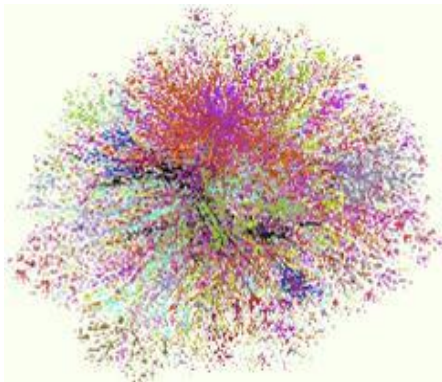


# COMP9311 Advanced Topics - Graph Data Analytics

# Why graphs ?

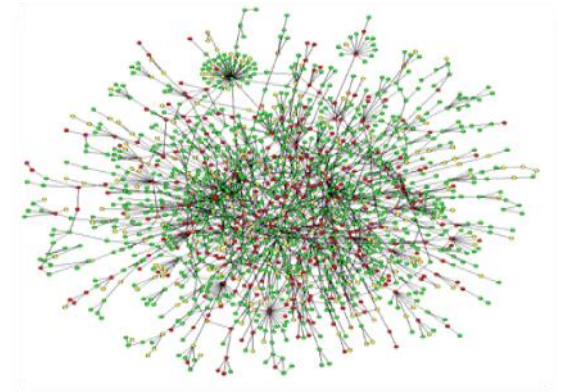
- Common model across different fields



Web Graph



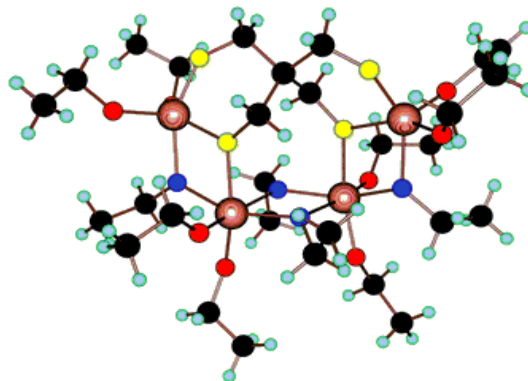
Social Network



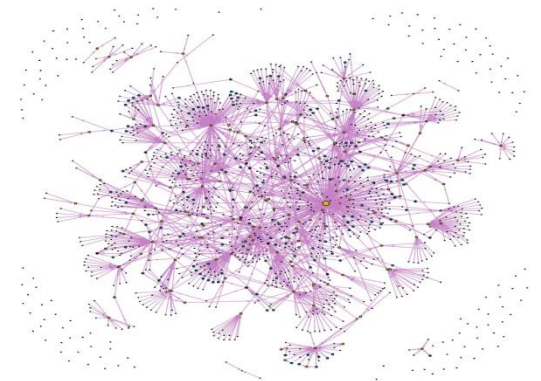
Protein Interaction Network



Road Network



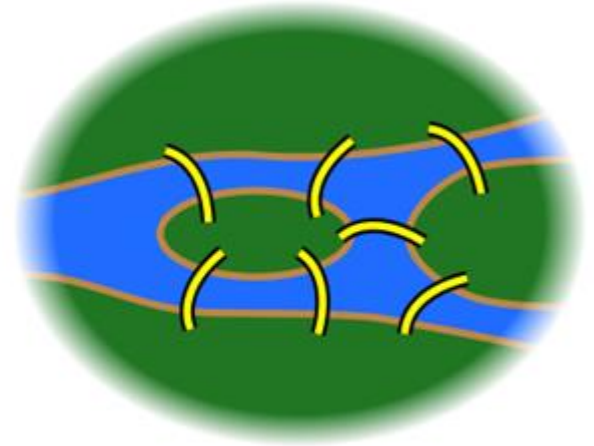
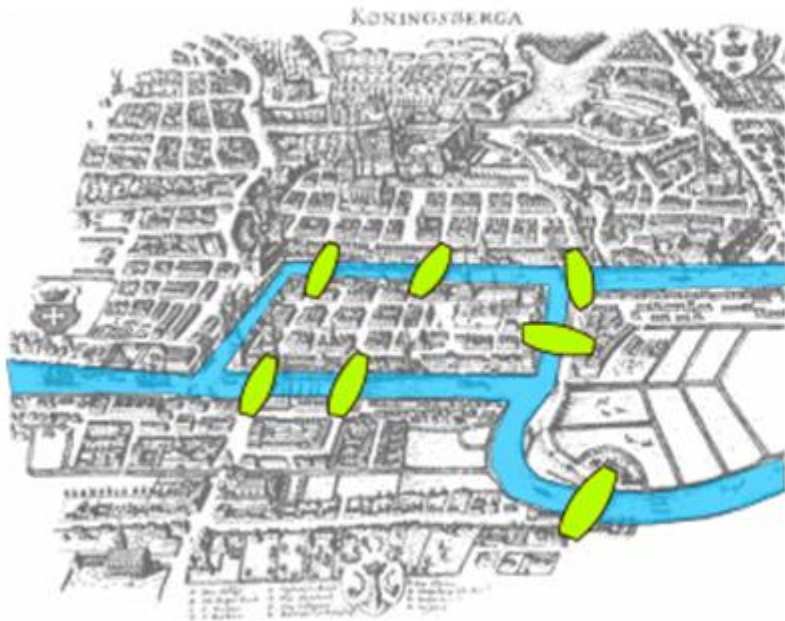
Chemical Compound



Ontology Graph

# History

- Seven Bridges of Königsberg (1736)



# Big Graphs

Google • 70+ billion facts in knowledge graphs in 2016



- 2+ billion active users in 2018
- 190 friends/user on average



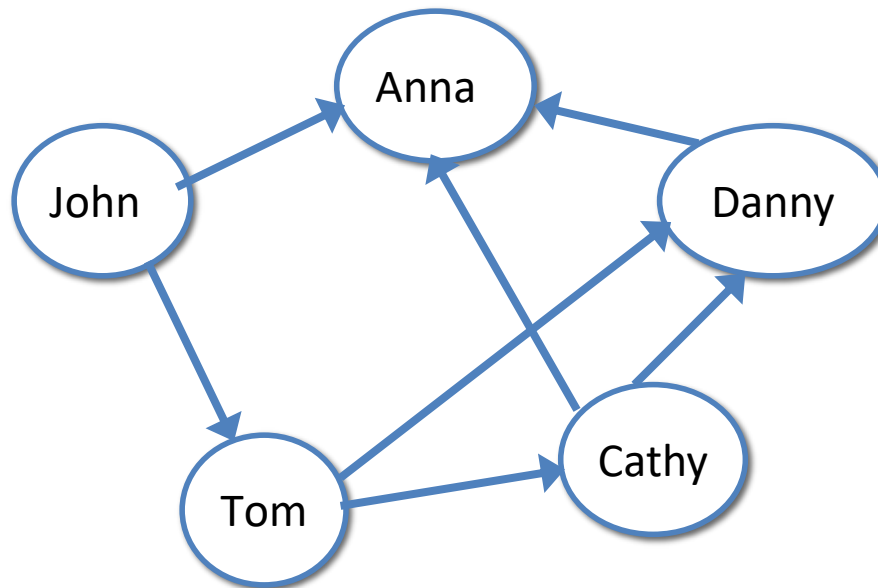
- 1.5+ billion users in 2018

# What's a Graph?

- $G = (V, E)$ , where
  - $V$  represents the set of vertices (nodes)
  - $E$  represents the set of edges (links)
  - Both vertices and edges may contain additional information
- Different types of graphs:
  - Directed vs. undirected edges
  - Presence or absence of cycles
- Graphs are everywhere:
  - Hyperlink structure of the Web
  - Physical structure of computers on the Internet
  - Interstate highway system
  - Social networks

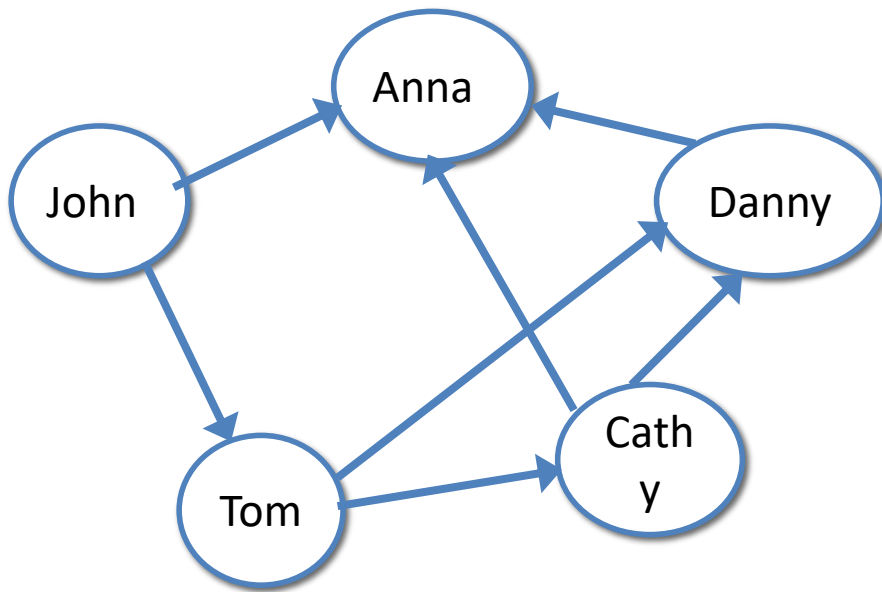
# What's a Graph?

- $G = (V, E)$ , where
  - $V$  represents the set of vertices (entities)
  - $E$  represents the set of edges (relations)
  - Both vertices and edges may contain additional information

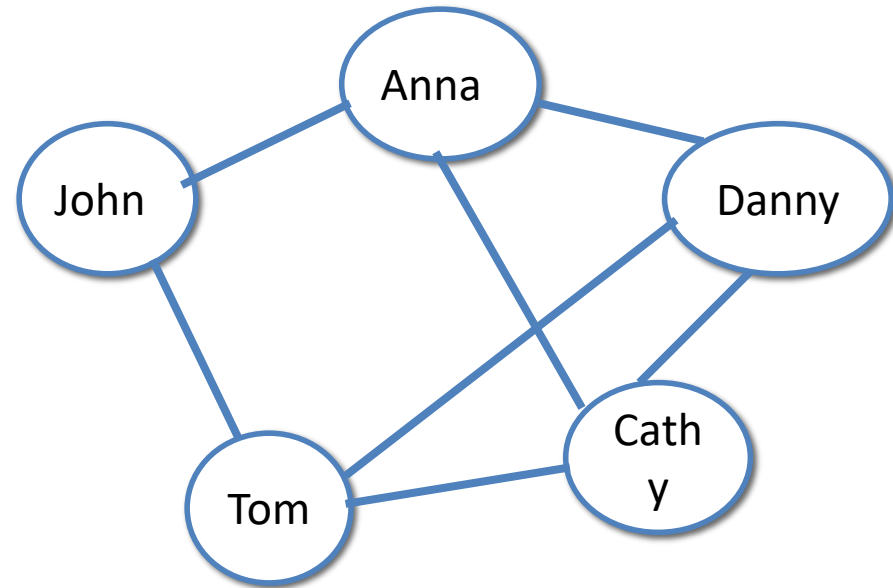


A twitter network

# Directed vs Undirected



A twitter network



A Facebook network

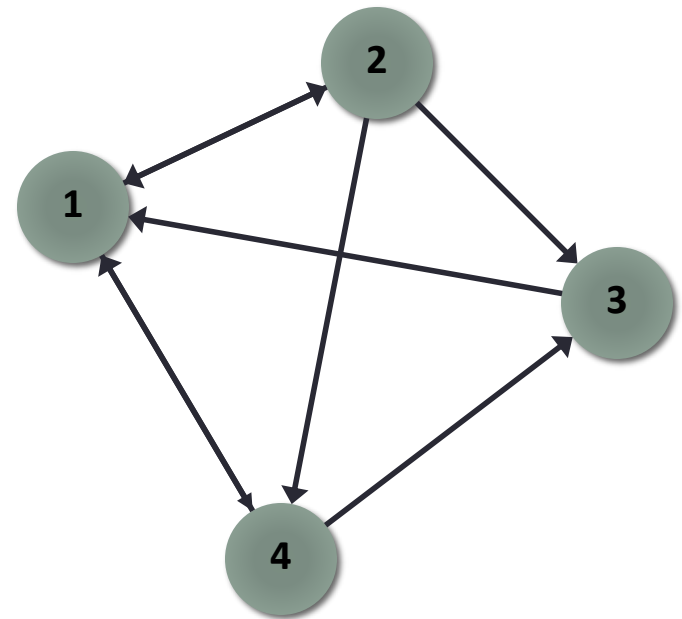
# Representing Graphs

- **Adjacency Matrices:** Represent a graph as an  $n \times n$  square matrix  $M$

–  $n = |V|$

–  $M_{ij} = 1$  means a edge from vertex  $i$  to  $j$

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



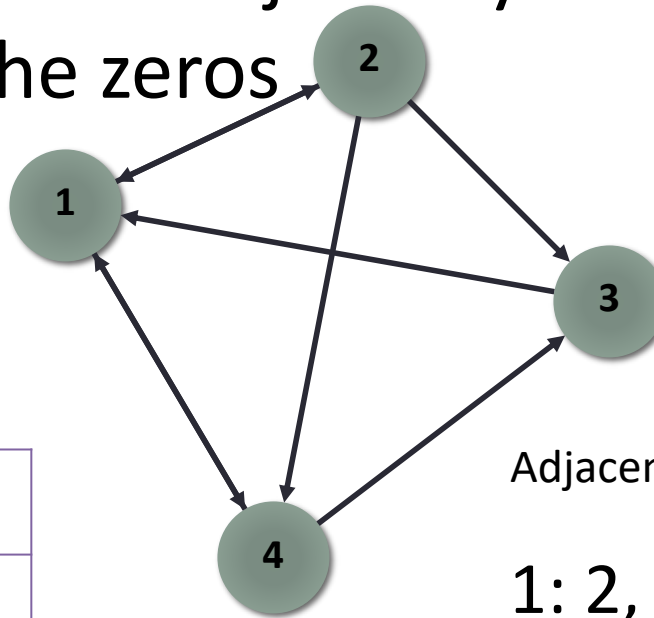


# Representing Graphs

- Adjacency Lists: Take adjacency matrices... and throw away all the zeros

Adjacency Matrix

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



Adjacency List

1: 2, 4

2: 1, 3, 4

3: 1

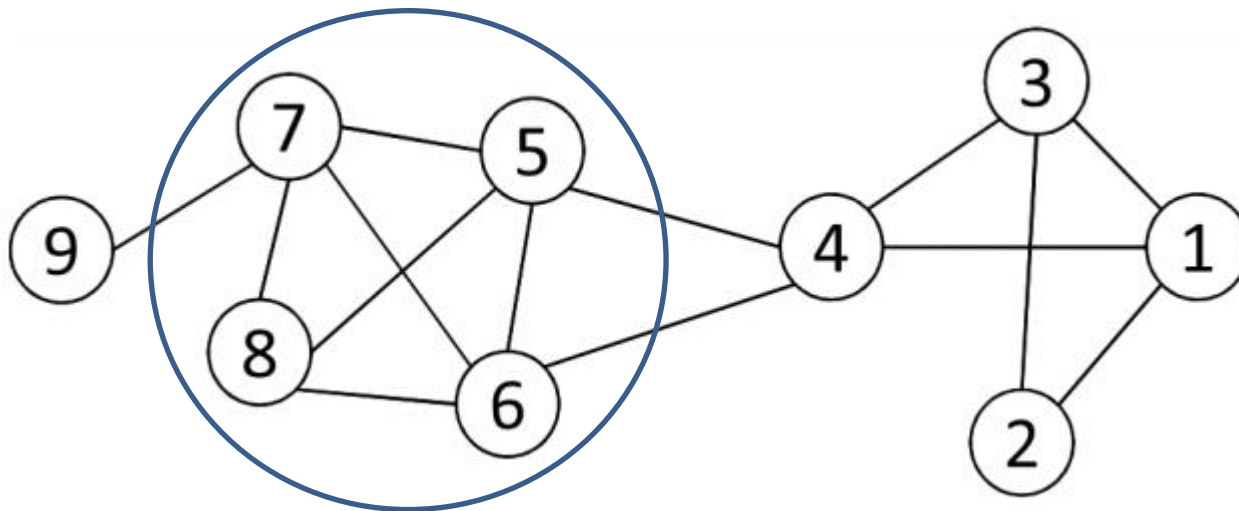
4: 1, 3



# Structural Analysis of Graphs

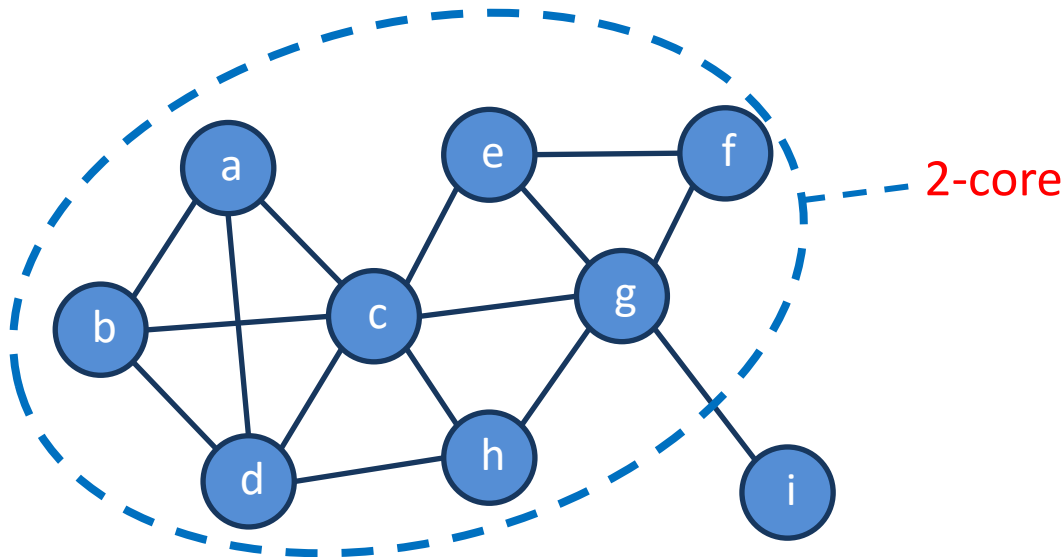
# Cliques

- Every pair of vertices pair is connected
- A clique is called maximal clique if there exist no other bigger cliques that contain it
- Also called complete graph



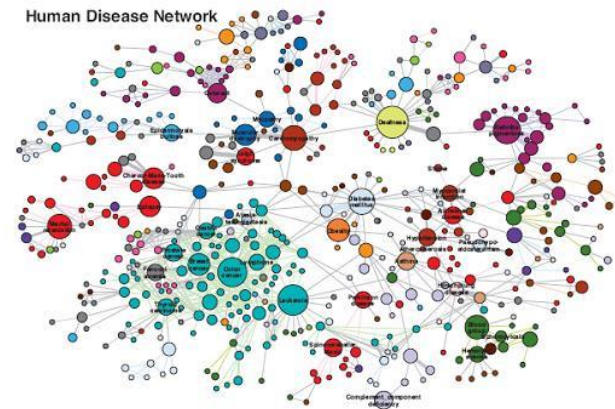
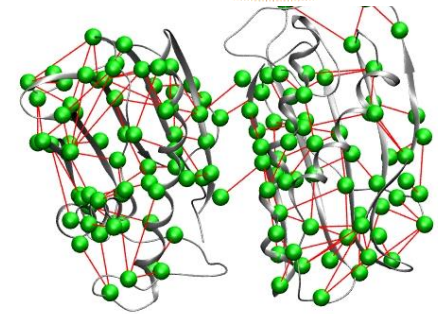
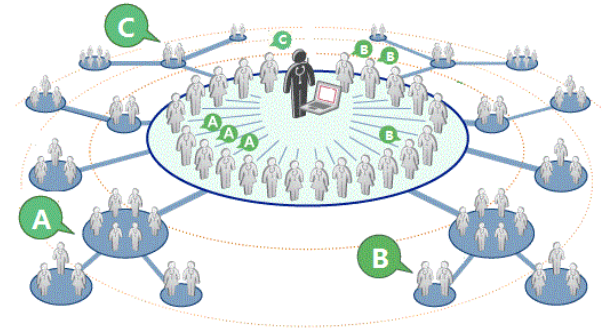
# $k$ -Core

- Given a graph  $G$ , the  $k$ -core of  $G$  is a subgraph where each node has at least  $k$  neighbors (i.e.,  $k$  adjacent nodes, or a degree of  $k$ ).



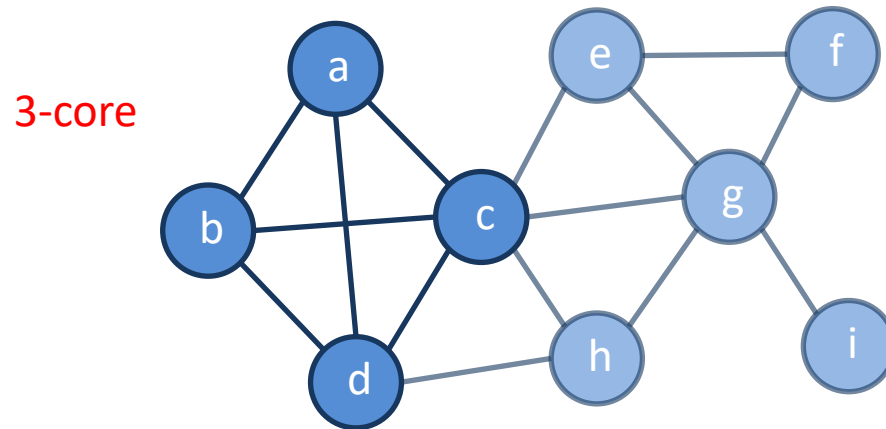
# Applications

- Community detection
- Social contagion
- User engagement
- Event detection
- Network analysis and visualization
- Influence study
- Graph clustering
- Protein function prediction
- Human Cerebral Cortex
- .....



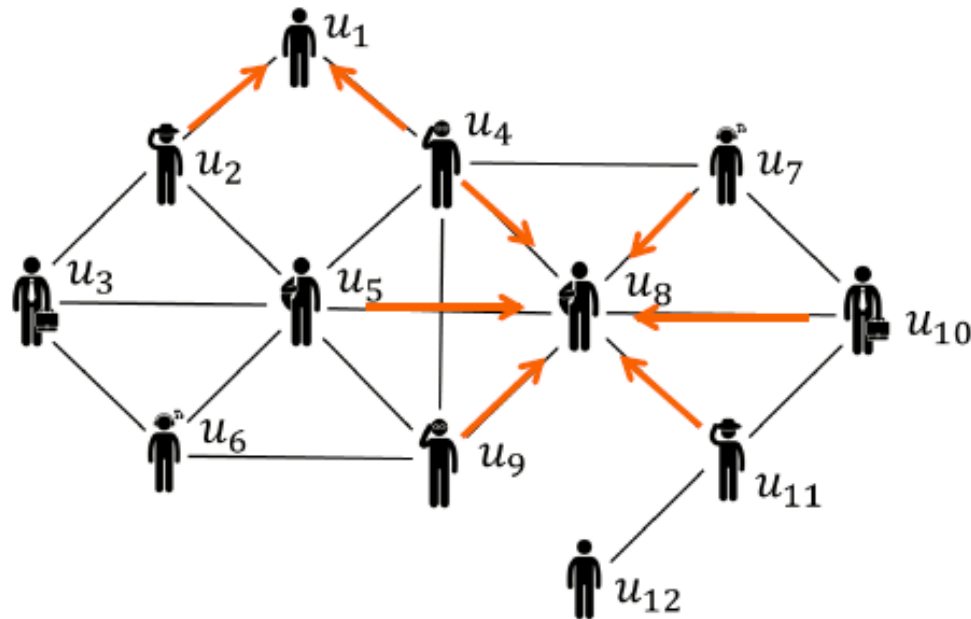
# Compute $k$ -Core

- Given a graph  $G$ , the  $k$ -core of  $G$  can be computed by recursively deleting every node and its adjacent edges if its degree is less than  $k$ .



# Why study k-core ?

The engagement of a user is influenced by the number of her engaged friends.

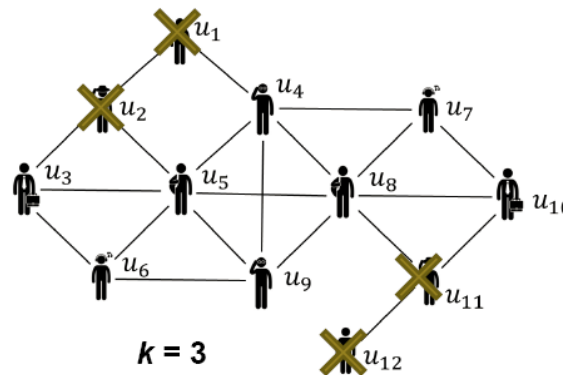


K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma. Preventing unraveling in social networks: the anchored k-core problem. *SIAM Journal on Discrete Mathematics*, 29(3):1452–1475, 2015.

# Why study $k$ -core ?

Assume a user will leave if less than  $k$  friends in the group

An equilibrium: a group has the minimum degree of  $k$ , namely  $k$ -core

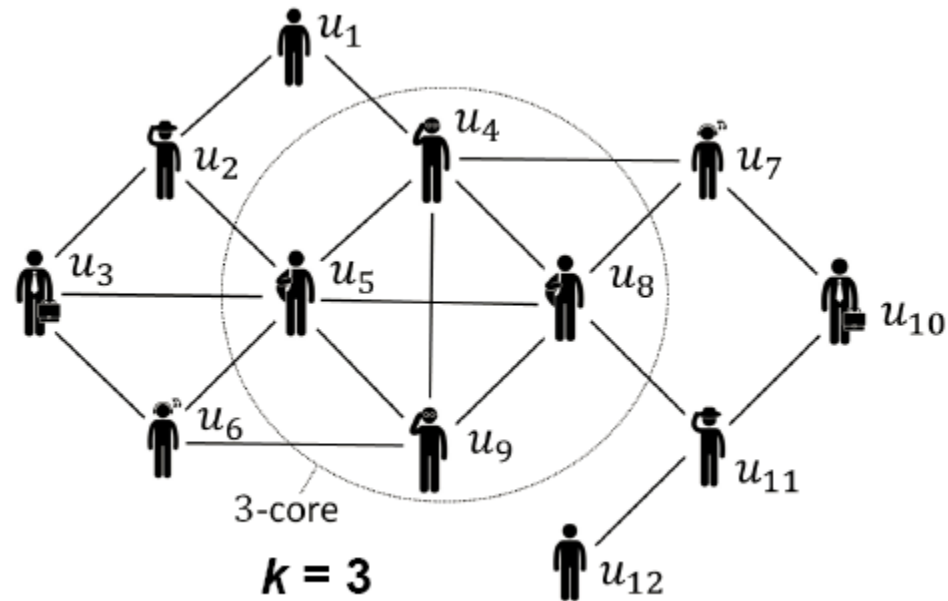


K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma. Preventing unraveling in social networks: the anchored  $k$ -core problem. *SIAM Journal on Discrete Mathematics*, 29(3):1452–1475, 2015.



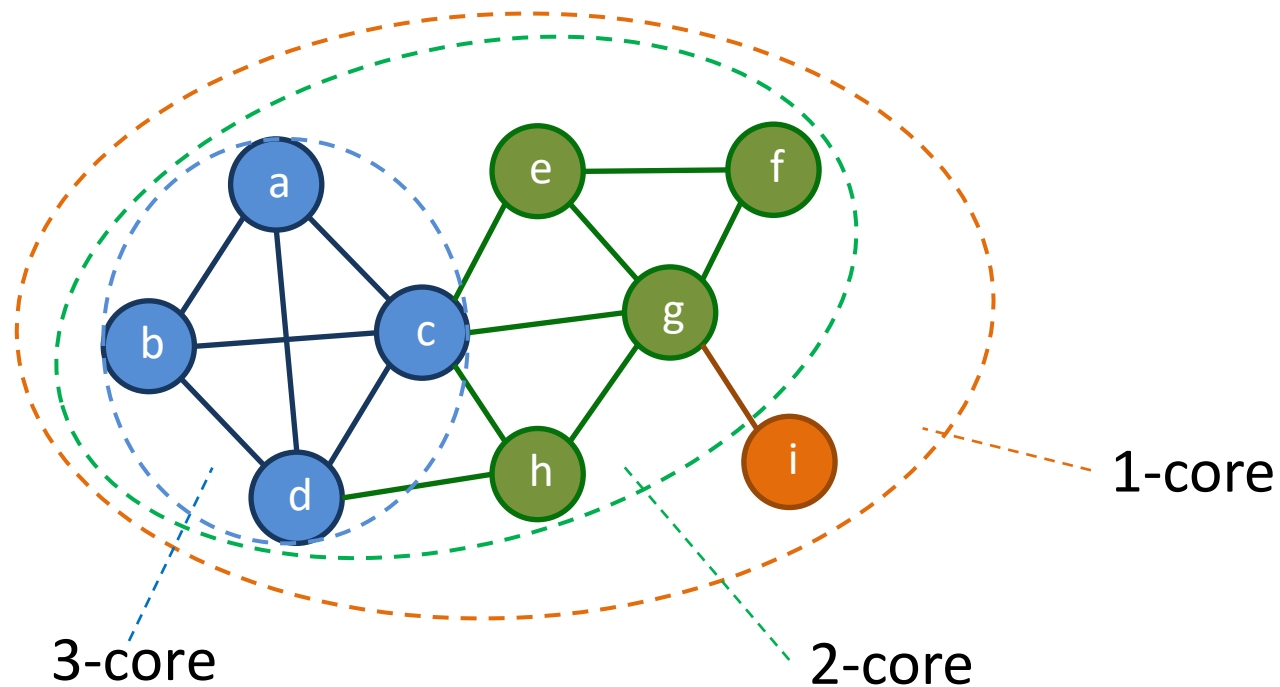
# Why study k-core ?

A stable social group tends to be a k-core in the network



# $k$ -Core Decomposition

- **Core number** of a node  $v$ : the largest value of  $k$  such that there is a  $k$ -core containing  $v$ .
- Core decomposition: compute the **core number** of each node in  $G$ .

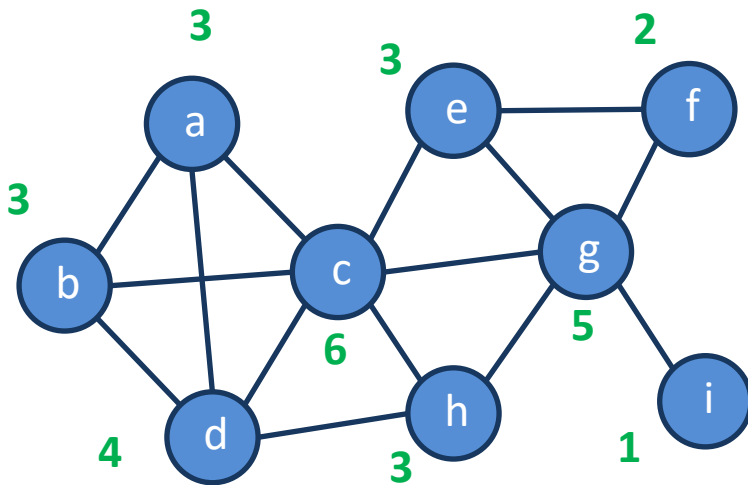


# In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:
- BZ algorithm computes core numbers by recursively deleting the vertex with the lowest degree.
- The core number of a node is exactly its degree at the time the node is deleted.

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

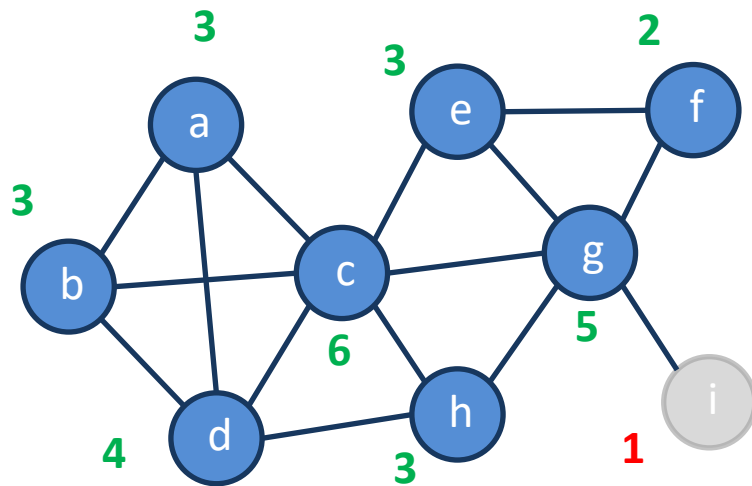
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1 compute the degrees of vertices;
1.2 order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2   for each  $v \in \mathcal{V}$  in the order do begin
2.1      $\text{core}[v] := \text{degree}[v]$ ;
2.2     for each  $u \in \text{Neighbors}(v)$  do
2.2.1       if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1          $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2         reorder  $\mathcal{V}$  accordingly
2.2.2       end
2.3   end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

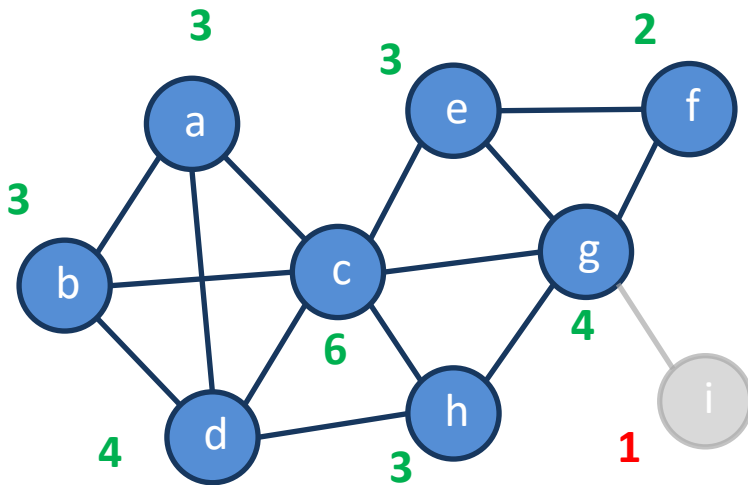
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1 compute the degrees of vertices;
1.2 order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2   for each  $v \in \mathcal{V}$  in the order do begin
2.1      $\text{core}[v] := \text{degree}[v]$ ;
2.2     for each  $u \in \text{Neighbors}(v)$  do
2.2.1       if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1          $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2         reorder  $\mathcal{V}$  accordingly
2.2.2       end
2.2     end
2.3   end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

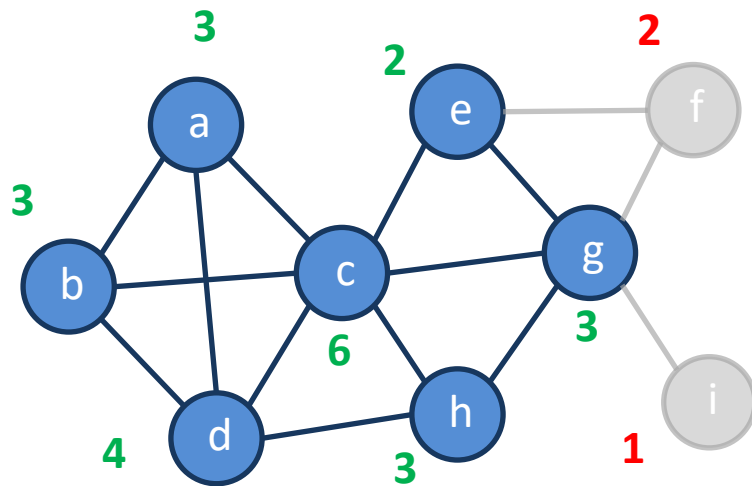
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1 compute the degrees of vertices;
1.2 order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2 for each  $v \in \mathcal{V}$  in the order do begin
2.1    $\text{core}[v] := \text{degree}[v]$ ;
2.2   for each  $u \in \text{Neighbors}(v)$  do
2.2.1     if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1        $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2       reorder  $\mathcal{V}$  accordingly
2.2.2     end
end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

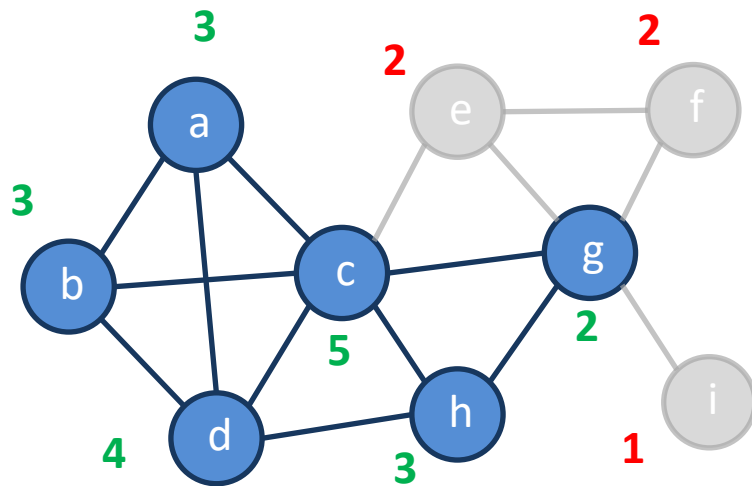
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.1.2        end
2.2      end
2    end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

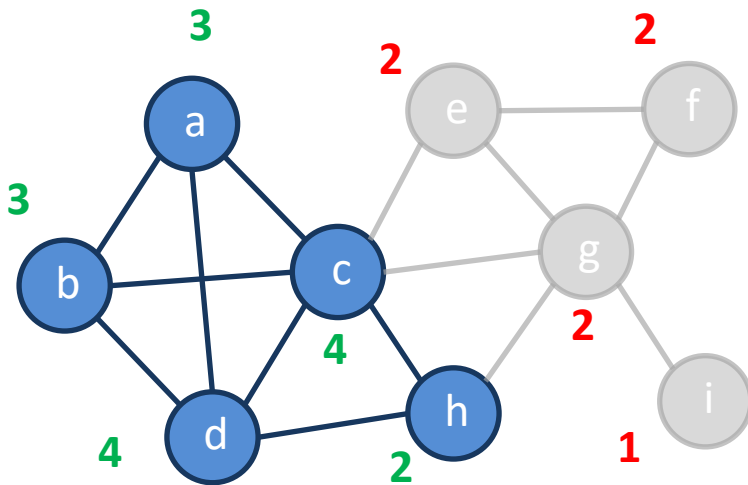
OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.1.2        end
2.2      end
2    end;
```



# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

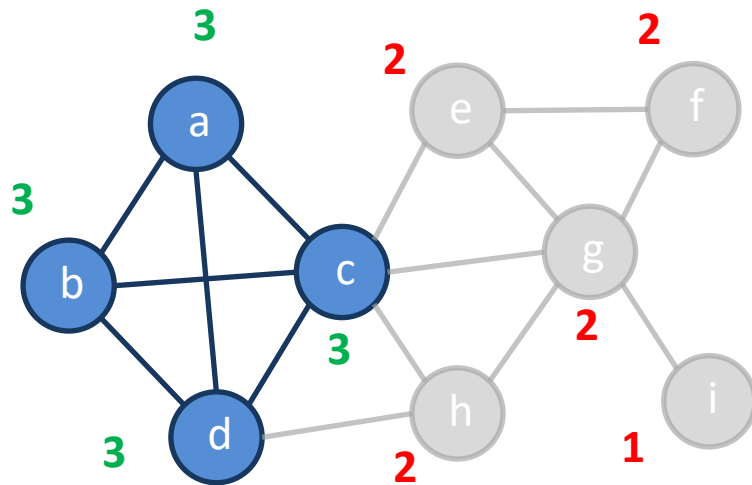
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.1.2        end
2.2      end
2    end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

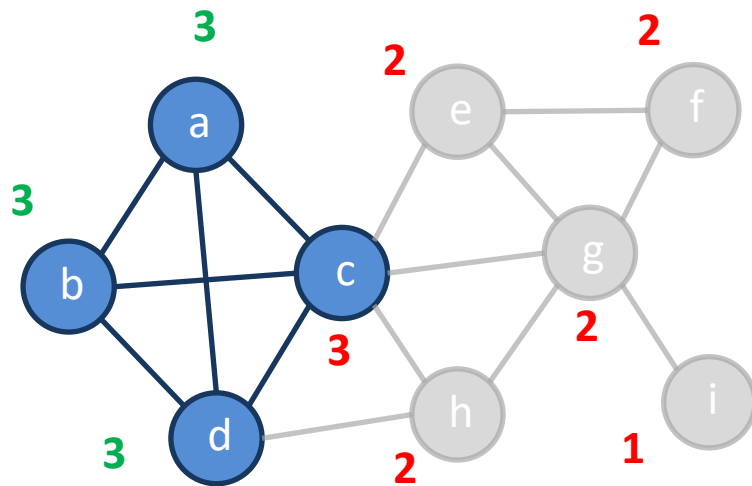
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.2        end
2.2.3      end
2.2.4    end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

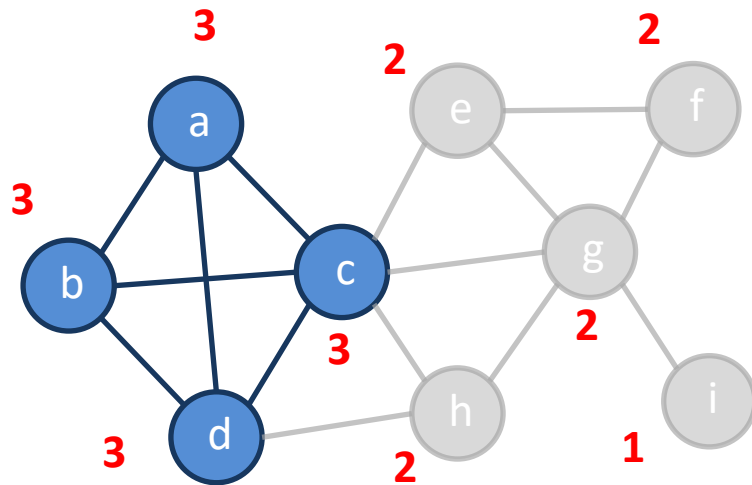
INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.2        end
2.2.3      end
2.2.4    end;
```

# In-Memory Core Decomposition Algorithm

## •Batagelj and Zaversnik Algorithm:



Number in green color: degree

Number in red color: core number

**Algorithm 1** In the algorithm the core number of vertex  $v$ ,  $\text{core}(v)$ , is represented by the table element  $\text{core}[v]$ , and its degree by the table element  $\text{degree}[v]$ .

INPUT: graph  $\mathcal{G} = (\mathcal{V}, \mathcal{L})$  represented by lists of neighbors  $\text{Neighbors}(v)$  for each vertex  $v$

OUTPUT: table  $\text{core}$  with core number  $\text{core}[v]$  for each vertex  $v$

```
1.1  compute the degrees of vertices;
1.2  order the set of vertices  $\mathcal{V}$  in increasing order of their degrees;
2    for each  $v \in \mathcal{V}$  in the order do begin
2.1       $\text{core}[v] := \text{degree}[v]$ ;
2.2      for each  $u \in \text{Neighbors}(v)$  do
2.2.1        if  $\text{degree}[u] > \text{degree}[v]$  then begin
2.2.1.1           $\text{degree}[u] := \text{degree}[u] - 1$ ;
2.2.1.2          reorder  $\mathcal{V}$  accordingly
2.2.2        end
2.2.3      end
2.2.4    end;
```

# More Applications of k-core

- Network modeling and analysis
- Reasoning the collapse of a network
- Prevent Network Unraveling
- Discovering Influential Nodes
- Community Discovery
- Anomaly Detection
- .....

# Learning Outcome

- The representation of graph: adjacency matrix and adjacency list
- K-core: definition and computation