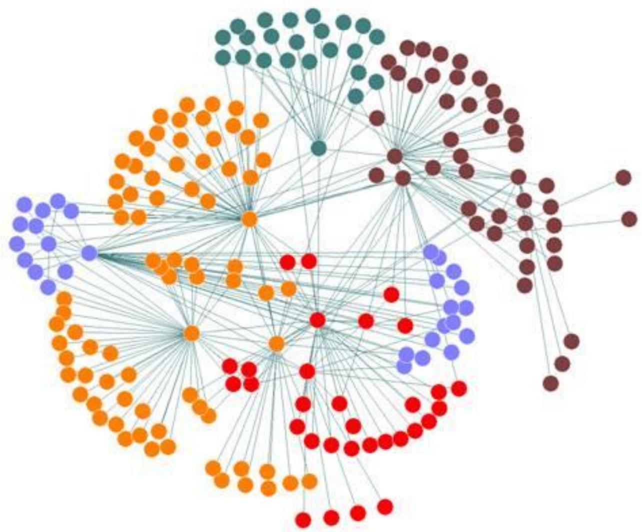
TEL AVIV UNIVERSITY

Algorithms and Applications in Social Networks



2019/2020, Semester B

Slava Novgorodov

Lesson #5

- Newman-Girvan betweenness computation
- Overlapping communities
- Communities detection algorithms
- More methods for community detection

Newman-Girvan: Betweenness

Newman-Girvan algorithm

Algorithm: Newman-Girvan, 2004

Input: graph G(V,E)

Output: Dendrogram

repeat

  For all $e \in E$ compute edge betweenness $C_B(e)$;

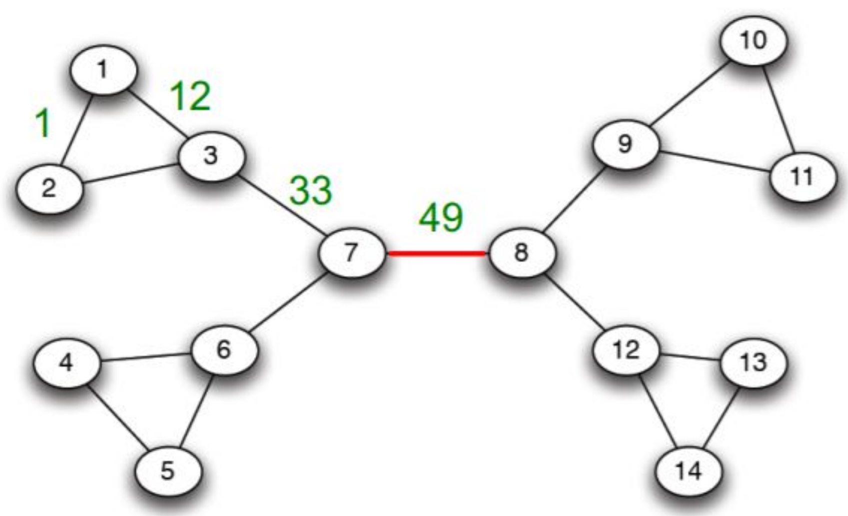
  remove edge e_i with largest $C_B(e_i)$;

until *edges left*;

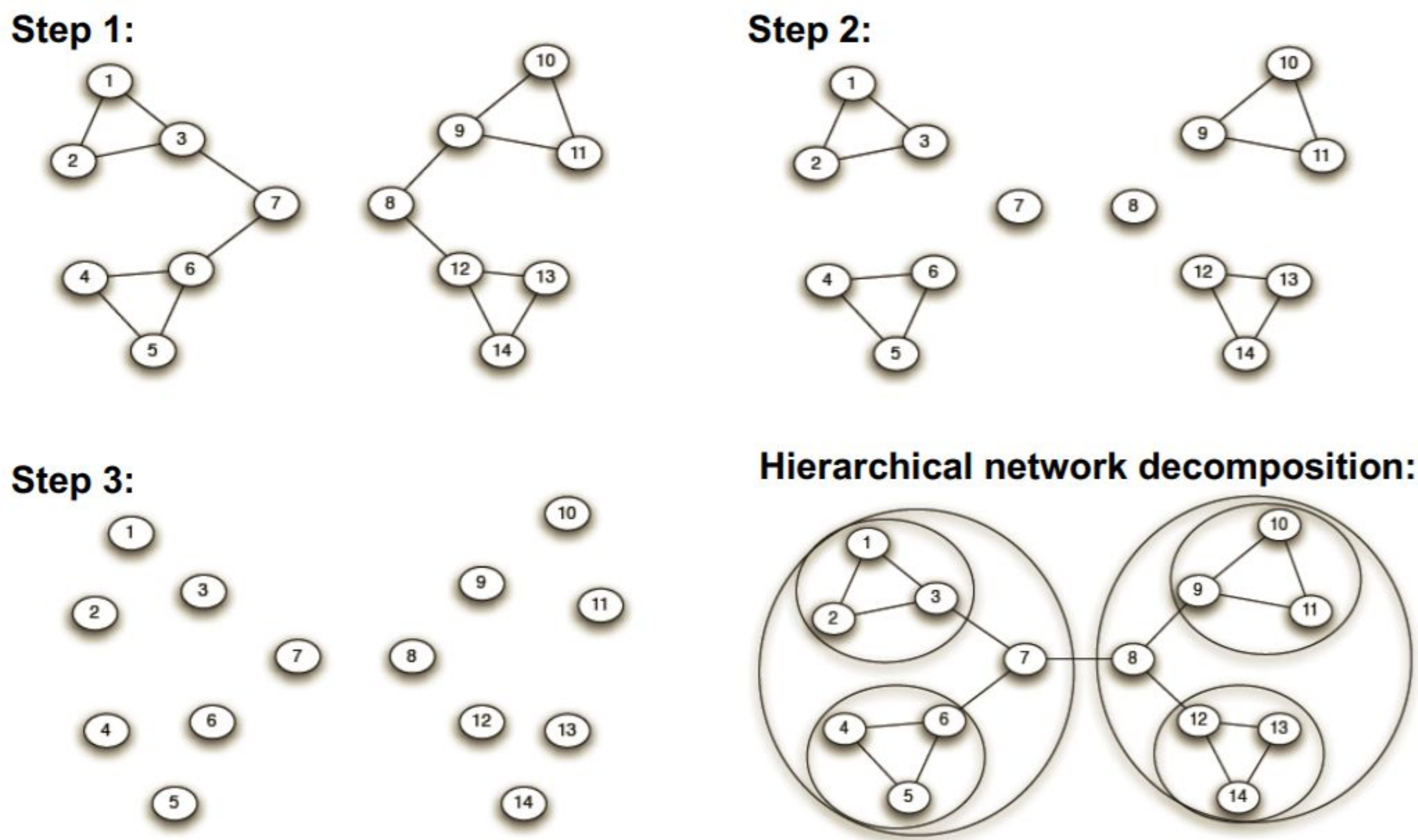
Edge Betweeness

- Number of shortest paths going via edge e

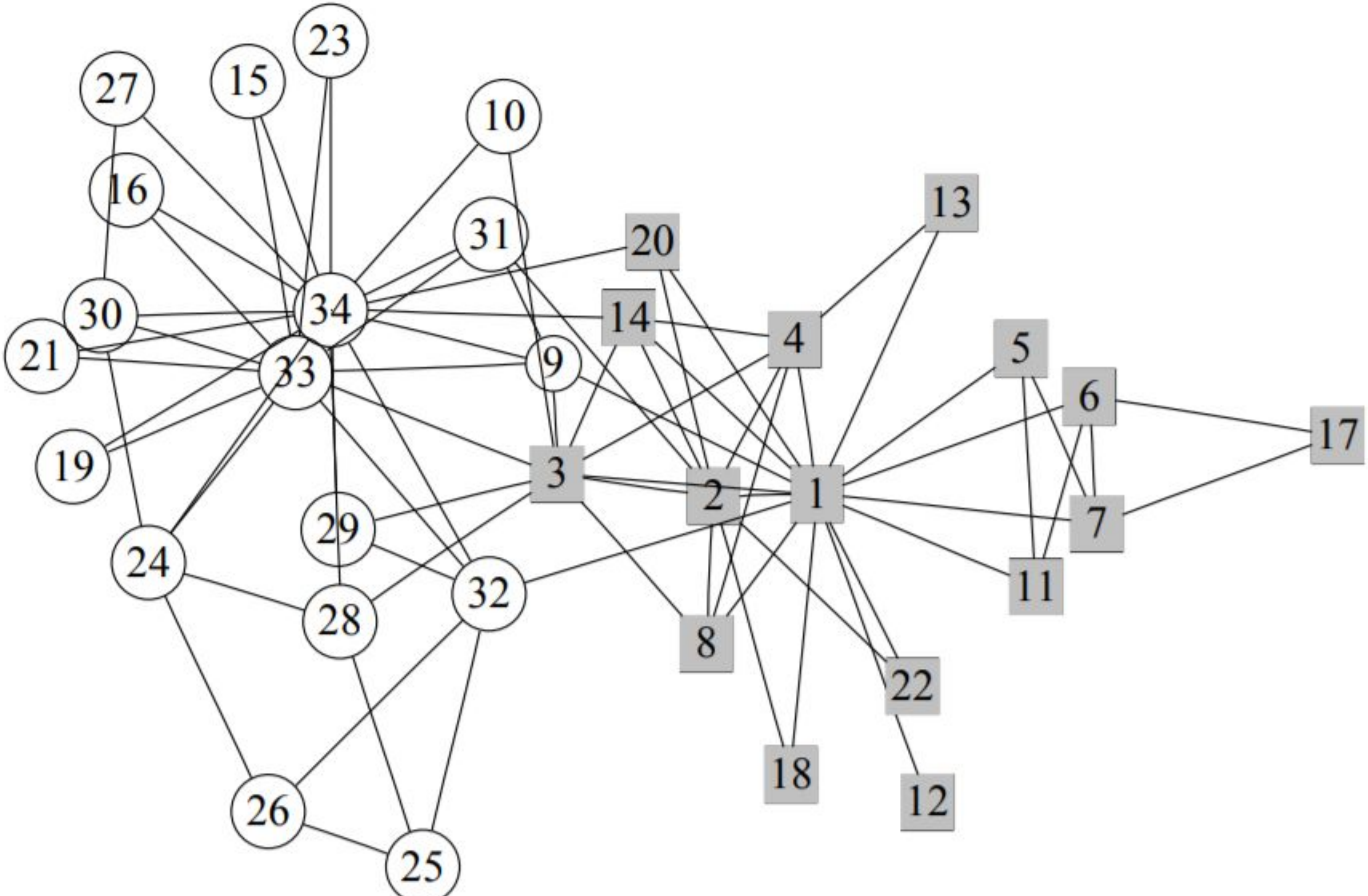
$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$



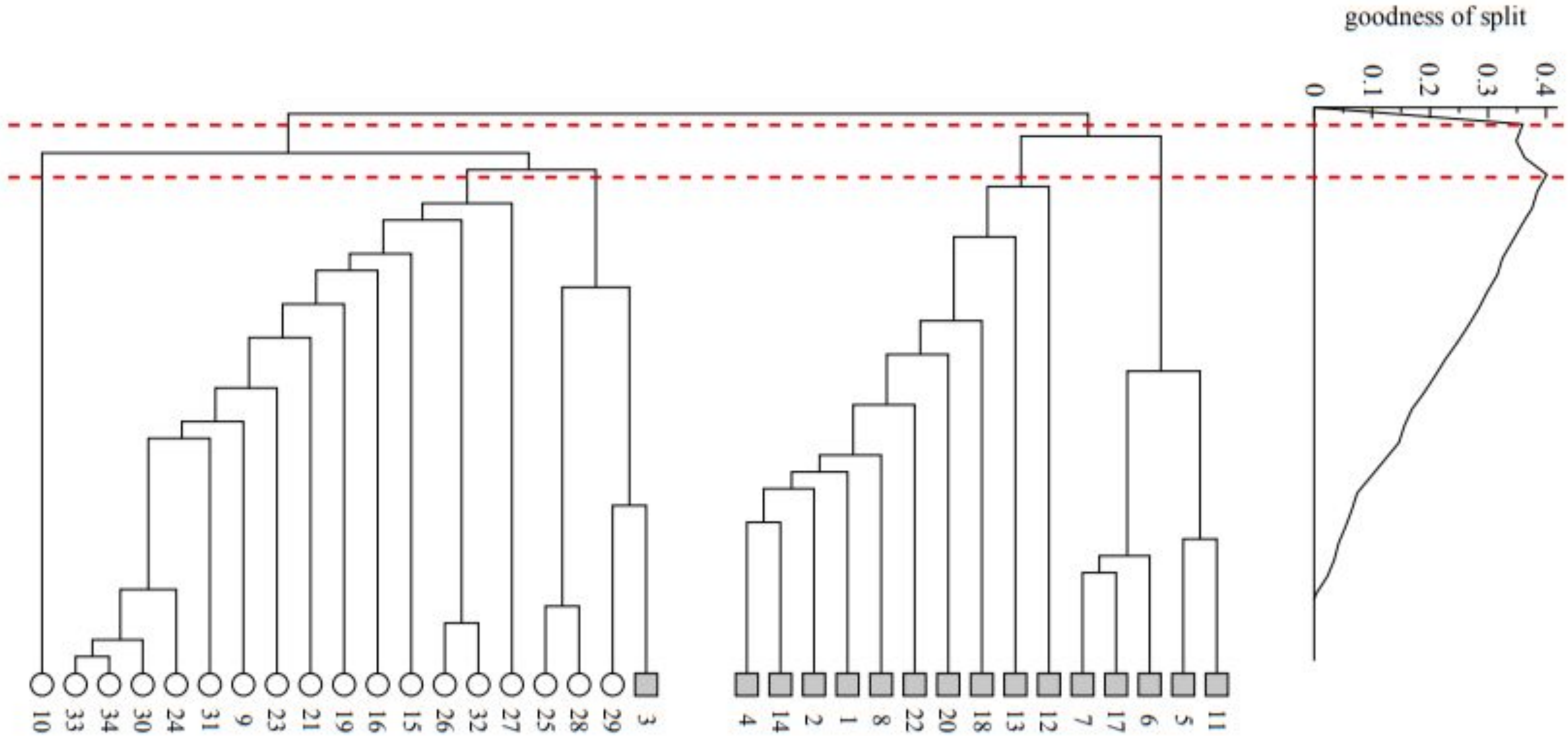
Step-by-step


Step 1:
Step 2:
Step 3:
Hierarchical network decomposition:
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Karate club example

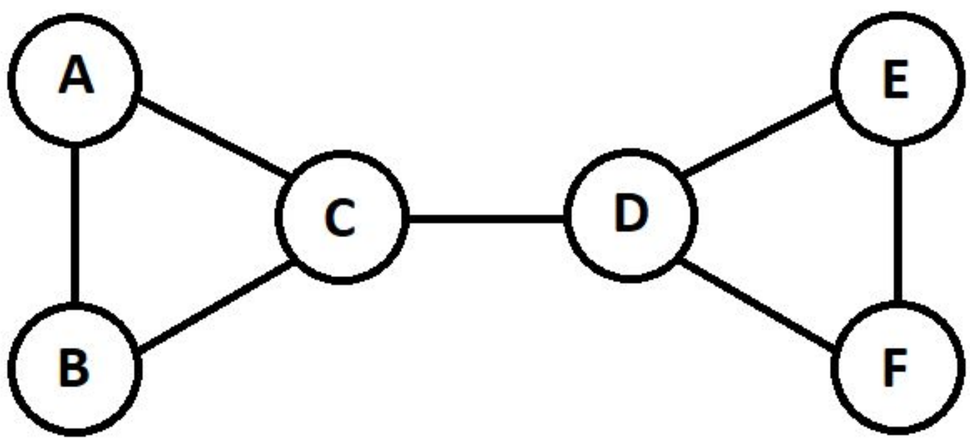


Karate club example

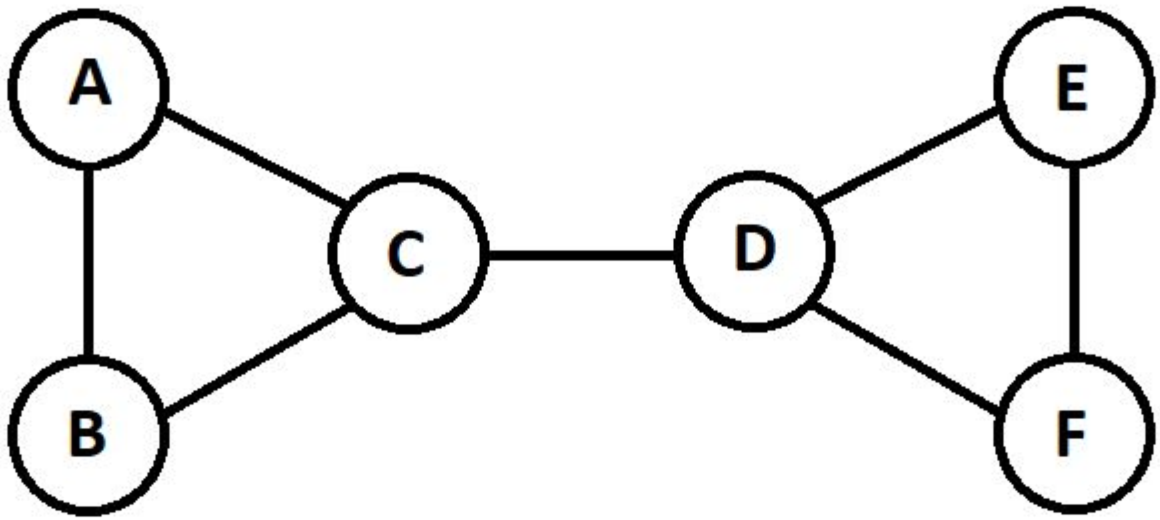


How to compute betweenness?

- Naïve approach: find all shortest pathes and compute brute-force
- Better approach: BFS based algorithm
- Example:



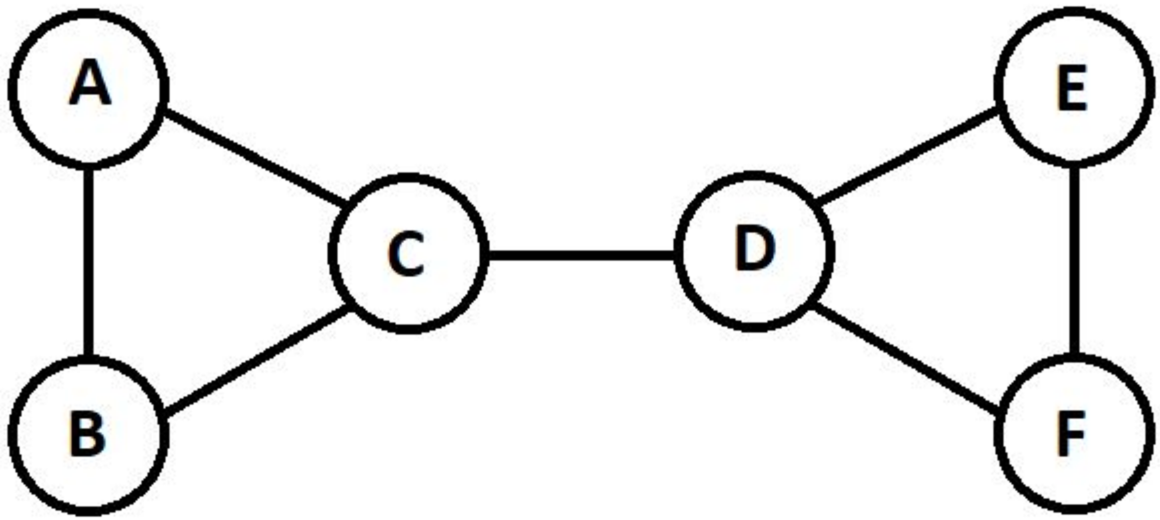
Betweenness computation



Run BFS from every node

Start from A

Betweenness computation

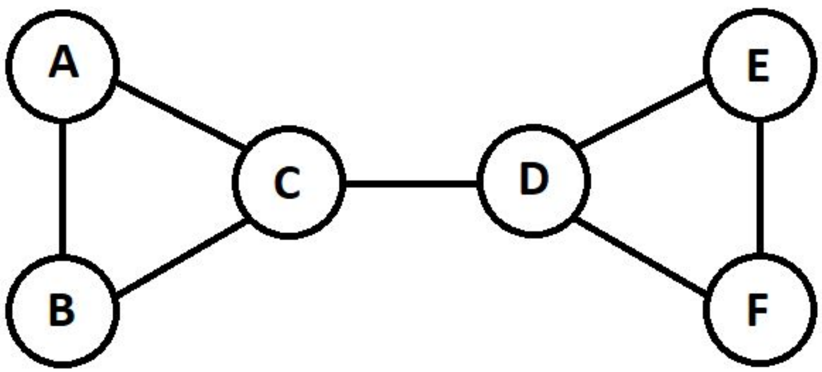


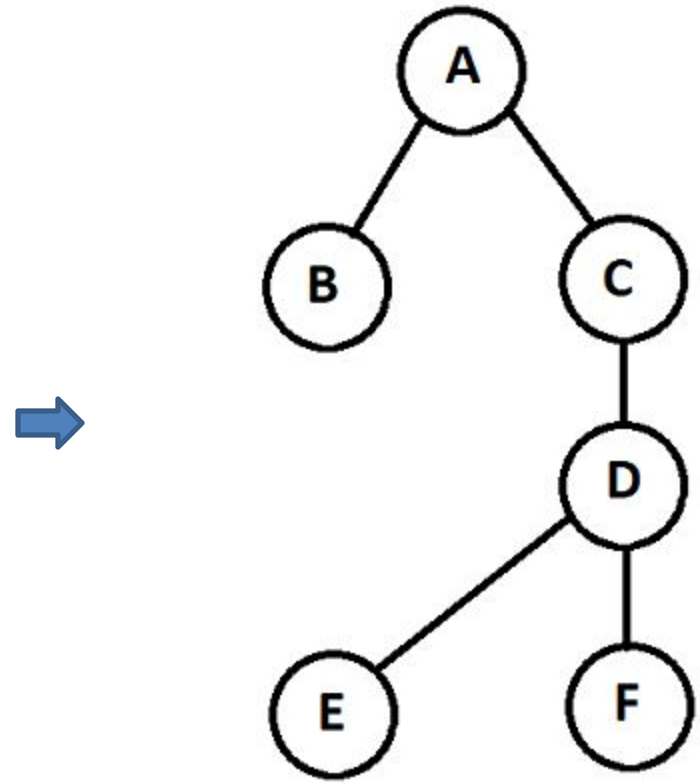
Symmetric case for:

1) A, B, E, F
2) C, D

Betweenness computation

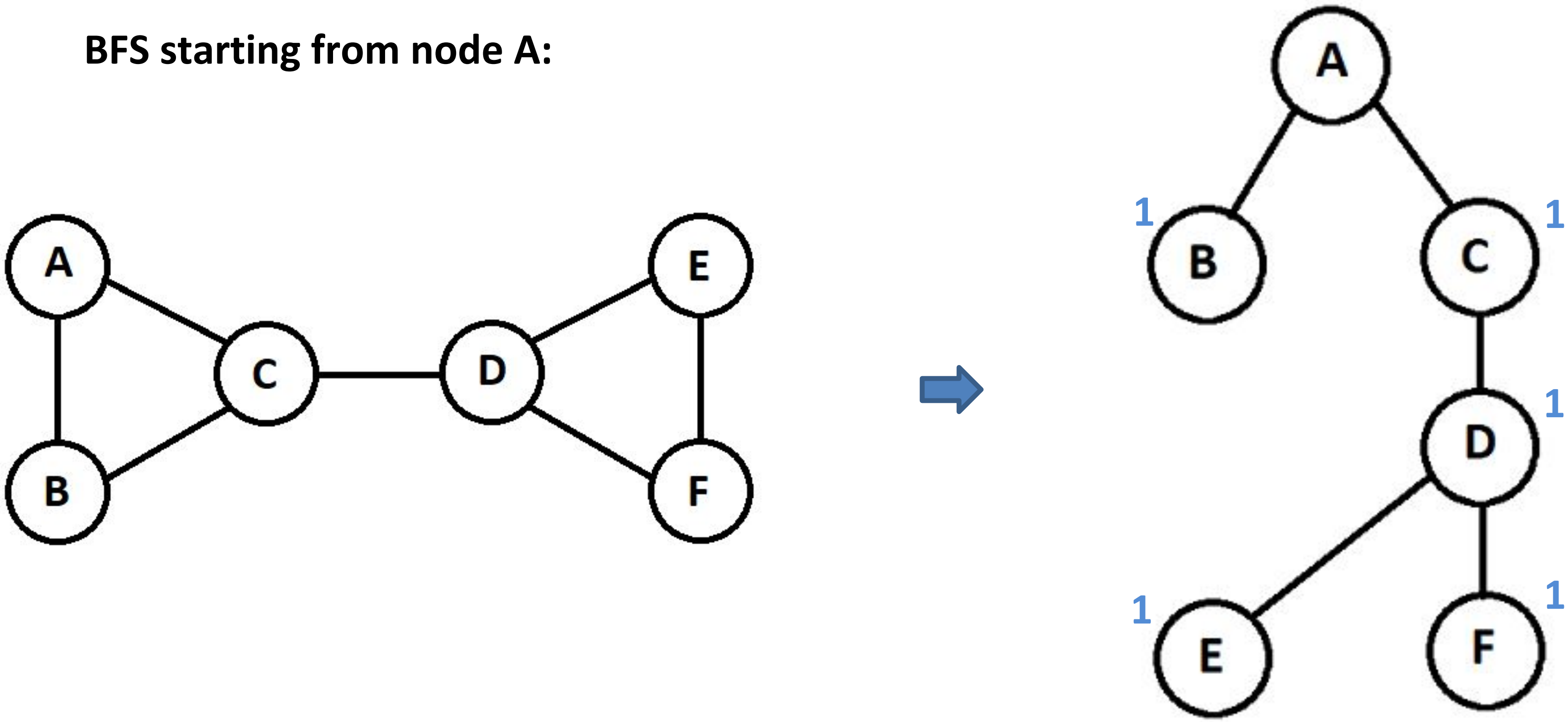
BFS starting from node A:





Betweenness computation

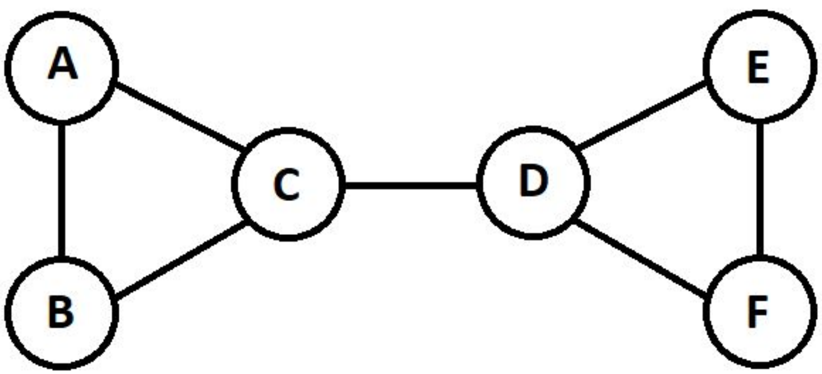
BFS starting from node A:

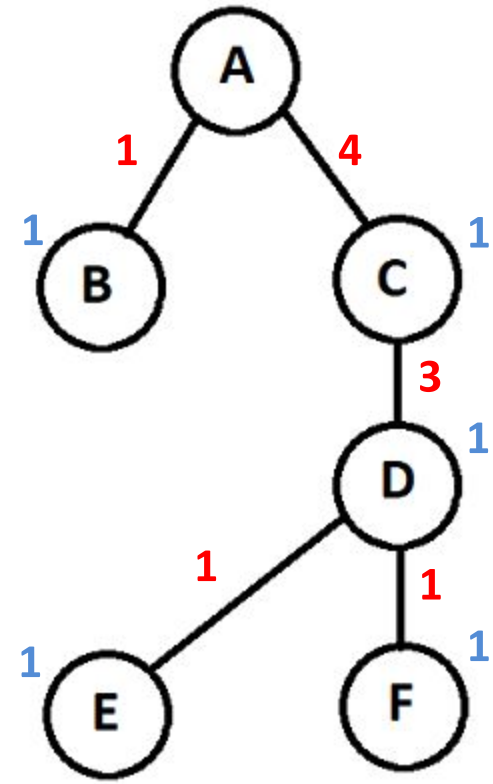


Weights of the nodes – top down, based on
Numbers and weights of parents

Betweenness computation

BFS starting from node A:

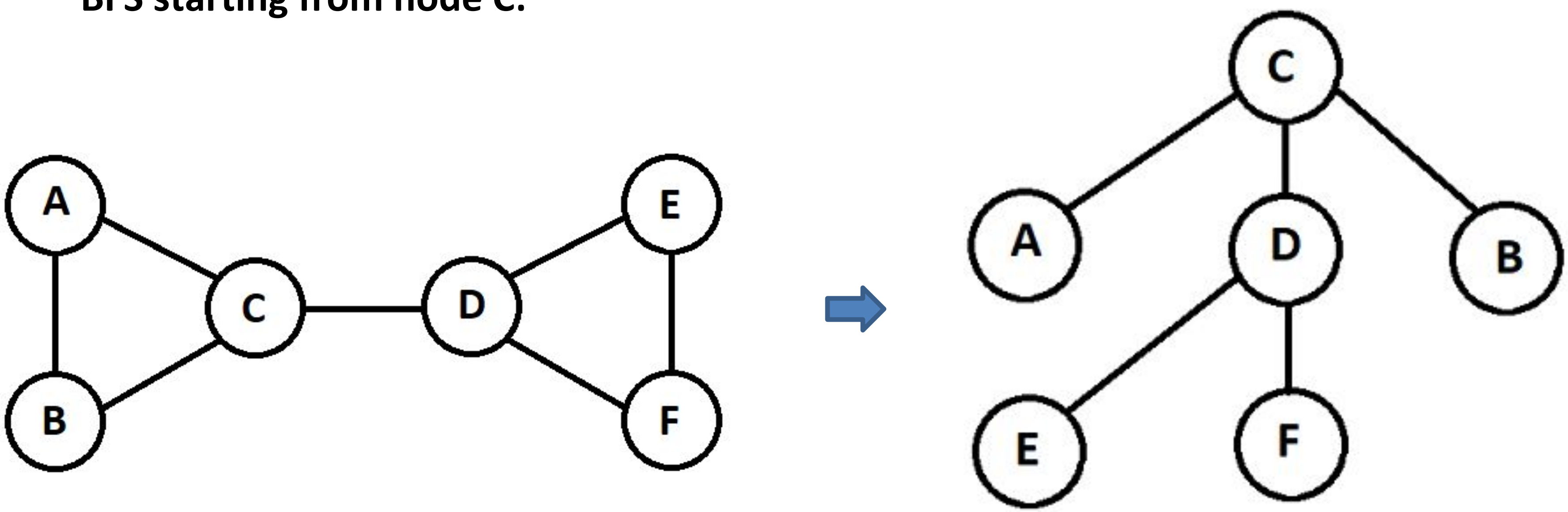




Weights of the edges – bottom up:
Weighted split of the weight between parents + 1

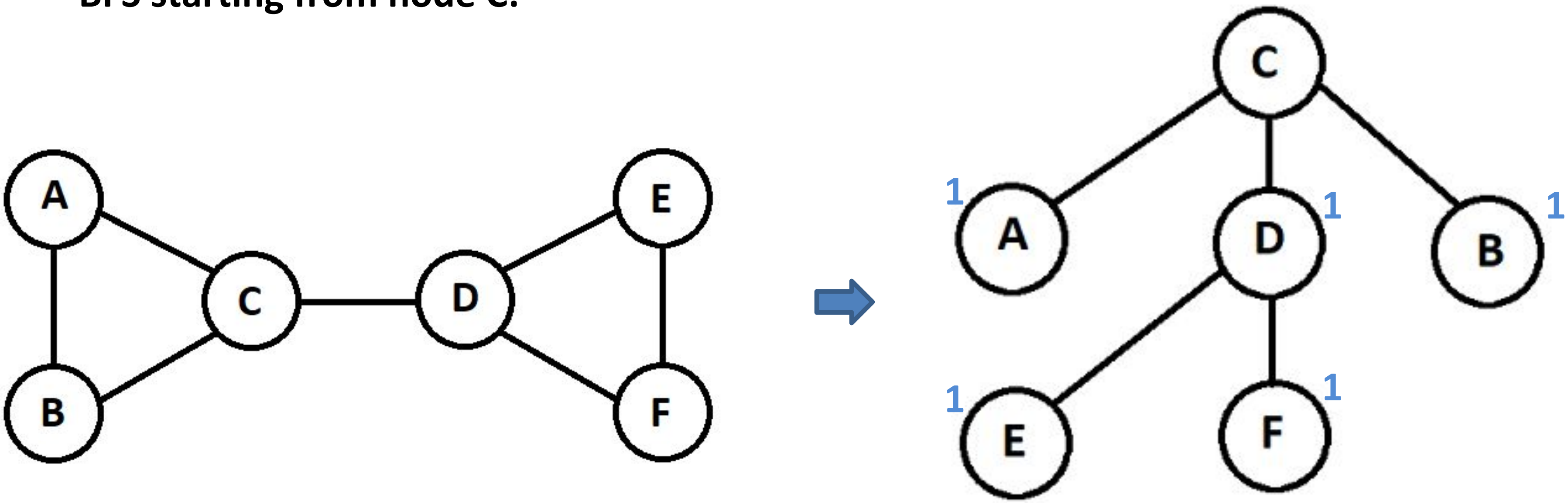
Betweenness computation

BFS starting from node C:



Betweenness computation

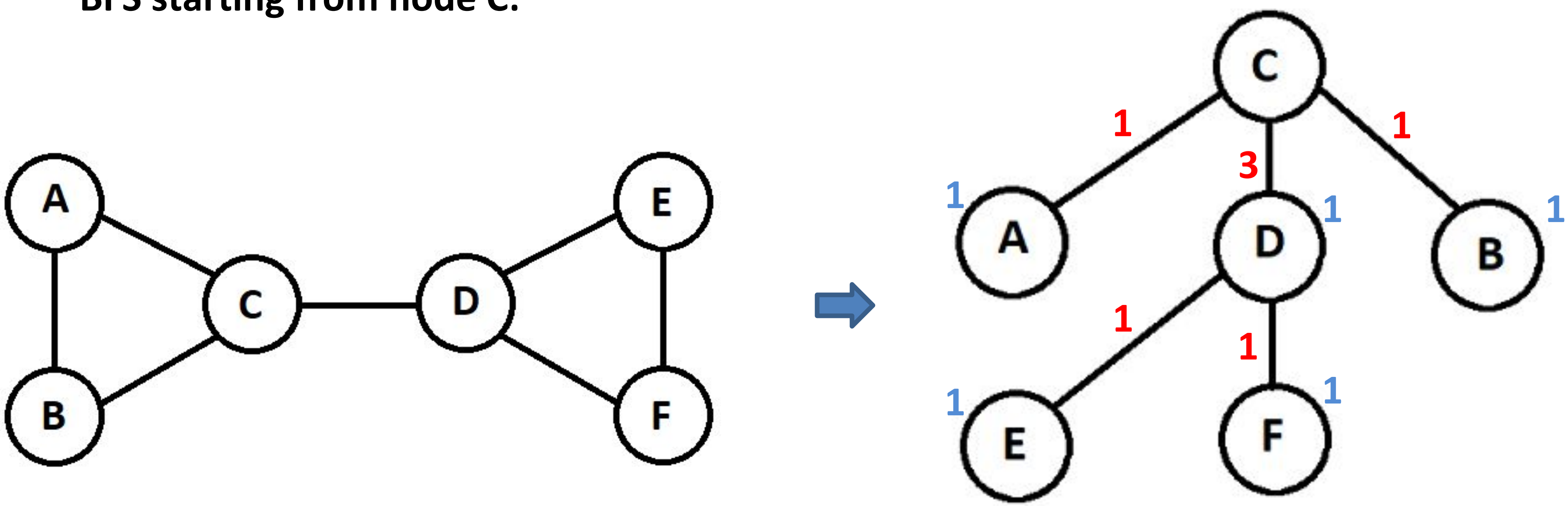
BFS starting from node C:



Weights of the nodes – top down, based on
Numbers and weights of parents

Betweenness computation

BFS starting from node C:



Weights of the edges – bottom up:
Weighted split of the weight between parents + 1

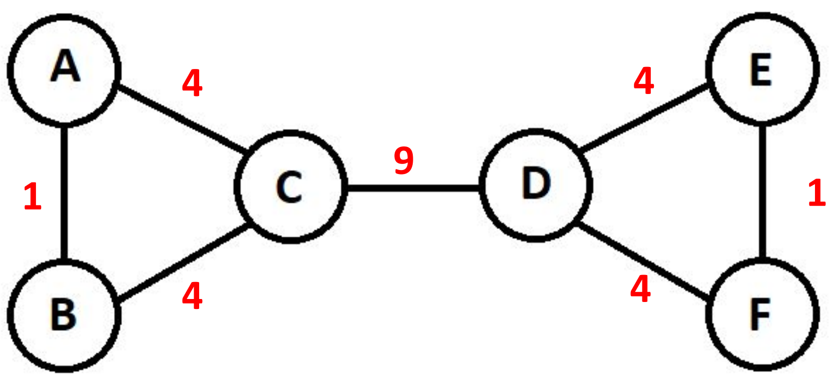
Betweenness computation

Edge betweenness – sum (/2) of edge weights on all BFS graphs

EB(A, B) = (1+1)/2 = 1

EB(A, C) = (4+1+1+1+1)/2 = 4

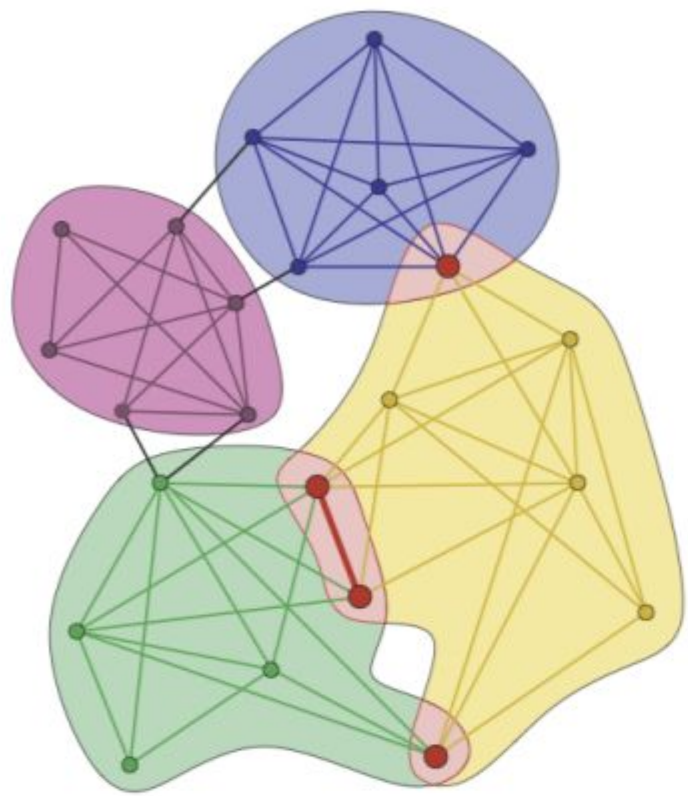
EB(C, D) = (3+3+3+3+3+3)/2 = 9



Overlapping communities

Overlapping Communities

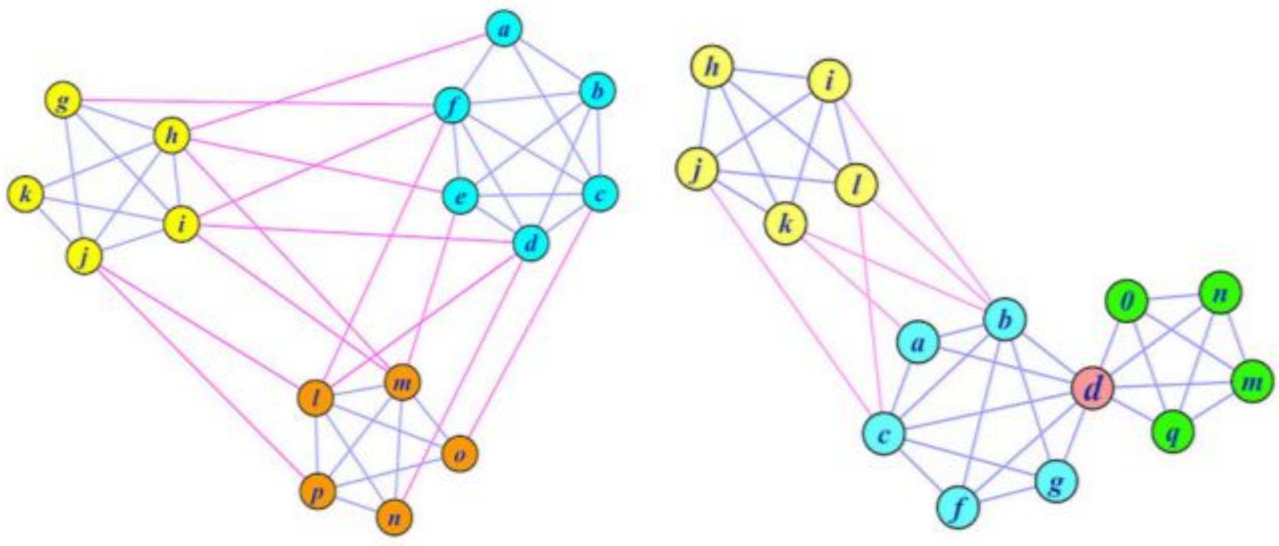
In opposite to non-overlapping community detection algorithms, where each node gets a unique label (and belongs to one community), nodes may belong to several communities



Communities

(a) Non-overlapping communities

(b) Overlapping (on “d” node) communities

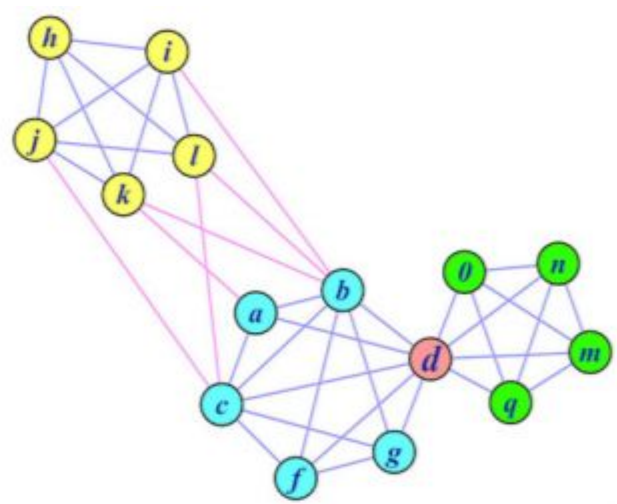


(a) (b)

Communities

Idea: duplicate nodes, and use Newman-Girvan

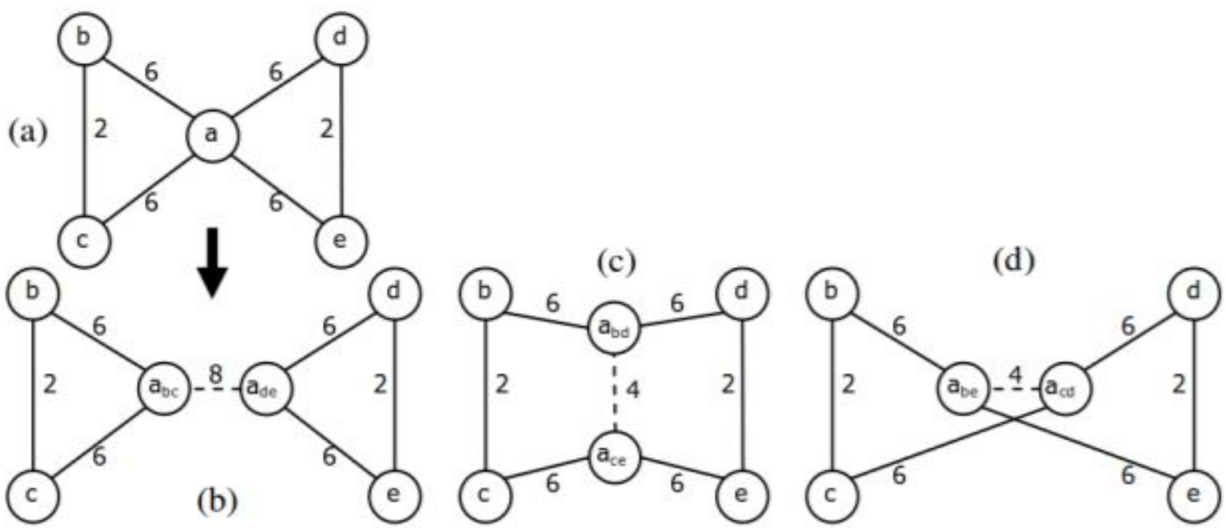
Which nodes to duplicate?



CONGO Algorithm

Cluster-Overlap Newman Girvan Optimized algorithm

Similar to Edge Betweenness – Split Betweenness



CONGO Algorithm

1. Compute Edge Betweenness for each edge and split betweenness for each node
2. Find node/edge with maximum betweenness
3. Remove the edge / Split the node
4. Recalculate 1
5. Repeat until no edges left

CONGO Algorithm

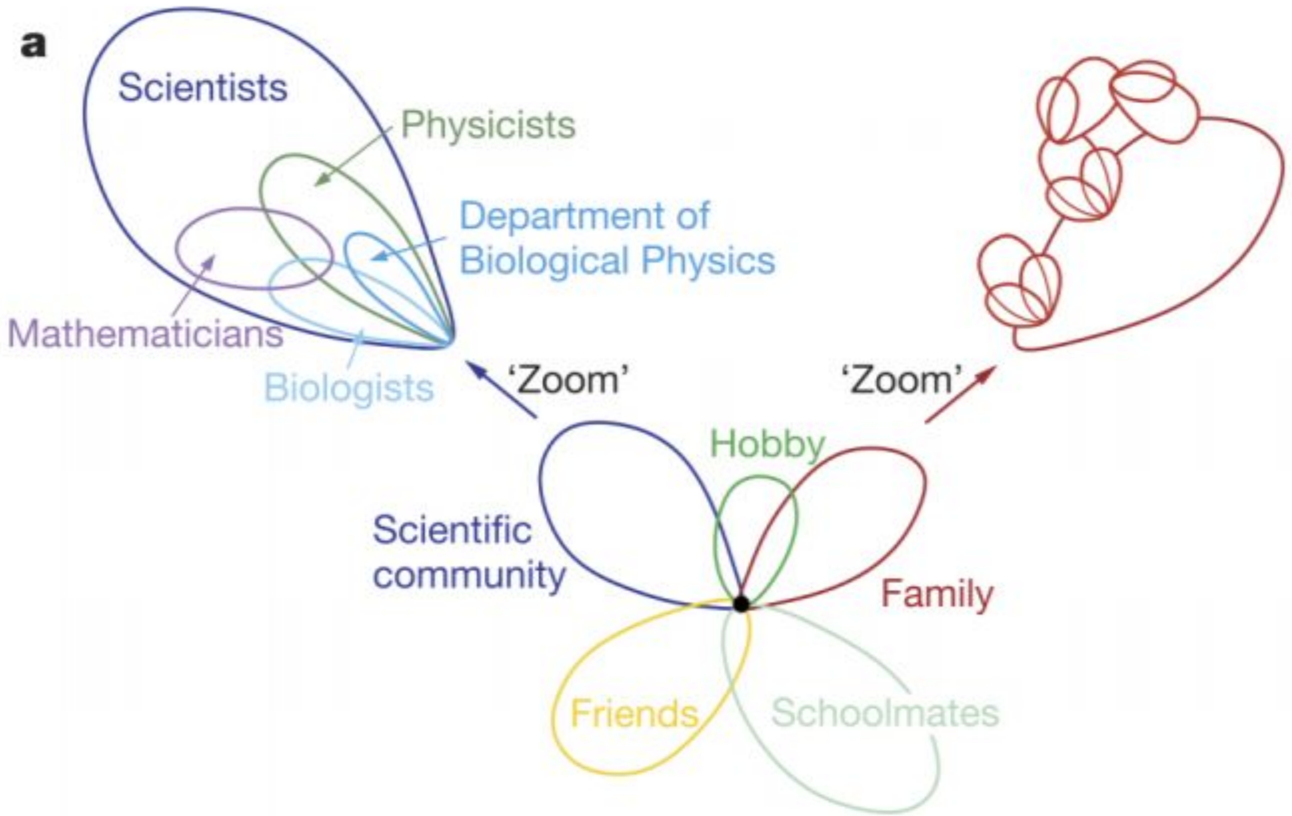
Pros:

- Similar to NG algorithm for non-overlap communities

Cons:

- Expensive computation

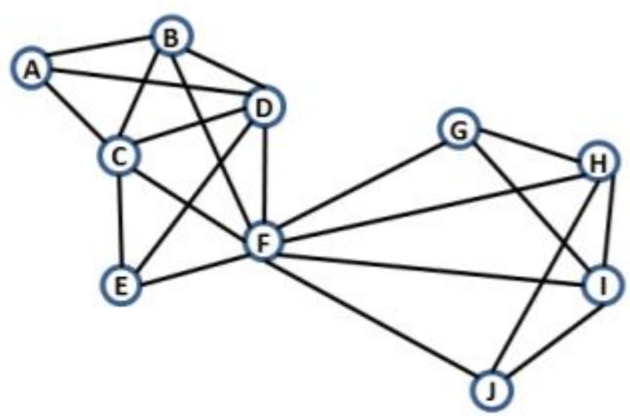
Overlapping Communities



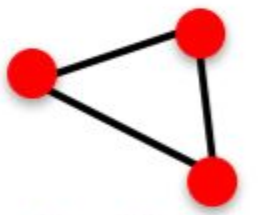
k-clique community

Definitions:

1. k-clique is a clique of k nodes
2. Adjacent k-cliques: if they share k-1 nodes
3. k-clique community – k-cliques that can be reached from each other via series of adjacent k-cliques

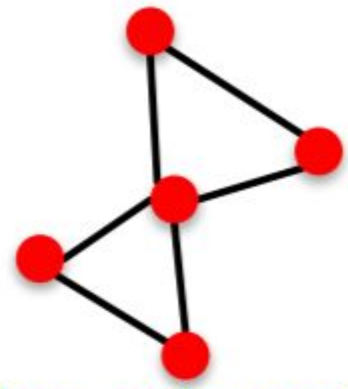


k-clique community

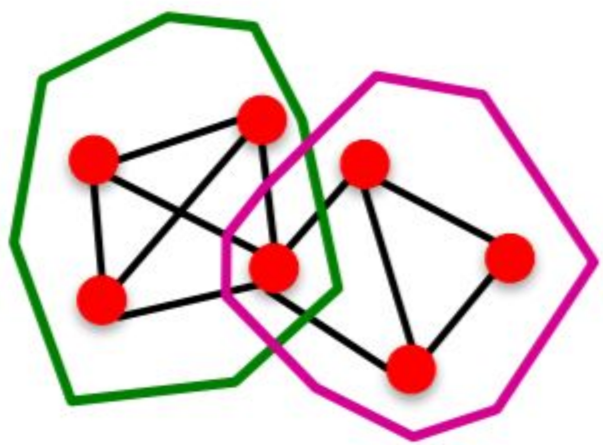


3-clique

Adjacent
3-cliques

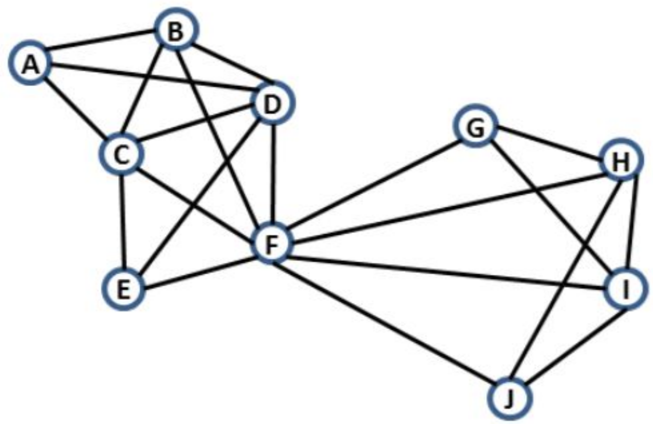


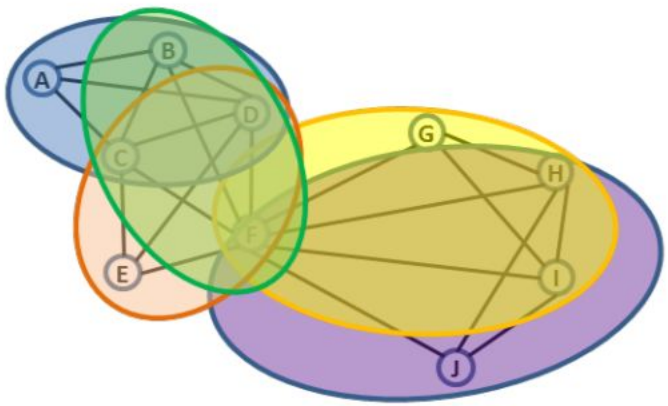
Non-adjacent
3-cliques



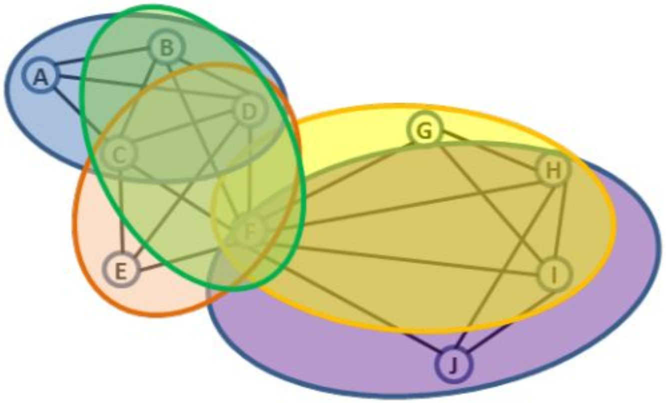
Two overlapping *3*-clique communities

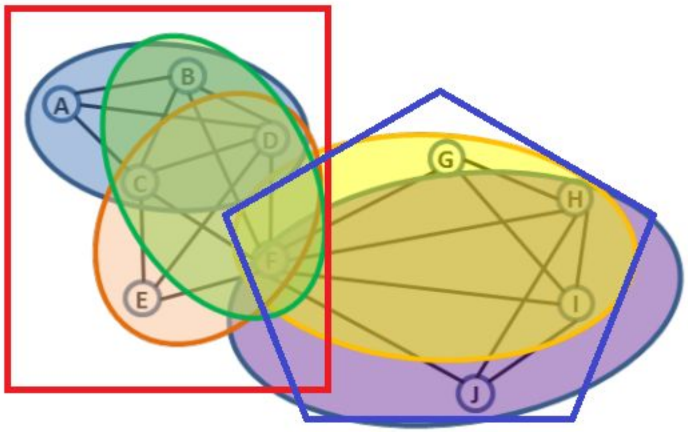
k-clique community





k-clique community





k-clique percolation method

By Palla et al. 2005:

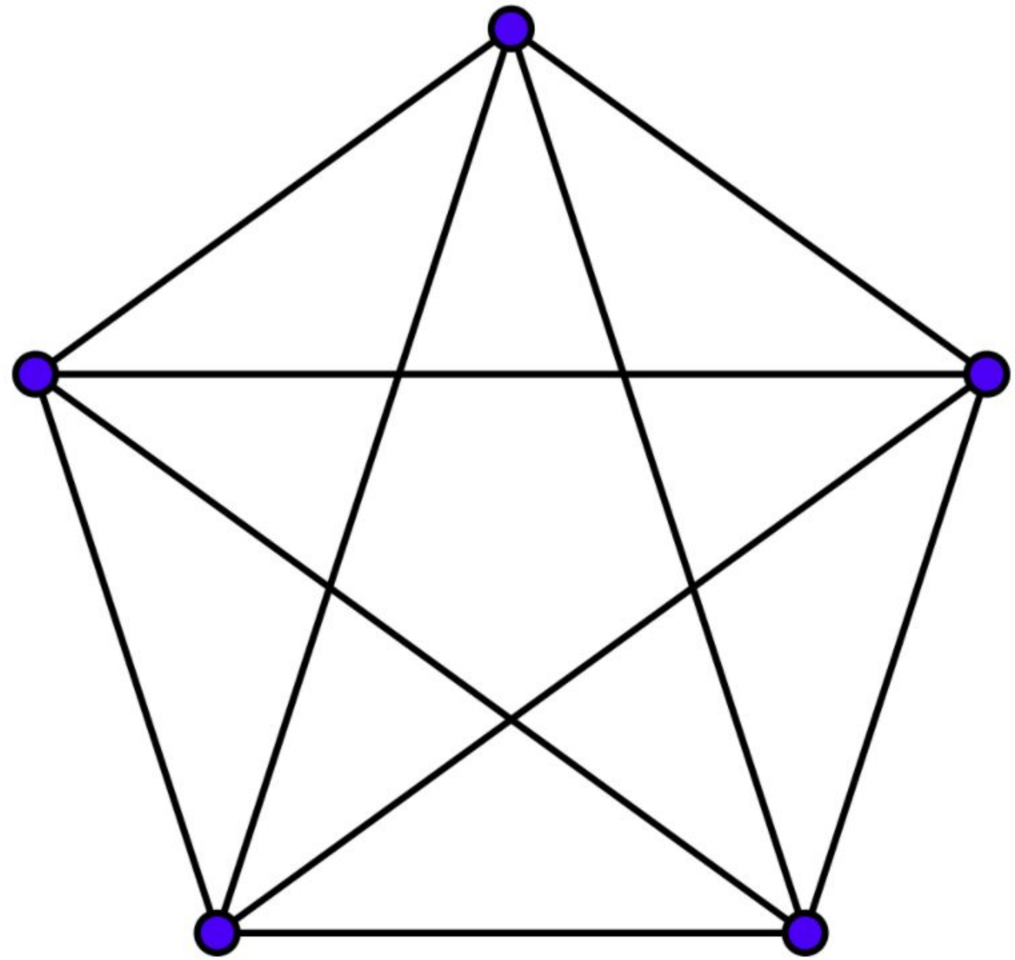
- Find all maximal cliques
- Create clique overlap matrix
- Threshold matrix with k-1
- Communities are connected components

Cliques and maximal cliques

5-clique

How many 4-cliques?

What can we say about them?



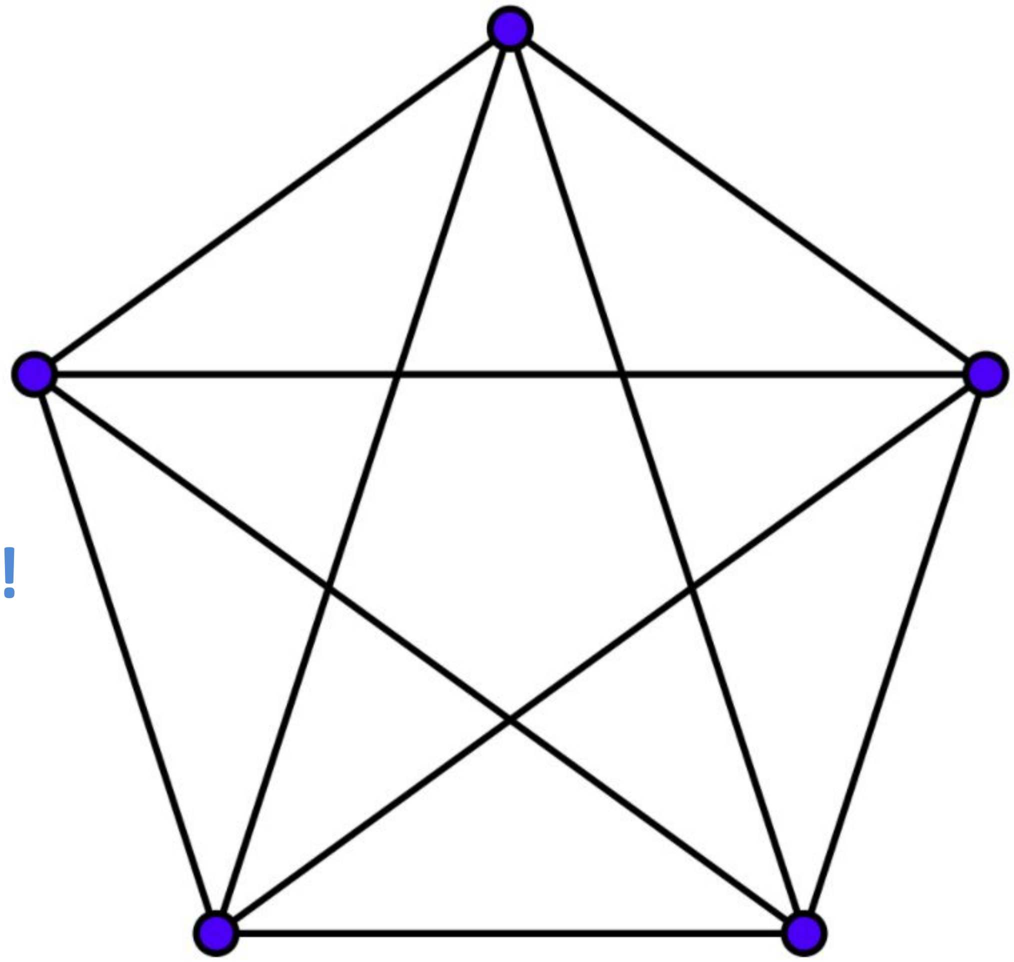
Cliques and maximal cliques

5-clique

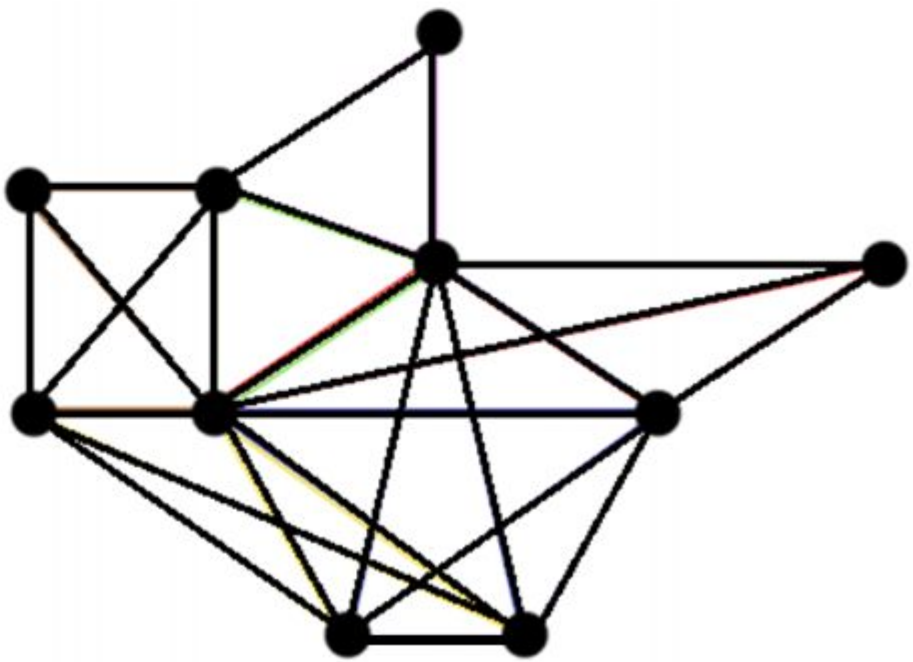
How many 4-cliques?

What can we say about them?

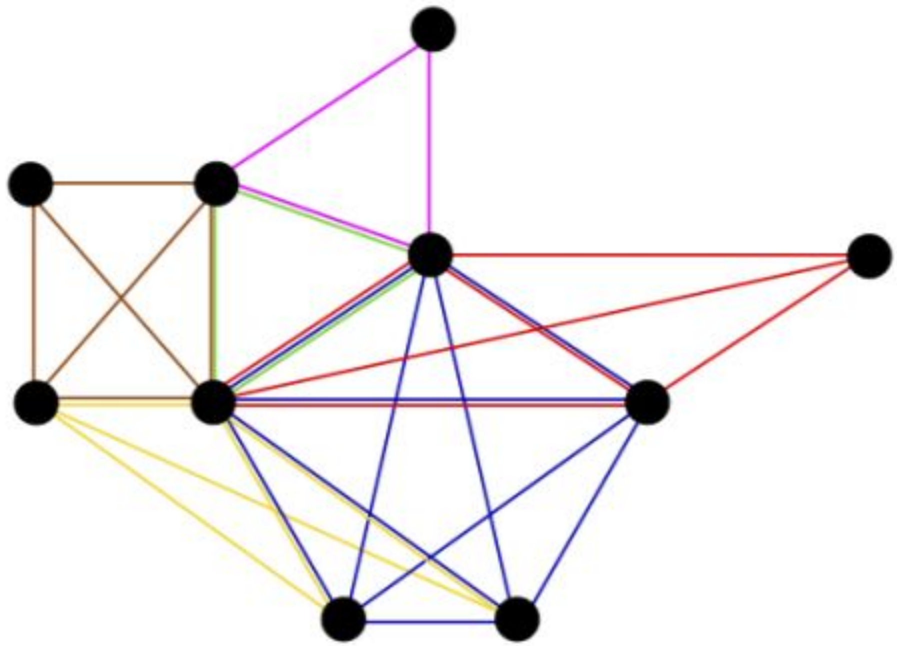
All 4-cliques are adjacent!



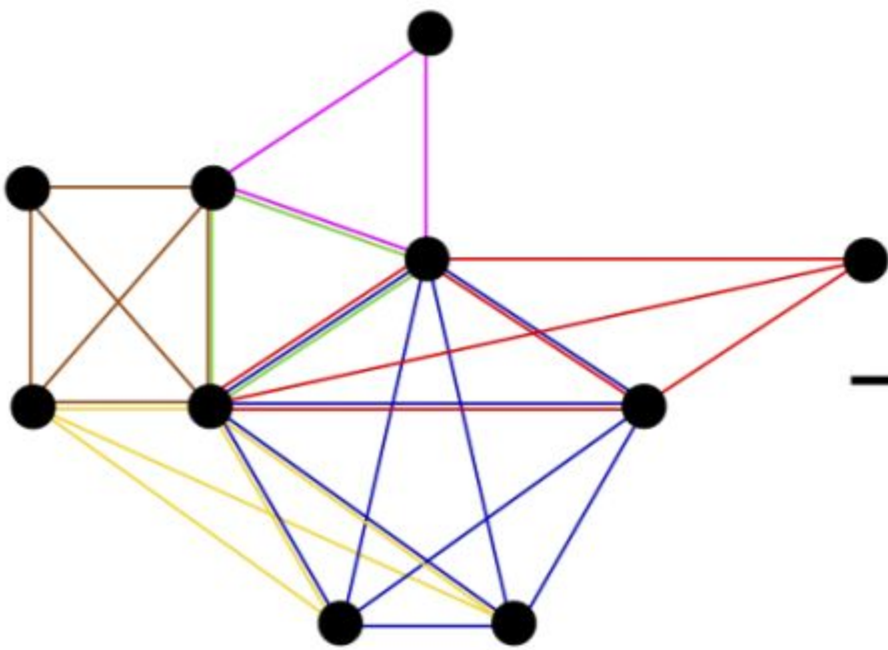
Example



Example

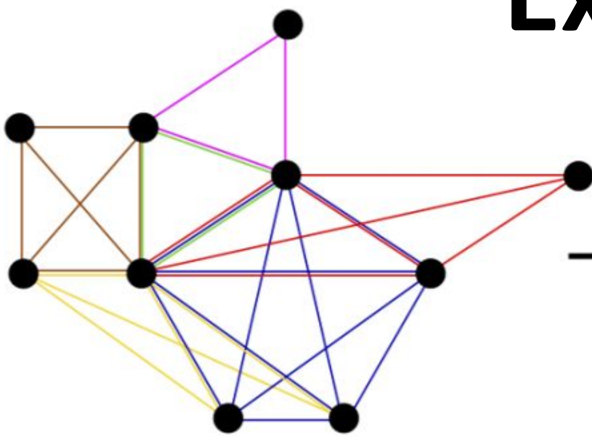


Example



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

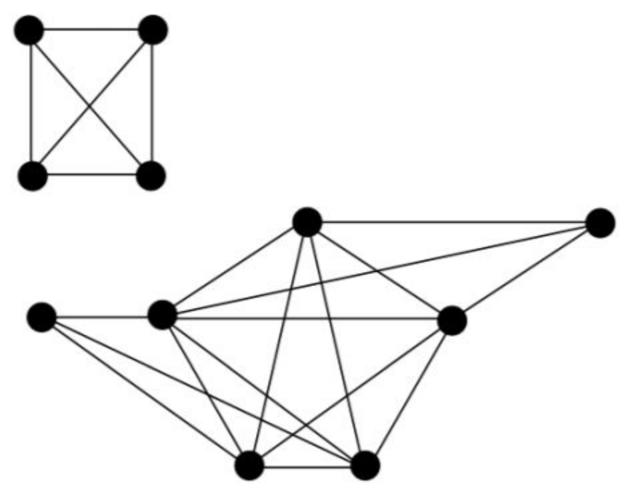
Example



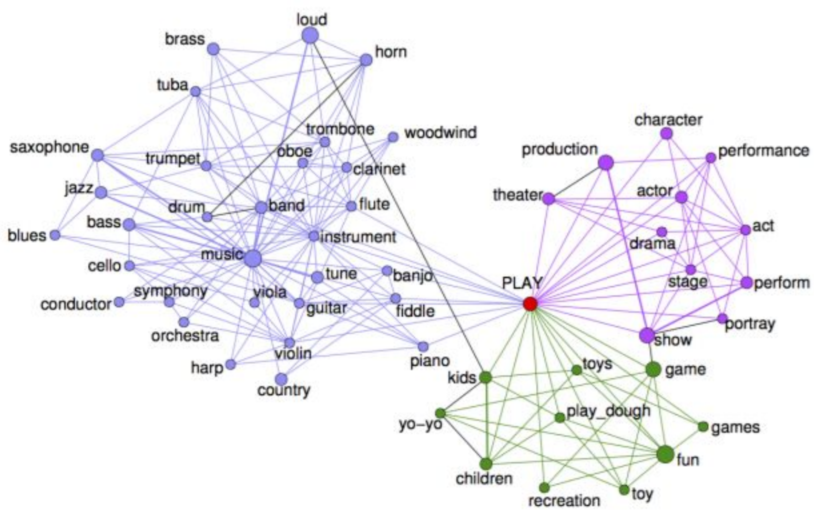
	blue	red	green	magenta	yellow	brown
blue	5	3	2	1	3	1
red	3	4	2	1	1	1
green	2	2	3	2	1	2
magenta	1	1	2	3	0	1
yellow	3	1	1	0	4	2
brown	1	1	2	1	2	4

k=4

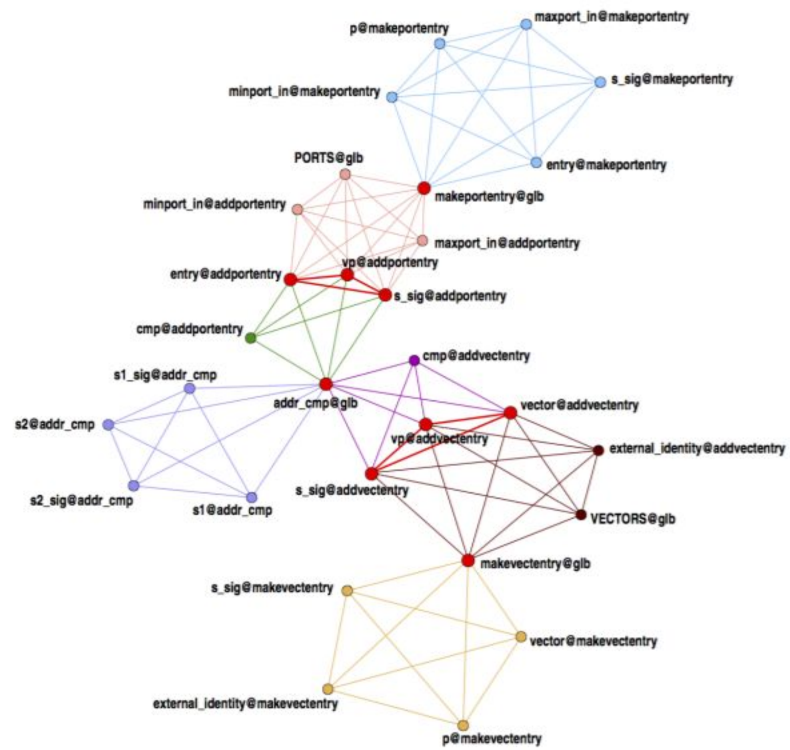
	blue	red	green	magenta	yellow	brown
blue	1	1	0	0	1	0
red	1	1	0	0	0	0
green	0	0	0	0	0	0
magenta	0	0	0	0	0	0
yellow	1	0	0	0	1	0
brown	0	0	0	0	0	1



More examples

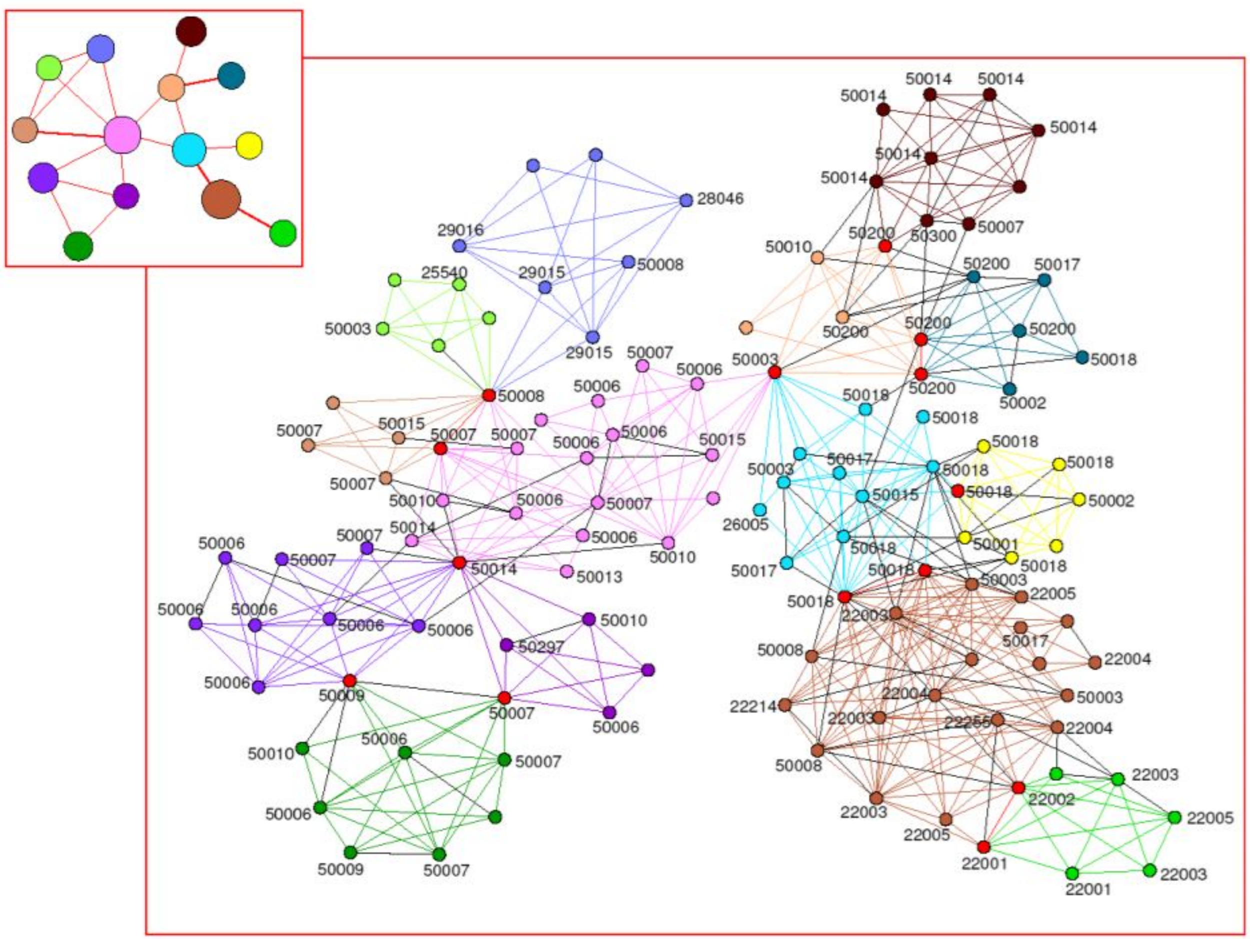


$k = 4$



$k = 5$

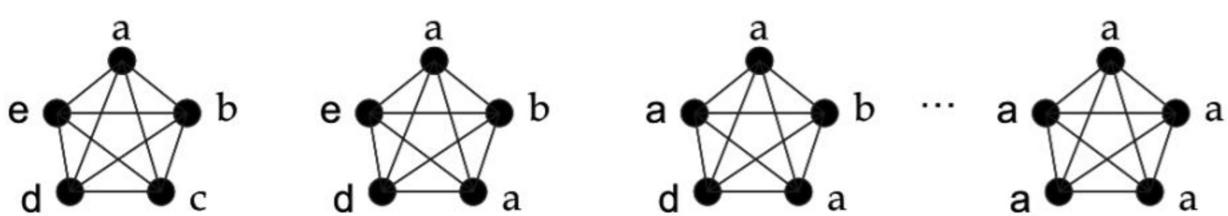
Phone call network



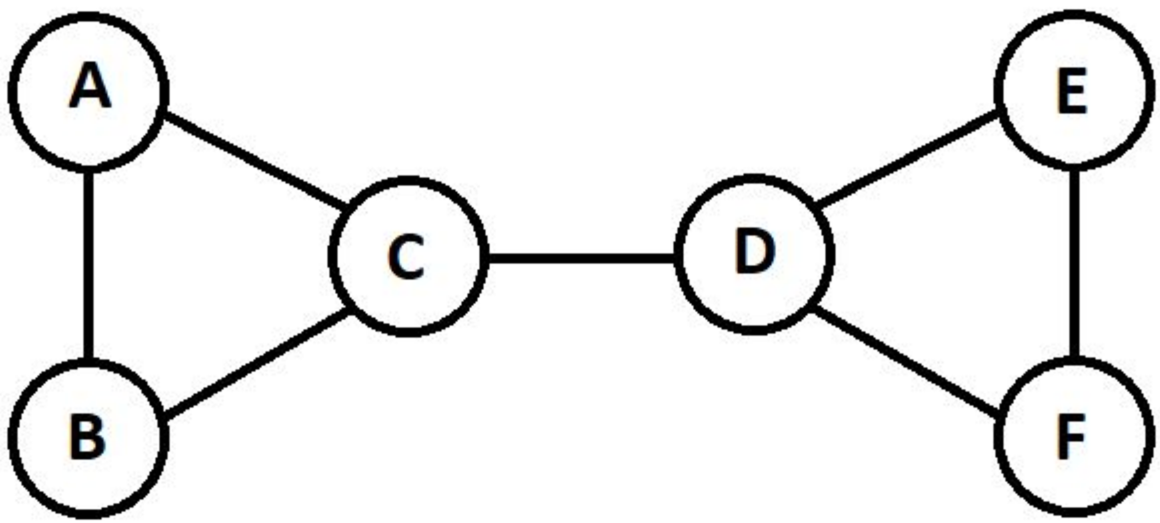
More methods

Label propagation

- Initialize labels on all nodes
- Randomized node order
- For every node replace its label with occurring with the highest frequency among neighbors (ties are broken uniformly randomly).
- If every node has a label that the maximum number of their neighbors have, then stop the algorithm

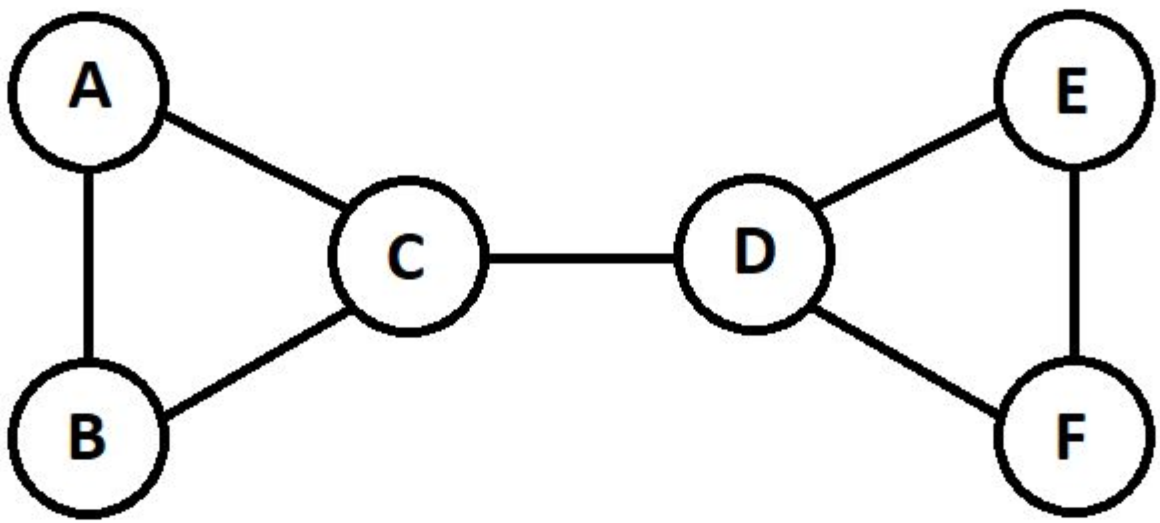


Example



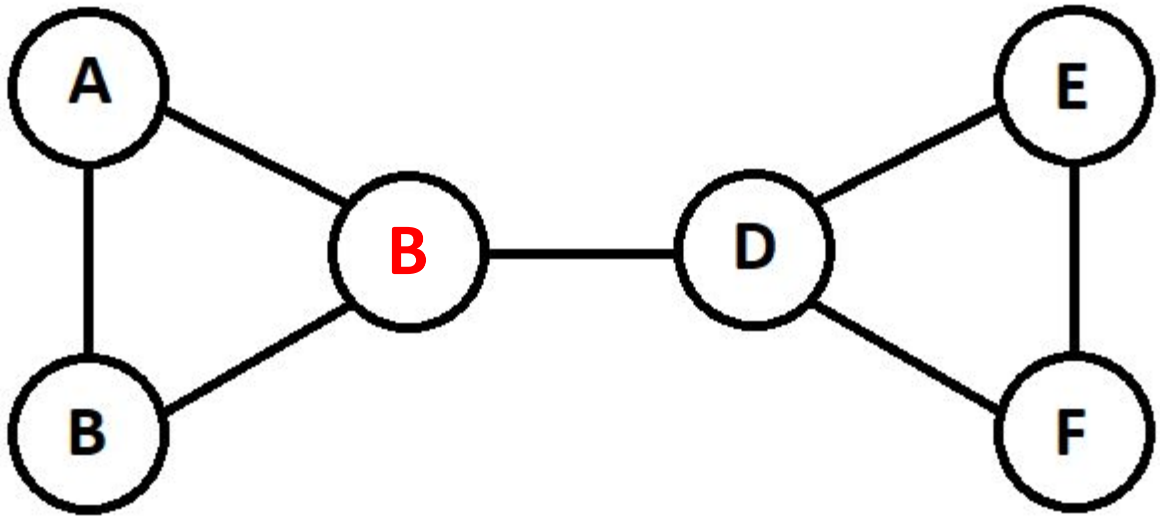
Example

Start from a random node see if it changes it’s label…



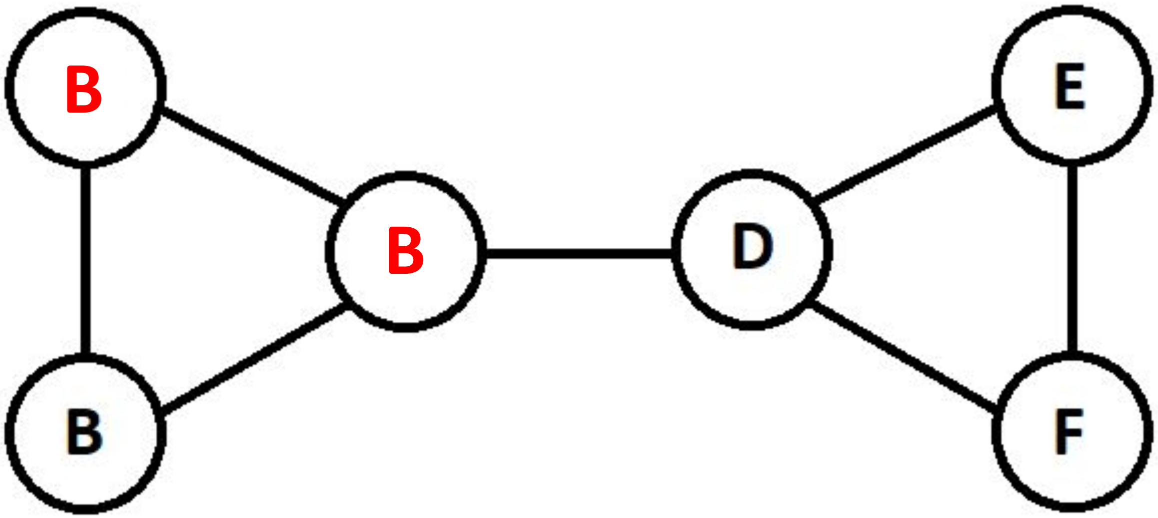
Example

C --> B



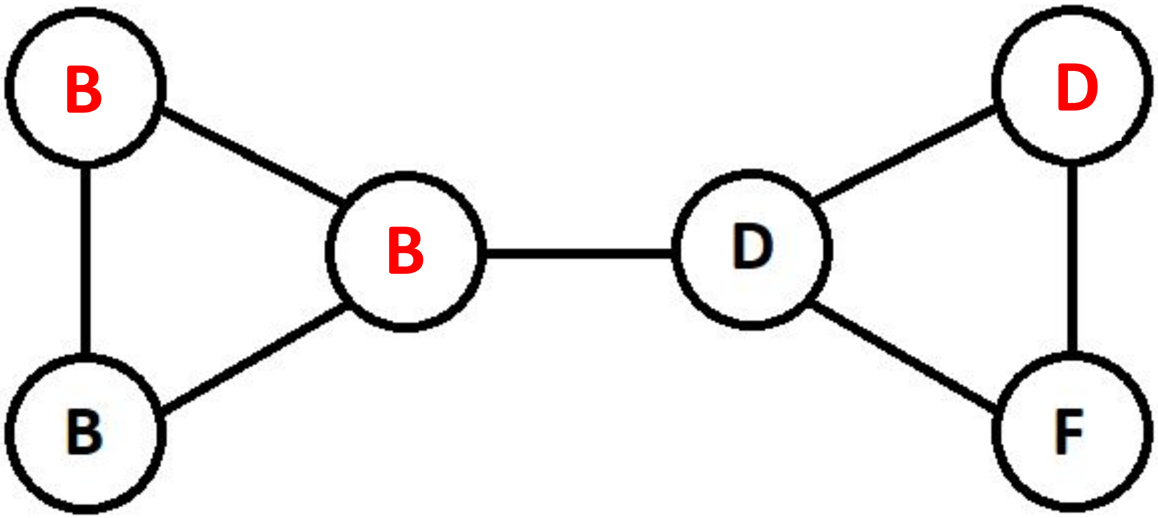
Example

A --> B



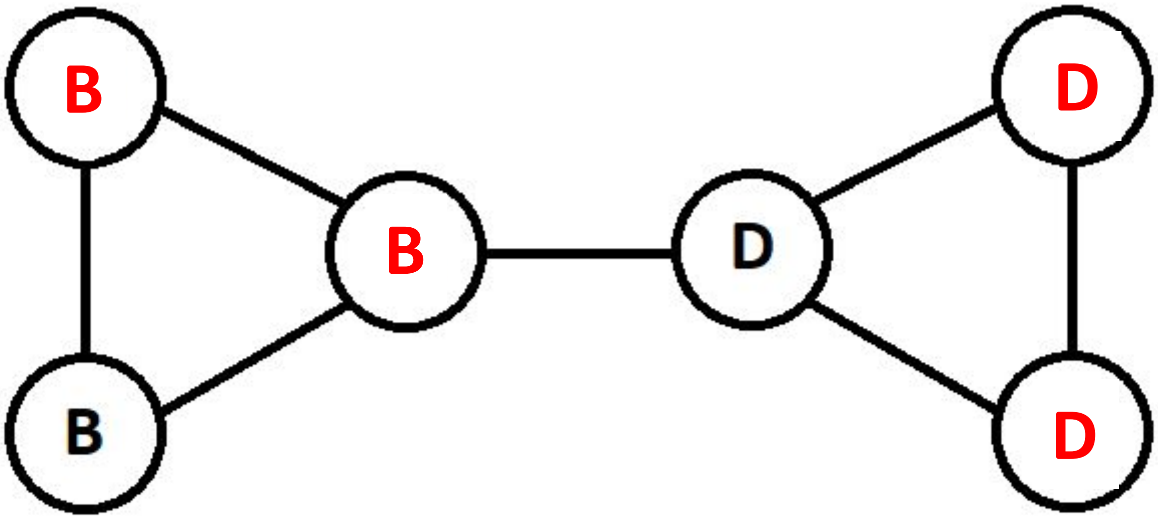
Example

E --> D

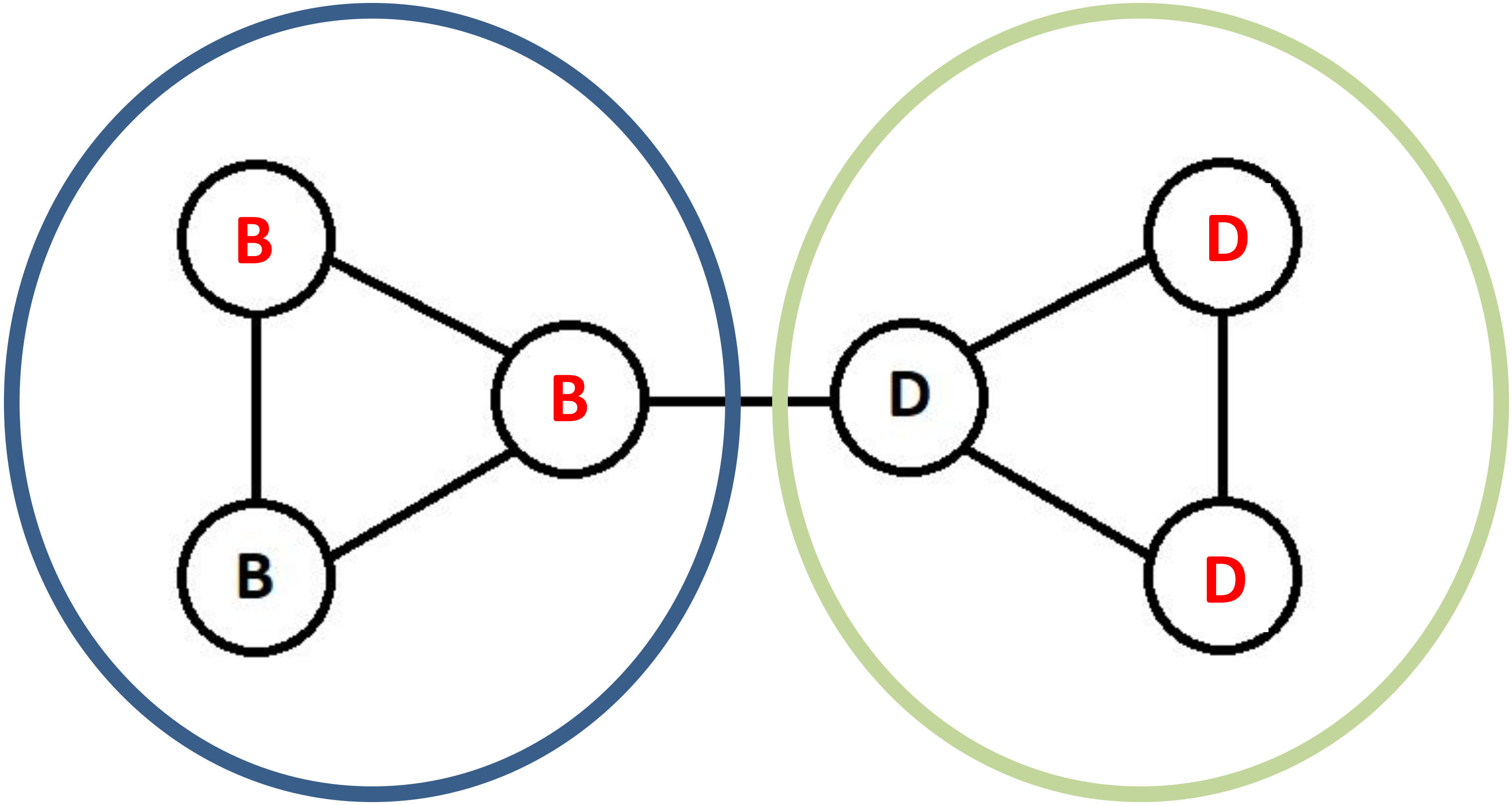


Example

F --> D



Example



Airports and flights example

Thank you!

Questions?