# Network representations, basic network algorithms

You are given four networks in Pajek format that was presented in lectures.

- Tiny toy network for testing (toy.net)
- Zachary karate club network (karate_club.net)
- iMDB actors collaboration network (collaboration_imdb.net)
- A small part of Google web graph (www_google.net)

## I. Adjacency list representation

1. **(code)** Assume that all networks are undirected. Implement your own adjacency list representation of the networks as an array of lists and represent all four networks.

2. **(discuss)** Now, assume that all networks are directed. How would you extend your network representation?

3. **(discuss)** Does your network representation allow for multiple links between the nodes, loops on nodes and isolated nodes?

## II. Basic network statistics

1. **(code)** Compute basic statistics of all four networks. Namely, the number of nodes $n$ and links $m$, the average node degree $\langle k \rangle = 2m/n$ and the undirected density $\rho = m/\binom{n}{2}$. Are the results expected?

2. **(code)** Compute the number of isolated nodes and the number of pendant nodes (i.e. degree-$1$ nodes), and the maximum node degree $k_{\max}$. How do the values of $k_{\max}$ compare to $\langle k \rangle$?

3. **(discuss)** What is the time complexity of the computations above?

## III. Network connected components

1. **(discuss)** Study the following algorithm for computing (weakly) connected components $\{C\}$ by any order link traversal. Does the algorithm implement breadth-first or depth-first search? Why? What is the time complexity of the algorithm?

```
input   graph G, nodes N
output  network components {C}
{C} ← empty list
while not N empty do
    {C}.add(component(G, N, N.next()))
return {C}
```

```
input   graph G, nodes N, node i
output  weak component C
C ← empty list
S ← empty stack
N.remove(S.push(i))
while not S empty do
    C.add(i ← S.pop())
    for neighbors j ∈ Γ_i do
        if N.remove(j) then
            S.push(j)
return C
```

2. **(code)** Implement the algorithm and compute the number of (weakly) connected components $s$ and the size of the largest (weakly) connected component $S$ of all four networks. Are the results expected?

3. **(discuss)** How could you further improve the algorithm to *only* compute $s$ and $S$?