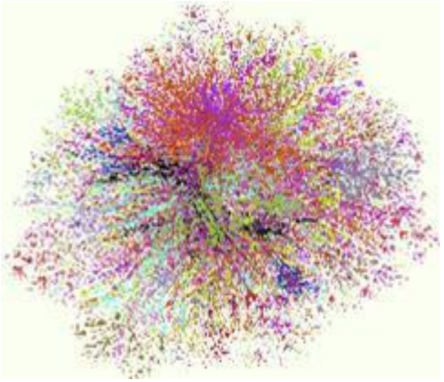
COMP9311 Advanced Topics - Graph Data Analytics

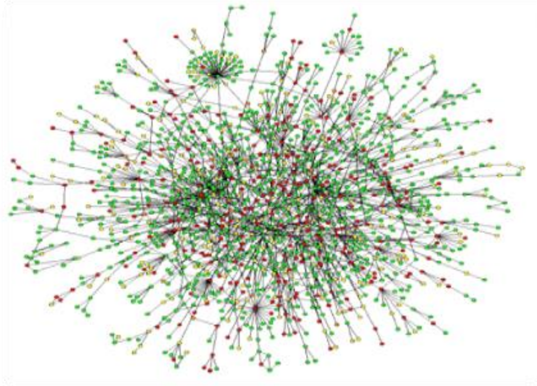
Why graphs ?

- Common model across different fields



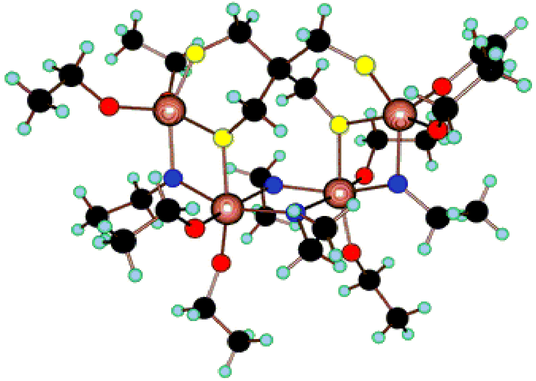
Web Graph

Social Network

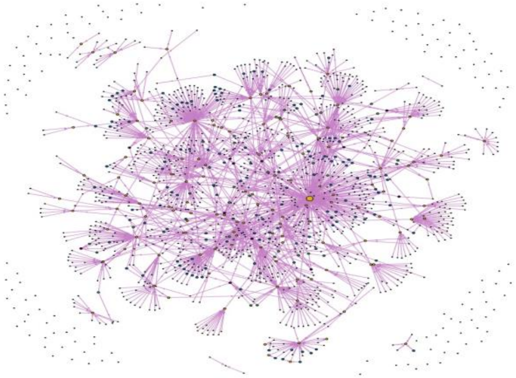


Protein Interaction Network

Road Network



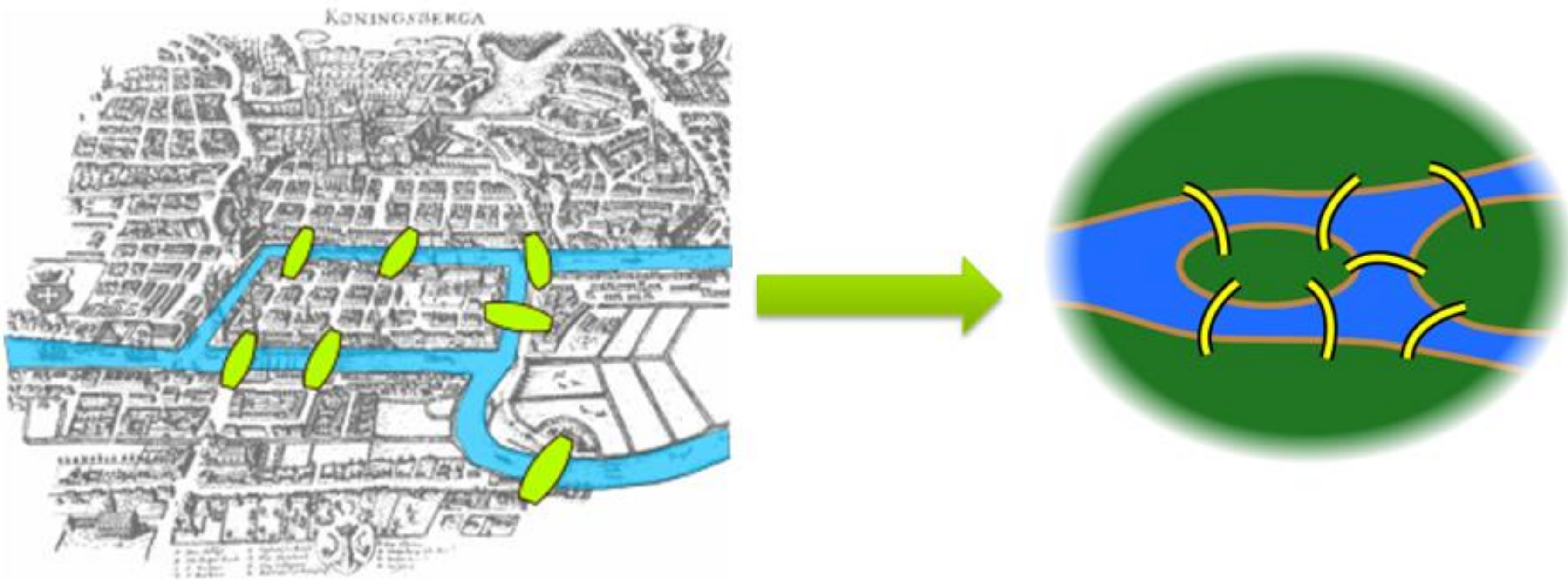
Chemical Compound



Ontology Graph

History

- Seven Bridges of Königsberg (1736)



Big Graphs

- 70+ billion facts in knowledge graphs in 2016

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

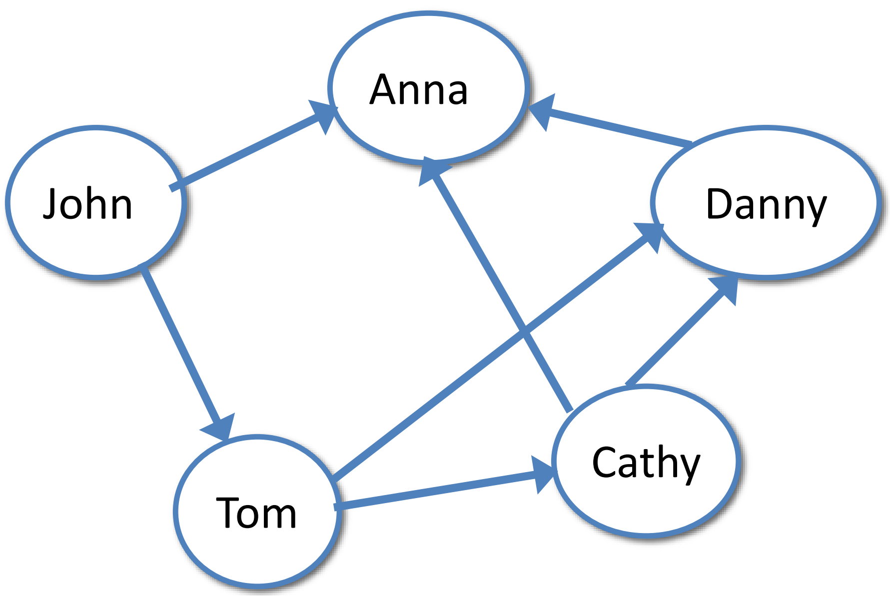
- 1.5+ billon users in 2018

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

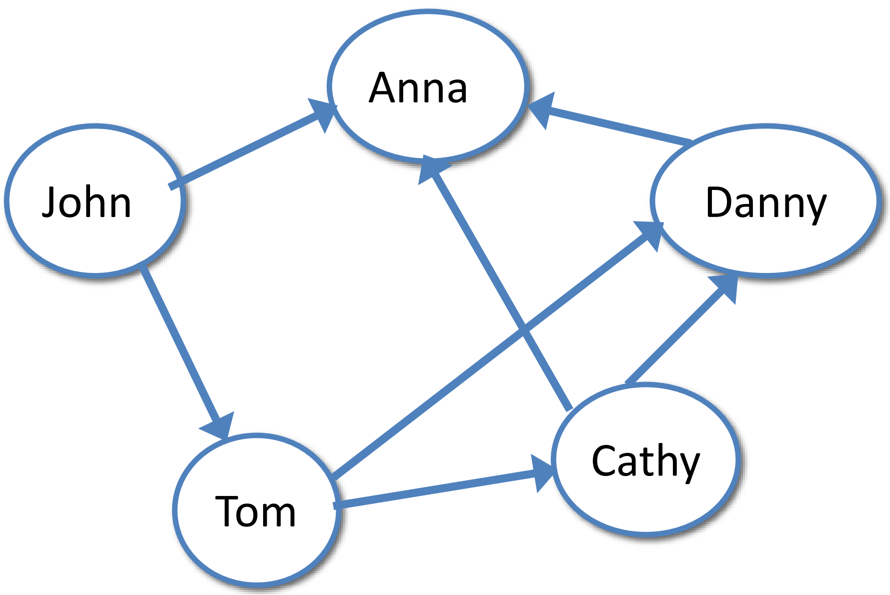
What's a Graph?

- 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

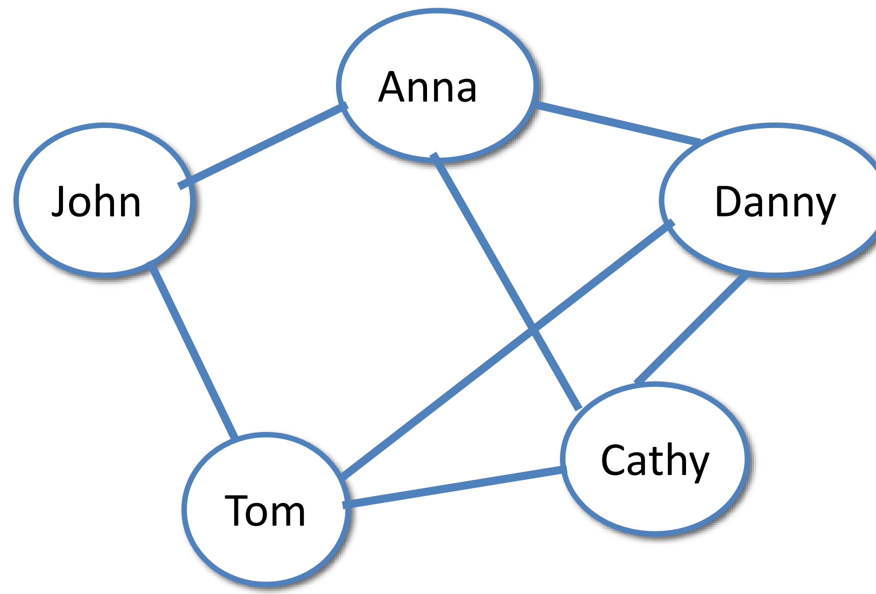


A twitter network

Directed vs Undirected



A twitter network

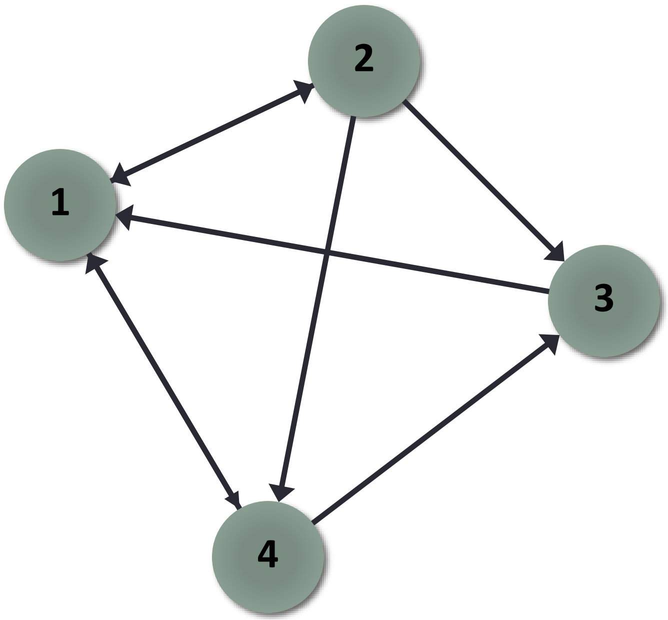


A Facebook network

Representing Graphs

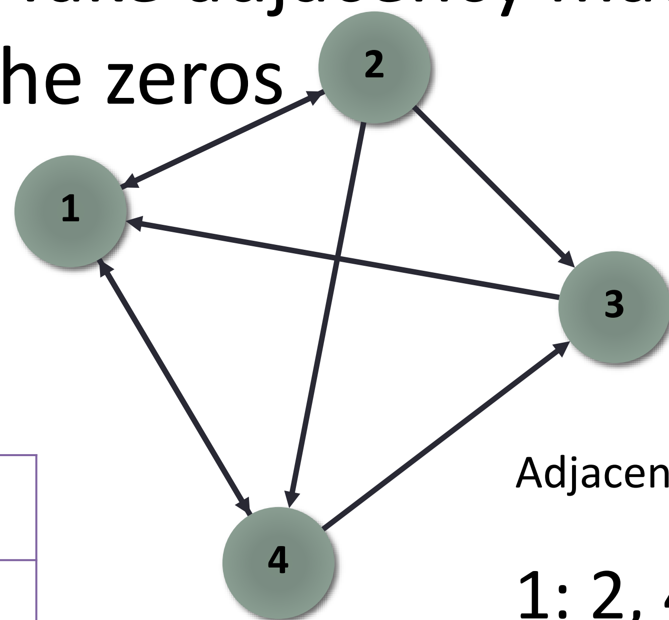
- Adjacency Matrices: Represent a graph as an n x n square matrix M
  - $n = |V|$
  - $M_{ij} = 1$ means a edge from vertex i to j

	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

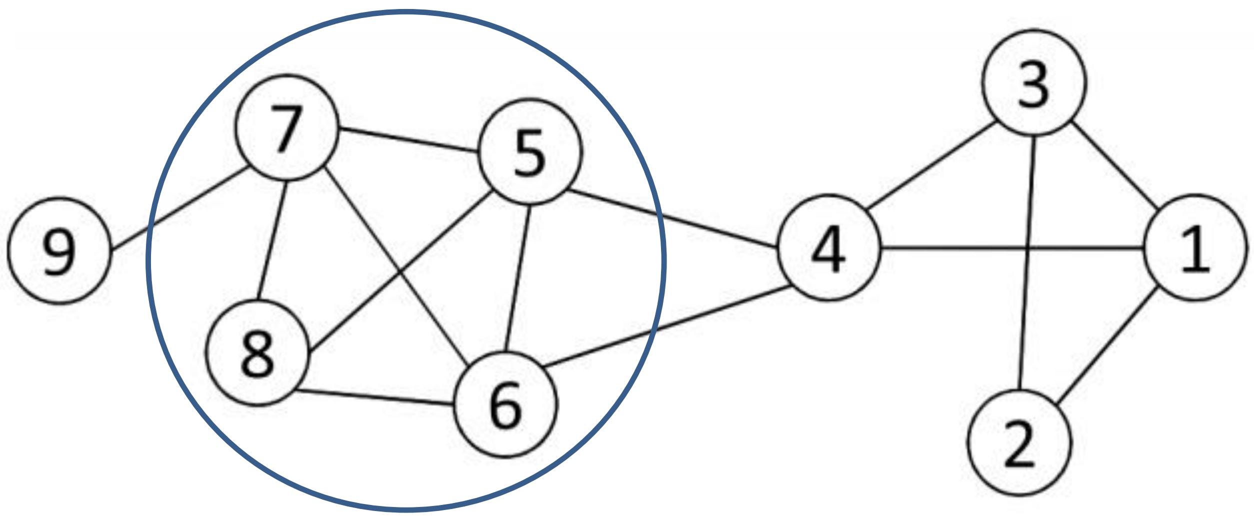
Adjacency List

1: 2, 4
2: 1, 3, 4
3: 1
4: 1, 3

Structural Analysis of Graphs

Cliques 团

- 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

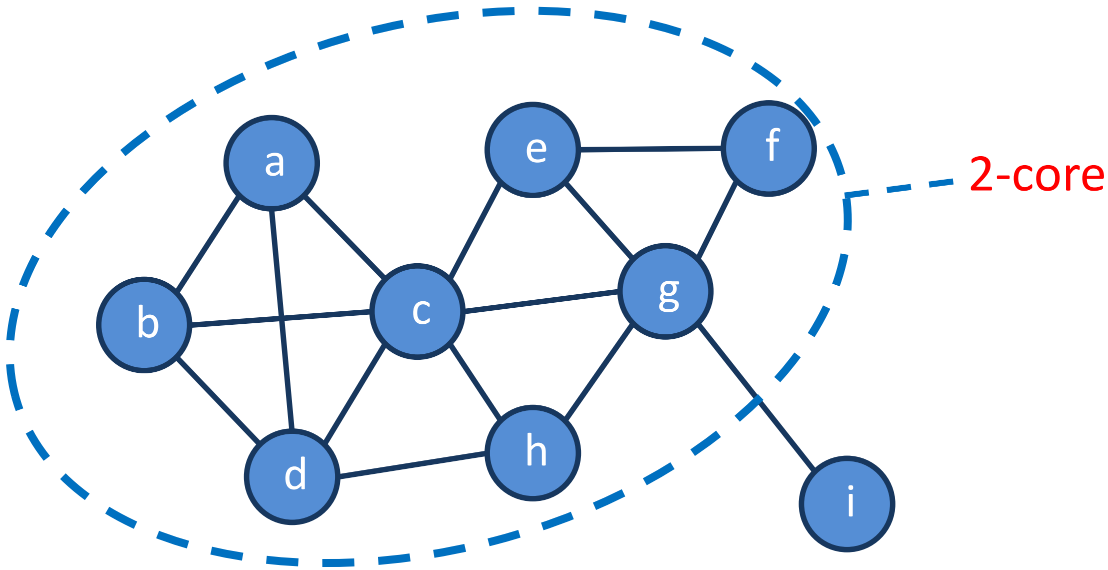


R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949

k-Core

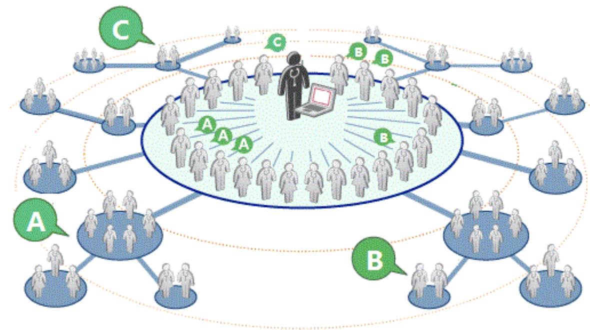
- Given a graph *G*, the *k*-core of *G* is a subgraph where each node has at least *k* neighbors (i.e., *k* adjacent nodes, or a degree of *k*).

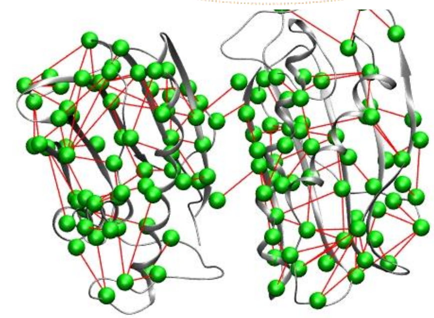
相邻



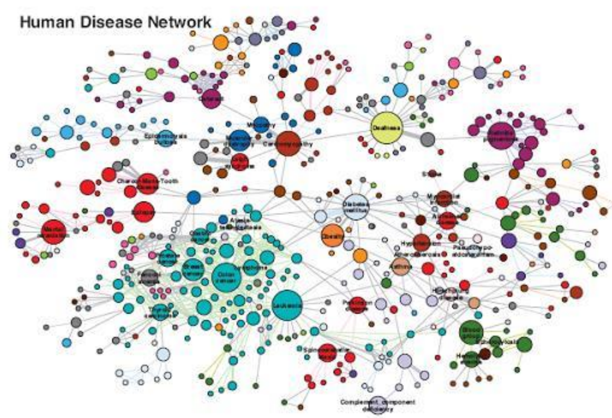
S. B. Seidman. Network structure and minimum degree. Social networks, 5(3):269–287, 1983.

Applications



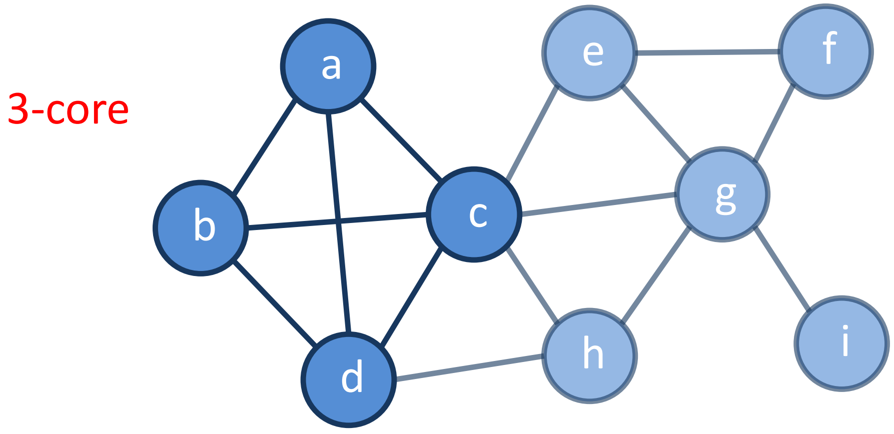


- Community detection
- Social contagion
- User engagement
- Event detection
- Network analysis and visualization
- Influence study
- Graph clustering
- Protein function prediction
- Human Cerebral Cortex
- ……



Compute *k*-Core

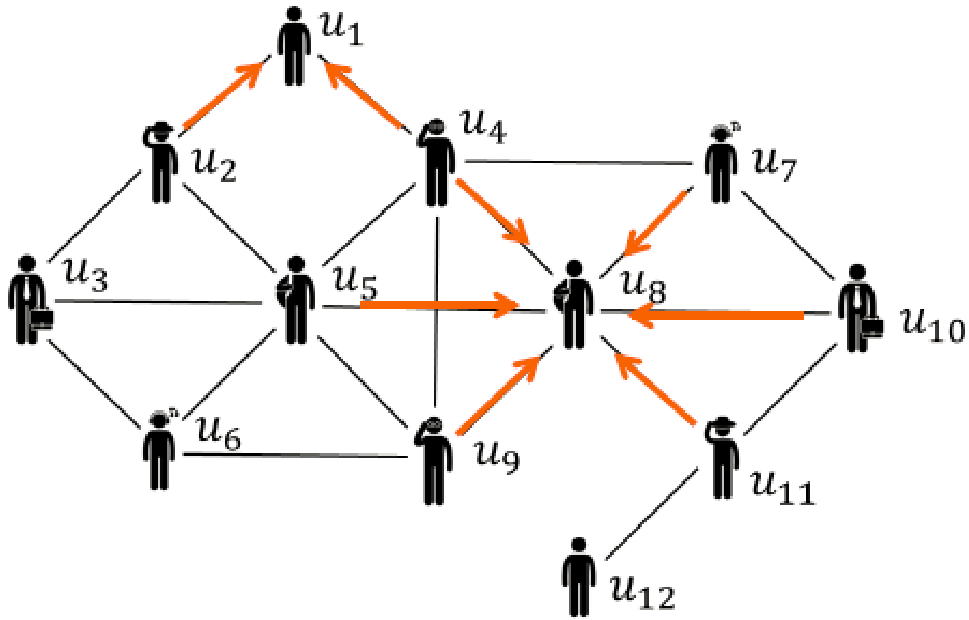
- Given a graph *G*, the *k*-core of *G* can be computed by recursively deleting every node and its adjacent edges if its degree is less than *k*.



S. B. Seidman. Network structure and minimum degree. Social networks, 5(3):269–287, 1983.

Why study k-core ?

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

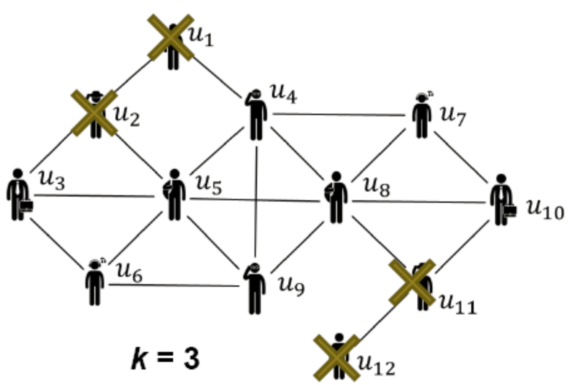


K. Bhawalkar, J. Kleinberg, K. Lewi, T. Roughgarden, and A. Sharma. Preventing unraveling in social networks: the anchored k-core problem. *SIAM Journal on Discrete Mathematics*, 29(3):1452–1475, 2015.

Why study k-core ?

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

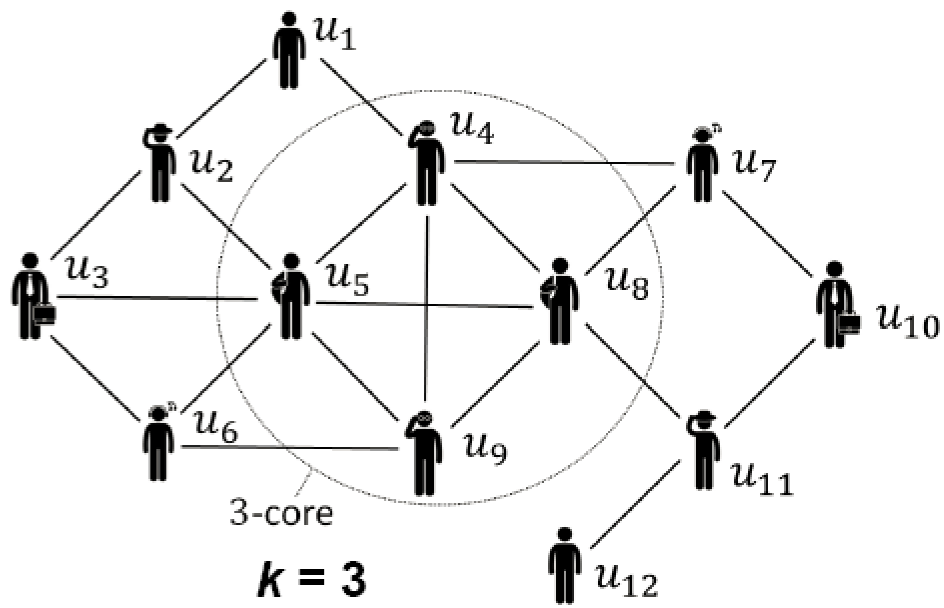
An equilibrium: a group has the minimum degree of *k,* namely *k-core*



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 ?

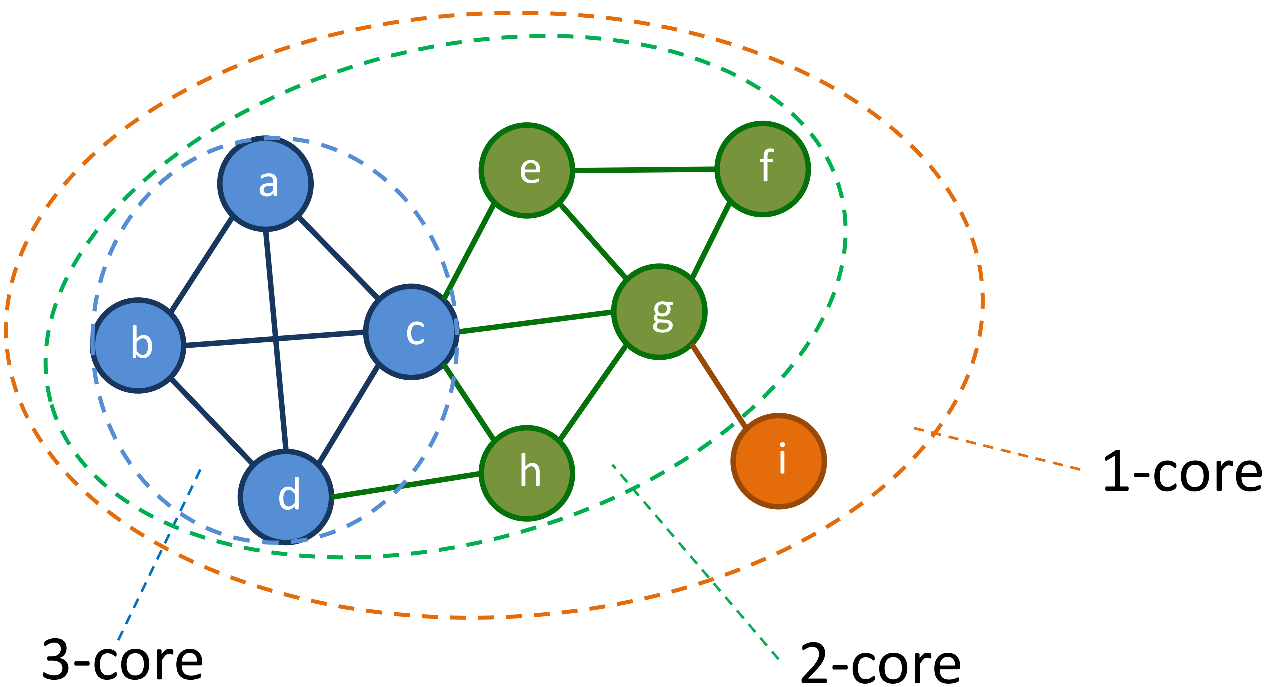
A stable social group tends to be a k-core in the network



k-Core Decomposition

分解

- **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.*



- *k*-core decomposition will not be assessed in final exam

In-Memory Core Decomposition Algorithm

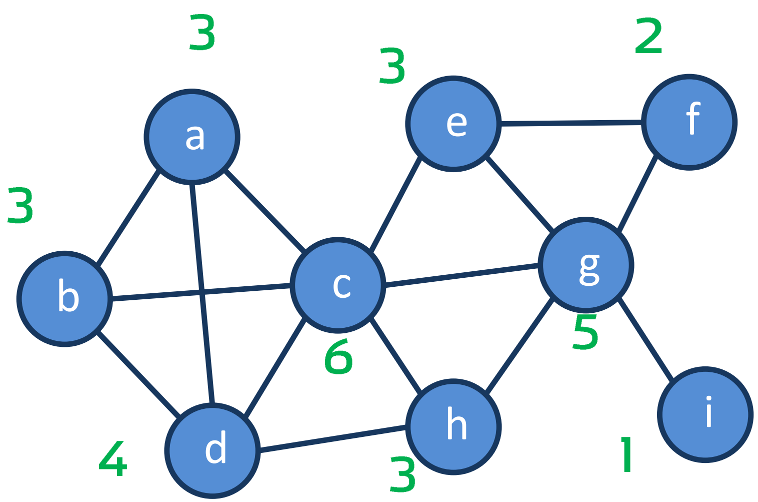
- Batagelj and Zaversnik Algorithm:
- BZ algorithm computes core numbers by recursively deleting the vertex with the lowest degree.

- The core number of a node is exactly its degree at the time the node is deleted.

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

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

OUTPUT: table $core$ with core number $core[v]$ 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 $core[v] := degree[v]$;
2.2 for each $u \in Neighbors(v)$ do
2.2.1 if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u] := degree[u] - 1$;
2.2.1.2 reorder $\mathcal{V}$ accordingly
end
end;

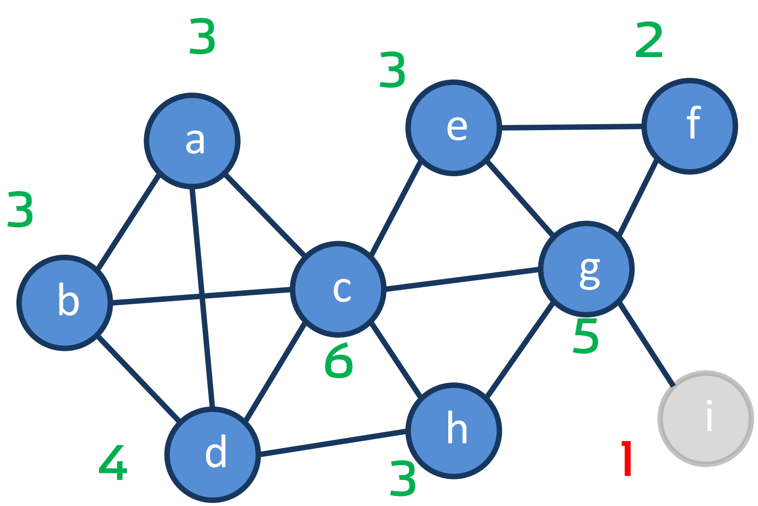
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ for each vertex v

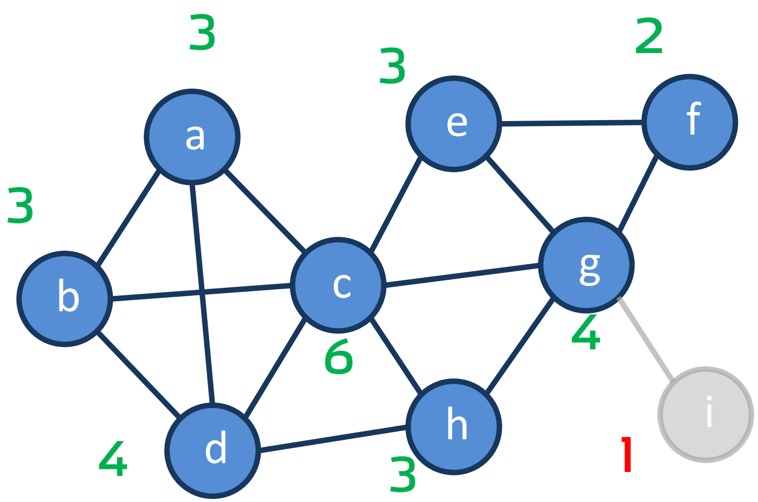
1.1 compute the degrees of vertices;
1.2 order the set of vertices $\mathcal{V}$ in increasing order of their degrees;
2 for each $v \in \mathcal{V}$ in the order do begin
2.1 $core[v] := degree[v]$;
2.2   for each $u \in Neighbors(v)$ do
2.2.1   if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u]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

Number in green color: degree
Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ 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 $core[v] := degree[v]$;
2.2   for each $u \in Neighbors(v)$ do
2.2.1   if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u] := degree[u] - 1$;
2.2.1.2   reorder $\mathcal{V}$ accordingly
            end
end;

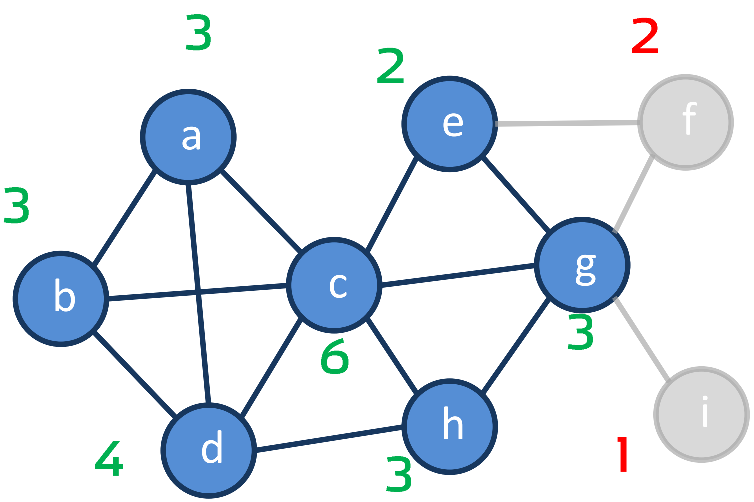
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

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

OUTPUT: table $core$ with core number $core[v]$ 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 $core[v] := degree[v]$;
2.2 for each $u \in Neighbors(v)$ do
2.2.1 if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u] := degree[u] - 1$;
2.2.1.2 reorder $\mathcal{V}$ accordingly
end
end;

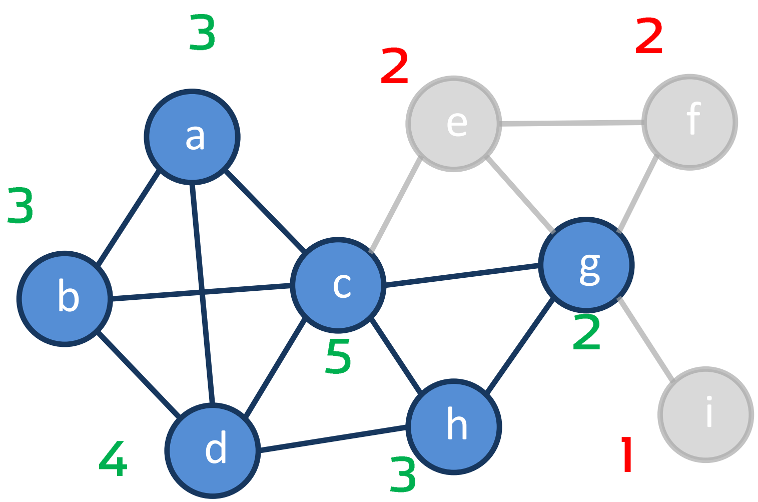
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ for each vertex v

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

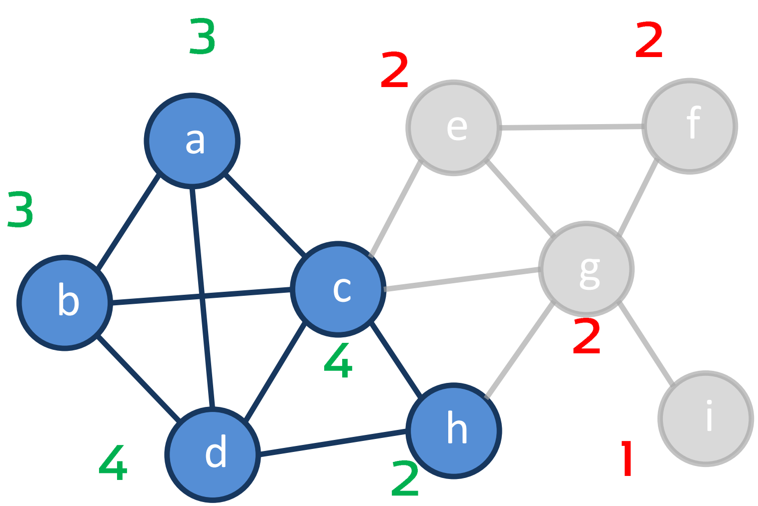
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ for each vertex v

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

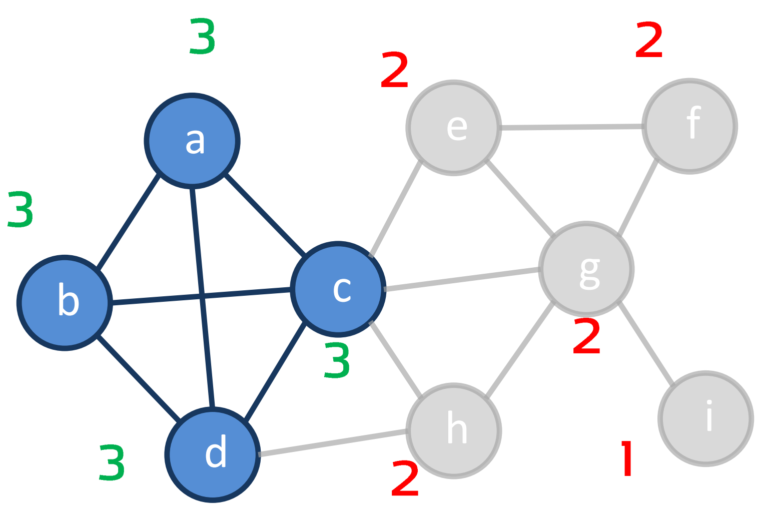
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ 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 $core[v] := degree[v]$;
2.2 for each $u \in Neighbors(v)$ do
2.2.1 if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u] := degree[u] - 1$;
2.2.1.2 reorder $\mathcal{V}$ accordingly
end
end;

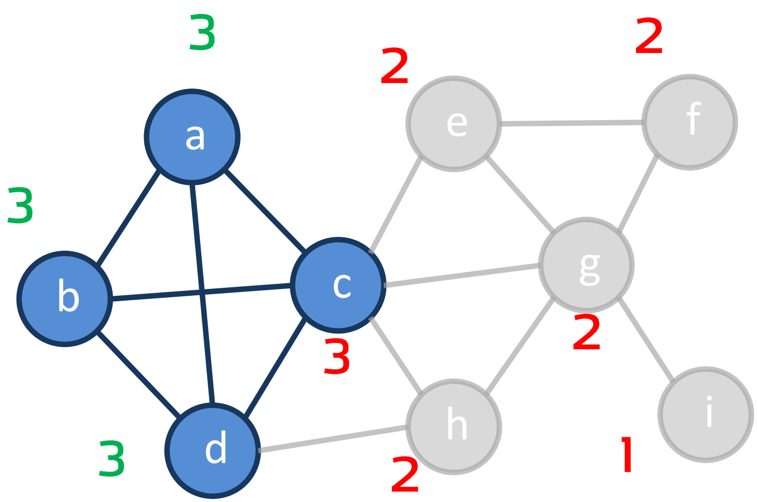
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

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

OUTPUT: table $core$ with core number $core[v]$ 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 $core[v] := degree[v]$;
2.2 for each $u \in Neighbors(v)$ do
2.2.1 if $degree[u] > degree[v]$ then begin
2.2.1.1 $degree[u] := degree[u] - 1$;
2.2.1.2 reorder $\mathcal{V}$ accordingly
end
end;

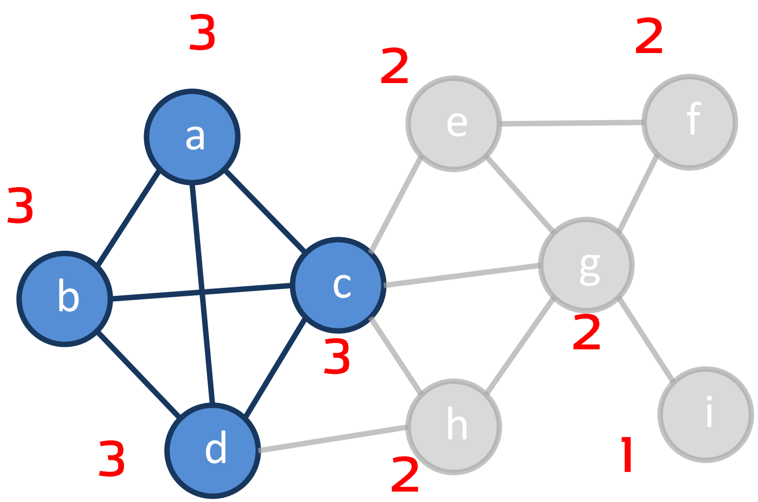
Number in green color: degree

Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

In-Memory Core Decomposition Algorithm

- Batagelj and Zaversnik Algorithm:



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

INPUT: graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ represented by lists of neighbors $Neighbors(v)$ for each vertex v
OUTPUT: table $core$ with core number $core[v]$ 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 $\quad core[v] := degree[v]$;
2.2 $\quad$ for each $u \in Neighbors(v)$ do
2.2.1 $\qquad$ if $degree[u] > degree[v]$ then begin
2.2.1.1 $\qquad\quad degree[u] := degree[u] - 1$;
2.2.1.2 $\qquad\quad$ reorder $\mathcal{V}$ accordingly
$\qquad$ end
$\quad$ end;

Number in green color: degree
Number in red color: core number

V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049, 2003

Learning Outcome

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