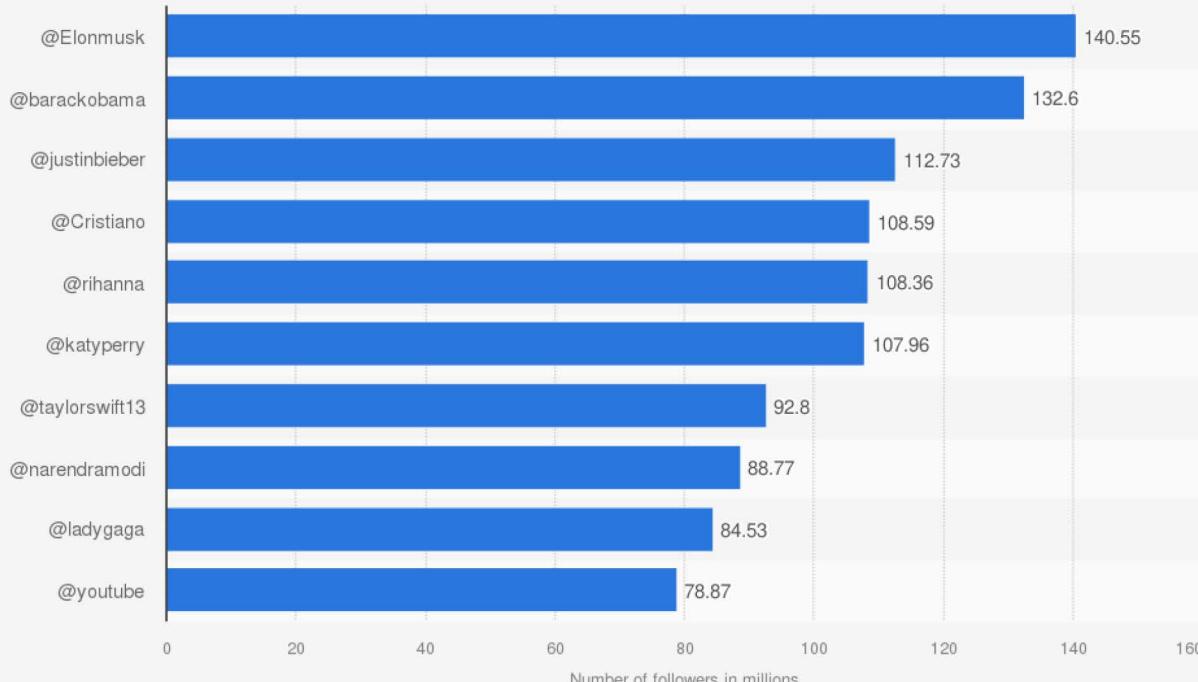
Washington, DC [barackobamabooks.com](#) Born August 4, 1961

Joined March 2007

546.9K Following 131.9M Followers

Followed by Nastaran Naseri, Paul Kennedy, and 580 others you follow

X (formerly Twitter) accounts with the most followers worldwide as of August 2023 (in millions)



Source

Socialtracker
© Statista 2023

Additional Information:

Worldwide; Socialtracker; August 2023

Justin Bieber ✅

31.1K posts



Justin Bieber ✅

@justinbieber

JUSTICE the album out now

Joined March 2009

275.7K Following 111.5M Followers

Followed by THE INTERESTING NETWORK, Getty Images Sport, and 75 others you follow

Cristiano Ronaldo ✅

4,006 posts



Cristiano Ronaldo ✅

@Cristiano

This Privacy Policy addresses the collection and use of personal information - [cristianoronaldo.com/terms](#)

[facebook.com/cristiano](#) Joined June 2010

68 Following 110.4M Followers

Followed by Marc Waters, The Sir Bobby Charlton Foundation, and 46 others you follow

Lideri mišljenja (*opinion leaders*)

- Lideri mišljenja su pojedinci u društvu koji se informiraju kroz razne medije i druge interakcije, te koji zatim oblikuju mišljenja i prenose informacije i utječu na druge pojedince koji su manje izravno informirani [Lazarsfeld, Berelson & Gaudet]



kako su pojedinci donosili odluke o glasovanju Ohiju u predsjedničkoj kampanji u SAD-u 1940. godine?

Što su otkrila rana istraživanja [Katz & Lazarsfeld] ?

- Iako su ponekad su lideri mišljenja imali viši društveni status, bilo je mnogo slučajeva u kojima su lideri bili na istom društvenom statusu kao i oni na koje su utjecali, posebno kad su bile u pitanju razne kućanske odluke
- Lideri mišljenja su se često isticali njihovom društvenošću i veličinom njihovih obitelji (što je u korelaciji s njihovom dobi i iskustvom)

Što još utječe na stvaranje mišljenja?

- Obitelj
- Obrazovanje
- Religija
- Različite organizacije



Na koji način vi danas stvarate vaše mišljenje?



Kako kompleksne mreže mogu objasniti proces stvaranja mišljenja?

- **Bayesian model učenja**
 - ponovljene akcije, međusobno promatranje
 - rationalno učenje
 - pojedinci promatraju akcije i rezultate koje su doživjeli njihovi susjedi i informacije na sofisticiran način
 - pruža uvjete pod kojim pojedinci s vremenom počinju djelovati slično
- **DeGroot model učenja**
 - ponovljenja komunikacija, „naivno“ ažuriranje
 - „kratkovidno“ učenje
 - model se temelji na mnogo naivnijem, ali još uvijek prirodnom obliku ažuriranja, gdje pojedinci razmjenjuju informacije sa svojim susjedima tijekom vremena, te zatim ažuriraju uzimajući težinski prosjek onoga što čuju

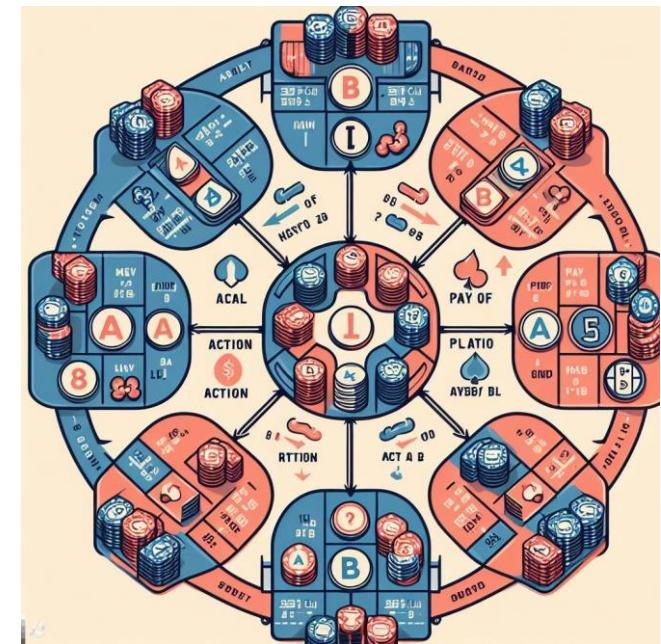
Bayesian model učenja: učenje promatranjem

- Središnji zaključak kod učenja promatranjem je da ako agenti mogu promatrati međusobne radnje i rezultate tijekom vremena, a svi agenti imaju iste preferencije i suočavaju se s istim oblikom neizvjesnosti, onda završavaju sa sličnim nagradama tijekom vremena
- Ideja je da agent koji radi znatno lošije od susjeda to mora shvatiti tijekom vremena, i na kraju će promijeniti postupke i početi raditi kao i susjed
- To onda implicira da svi povezani agenti završavaju s istim ograničenim nagradama
- To ne znači da svi oni nauče raditi najbolje moguće akcije
 - može se dogoditi da svi završe sa sub-optimalnim nagradama
- Međutim, ako uz to, agenti počinju s dovoljnom raznolikošću mišljenja tako da imaju poticaj za eksperimentiranje s različitim radnjama, tada će imati veliku vjerovatnost približavanja pravoj akciji
- Naravno, ova teorija se oslanja na veliku količinu stacionarnosti u okruženju, te sličnosti u preferencijama i situacijama među agentima; ali u isto vrijeme pruža početno mjerilo



Bayesian model učenja [Bala & Goyal]

- n igrača u neusmjerenoj komponenti g
- Odabiranje akcije A ili B svaku rundu
- A nagrađuje 1 (sigurno), dok B nagrađuje 2 s vjerojatnošću p i 0 s vjerojatnošću $1 - p$

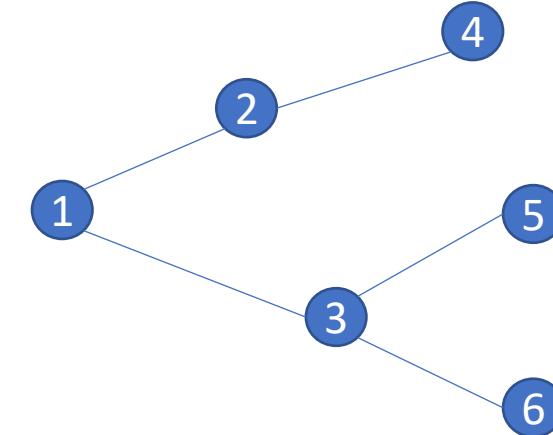


Bayesian model učenja: proces učenja

- Igrač dobiva nagradu svaku rundu na temelju svog odabira
- Igrač također promatra odabire svojih susjeda
- Igrač maksimizira diskontirani niz očekivanih nagrada $E \left[\sum_t \delta^t \pi_{it} \right]$
- p je nepoznat i može poprimiti konačan skup vrijednosti

Izazovi Bayesian model učenja

- Strategije različitih igrača



- Eksperimentiranje

1

1

2

Bayesian model učenja: pretpostavka

- Ako p nije točno $1/2$, tada s vjerojatnosti 1 postoji vrijeme takvo da svi agenti u datoј komponenti igraju samo jednu radnju (i svi igraju istu radnju) od tog vremena nadalje

Bayesian model učenja: dokaz pretpostavke

- Prepostavimo suprotno
- Neki agent u nekoj komponenti igra B beskonačno često
- Taj agent će konvergirati pravom uvjerenju prema zakonu velikih brojeva
- Mora biti da uvjerenje konvergira prema $p > 1/2$, ili bi taj agent prestao igrati B
- Uz vjerojatnost 1, svi agenti koji vide B igrano beskonačno često konvergiraju do uvjerenja da B nadgrađuje 2 s vjerojatnošću $p > 1/2$
- Susjedi agenta moraju igrati B , nakon nekog vremena, i tako dalje
- Svi agenti moraju igrati B od nekog vremena

Bayesian model učenja: igranje ispravne akcije?

- Ako je B ispravna akcija, onda igrač igra ispravnu akciju ako konvergira k tome, ali možda ne konvergira
- Ako je A ispravna akcija, tada mora konvergirati ispravnoj akciji
- Vjerojatnost konvergencije „ispravnoj“ akciji?
 - proizvoljno visoka ako svaka akcija ima nekog agenta koji u početku ima proizvoljno visoko inicialno vjerovanje da je upravo ta akcija najbolja

Bayesian model učenja: ograničenja

- Homogenost akcija i nagrada među igračima
- Što ako imamo heterogenost akcija i nagrada?
- Ponavljane radnje tijekom vremena
 - Što ako imamo samo jedan pokušaj?
- Stacionarnost
- Mreže ovdje ne igraju bitnu ulogu

DeGroot model: imitacija i društveni utjecaj

- Ponovljena komunikacija
- Informacije dolaze samo jednom
- Kako se informacije šire?
- Tko ima utjecaj, brzina konvergencije, utjecaj strukture mreže



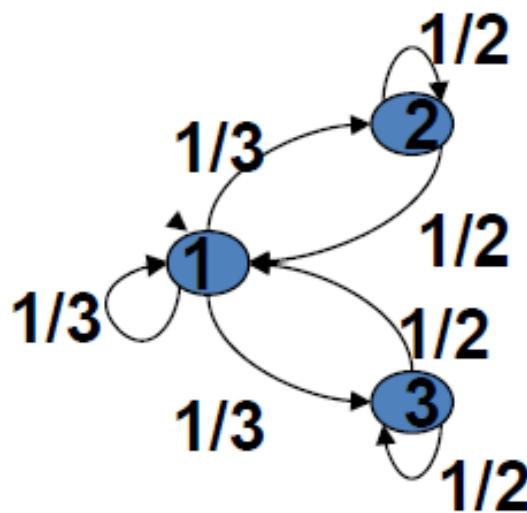
DeGroot model: model ograničene racionalnosti

- Opetovano uprosječenje uvjerenja o sebi kroz komunikaciju sa susjedima
- Non-Bayesian ako se težine ne prilagođavaju tijekom vremena
- Moguće je podcijeniti „težinu“ susjeda (baš kao u eksperimentima)
 - „Tvrdoglavi igrači“

DeGroot model društvene interakcije [DeGroot]

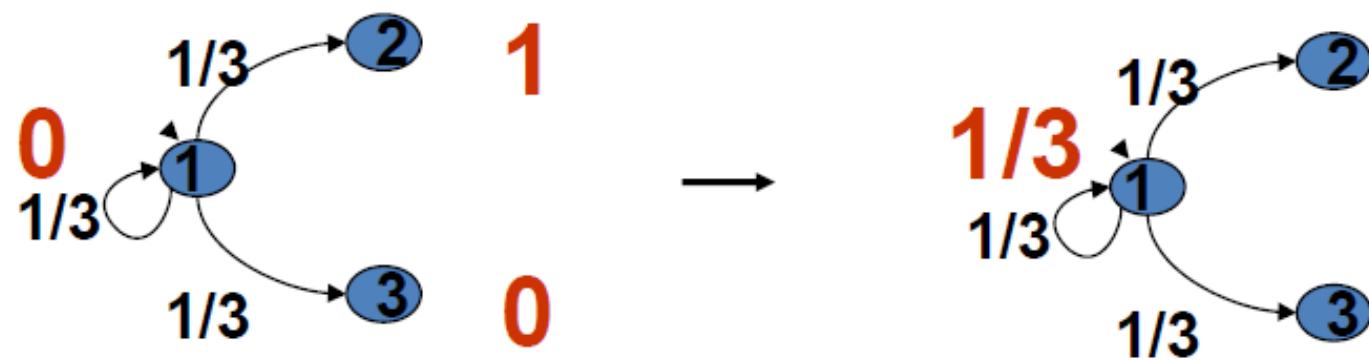
- Igrači $\{1, \dots, n\}$
- T težinska usmjerena mreža, stohastička matrica
- Počinjemo s uvjerenjima $b_i(0)$ u $[0,1]$
 - to također mogu biti vektori
- Ažuriranje: $b_i(t) = \sum_j T_{ij} b_j(t-1)$

DeGroot model: primjer (1)

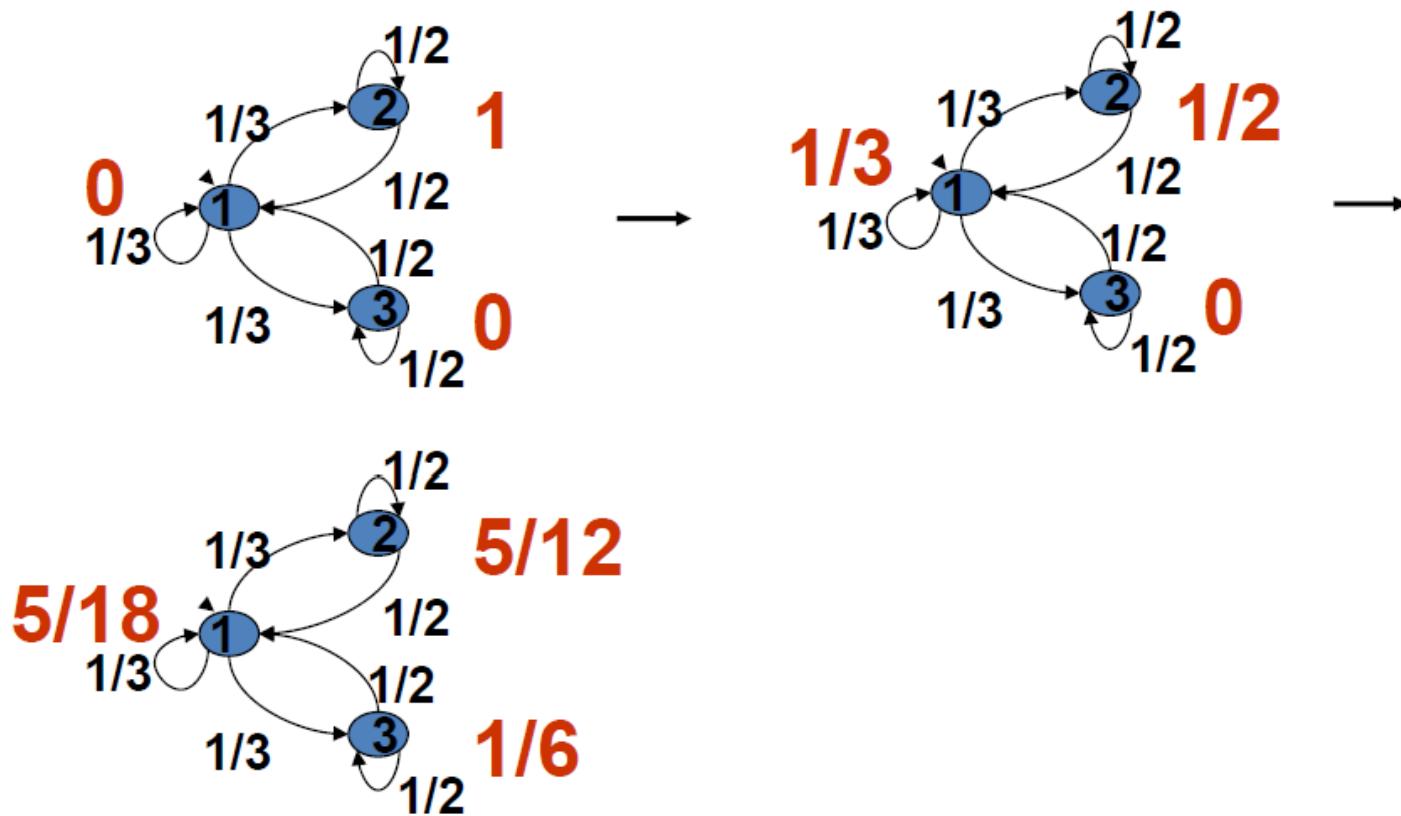


$$T = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \end{bmatrix}$$

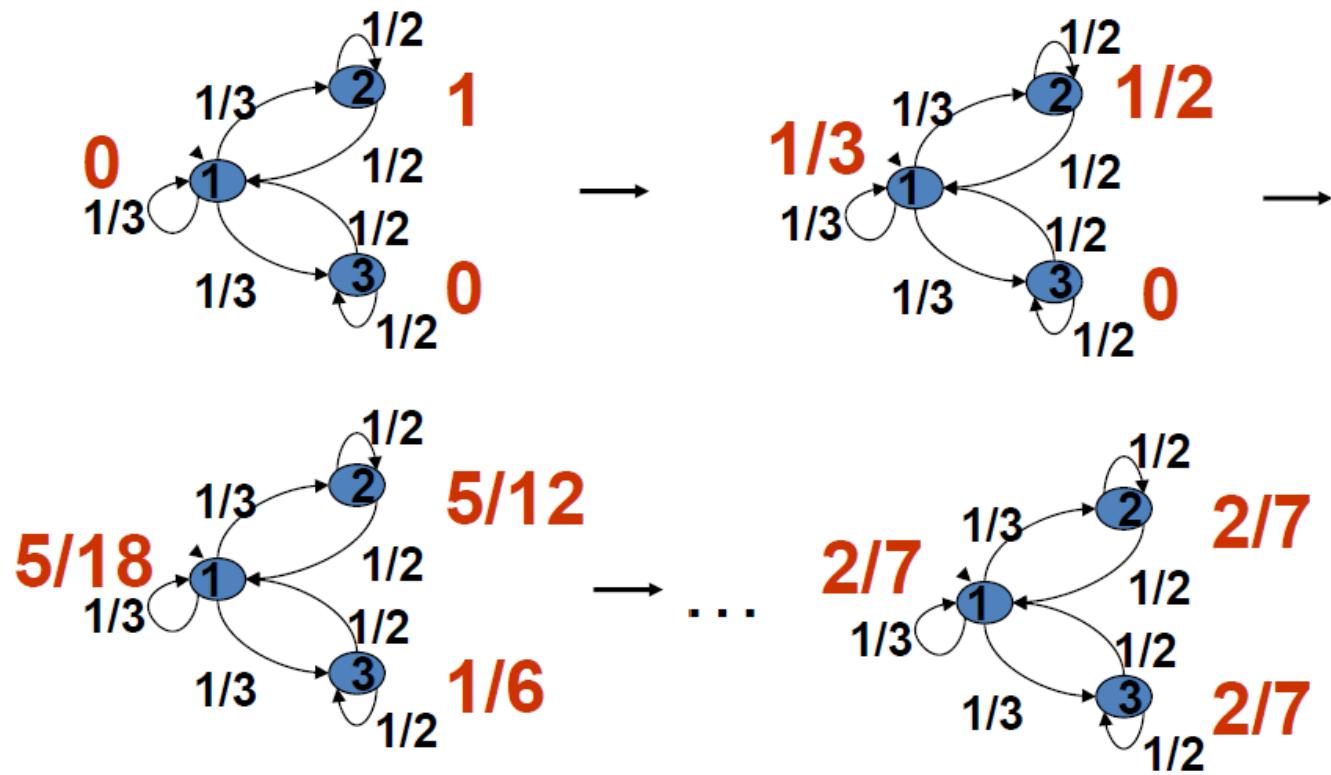
DeGroot model: ažuriranje (1)



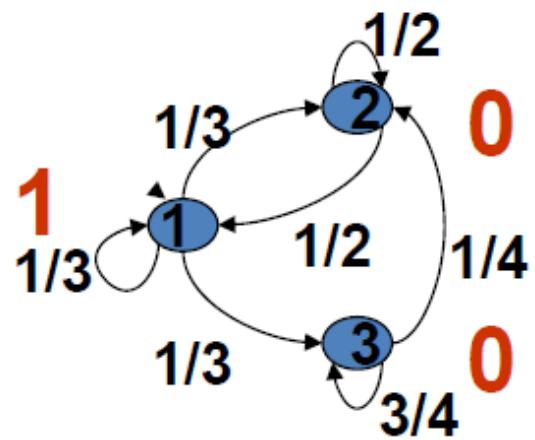
DeGroot model: ažuriranje (2)



DeGroot model: ažuriranje (3)

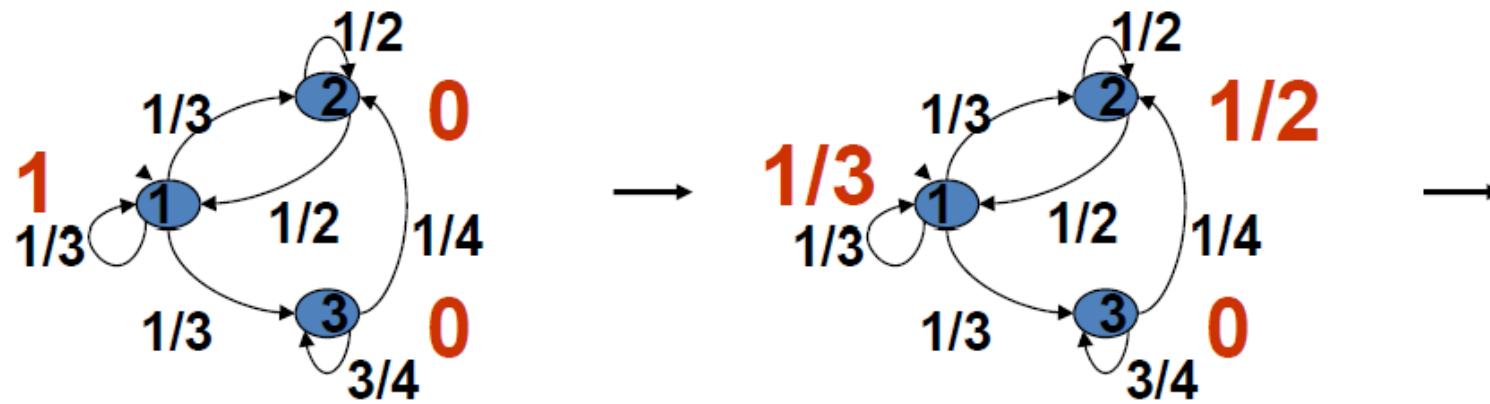


DeGroot model: primjer (2)



$$T = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 0 & 1/4 & 3/4 \end{pmatrix}$$

DeGroot model: ažuriranje (1)



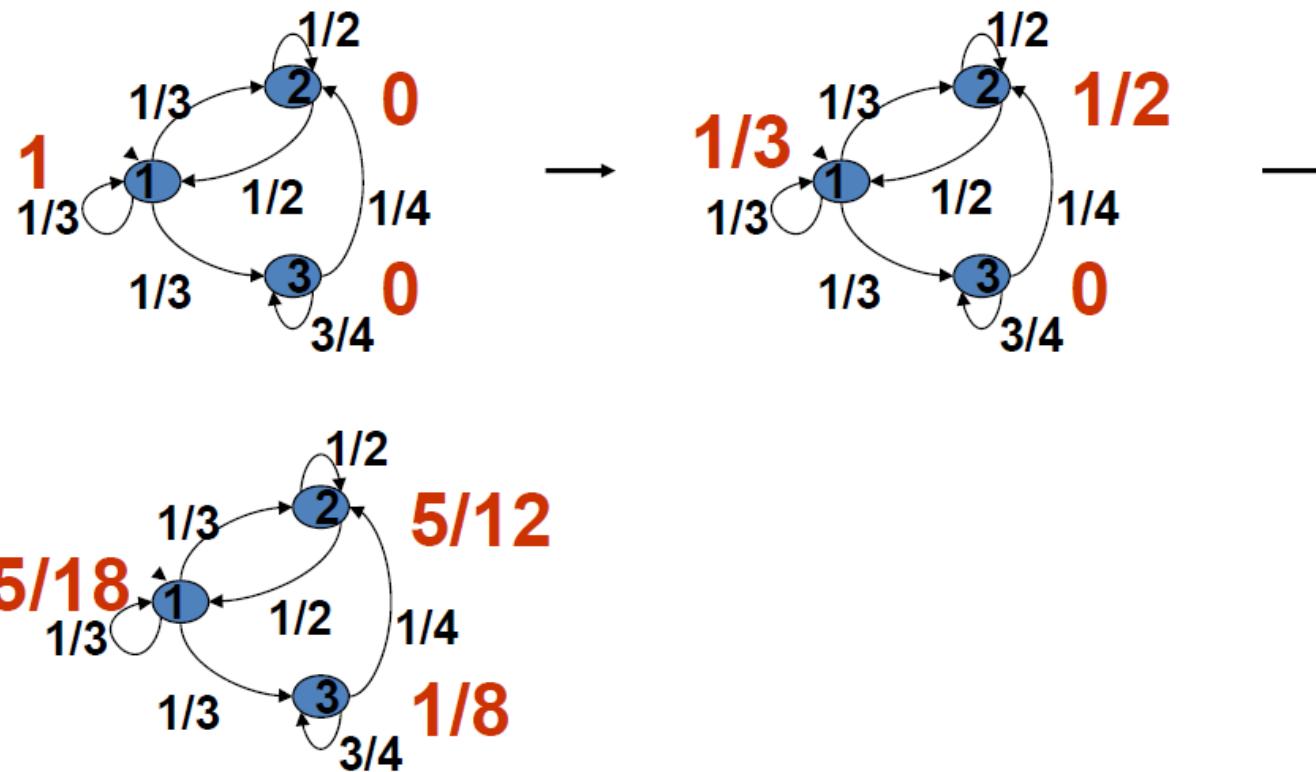
DeGroot model: ažuriranje (2)



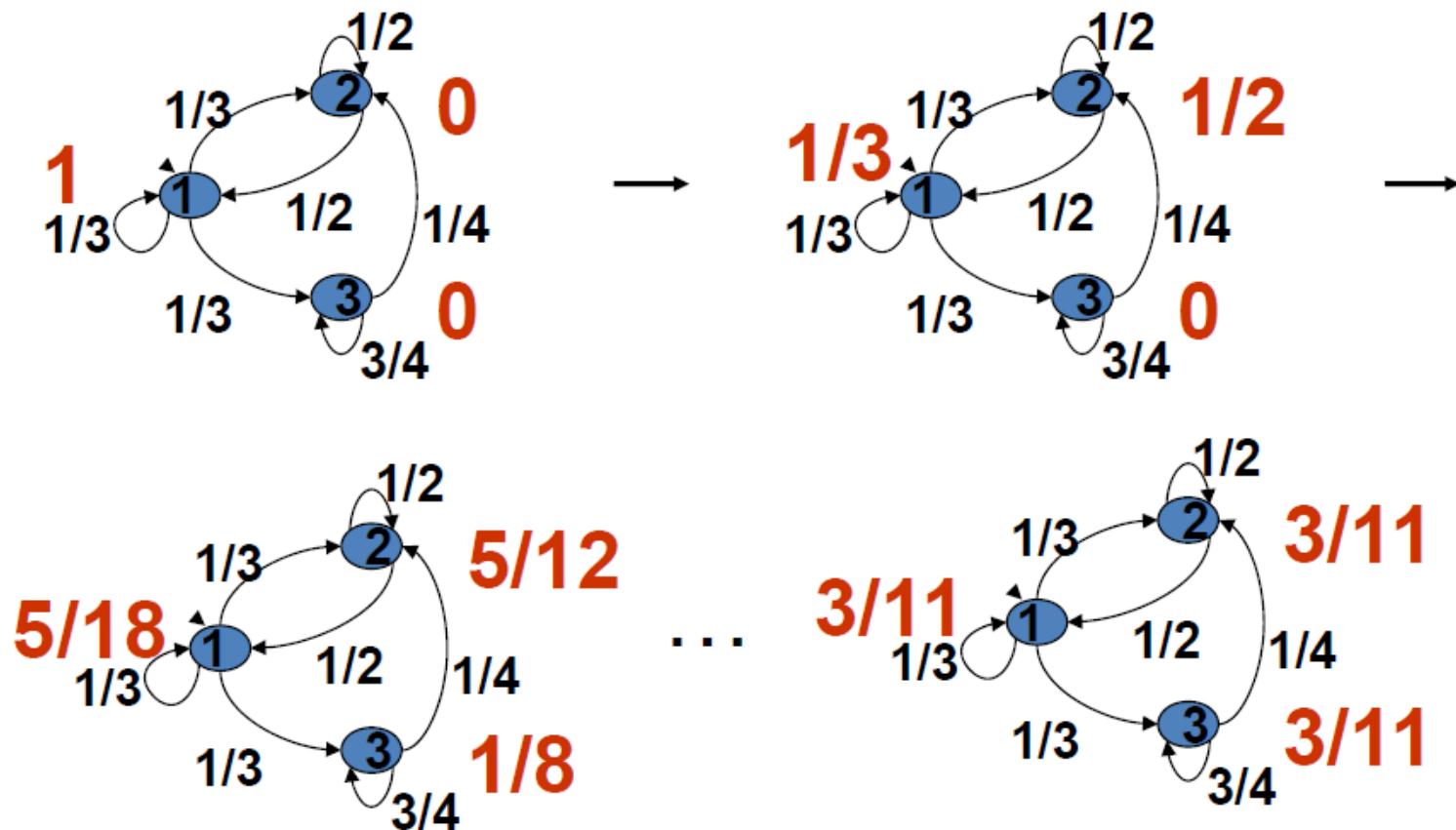
Pitanje:

Kolike će biti vrijednosti uvjerenja
svakog igrača nakon sljedećeg
ažuriranja?

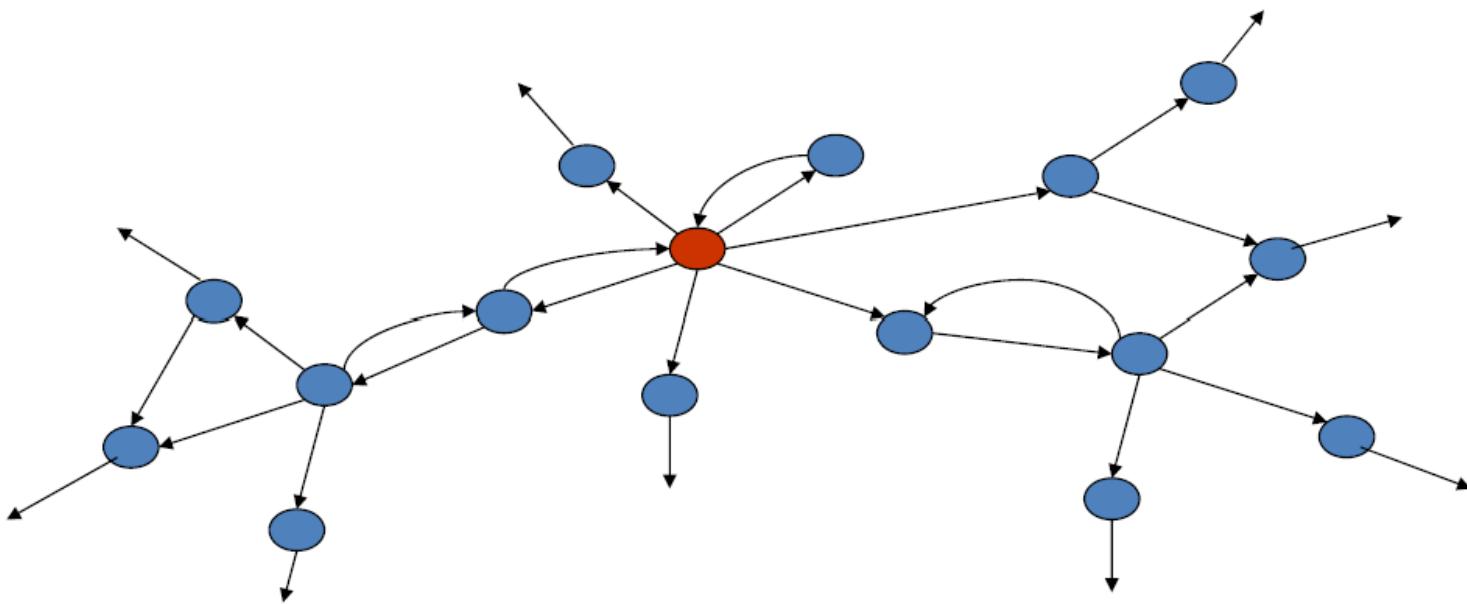
DeGroot model: ažuriranje (2)



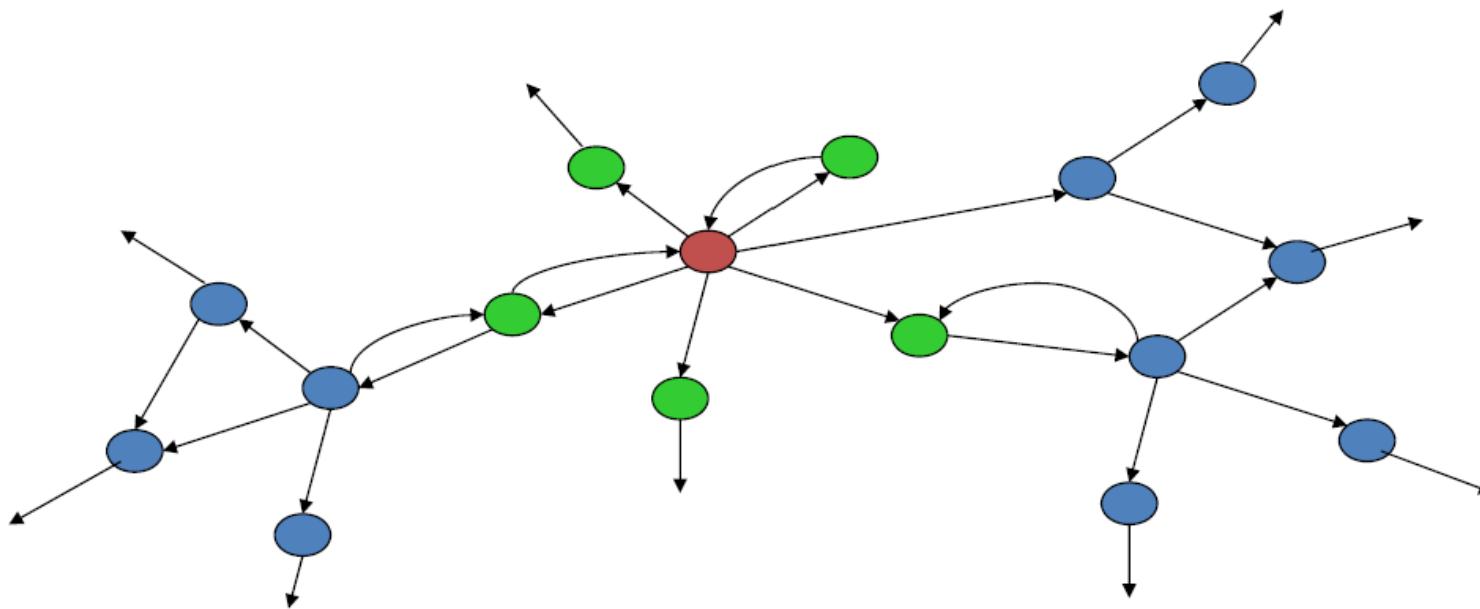
DeGroot model: ažuriranje (3)



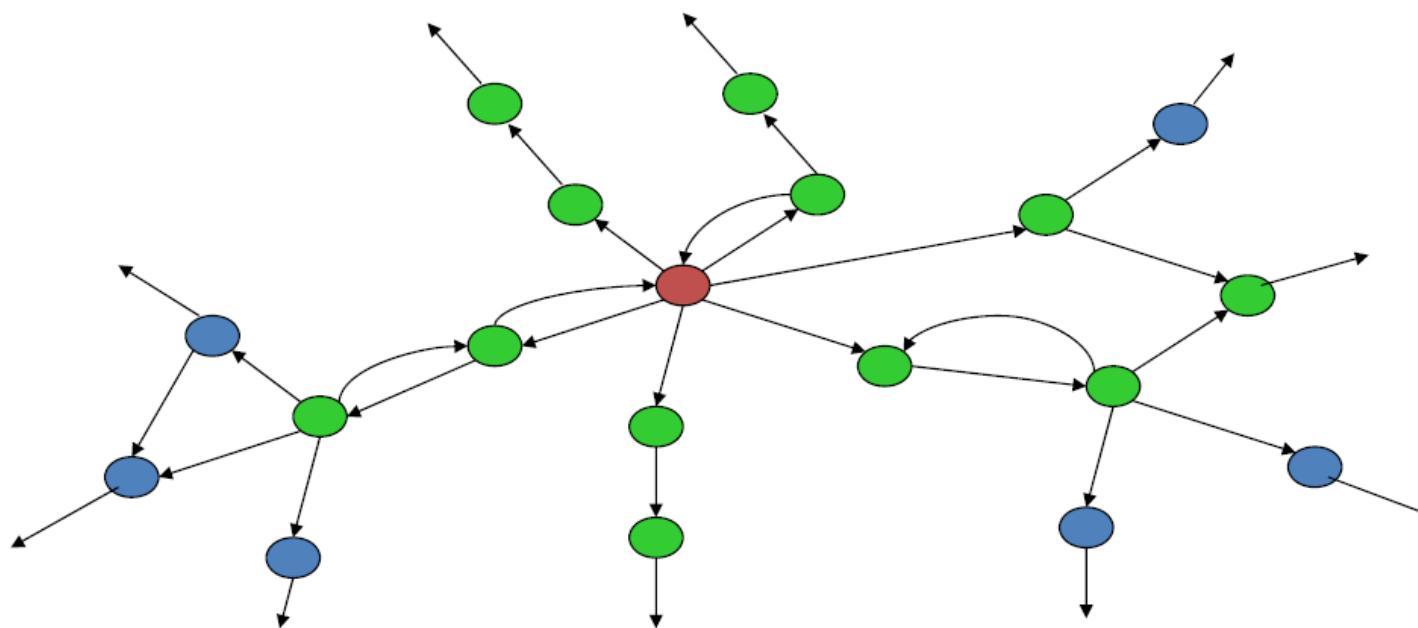
DeGroot model: utjecaj susjeda



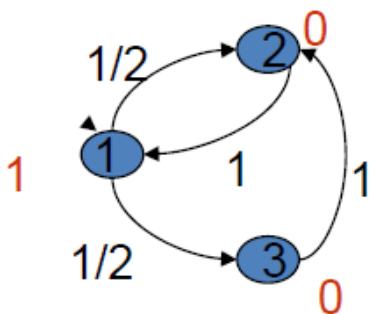
DeGroot model: utjecaj susjedovog susjeda



DeGroot model: utjecaj mreže



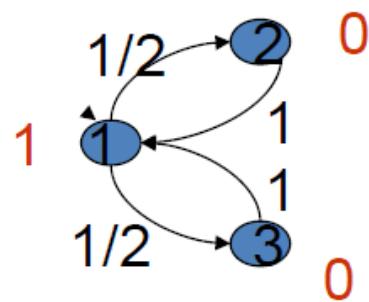
DeGroot model: konvergencija



$$T = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} b(0) &= \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} & b(1) &= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1/2 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 3/4 \\ 1/2 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1/4 \\ 3/4 \\ 1/2 \end{pmatrix} \dots \rightarrow \begin{pmatrix} 2/5 \\ 2/5 \\ 2/5 \end{pmatrix} \end{aligned}$$

DeGroot model: nekonvergencija



$$T = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$b(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad b(1) = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \dots \rightarrow$$

DeGroot model: konsenzus

- Konvergencija prema (normaliziranom) eigenvektoru težinskih suma originalnih uvjerenja

$$T = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$T^3 = \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \end{pmatrix}$$

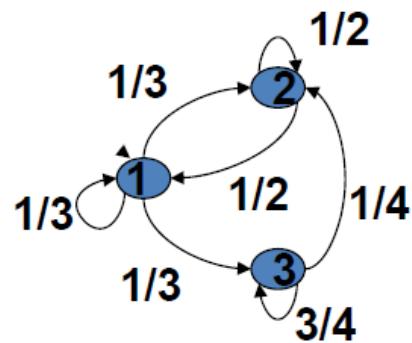
$$T^5 = \begin{pmatrix} 1/2 & 3/8 & 1/8 \\ 1/4 & 1/2 & 1/4 \\ 1/2 & 1/4 & 1/4 \end{pmatrix}$$

$$T^2 = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \end{pmatrix}$$

$$T^4 = \begin{pmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1/4 & 1/4 \\ 1/2 & 1/2 & 0 \end{pmatrix}$$

$$T^\infty = \begin{pmatrix} \textcolor{red}{2/5} & \textcolor{red}{2/5} & \textcolor{red}{1/5} \\ 2/5 & 2/5 & 1/5 \\ 2/5 & 2/5 & 1/5 \end{pmatrix}$$

DeGroot model: konsenzus



$$\lim_t \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 0 & 1/4 & 3/4 \end{pmatrix}^t b(0)$$

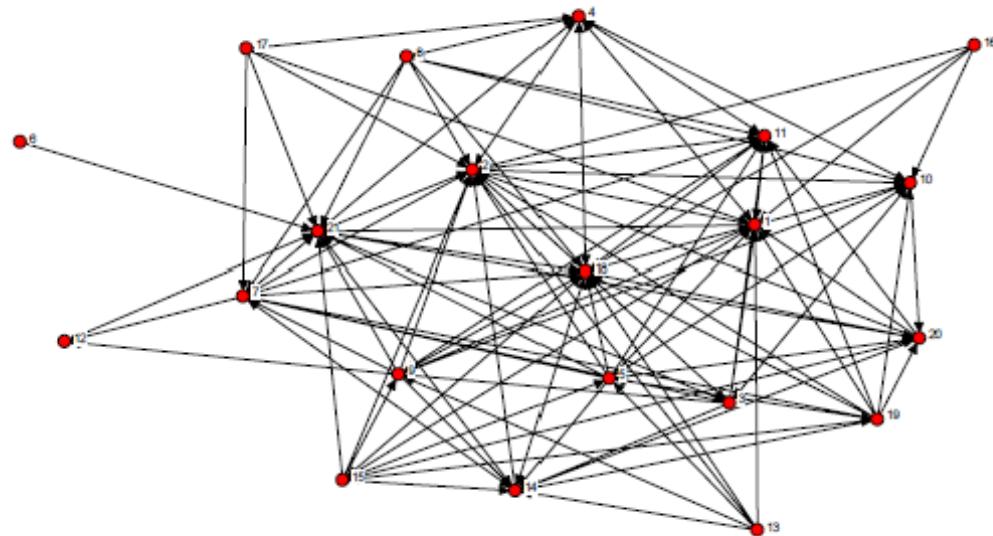
$$\lim_t \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 0 & 1/4 & 3/4 \end{pmatrix}^t = \begin{pmatrix} 3/11 & 4/11 & 4/11 \\ 3/11 & 4/11 & 4/11 \\ 3/11 & 4/11 & 4/11 \end{pmatrix}$$

DeGroot model: utjecaj (*influence*)

- Centralnost eigenvektora

$$\begin{array}{l} \text{Limit} \quad \left(\begin{array}{ccc|c} 1/3 & 1/3 & 1/3 & t \\ 1/2 & 1/2 & 0 & \\ 0 & 1/4 & 3/4 & \end{array} \right) \times \left(\begin{array}{c} 1 \\ 0 \\ 0 \end{array} \right) = \left(\begin{array}{c} 3/11 \\ 3/11 \\ 3/11 \end{array} \right) \\ \text{Limit} \quad \left(\begin{array}{ccc|c} 1/3 & 1/3 & 1/3 & t \\ 1/2 & 1/2 & 0 & \\ 0 & 1/4 & 3/4 & \end{array} \right) \times \left(\begin{array}{c} 0 \\ 1 \\ 0 \end{array} \right) = \left(\begin{array}{c} 4/11 \\ 4/11 \\ 4/11 \end{array} \right) \\ \text{Limit} \quad \left(\begin{array}{ccc|c} 1/3 & 1/3 & 1/3 & t \\ 1/2 & 1/2 & 0 & \\ 0 & 1/4 & 3/4 & \end{array} \right) \times \left(\begin{array}{c} 0 \\ 0 \\ 1 \end{array} \right) = \left(\begin{array}{c} 4/11 \\ 4/11 \\ 4/11 \end{array} \right) \end{array}$$

Mreža savjeta [Krackhardt]



label	s	level	dept.	age	tenure
1	0.048	3	4	33	9.3
2	0.132	2	4	42	19.6
3	0.039	3	2	40	12.8
4	0.052	3	4	33	7.5
5	0.002	3	2	32	3.3
6	0.000	3	1	59	28
7	0.143	1	0	55	30
8	0.007	3	1	34	11.3
9	0.015	3	2	62	5.4
10	0.024	3	3	37	9.3
11	0.053	3	3	46	27
12	0.051	3	1	34	8.9
13	0.000	3	2	48	0.3
14	0.071	2	2	43	10.4
15	0.015	3	2	40	8.4
16	0.000	3	4	27	4.7
17	0.000	3	1	30	12.4
18	0.106	2	3	33	9.1
19	0.002	3	2	32	4.8
20	0.041	3	2	38	11.7
21	0.201	2	1	36	12.5

THE WISDOM OF CROWDS

JAMES SUROWIECKI

WITH A NEW AFTERWORD BY THE AUTHOR



Mudrost masa (*wisdom of crowds*)

- Ukoliko ne postoji nitko u mreži tko ima značajan utjecaj, tada govorimo o scenariju „mudrosti masa”
- Mudrost masa je ideja da su velike grupe ljudi kolektivno pametnije od pojedinačnih stručnjaka kada je riječ o rješavanju problema, donošenju odluka, inovacijama i predviđanjima
 - ideja je da stajalište pojedinca može biti inherentno pristrano, dok uzimanje prosječnog znanja gomile može rezultirati uklanjanjem pristranosti ili buke kako bi se dobio jasniji i koherentniji rezultat

Želim znati više

- M. Jackson: „Social and Economic Networks” (2008)
 - Poglavlje 8 (Learning and Networks)

Kompleksne mreže

8. predavanje

Igre u mrežama (peer utjecaj, odnos mrežne strukture i ponašanja)

Peer utjecaj



- Koje proizvode / usluge kupujemo?
- Koju profesiju izabiremo?
- Koji strani jezik učimo?
- ...

Glavno pitanje

- Kako struktura (društvene) mreže utječe na ponašanje?
 - Ako pojedinac promijeni s kime ima društvenu interakciju, koji će biti utjecaj na njegovo/njezino ponašanje?
 - Inicijalno područje istraživanja u sociologiji, ali danas sve više u ekonomiji i računarstvu

Sadržaj

- Utjecaj cijelog društva na odluke pojedinca
- Utjecaj mrežne strukture na odluke pojedinca
 - Strateške komplementarnosti
 - Strateški supstituti
- Utjecaj dinamike mreže na ponašanje pojedinca

Primjer 1: Utjecaj cijelog društva na odluke pojedinca (1)

- Dvije moguće akcije su označene kao 0 i 1
- Vrijeme napreduje u diskretnim razdobljima $t \in \{1, 2, \dots\}$
- Stanje sustava opisuje se brojem pojedinaca koji poduzimaju akciju 1, označeno s s_t , na kraju razdoblja t
- Tranzicija između diskretnih razdoblja: $\Pr(s_{t+1} = s' | s_t = s)$

Primjer 1: Utjecaj cijelog društva na odluke pojedinca (2)

- Markovljev lanac
- Ravnomjerno nasumično biramo pojedinca koji ažurira svoju akciju na temelju trenutnog broja ljudi u društvu koji poduzimaju akciju 0 ili 1
 - Primjerice, pojedinac odlučuje hoće li otići na plažu ili ne



Ponašanje mrava: imitacija i *herdanje* [Kirman]

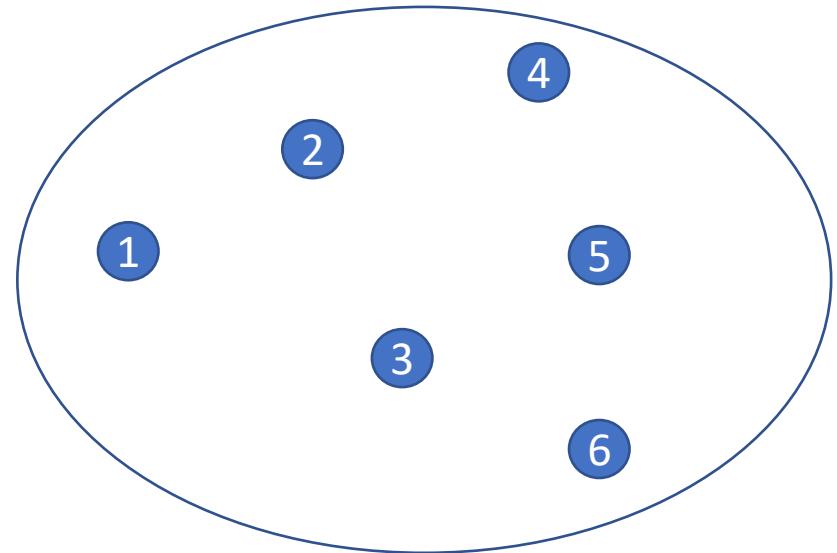
- Kirmanov rad bio je motiviran zapažanjem da mravi imaju tendenciju okupljanja (*herd*) na izvorima hrane koje iskorištavaju čak i kada imaju izbor nekoliko različitih jednako korisnih izvora



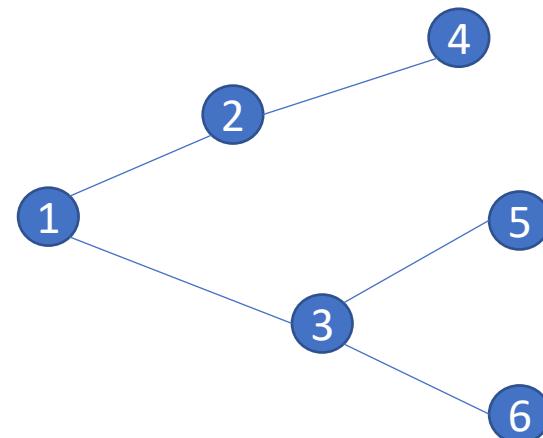
Utjecaj mrežne strukture (1)

- Dok jednostavan Markovljev model društvenih interakcija daje uvid u široke obrasce društvenog ponašanja, ne uključuje mikro-detalje tko s kim komunicira
- Takvi mrežni odnosi mogu imati dubok učinak na proces
- Da bismo uključili umrežene interakcije, potrebna nam je bogatija struktura

6 pojedinaca u društvu



vs



6 pojedinaca u društvenoj mreži

Utjecaj mrežne strukture (2)

- Pojedinci opet biraju između dvije akcije 0 i 1
- Razlika je da sada društveno stanje mora pratiti koji pojedinci poduzimaju koje akcije
 - Društveno stanje je tako n-dimenzionalni vektor $x(t)$, gdje je $x_i(t)$ za $i \in \{1, 2, \dots\}$ akcija koju pojedinac i radi u trenutku t
 - Interakcija je opisana s w , što je $n \times n$ -dimenzionalna matrica, gdje je $w_{ij} = [0, 1]$ težina koja opisuje vjerojatnost da je izbor pojedinca i u trenutku $t + 1$ akcija koju je pojedinac j poduzeo u trenutku t

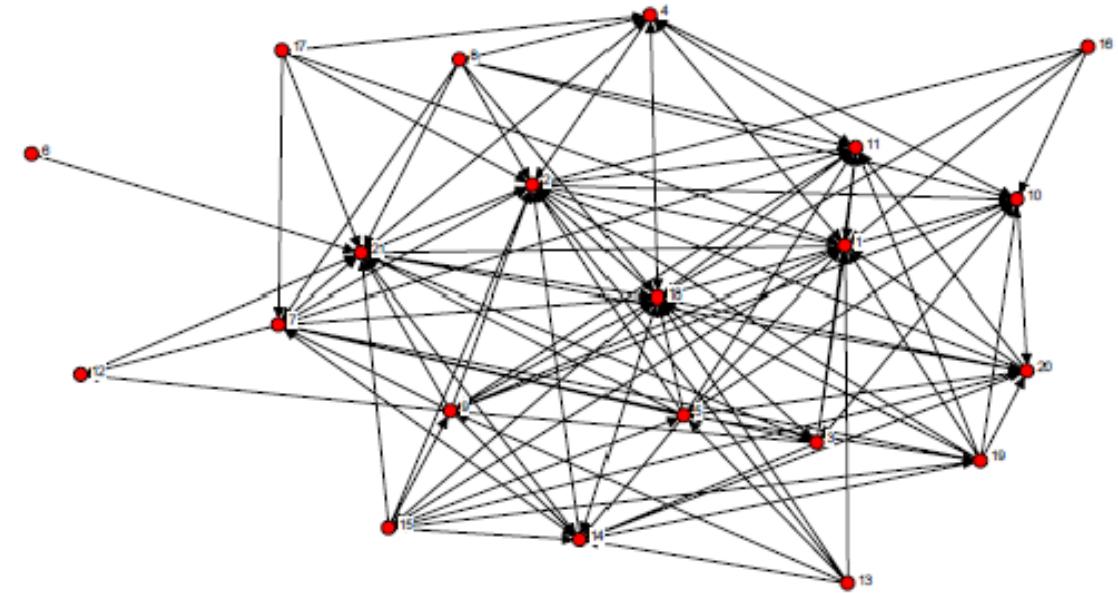
Utjecaj mrežne strukture (3)

- Nadalje, dopustimo vjerojatnost, označenu kao $\varepsilon_i(1)$, da pojedinac i izabere akciju 1 neovisno o stanju sustava i vjerojatnost, označenu kao $\varepsilon_i(0)$, da izabere radnju 0 neovisno o stanju sustava
- Tada vrijedi:

$$\Pr(x_i(t+1) = 1 | x(t)) = \varepsilon_i(1) + (1 - \varepsilon_i(1) - \varepsilon_i(0)) \sum_j w_{ij} x_j(t)$$

Primjer 2: Mreža savjeta među menadžerima [Krackhardt] (1)

Akcija: odlazak u bar nakon posla



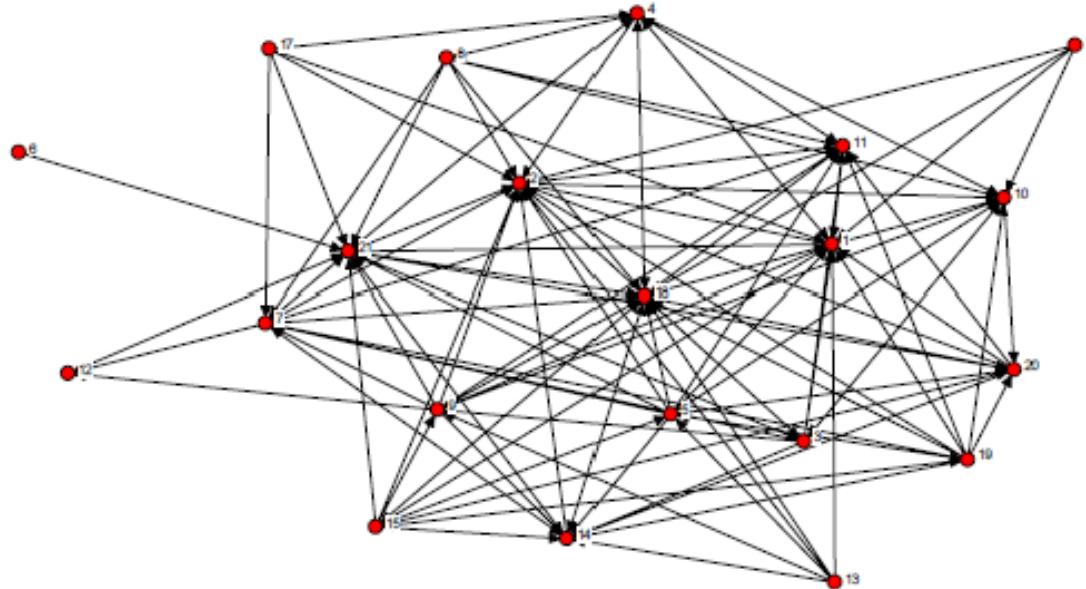
Primjer 2: Mreža savjeta među menadžerima

[Krackhardt] (2)

- Menadžer s *outdegree* vrijednosti d odabire otići u bar s vjerojatnošću $1/(d + 2)$, a ne otići s vjerojatnošću $1/(d + 2)$, a s preostalom vjerojatnošću $d/(d + 2)$ uniformno nasumično odabire jednog od njegovih/njezinih susjeda te zatim odlazi u bar ako je taj susjed otišao u bar prethodni dan
- Ovo vrijedi za sve menadžere, osim za menadžere najviše razine (označene kao čvorovi 2, 7, 14, 18 i 21) koji su pristrani prema odlasku u bar
 - Menadžeri najviše razine koriste slično pravilo osim što koriste težine $1/(d + 1)$ i ne postavljaju nikakve težine na radnju 0
- Na temelju ovoga možemo izračunati učestalost kojom će svaki menadžer ići u bar na duge staze

Primjer 2: Mreža savjeta među menadžerima

[Krackhardt] (3)



label	Prob of 1	level	dept.	age	tenure
1	0.667	3	4	33	9.3
2	0.842	2	4	42	19.6
3	0.690	3	2	40	12.8
4	0.666	3	4	33	7.5
5	0.690	3	2	32	3.3
6	0.585	3	1	59	28
7	0.771	1	0	55	30
8	0.676	3	1	34	11.3
9	0.681	3	2	62	5.4
10	0.660	3	3	37	9.3
11	0.656	3	3	46	27
12	0.585	3	1	34	8.9
13	0.680	3	2	48	0.3
14	0.821	2	2	43	10.4
15	0.687	3	2	40	8.4
16	0.651	3	4	27	4.7
17	0.671	3	1	30	12.4
18	0.737	2	3	33	9.1
19	0.685	3	2	32	4.8
20	0.685	3	2	38	11.7
21	0.755	2	1	36	12.5

Utjecaj mrežne strukture na odluke pojedinca: definirajmo kanonski primjer

- Svaki pojedinac bira akciju x_i u $\{0,1\}$
- Nagrada pojedincu ovisi o tome
 - koliko susjeda bira svaku akciju
 - koliko susjeda pojedinac ima
- Razmotrimo slučajeve u kojima nagrada $u_{d_i}(x_i, m_{N_i})$ ovisi samo o $d_i(g)$ i $m_{N_i}(g)$ - broju susjeda pojedinca i koji su odabrali akciju 1

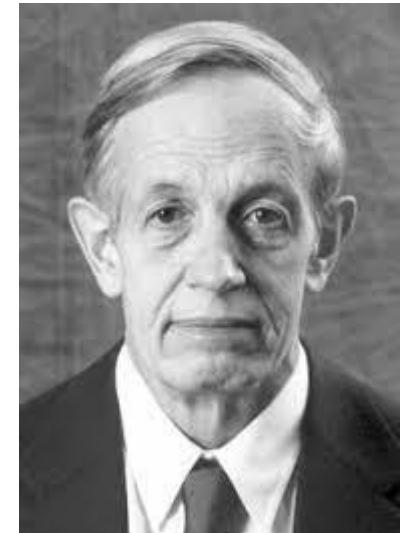
Definicija: ekvilibrij

Pitanje:

Definirajte jednom rečenicom pojam
ekvilibrija?

Definicija: ekvilibrij (teorija igara)

- **ekvilibrij** (lat. *aequilibrium*): ravnoteža, uravnoteženost
- **Nashova ravnoteža** je situacija u kojoj pojedinci ili igrači nemaju poticaj za promjenu svoje strategije uzimajući u obzir odluke svojih protivnika
- U Nashovoj je ravnoteži optimalna strategija koju je odabrao svaki od sudionika u sukobu ili igri, s obzirom na strategiju koju su odabrali ostali. Drugim riječima, nitko neće ništa dobiti ako odluči promijeniti svoju strategiju pod pretpostavkom da drugi pojedinci ne mijenjaju svoju
- Treba napomenuti da u Nashovoj ravnoteži najveći dobitak nije nužno postignut za sve pojedince ili igrače zajedno. Istina je samo da svaki optimalno reagira na strategiju ostalih. U mnogim slučajevima pojedinci bi željeli postići novu ravnotežu s većom dobiti, ali to ne uspijevaju jer se suočavaju s rizikom da budu izdani



John Nash
(1928-2015)

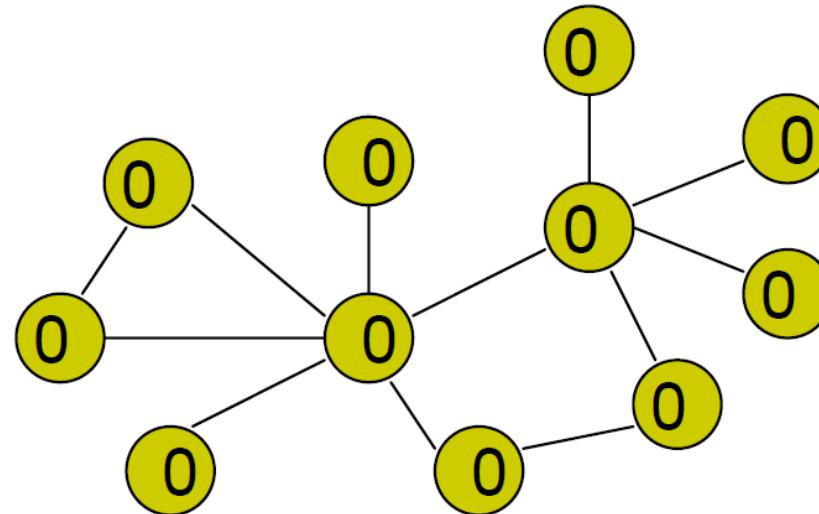
Primjer 3: Jednostavni komplement (1)

- Pojedinac i voljan je izabrati akciju 1 ako i samo ako to učini najmanje t njegovih/njezinih susjeda
- Nagrada akcije 0: $u_{d_i}(0, m_{N_i}) = 0$
- Nagrada akcije 1: $u_{d_i}(1, m_{N_i}) = -t + m_{N_i}$

Primjer 3: Jednostavni komplement (2)

- Pojedinac i voljan je izabrati akciju 1 ako i samo ako to učini najmanje 2 njegova/njezina susjeda

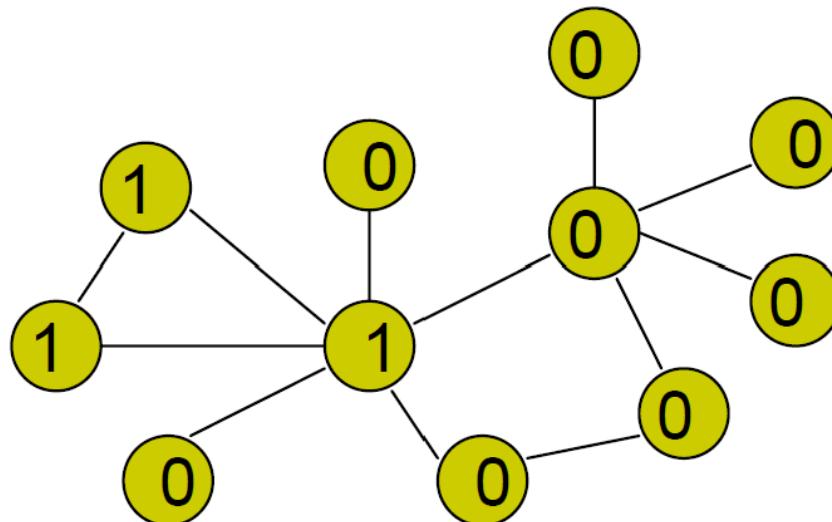
Ekvilibrij 1



Primjer 3: Jednostavni komplement (3)

- Pojedinac i voljan je izabrati akciju 1 ako i samo ako to učini najmanje 2 njegova/njezina susjeda

Ekvilibrij 2

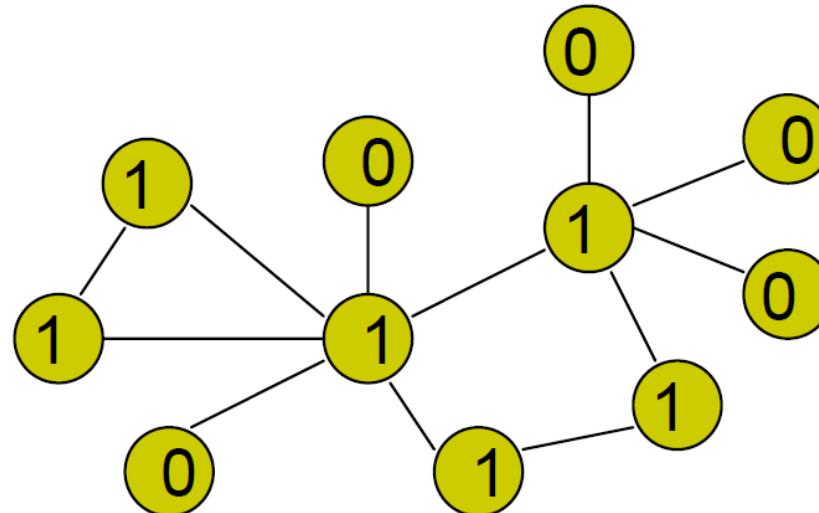


Primjer 3: Jednostavni komplement (4)

- Pojedinac *i* voljan je izabrati akciju 1 ako i samo ako to učini najmanje 2 njegova/njezina susjeda

Pitanje:

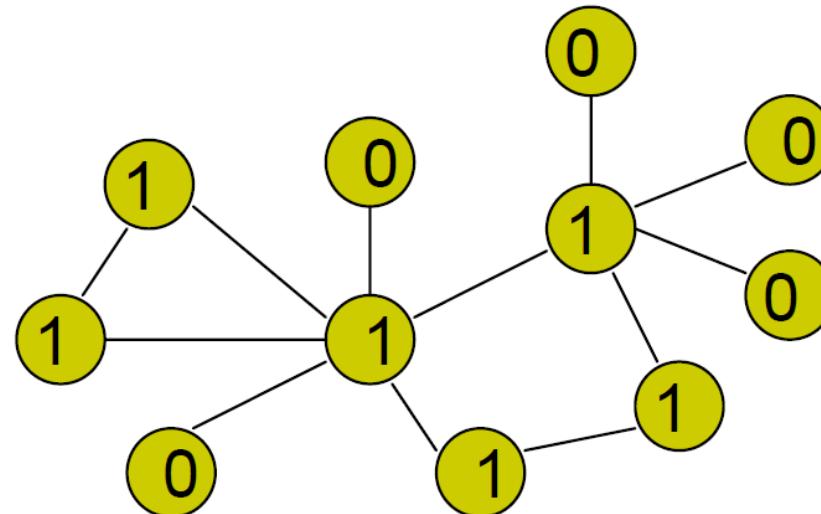
Je li ovo ekvilibrij?



Primjer 3: Jednostavni komplement (5)

- Pojedinac *i* voljan je izabrati akciju 1 ako i samo ako to učini najmanje 2 njegova/njezina susjeda

“Maksimalni
ekvilibrij”



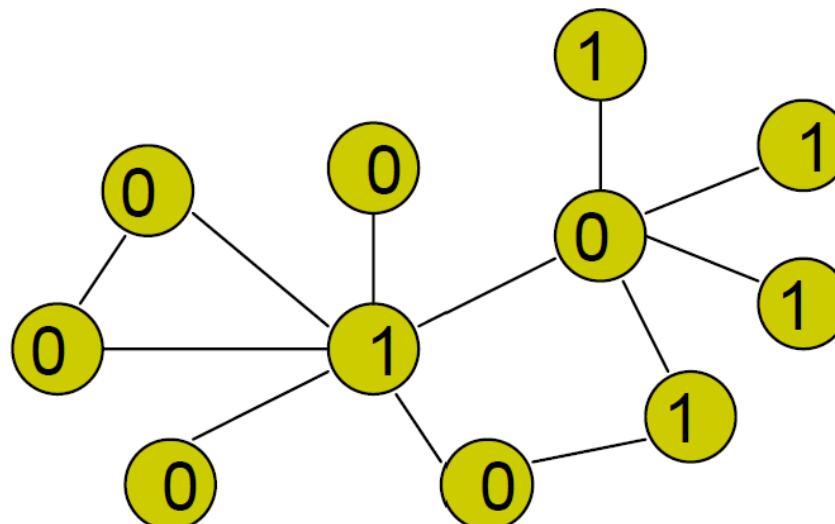
Primjer 4: *Best Shot* (1)

- Pojedinac i voljan je izabrati akciju 1 ako i samo ako to ne učini nijedan od njegovih/njezinih susjeda
- Nagrada akcije 0: $u_{d_i}(0, m_{N_i}) = 1 \text{ ako } m_{N_i} > 0$
 $\qquad\qquad\qquad = 0 \text{ ako } m_{N_i} = 0$
- Nagrada akcije 1: $u_{d_i}(1, m_{N_i}) = 1 - c$

Primjer 4: *Best Shot* društveno dobro (2)

- Pojedinac *i* voljan je izabrati akciju 1 ako i samo ako to ne učini nijedan od njegovih/njezinih susjeda

Ekvilibrij 1

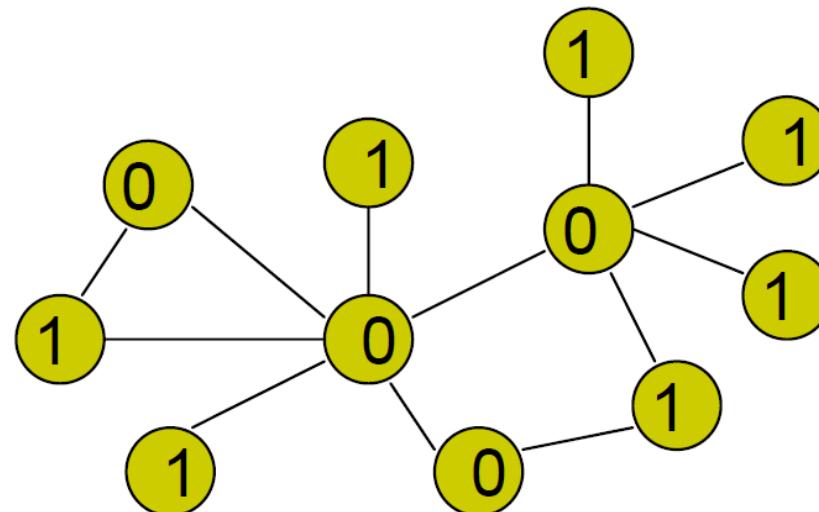


Primjer 4: *Best Shot* društveno dobro (3)

- Pojedinac *i* voljan je izabrati akciju 1 ako i samo ako to ne učini nijedan od njegovih/njezinih susjeda

Pitanje:

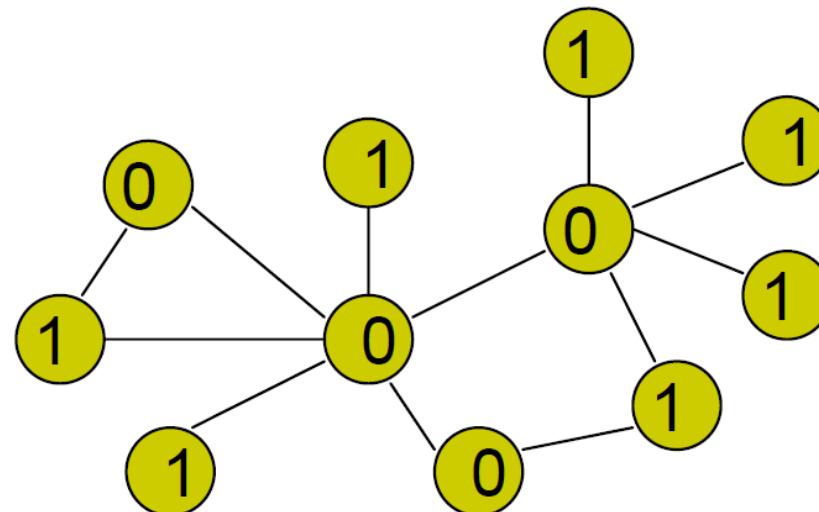
Je li ovo ekvilibrij?



Primjer 4: *Best Shot* društveno dobro (4)

- Pojedinac i voljan je izabrati akciju 1 ako i samo ako to ne učini nijedan od njegovih/njezinih susjeda

Ekvilibrij 2



(Strateški) komplementi vs supstituti (1)

- Strateški **komplementi** (za sve $d, m \geq m'$)

- Povećanje razlike

$$u_d(1, m) - u_d(0, m) \geq u_d(1, m') - u_d(0, m')$$

- Strateški **supstituti** (za sve $d, m \geq m'$)

- Smanjenje razlike

$$u_d(1, m) - u_d(0, m) \leq u_d(1, m') - u_d(0, m')$$

(Strateški) komplementi vs supstituti (2)

- Strateški **komplementi**

- Izbor mojih prijatelja da poduzmu akciju povećava moju relativnu dobit od poduzimanja te akcije (npr. prijatelj uči igrati video igru)



- Strateški **supstituti**

- Izbor za poduzimanje akcije od strane mojih prijatelja smanjuje moju relativnu nagradu za poduzimanje te akcije (npr. cimer kupuje TV ili frižider)



(Strateški) komplementi vs supstituti (3): primjeri

- Strateški **komplementi**

- odluke o obrazovanju
 - briga o broju susjeda, pristupu poslovima: uložiti ako to učini barem *k* susjeda
- pušenje i drugo ponašanje među tinejdžerima, vršnjacima
- usvajanje tehnologije – koliko je drugih kompatibilnih
- naučiti jezik
- varanje, doping



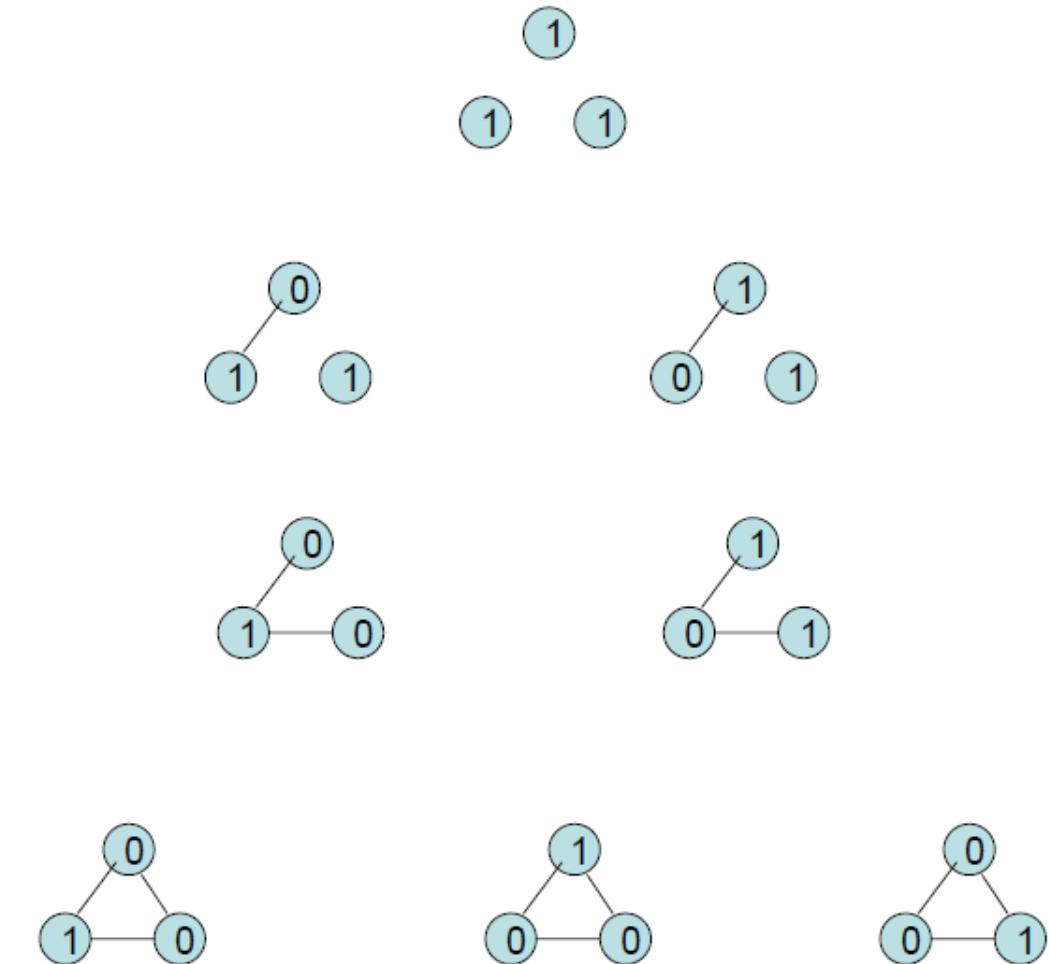
- Strateški **supstituti**

- skupljanje informacija
 - npr. isplata 1 ako je netko u susjedstvu obaviješten, trošak informiranja ($c < 1$)
- lokalna javna dobra (dijeljivi proizvodi, ...)
- konkurentske tvrtke (oligopol s lokalnim tržištima)



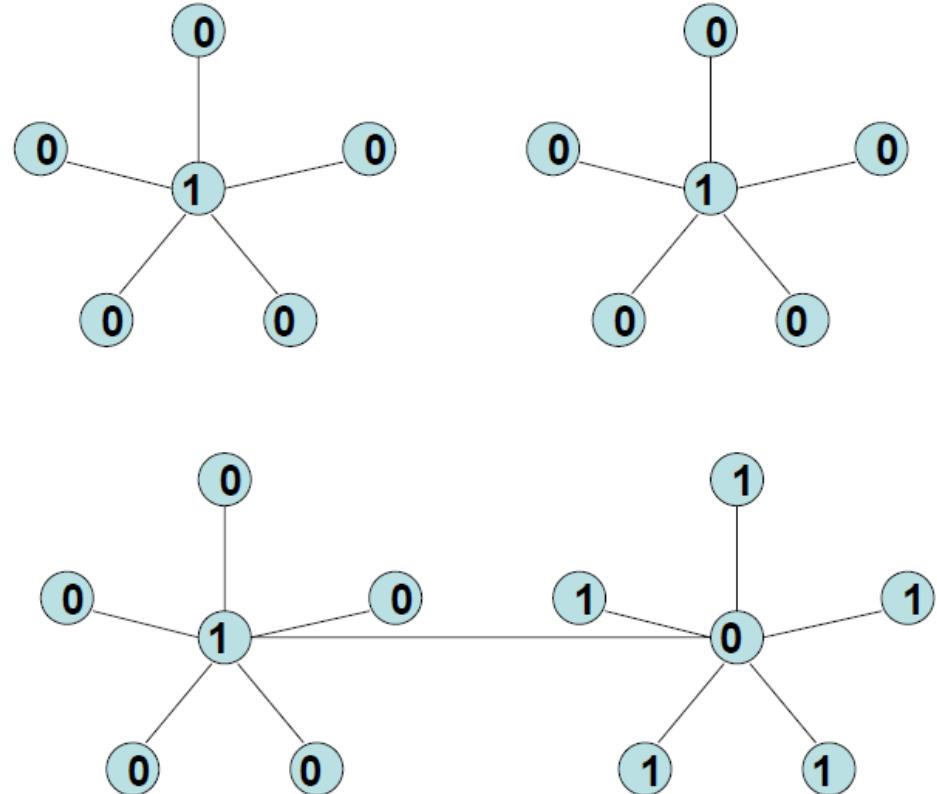
Utjecaj veza u mreži

- Postojanje različitih veza u mreži **mijenja ekvilibrij**
 - *best shot* scenarij



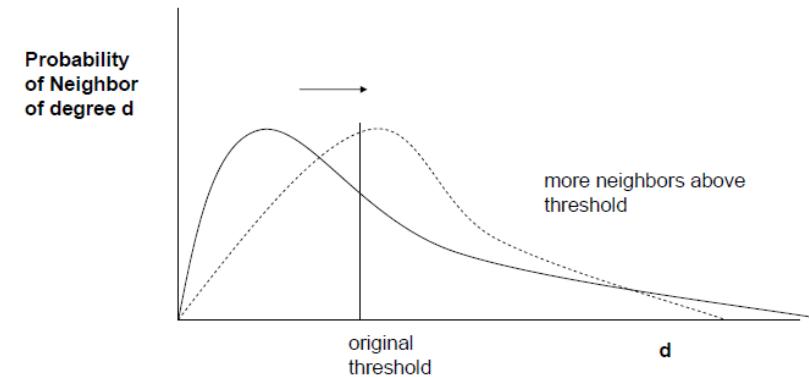
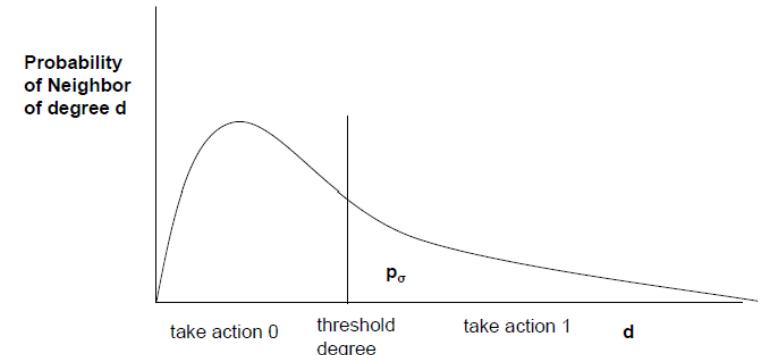
Utjecaj dinamike mreže

- Dodavanje nove veze u društvenoj mreži **mijenja ekvilibrij**
 - *best shot* scenarij

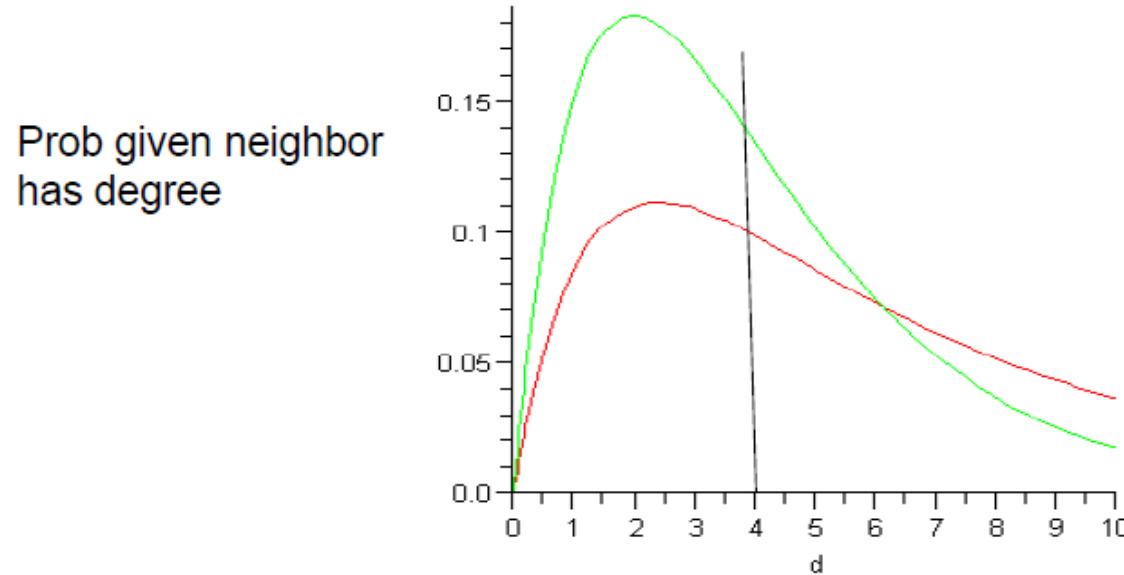


Utjecaj broja veza u mreži (1)

- Ponašanje pojedinca kao funkcija stupnja čvora (broja veza)
 - Scenarij strateške komplementarnosti



Utjecaj broja veza u mreži (2)

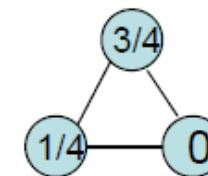
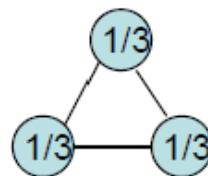
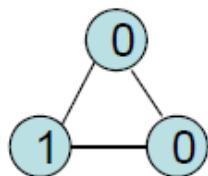
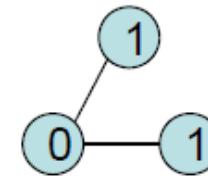
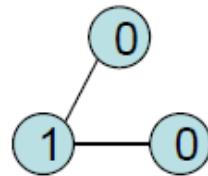


Crveno (mreža koautorstva): Goyal, S., M. van der Leij, and J.-L. Moraga-González (2006). Economics: an emerging small world, Journal of Political Economy

Zeleno (ljubavna mreža): Bearman, P., J. Moody, and K. Stovel (2004), Chains of Affection: The Structure of Adolescent Romantic and Sexual Networks, University of Chicago

Što kada imamo više od dvije moguće akcije?

[Bramoule & Kranton]



Želim znati više

- M. Jackson: „Social and Economic Networks” (2008)
 - Poglavlje 9 (Decisions, Behavior, and Games on Networks)

Kompleksne mreže

9. predavanje

Društvene mreže (definicija, razvoj, vrste)

Sadržaj

- Osnovno o društvenim mrežama
- Rana (evolucija) društvenih mreža
- Trendovi i statistika
- Analiza (podataka) društvenih mreža

Osnovno o društvenim mrežama

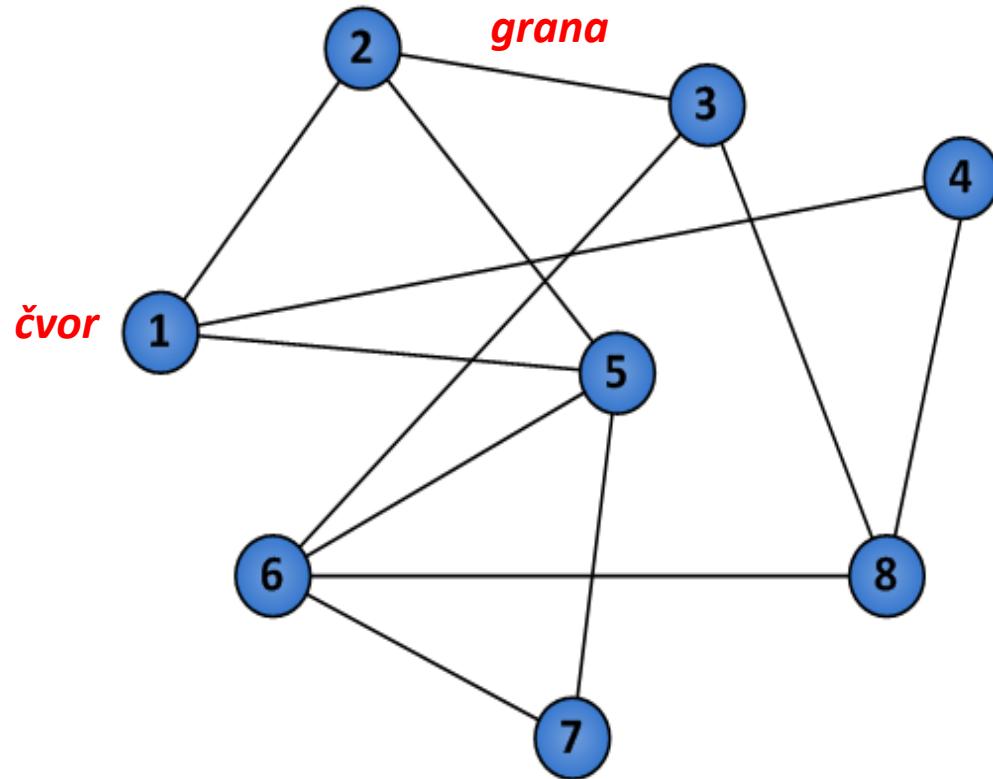
Definicija, temeljne postavke, povijest

Što je to mreža?



Što je to mreža?

- Skup **čvorova** međusobno povezanih **granama**



Što je to društvena mreža?



Što je to društvena mreža?

- Skup **entiteta** međusobno povezanih **odnosima**
 - Entiteti: *ljudi, grupe ljudi, organizacije, brandovi*
 - Odnosi: *poznanstvo, obiteljska povezanost, privlačnost, ...*

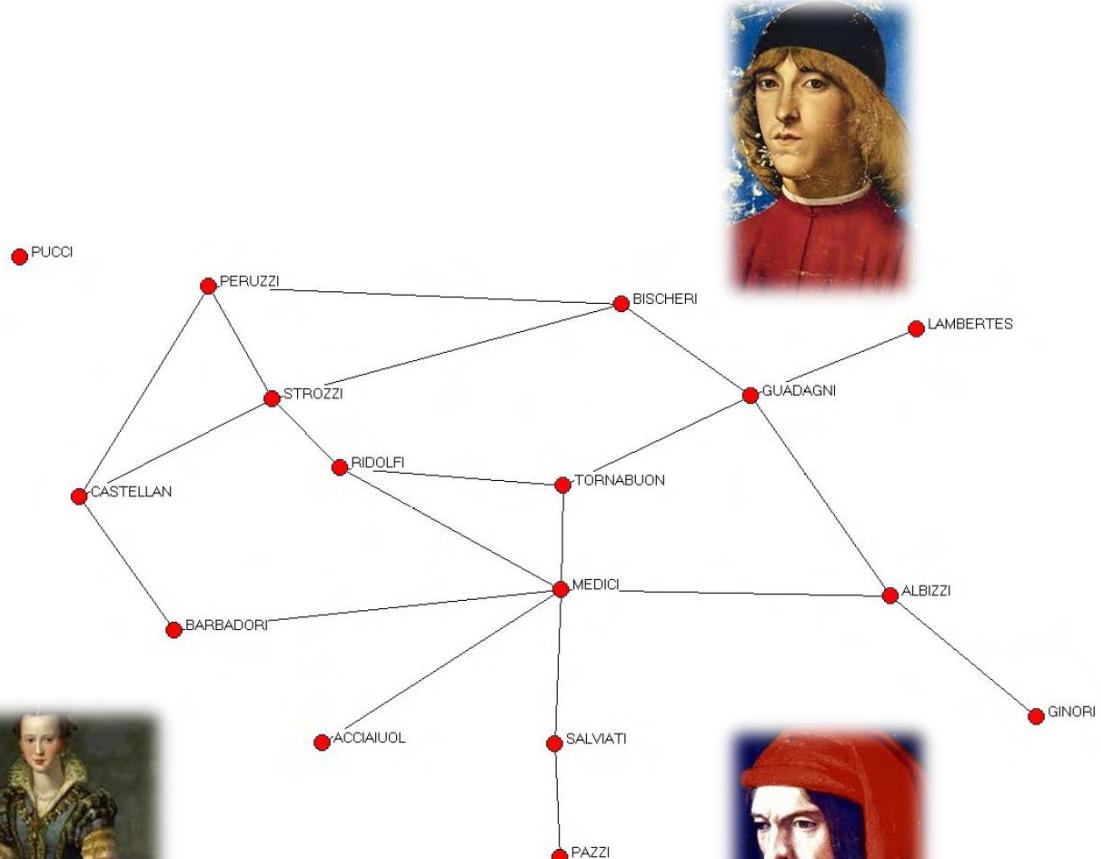


The hidden influence of social networks

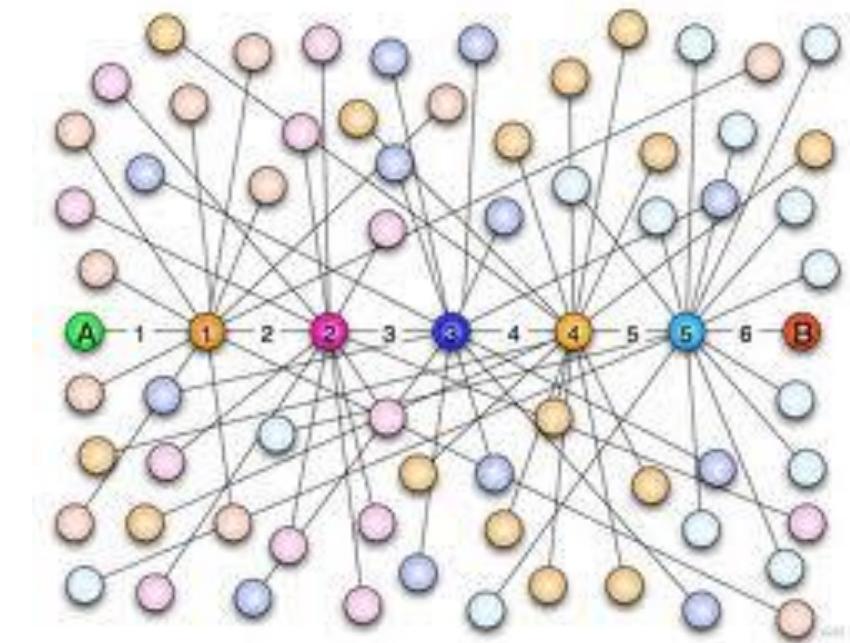
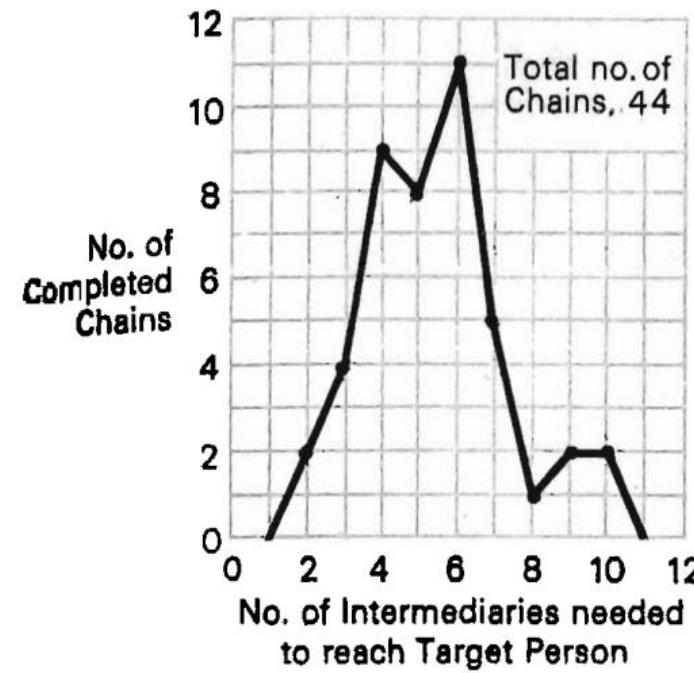
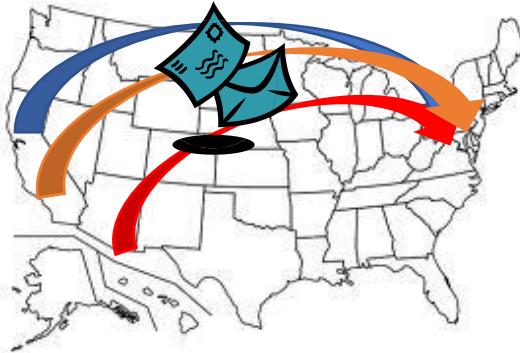


You The hidden influence of social networks

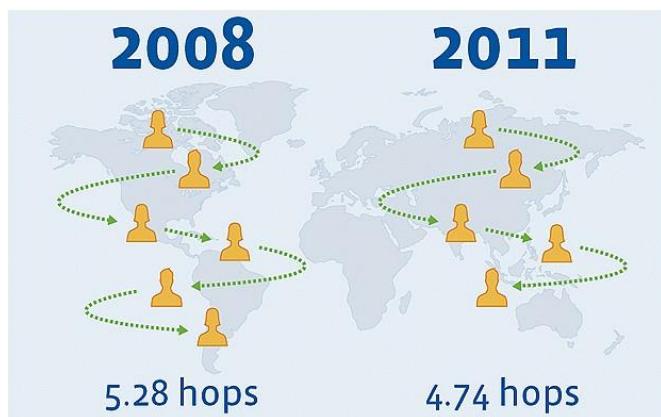
Povijest društvenih mreža (1): Utjecaj obitelji Medici (1430)



Povijest društvenih mreža (2): Milgramov „small world” eksperiment: „six degrees of separation”



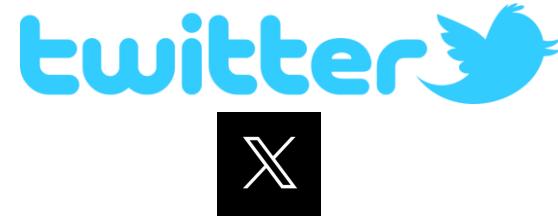
Six degrees of separation u Facebook eri



2016
3,57

Tipovi društvenih mreža

- Društvene mreže se mogu različito klasificirati
- **Eksplisitne društvene mreže**
 - Poznati odnosi među aktorima
 - Facebook, Twitter, LinkedIn i Instagram su dobri primjeri eksplisitnih društvenih mreža



- **Implicitne društvene mreže**

- Poveznice među aktorima mogu samo **dubljom analizom** dostupnih podataka biti utvrđene
- Amazon ili Netflix
 - Analizom logova mogu se zaključiti zajednički interesi poput političkih naklonosti ili hobija



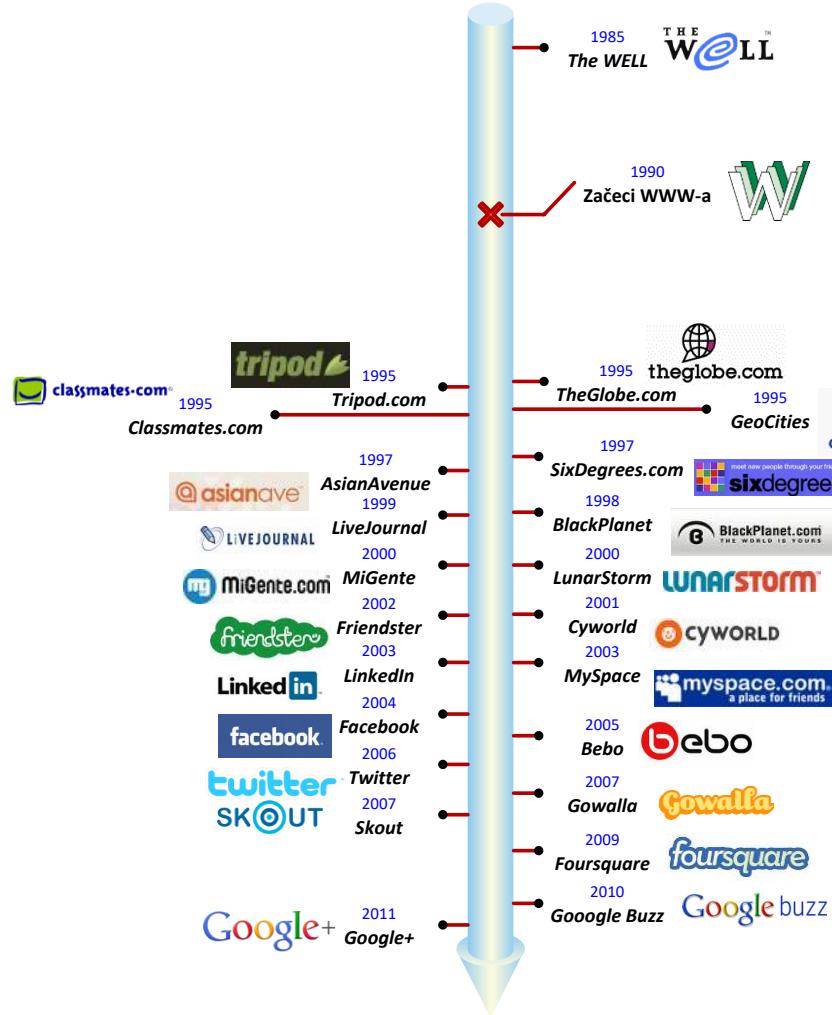
(Rana) evolucija društvenih mreža

1985. – 2008.

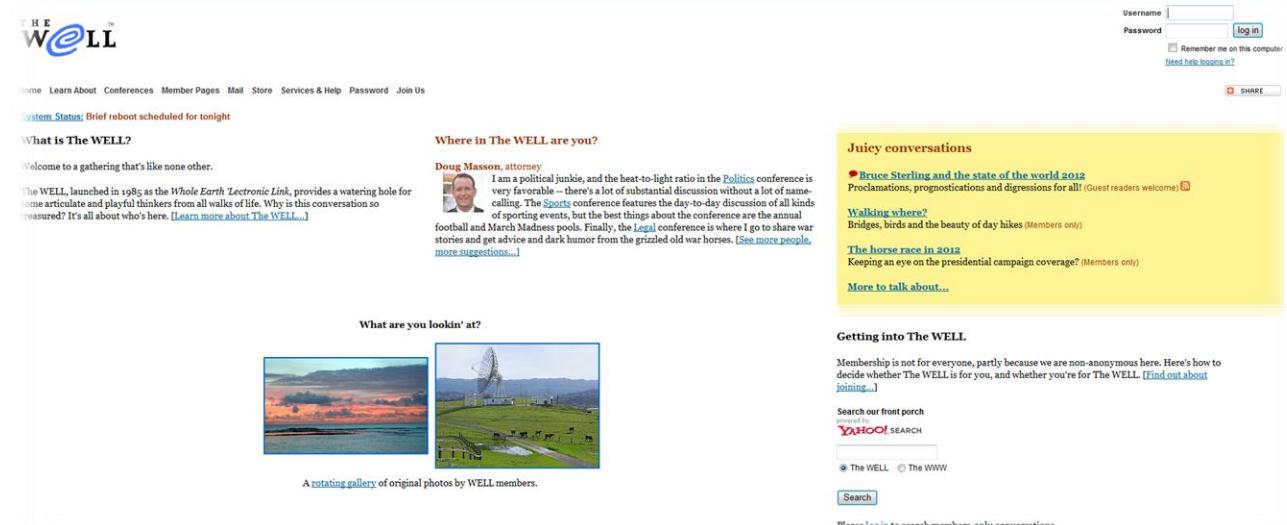
Koja je najstarija društvena mreža i kada je osnovana?



Evolucija društvenih mreža zasnovanih na Internetu (1)

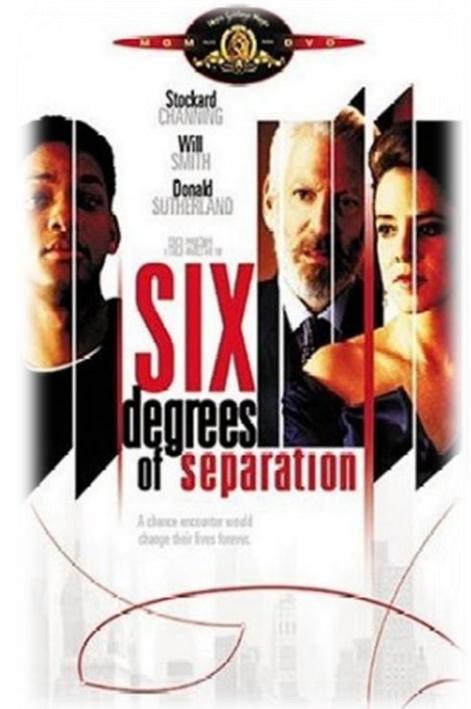


Evolucija društvenih mreža zasnovanih na Internetu (2): Društvene mreže prije WWW



Evolucija društvenih mreža zasnovanih na Internetu (3): Prva društvena mreža u razdoblju Web-a 1.0

- SixDegrees.com (1997 - 2000)
- Teorija „six degrees of separation”
- Profili, prijatelji, liste prijatelja



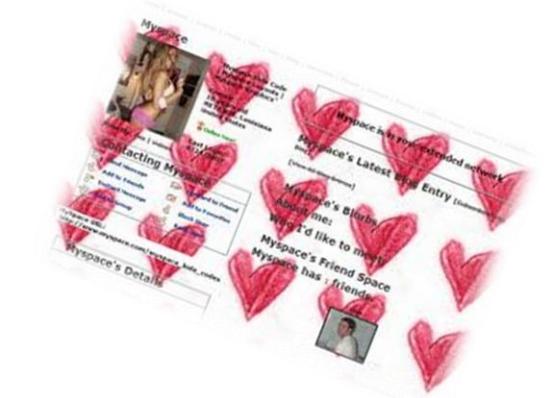
Evolucija društvenih mreža zasnovanih na Internetu (4): Društvene mreže u razdoblju Web-a 2.0

- 4 kategorije:
 - **Zabava**
 - Spajanje s ljudima radi druženja i zabave
 - SixDegrees.com, Friendster
 - **Posao**
 - Širenje poslovne mreže
 - LinkedIn, Plaxo, VisiblePath, Xing
 - **Razmjena sadržaja**
 - Djeljenje Web sadržaja sa ostalim korisnicima
 - Flickr, Last.FM, Youtube
 - **Mreže specifične domene**
 - Spajanje korisnika sličnih interesa
 - Dogster, Catster, Care2, MyChurch



Evolucija društvenih mreža zasnovanih na Internetu (6): MySpace (2003.)

- „Efekt mreže“ osam mjeseci nakon lansiranja
- Ime „MySpace“ proizlazi od opcije korisničkog uređivanja HTML koda
- Više od 80 milijuna korisnika u 2006.
- Veliki sigurnosni problemi



Evolucija društvenih mreža zasnovanih na Internetu (7): Facebook (2004.)

- 2004. Mark Zuckerberg skupa sa tri kolege lansira web stranicu Thefacebook.com
- Registracija zahtjeva harvard.edu e-mail domenu, ostali korisnici mogli su sudjelovati tek u kasnijim fazama
- U rujnu 2005. Thefacebook mjenja ime u Facebook



Evolucija društvenih mreža zasnovanih na Internetu (8): Facebook vs MySpace (>100M korisnika)

	facebook	myspace.com. a place for friends
Verifikacija korisnika	Ima verifikaciju	Nema verifikaciju
Baza korisnika	Vrhunski studenti u SAD-u	SAD
Dostupnost	Ekskluzivna mreža ("urban elite")	Dostupno svima
Reklame	Nema reklama	Puno reklama
Dizajn	Moderno	Kaotično

Evolucija društvenih mreža zasnovanih na Internetu (9)



VS.



- Facebook vs. MySpace (otvorena vs. zatvorena platforma)
 - MySpace nema otvoren API
 - Facebook ima otvoren API
 - Nezavisne kompanije razvijaju Facebook aplikacije
 - Slide, RockYou, zynga

slide®

RockYou!

zynga

Comment 1
Like 52
Tweet 6
Share 2
+1 1

Facebook Launches Facebook Platform; They are the Anti-MySpace

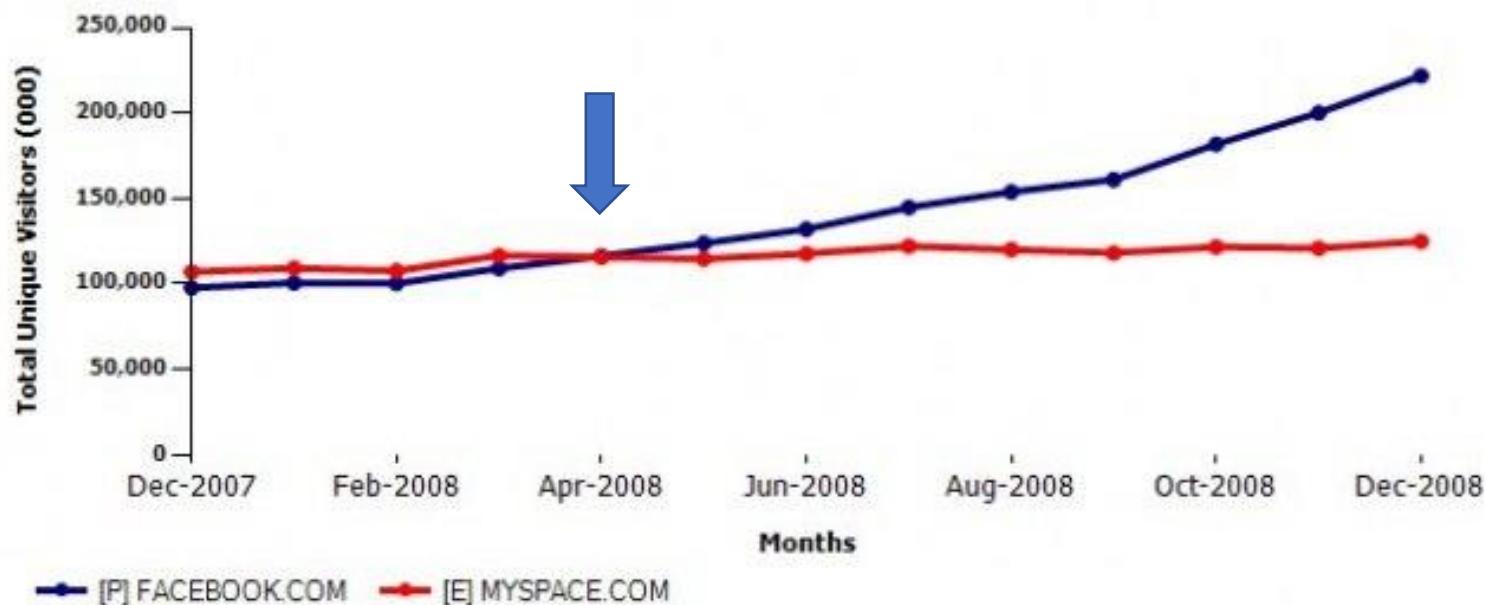
MICHAEL ARRINGTON Thursday, May 24th, 2007 1 Comments

Facebook is holding a massive press/developer event today in San Francisco to officially launch Facebook Platform. 750 or so people are here.

A number of third party applications will also be announced, including Microsoft, Amazon, Slide, RockYou, Box.net, Red Bull, Washington Post, Project Aaran, Prosser, Snanvine, like, PicksPal, Dinn, Plum and others. Seventy



Evolucija društvenih mreža zasnovanih na Internetu (10)



Trendovi i statistika

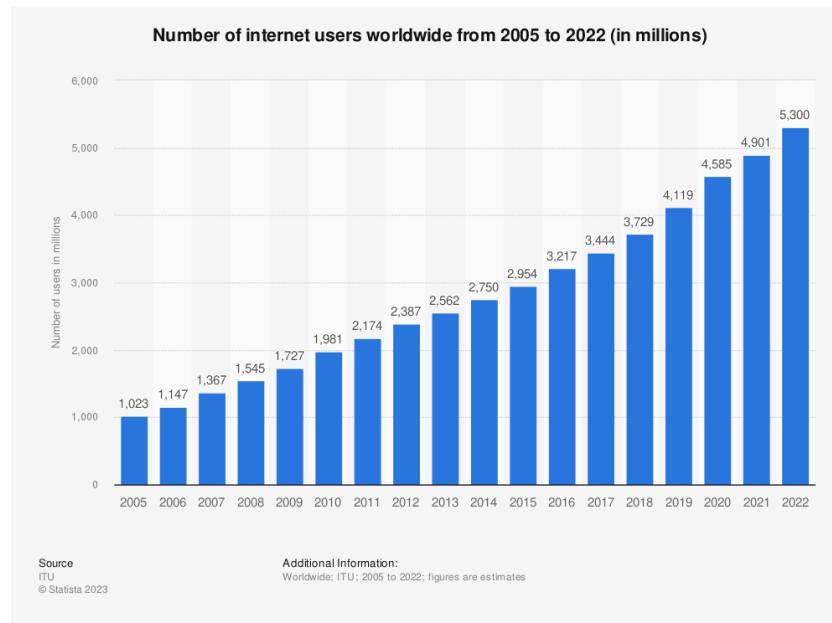
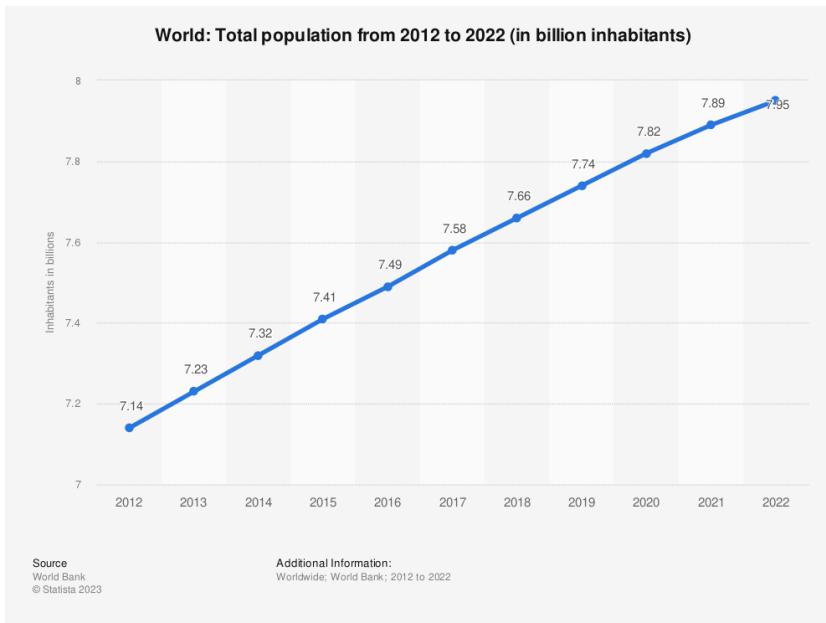
Ljudi vs korisnici Interneta



Ljudi vs korisnici Interneta



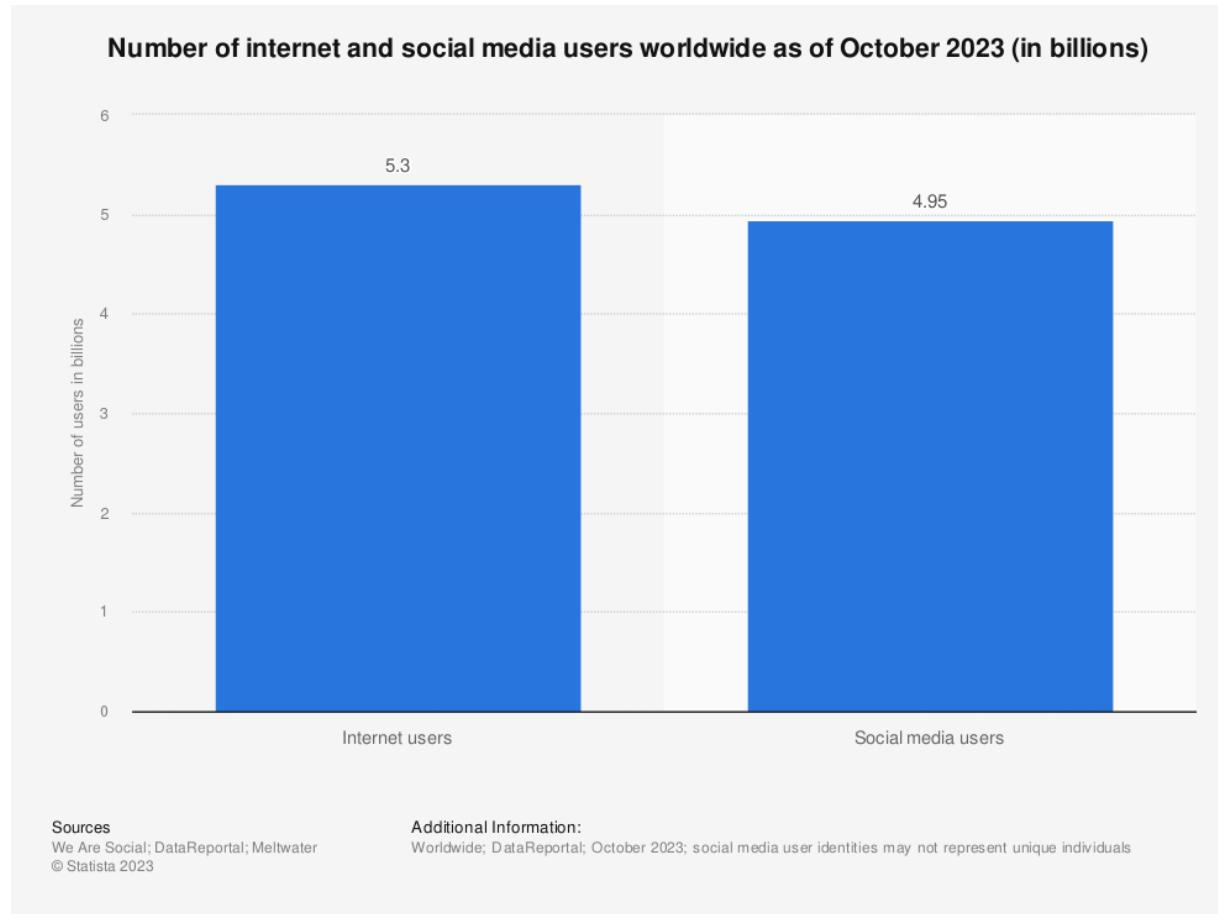
67%



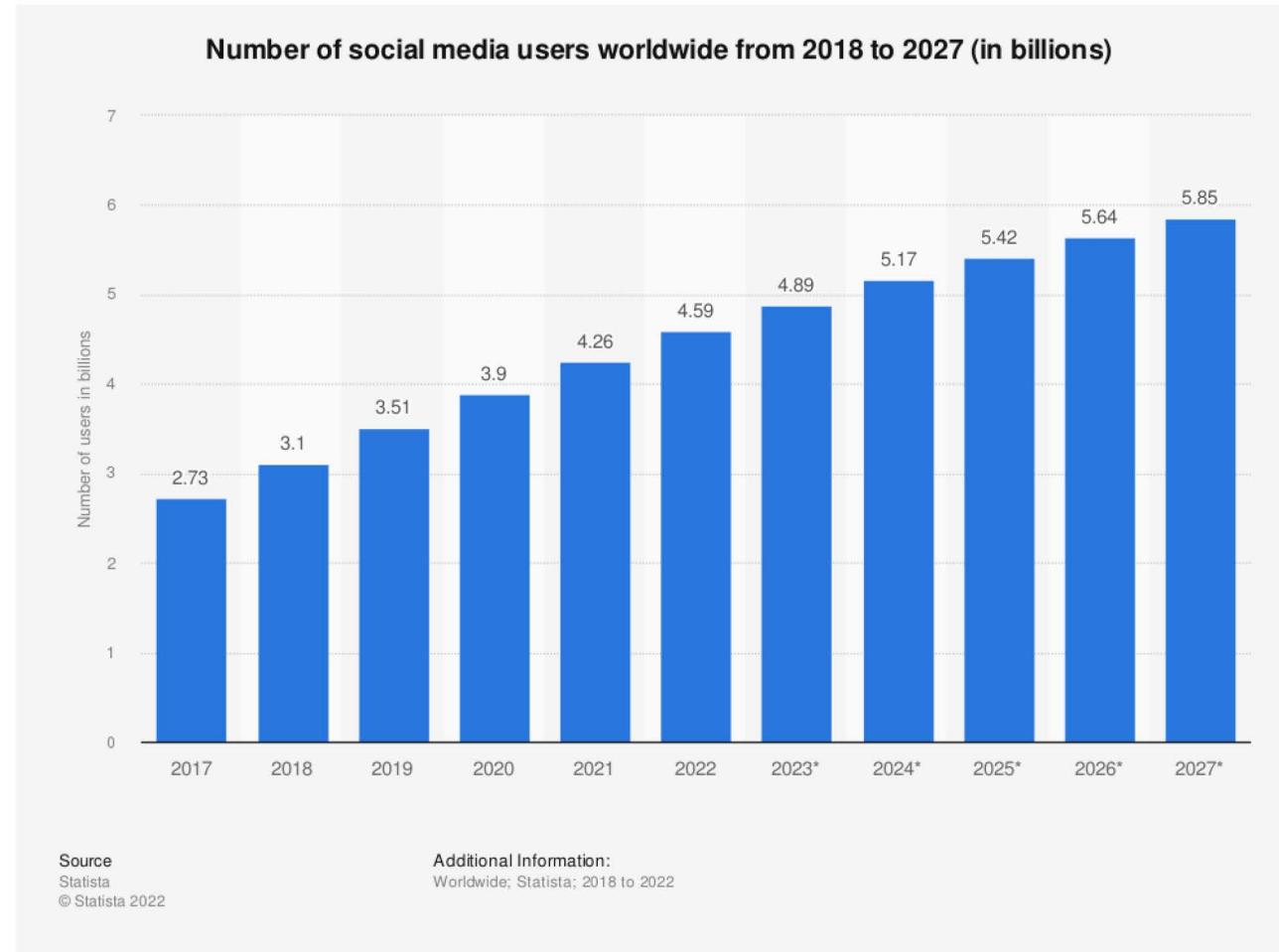
Korisnici Interneta vs korisnici društvenih mreža



Korisnici Interneta vs korisnici društvenih mreža



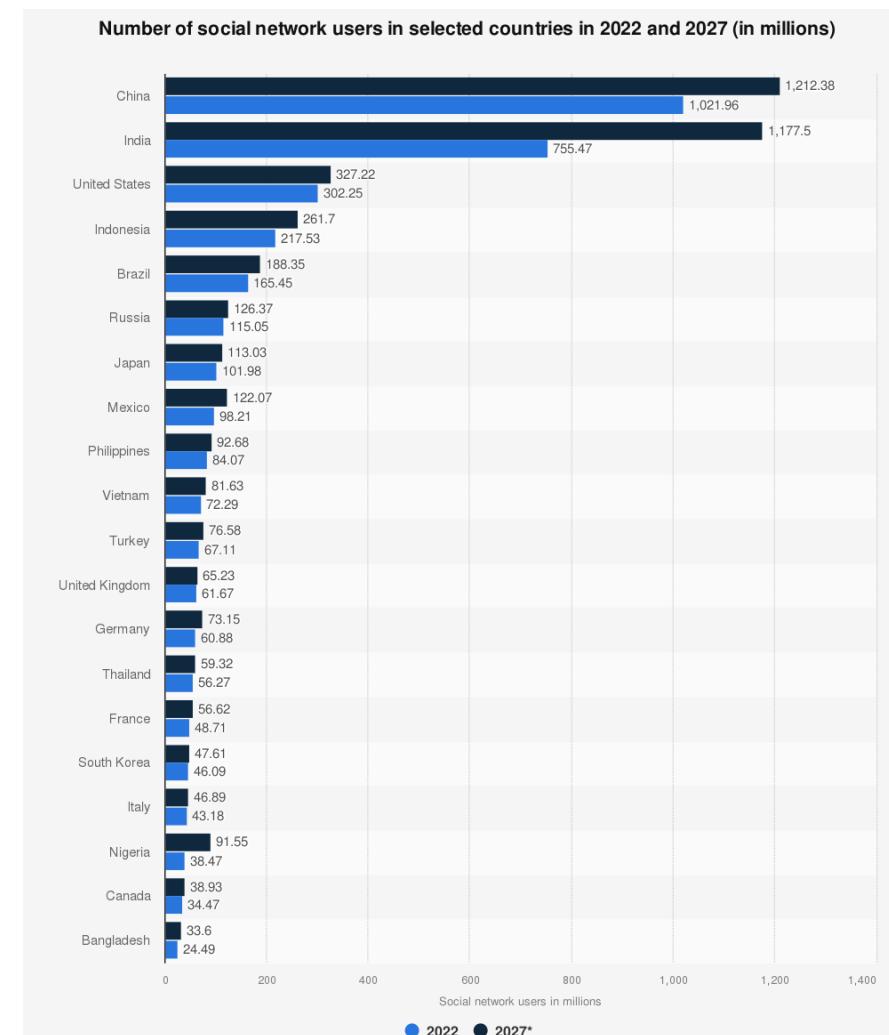
Projekcija rasta broja korisnika društvenih mreža



Distribucija korisnika društvenih mreža po državama (apsolutni broj) – top 3 zemlje



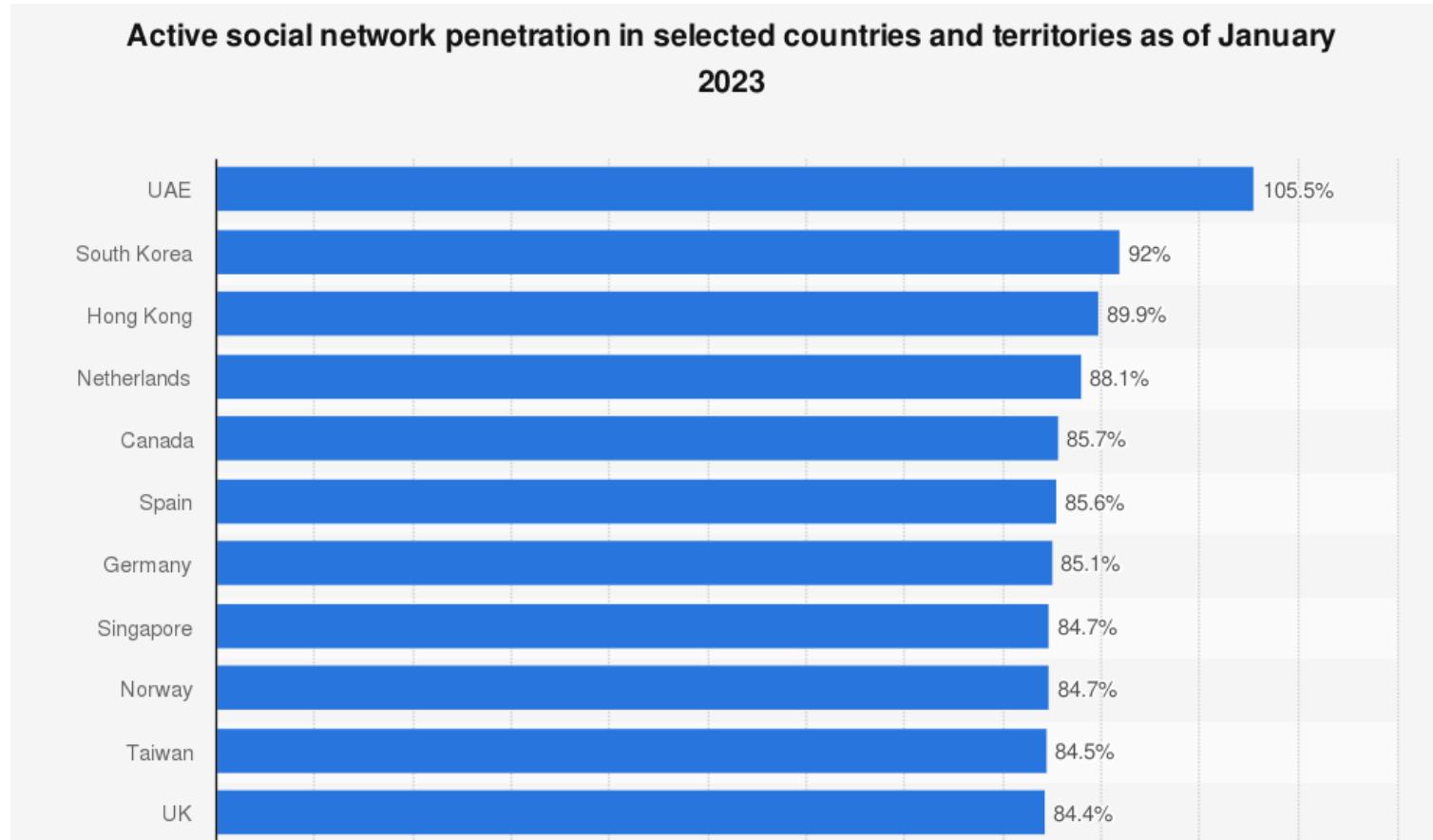
Distribucija korisnika društvenih mreža po državama (apsolutni broj)



Source
Statista
© Statista 2022

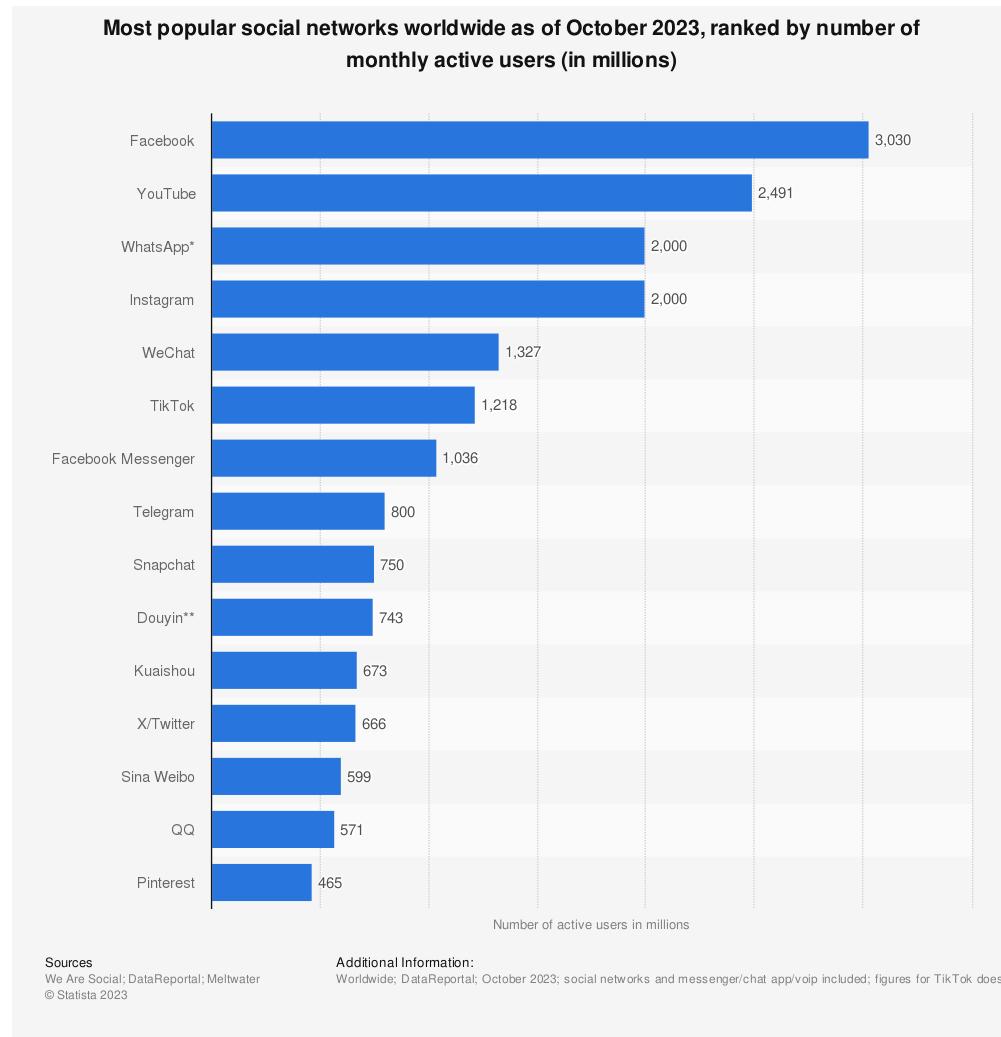
Additional Information:
Worldwide; Statista; 2022; internet users who use a social network site at least once a month

Distribucija korisnika društvenih mreža po državama (relativno vs populacija)



WW=59%

Najpopularnije društvene mreže



LinkedIn
310M

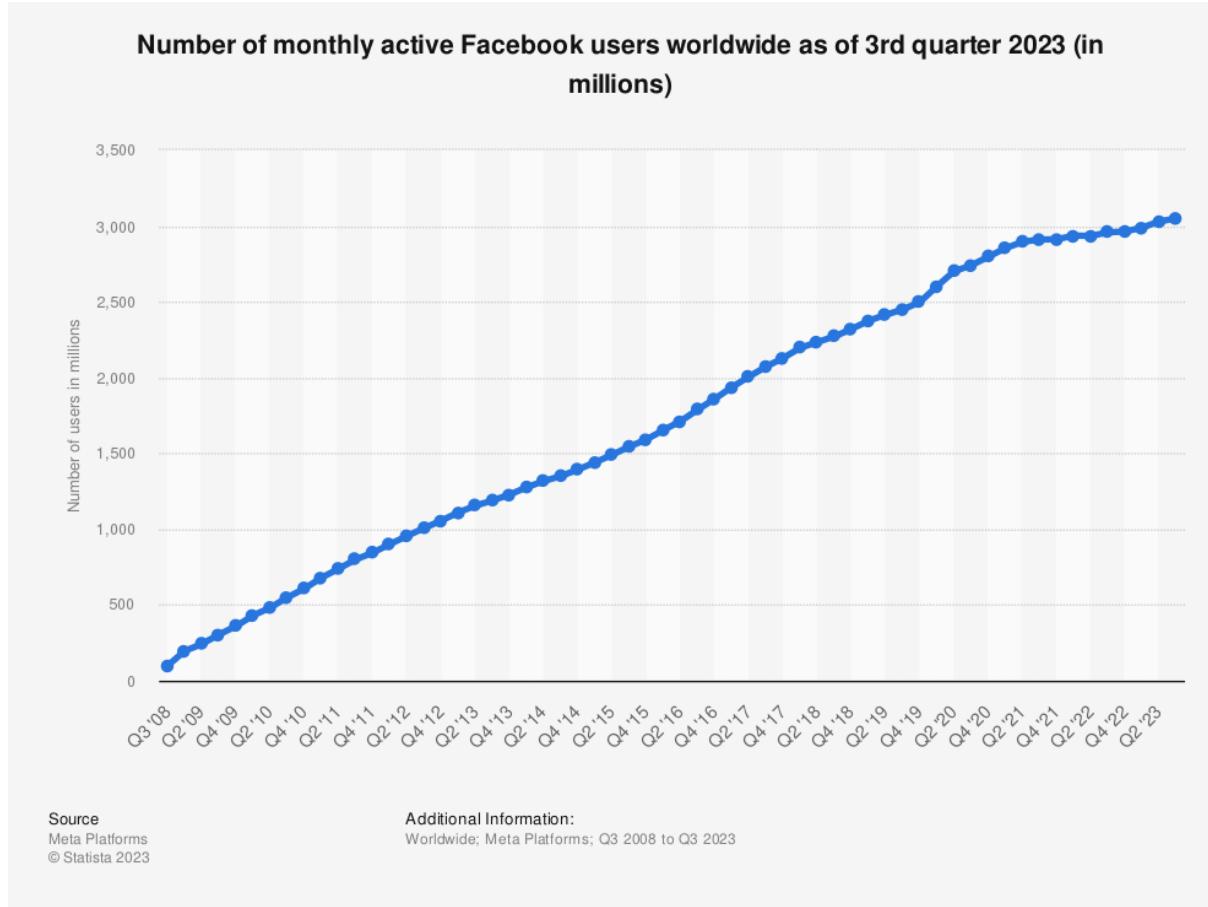


Facebook – broj korisnika WW & Hrvatska





Facebook rast broja korisnika (mjesечно aktivni)



CRO
1,9M

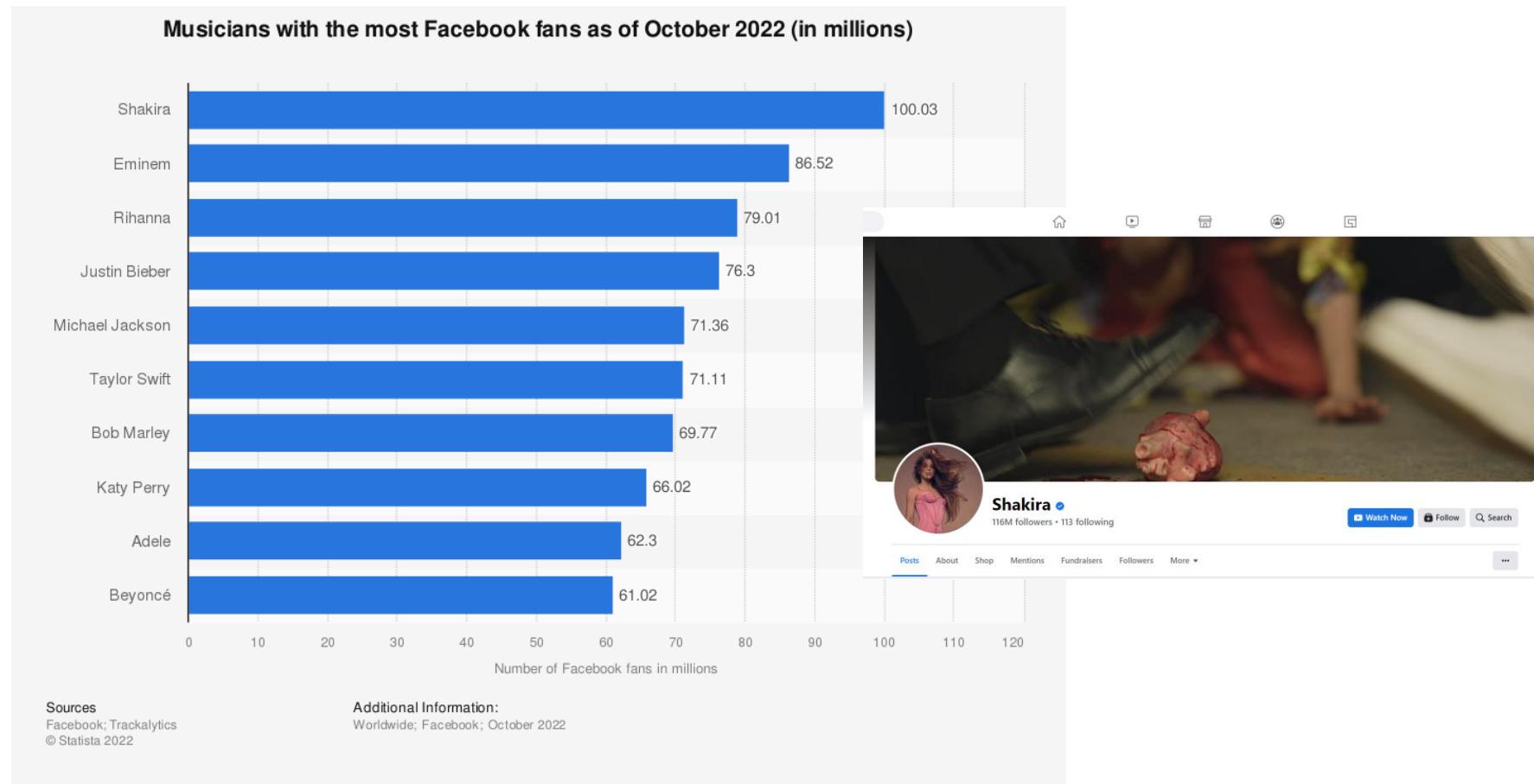


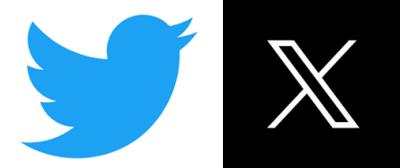
Facebook – najpopularnija umjetnica





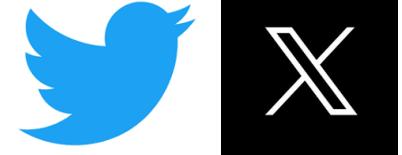
Facebook – najpopularnija umjetnica



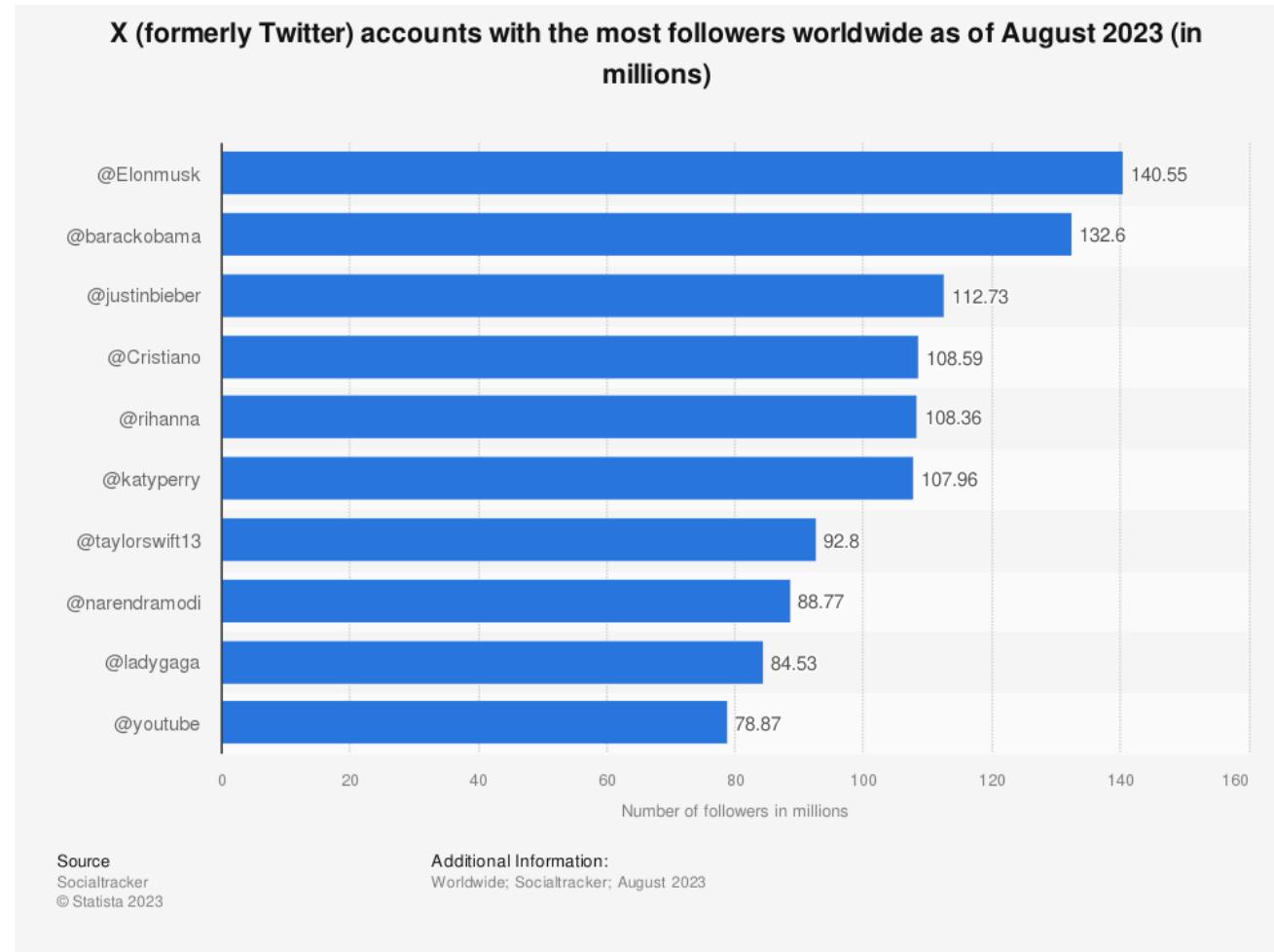


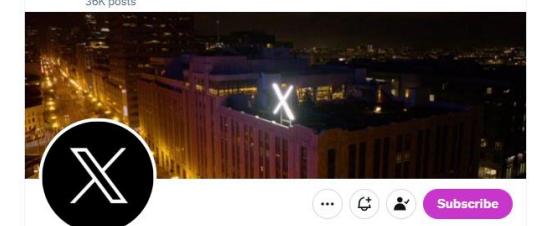
Twitter / X – najpopularniji profil





Twitter / X – najpopularniji profili

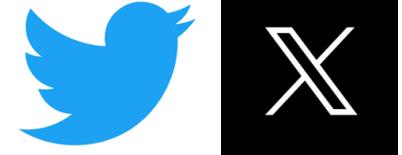




Elon Musk verified X
@elonmusk
(CTO) Chief Troll Officer
Joined June 2009
516 Following 168.6M Followers 139 Subscriptions



Barack Obama verified X
@BarackObama
Dad, husband, President, citizen.
Joined March 2007
546.2K Following 131.9M Followers



Twitter / X & Elon Musk (love & hate)

2022.



Elon Musk  19.7K Tweets

Elon Musk  @elonmusk

Perfume Salesman

⌚ Boring  Joined June 2009

123 Following 109.1M Followers

 Followed by Nastaran Naseri, Angler, and 580 others you follow

2024.



Elon Musk  X 36K posts

Elon Musk  @elonmusk

(CTO) Chief Troll Officer

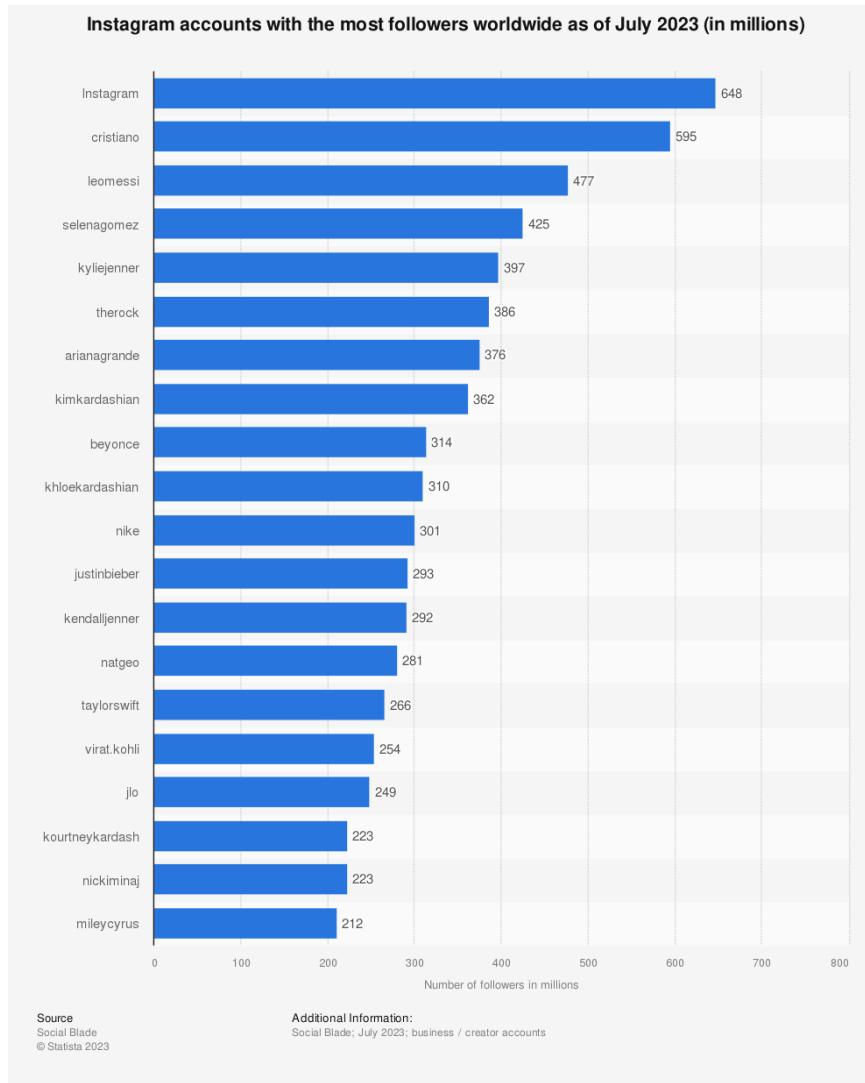
⌚ Tröllheim  Joined June 2009

516 Following 168.6M Followers 139 Subscriptions

\$44B



Instagam: Najpopularniji Instagram računi



cristiano Follow Message ...
3,607 posts 617M followers 579 following
Cristiano Ronaldo
Join my NFT journey on @Binance. Click the link below to get started.
ter.li/CRTForeverZone
Followed by mhdsdj, foxsports, roccolibroccoli + 21 more

leomessi Follow Message ...
1,156 posts 497M followers 308 following
Leo Messi
Sportsperson
Bienvenidos a la cuenta oficial de Instagram de Leo Messi / Welcome to the official Leo Messi Instagram account
themesslexperience.com
Followed by sportsillustrated, lidijabacic_lile, foxsports + 6 more

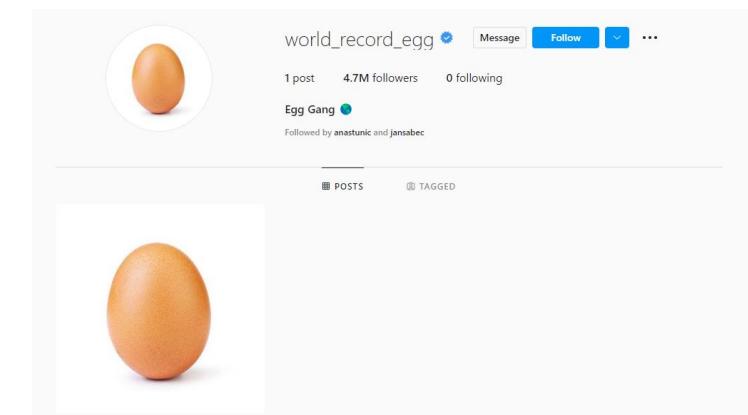
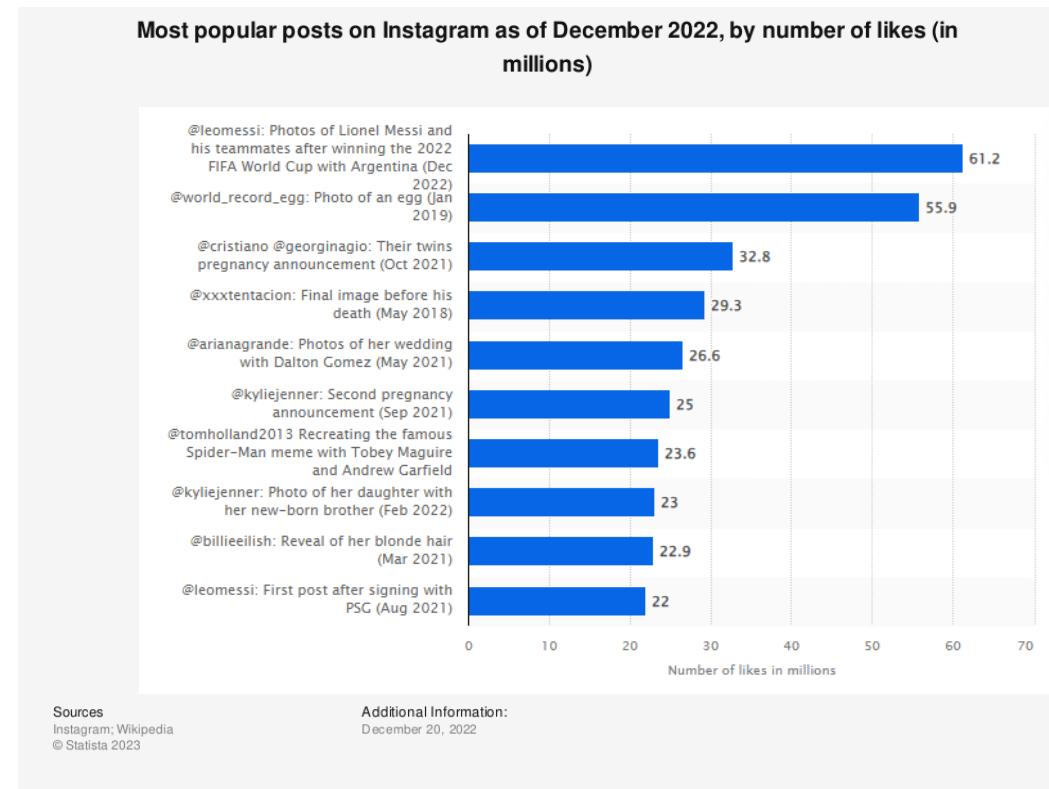
Selección FCB Familia

POSTS REELS TAGGED

POSTS REELS TAGGED



Instagram: Najpopularniji Instagram postovi



Analiza (podataka) društvenih mreža

Reprezentacija društvenih mreža: graf vs matrica

- Grafička reprezentacija
 - Koristi se za **prikaz interakcija individualaca** u društvenoj mreži
 - Ključno za razumijevanje povezanosti informacija, inovacija i pojava bolesti

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$



- Grane grafa mogu označavati
 - Društvenu interakciju, organizacijsku strukturu, fizičku bliskost, ili apstraktnije interakcije poput sličnosti

Analiza podataka (1)

- Pojava globalnih zajednica temeljenih na Webu
 - **Interes** za proučavanje ljudskih društvenih mreža
 - Ljudi iz cijelog svijeta izravno su ili neizravno povezani popularnim društvenim mrežama kao što su Facebook i Twitter/X
- Društvene mreže – moćan medij
 - **Razmjena pogleda i utjecaj na druge**
 - Brzo rastu
 - **Automatizirane** metode analize
- Proučavanje društvenih zajednica **izuzetno je multidisciplinarno** i zahtijeva stručnost od:
 - Računalne znanosti, socijologije, znanosti o ponašanju, matematike, statistike, ...

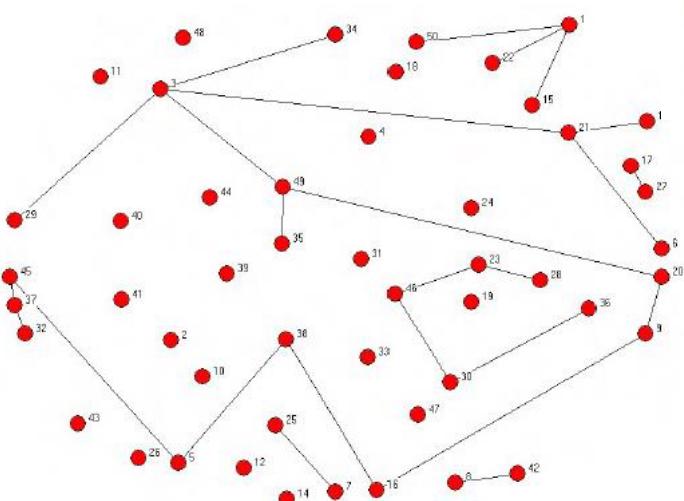


Analiza podataka (2)

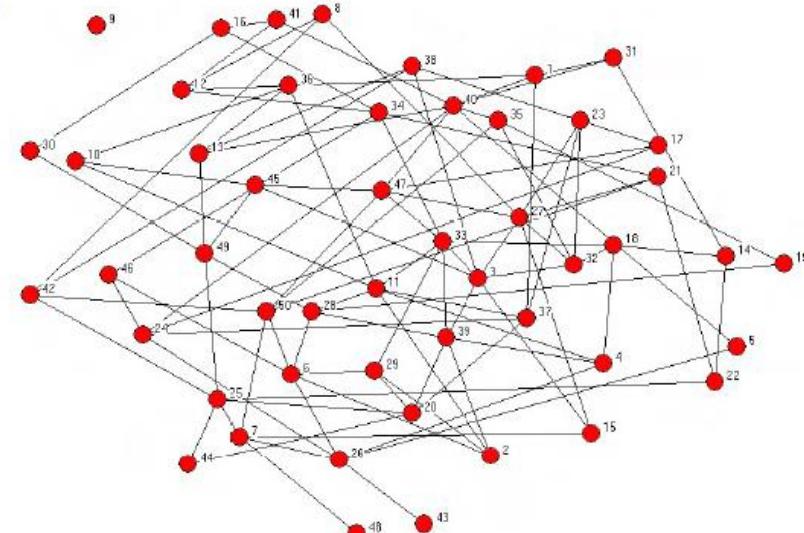
- Tehnike podatkovne analize
 - Primijenjene za stjecanje vrijednih otkrića s raširenim **društvenim i ekonomskim utjecajima** u mnogim područjima primjenjenih istraživanja društvenih mreža
- Iz društvene perspektive, područja primjene uključuju:
 - Antiterorističke strategije,
 - Analiza podataka o kriminalu,
 - Biomarkeri bolesti,
 - Hobi,
 - Obiteljski odnosi,
 - Društvene funkcije,
 - Zanimanja i
 - Prijateljstvo
- Iz ekonomske perspektive, analize mogu dovesti do određenih:
 - Ciljnih skupina kupaca,
 - Razvoja lijekova
 - ...



Načini formiranja društvenih mreža (1)

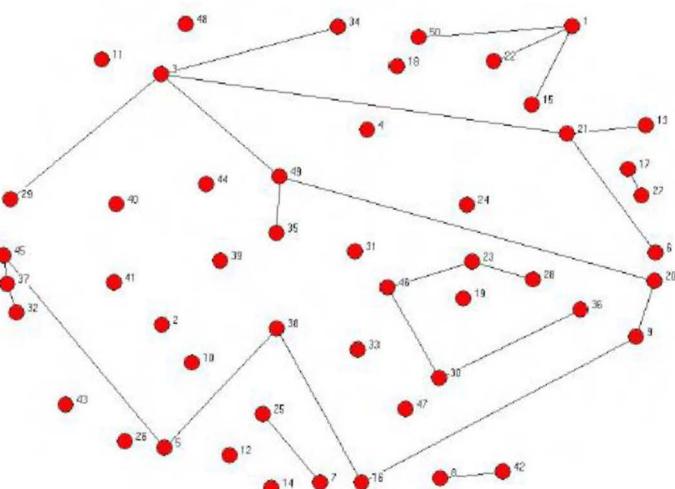


VS



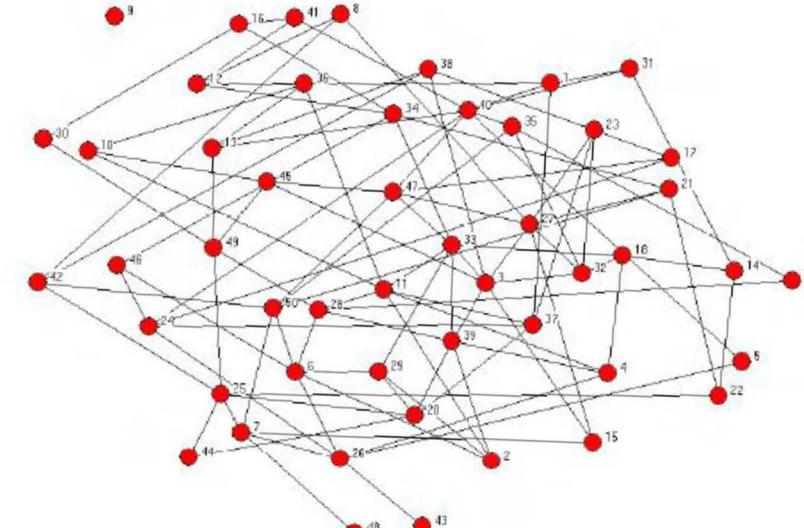
Načini formiranja društvenih mreža (2)

random, $p=0.02$

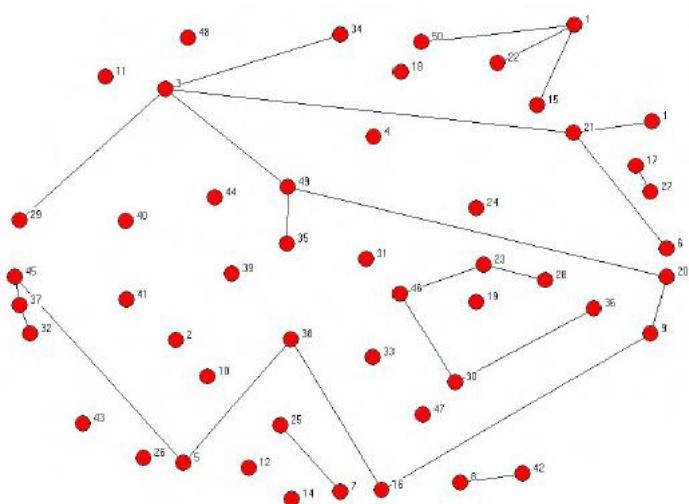


VS

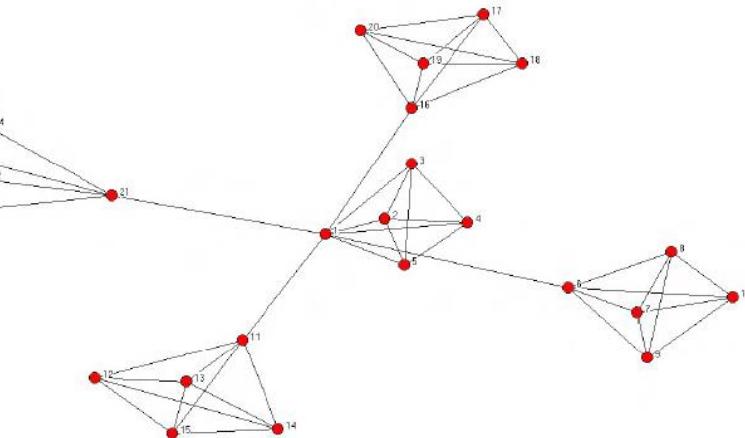
random, $p=0.08$



Načini formiranja društvenih mreža (3)

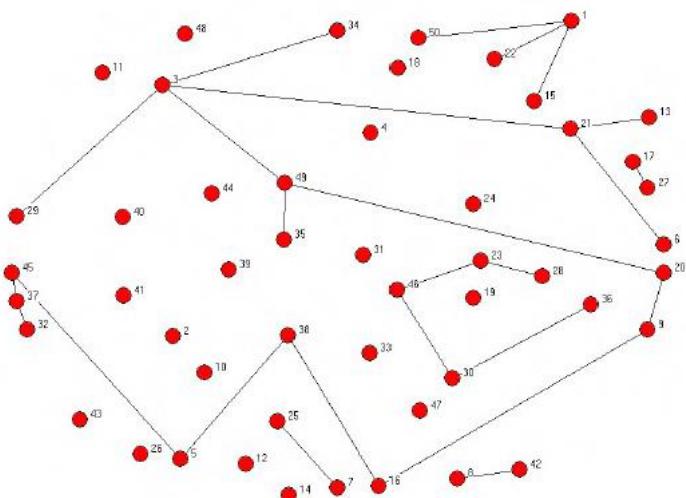


VS

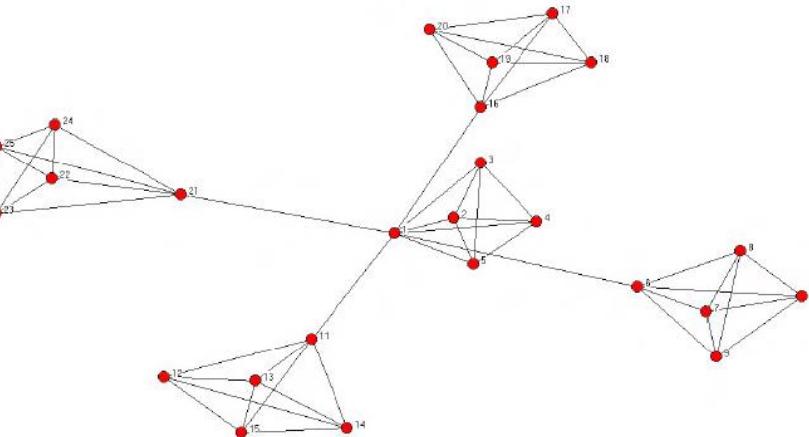


Načini formiranja društvenih mreža (4)

random



small world



vs

Analiza društvenih mreža (metode, metrike, alati)

