CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Energy Informatics

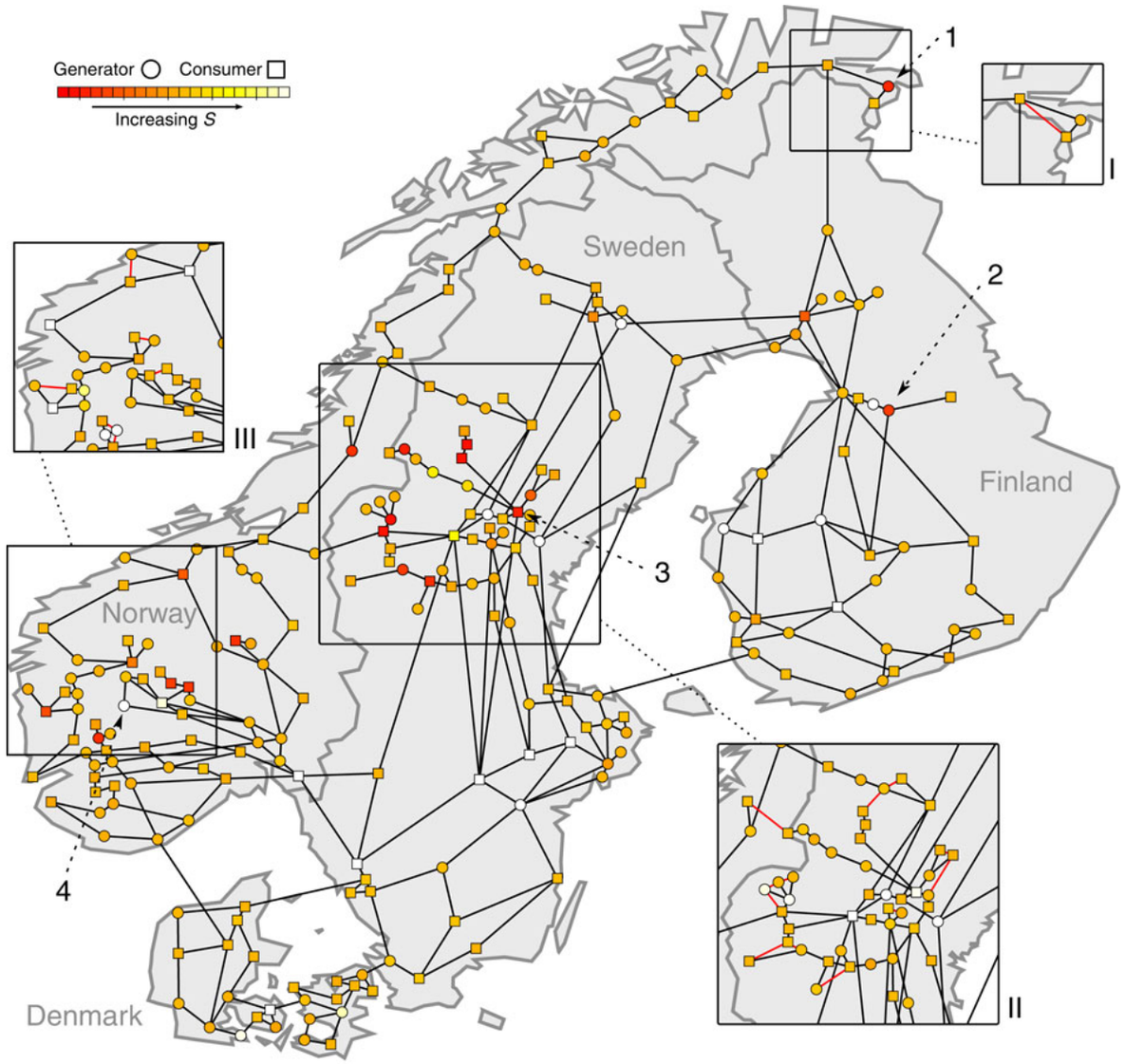
03 Network Algorithms

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg

UNI FREIBURG

Graph Theory in a Nutshell

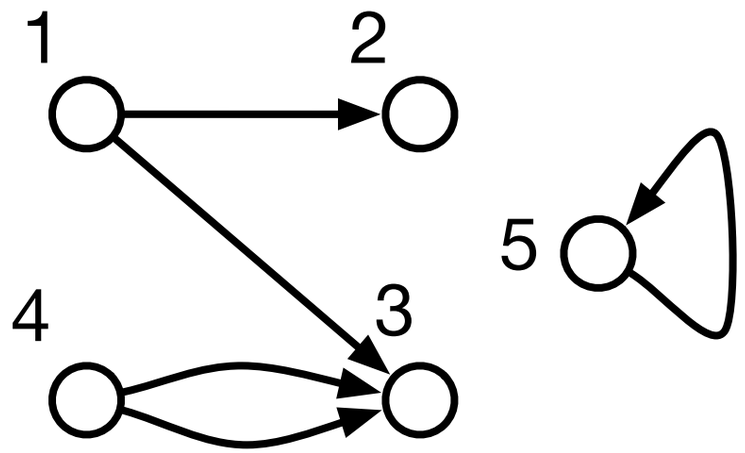
- A graph G=(V,E)
  - nodes/vertices V
  - edges E
    - connecting two nodes
- Variants
  - undirected/directed edge = lines/arrows
  - loops
    - edge connecting the node with itself
  - node/edge weights
    - mapping of numbers to the nodes/edges

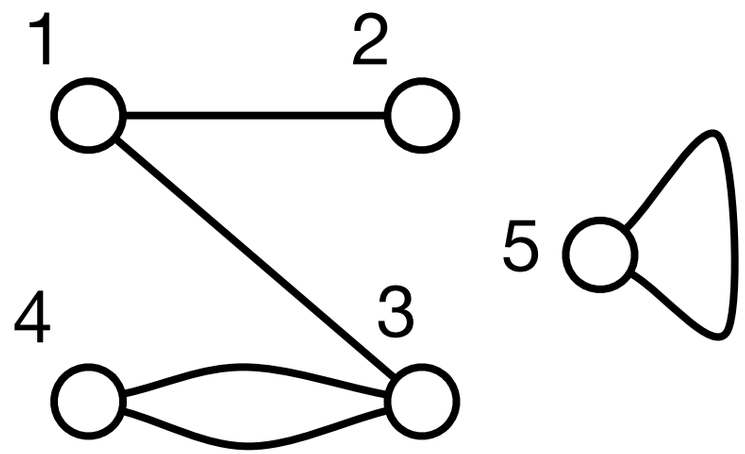


How dead ends undermine power grid stability
Peter J. Menck, Jobst Heitzig, Jürgen Kurths & Hans Joachim Schellnhuber
Nature Communications 5, Article number: 3969

Terms in Graphs

- Degree of a node
  - number of edges at a node u
- Regular graph
  - if the maximum degree = minimum degree in a graph
- Indegree/Out-degree
  - in case of directed graph (digraph), the number of edges pointing to/from a node u
- Two nodes u,v are **adjacent**
  - if they are connected via an edge
- A graph is simple, if there are no loops or no parallel edges
  - usually only simple graphs are considered

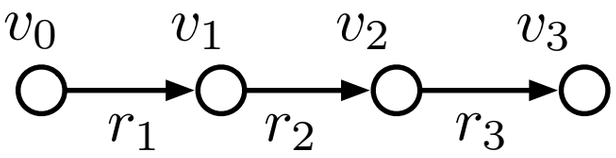




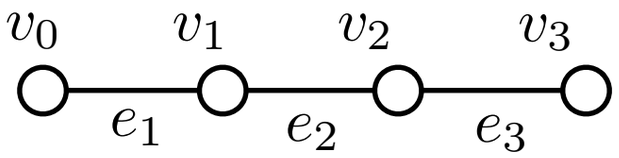
Paths

- A sequence of adjacent nodes is called a path
- Paths with same start and end are called cycles
- The length of a path is the number of edges passed
- A path is simple if no edge occurs twice
  - it is elementary if no node occurs twice

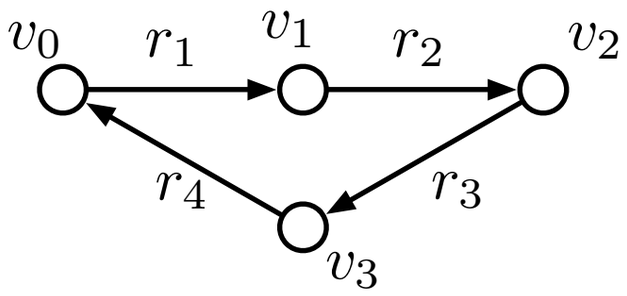
path of length 3



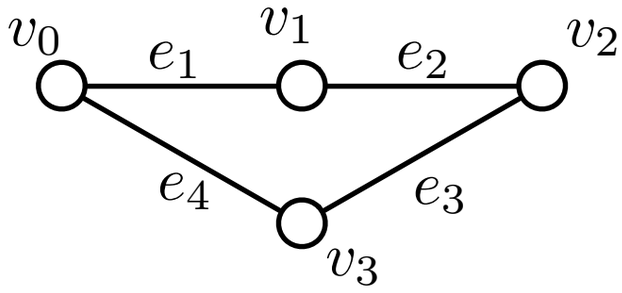
path of length 3



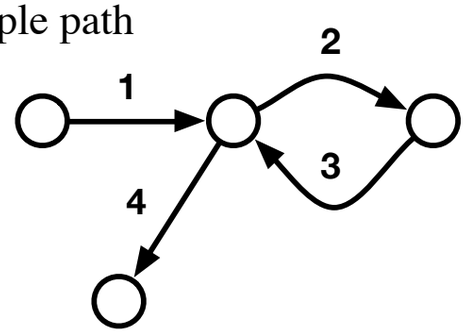
cycle of length 4



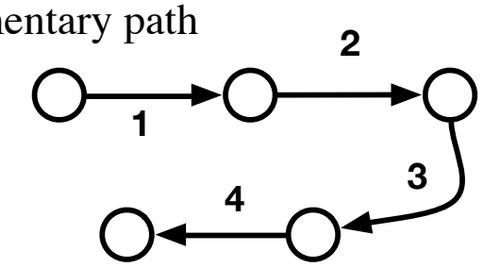
cycle of length 4



walk
simple path



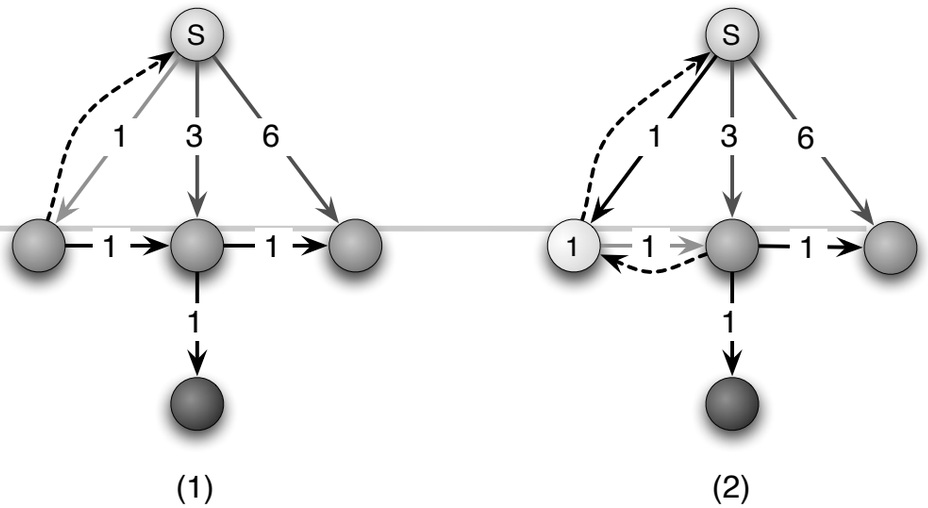
trail
elementary path



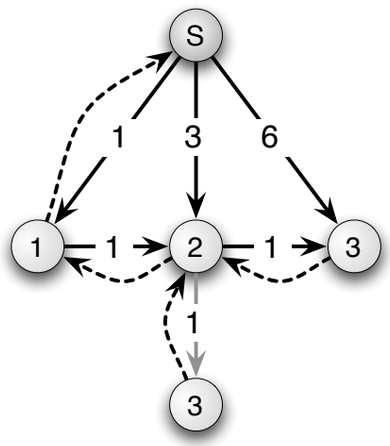
CoNe Freiburg

Shortest Paths

- Given
  - a graph G=(V,E)
  - start node s, target node t
- Compute the shortest path
- Dijkstras algorihm
  - Start with set S={s}
  - In each round
    - add the node u of the neighborhood of S
      - which has the shortest distance to s
    - store the edge used to u



(1) (2) (3) (4)

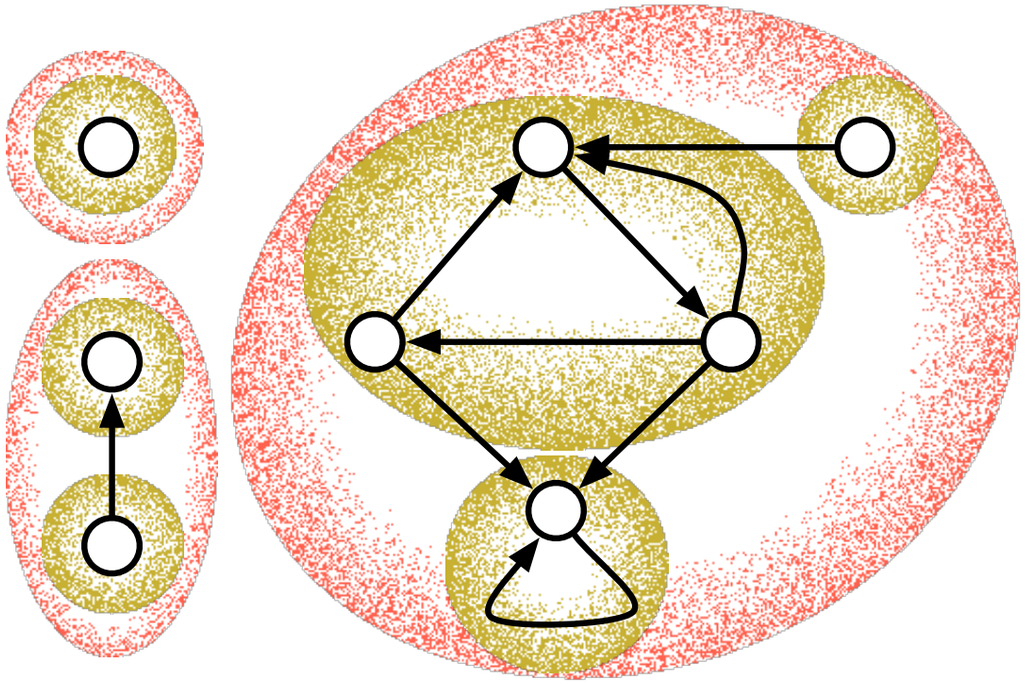


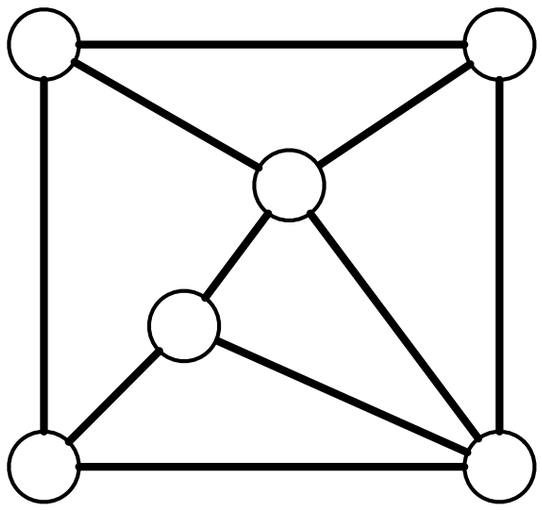
(5)

CoNe Freiburg

Connectivity

- An undirected graph is **connected**
  - if for all nodes u,v there exists a path connecting u and v
- A directed graph is **weakly connected**
  - if the corresponding undirected graph is connected
- A directed graph is **strongly connected**
  - if for all nodes u,v there exists a directed path connecting u and v
- A graph is **k-(vertex)-connected**, if there are k node disjoint paths between all nodes
- analog definition for d-edge connected

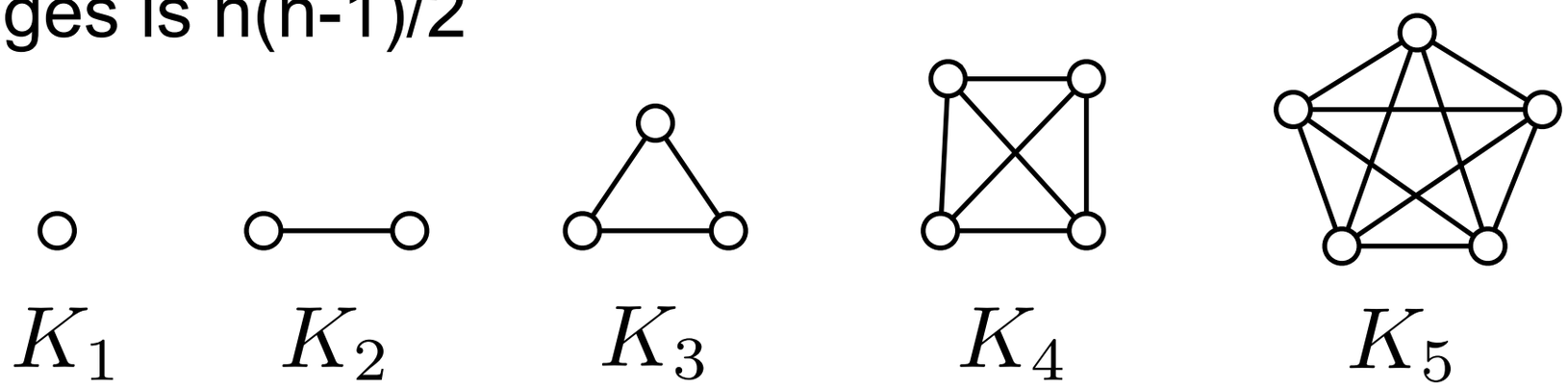




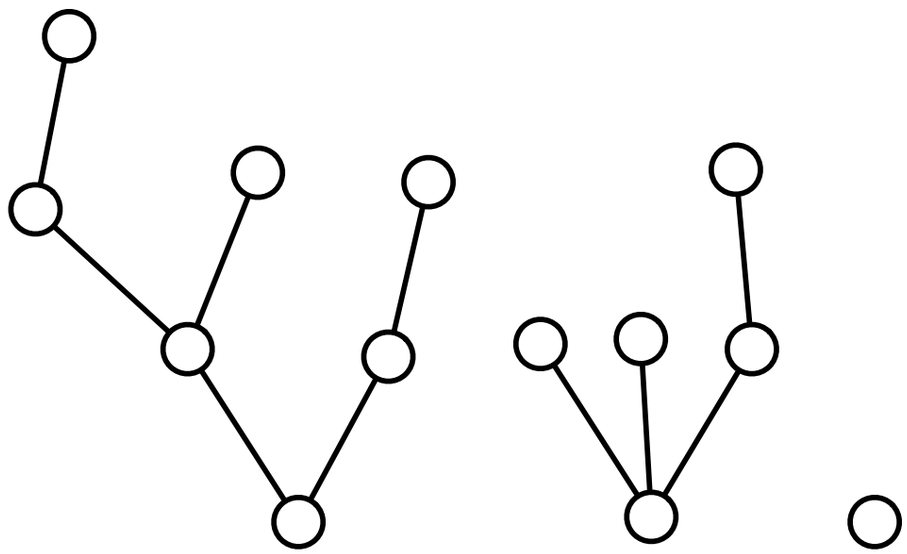
CoNe Freiburg

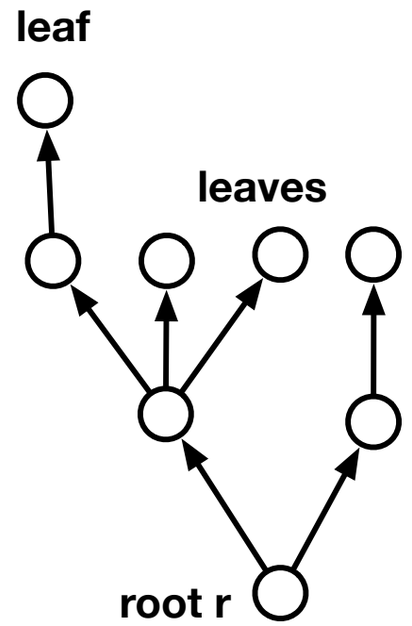
Special Graphs

- Complete (simple) undirected graphs
  - no. of edges is n(n-1)/2



- Trees are connected undirected graphs without cycles
  - sets of graphs are called forests

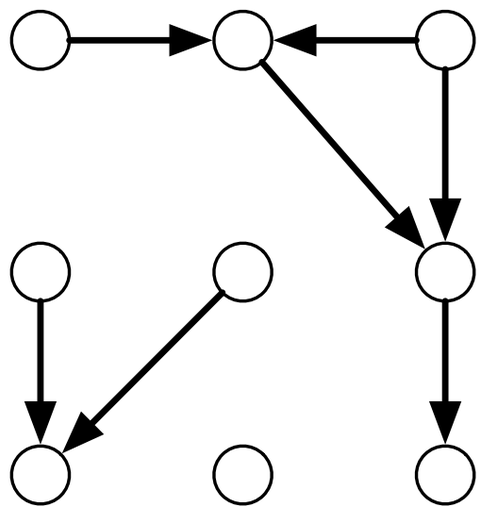




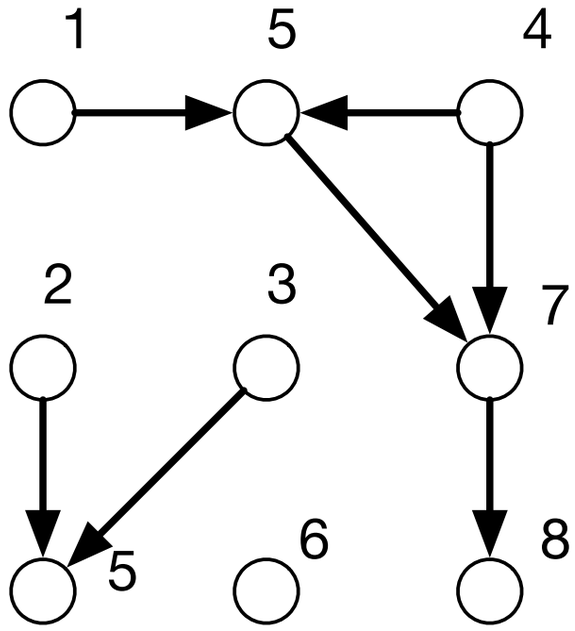
UNI FREIBURG

DAG: Directed Acyclic Graph

- Directed acyclic graph

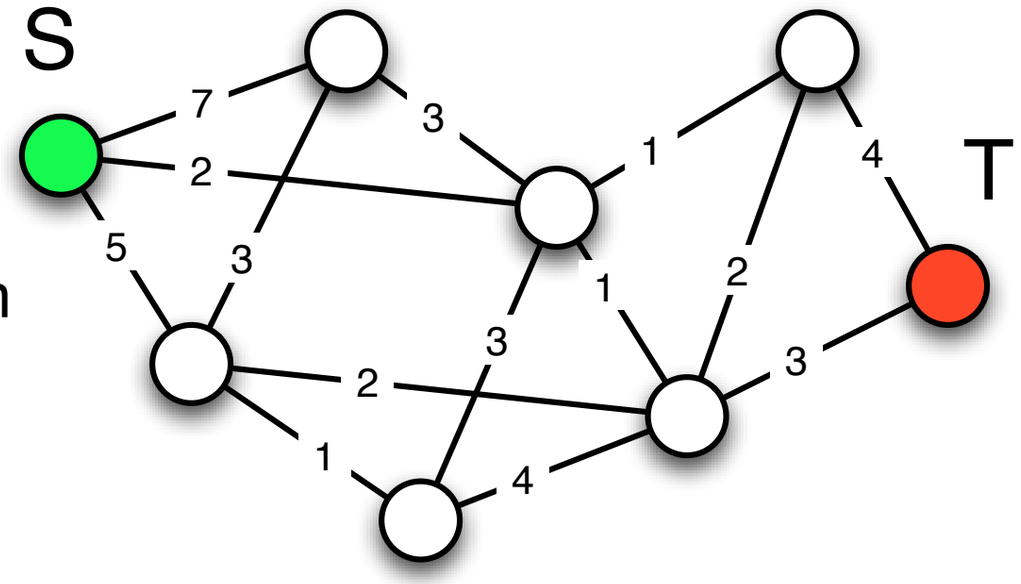


- Topologic Sorting
  - mapping f of {1,..,n} to V such that for edge (u,v)
    - f(u) < f(v)



Flows in Networks

- Motivation
  - Optimize flow from source to target
- Definition:
  - (Single-commodity) maximum flow problem
  - Given
    - a graph G=(V,E)
    - a capacity function w:E→R^+_0,
    - source set S and target set T
  - Find a maximum flow from S to T

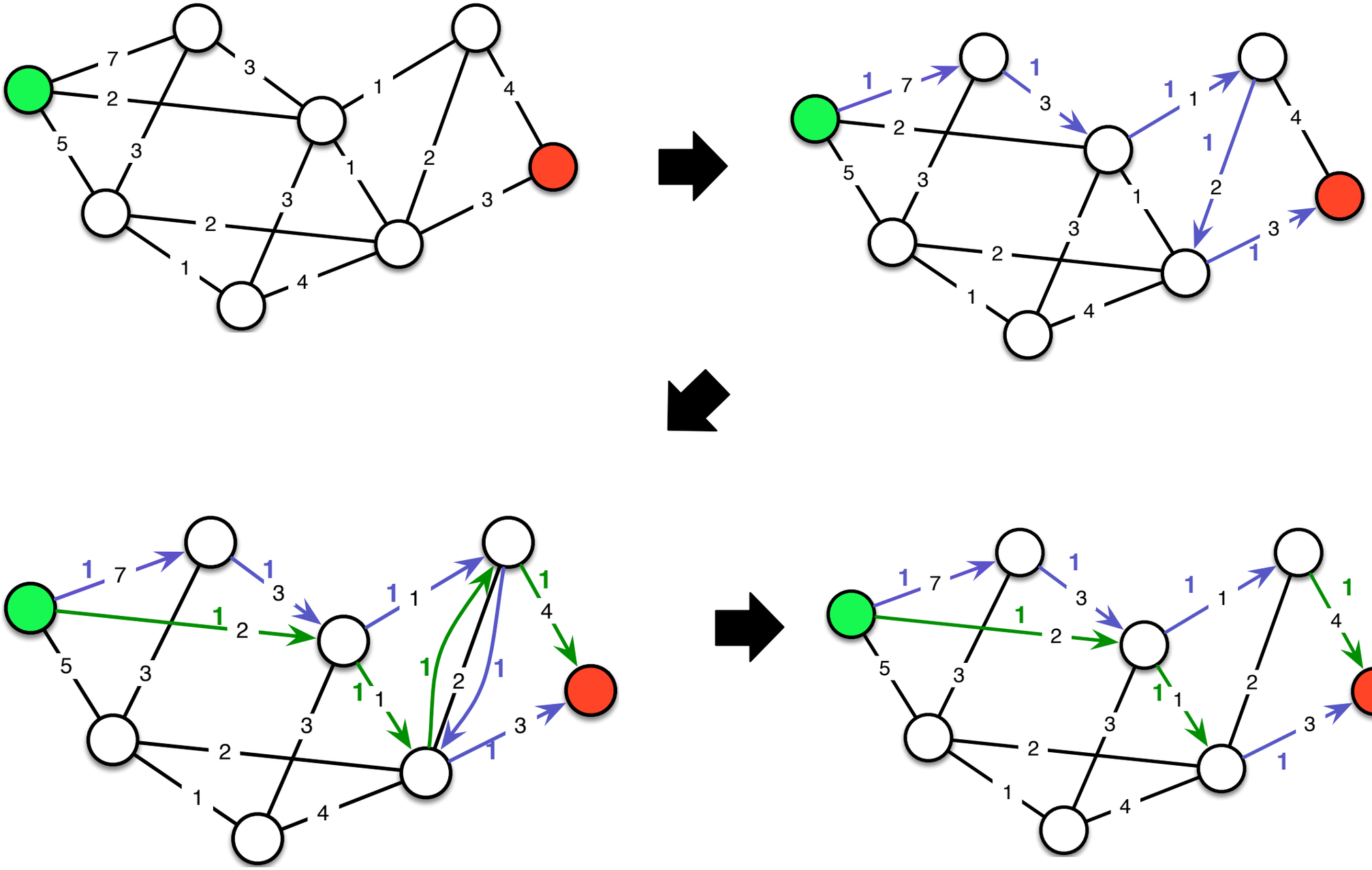


- A flow is a function f : E → R_0^+ such that
  - for all e ∈ E: f(e) ≤ w(e)
  - for all e ∉ E: f(e) = 0
  - for all u,v ∈ V: f(u,v)≥0

$$\forall u \in V \setminus (S \cup T) \quad \sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$$

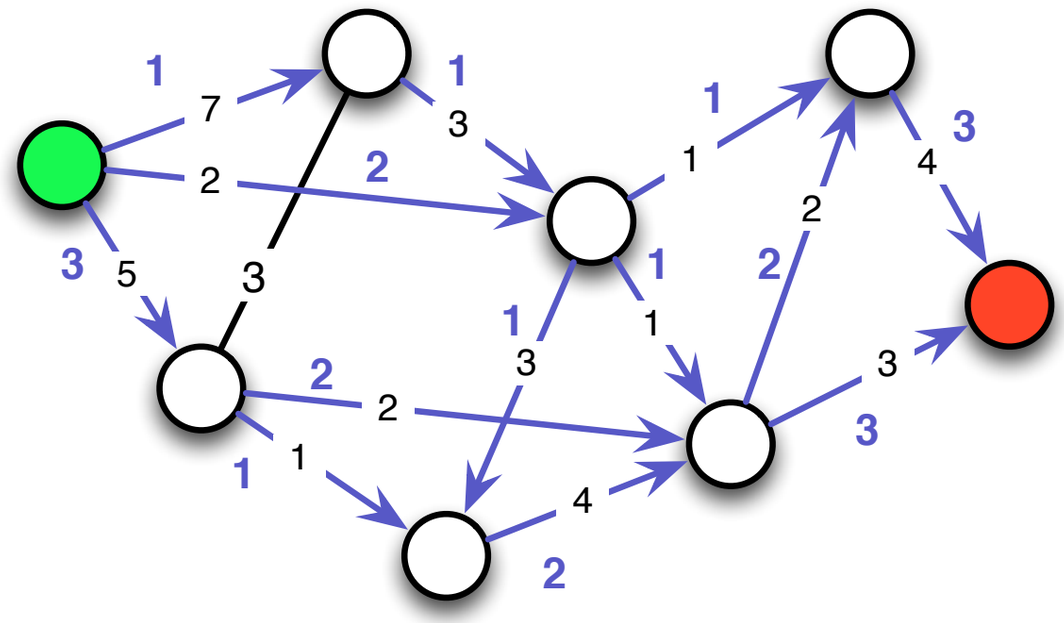
- Maximize flow $\sum_{u \in S} \sum_{v \in V} f(u,v)$

Flows in Networks



Computation of the Maximum Flow

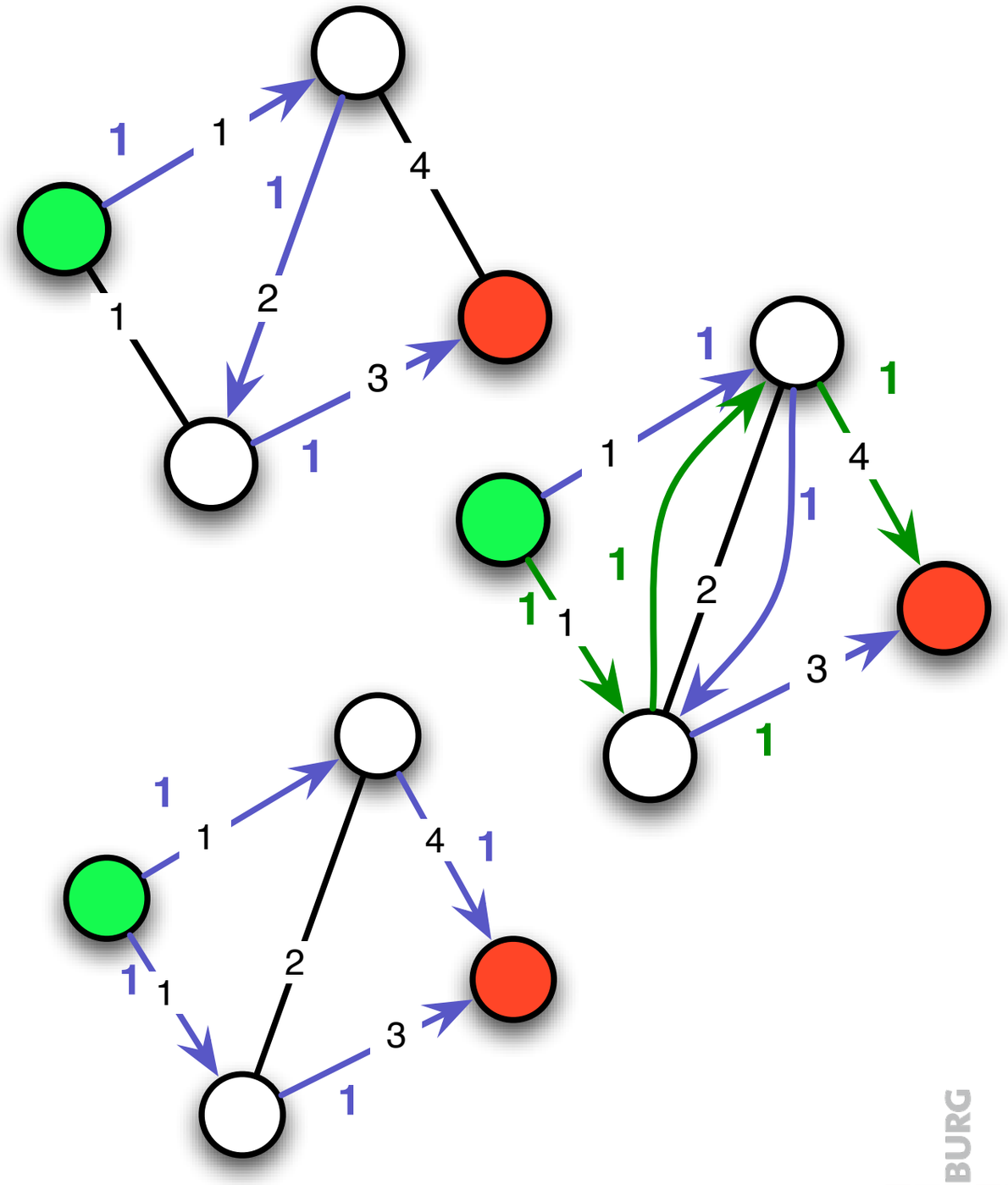
- Every natural pipe system solves the maximum flow problem
- Algorithms
  - Linear Programming
    - for real numbers
    - the flow is described by equations of a linear optimization problem
    - Simplex algorithm (or Ellipsoid method) can solve any linear equation system
  - Ford-Fulkerson
    - also for integers
    - as long as open paths exist, increase the flow on theses paths
      - open path: path which increases the flow
  - Edmonds-Karp
    - special case of Ford-Fulkerson
    - use BFS (breadth first search) to find open paths



CoNe Freiburg

Ford-Fulkerson

- Find a path from the source node to the target node
  - where the capacity is not fully utilized
  - or which reduces the existing flow
- Compute the maximum flow on this augmenting path
  - by the minimum of the flow that can be added on all paths
- Add the flow on the path to the existing flow
- Repeat this step until no flow can be added anymore



CoNe Freiburg

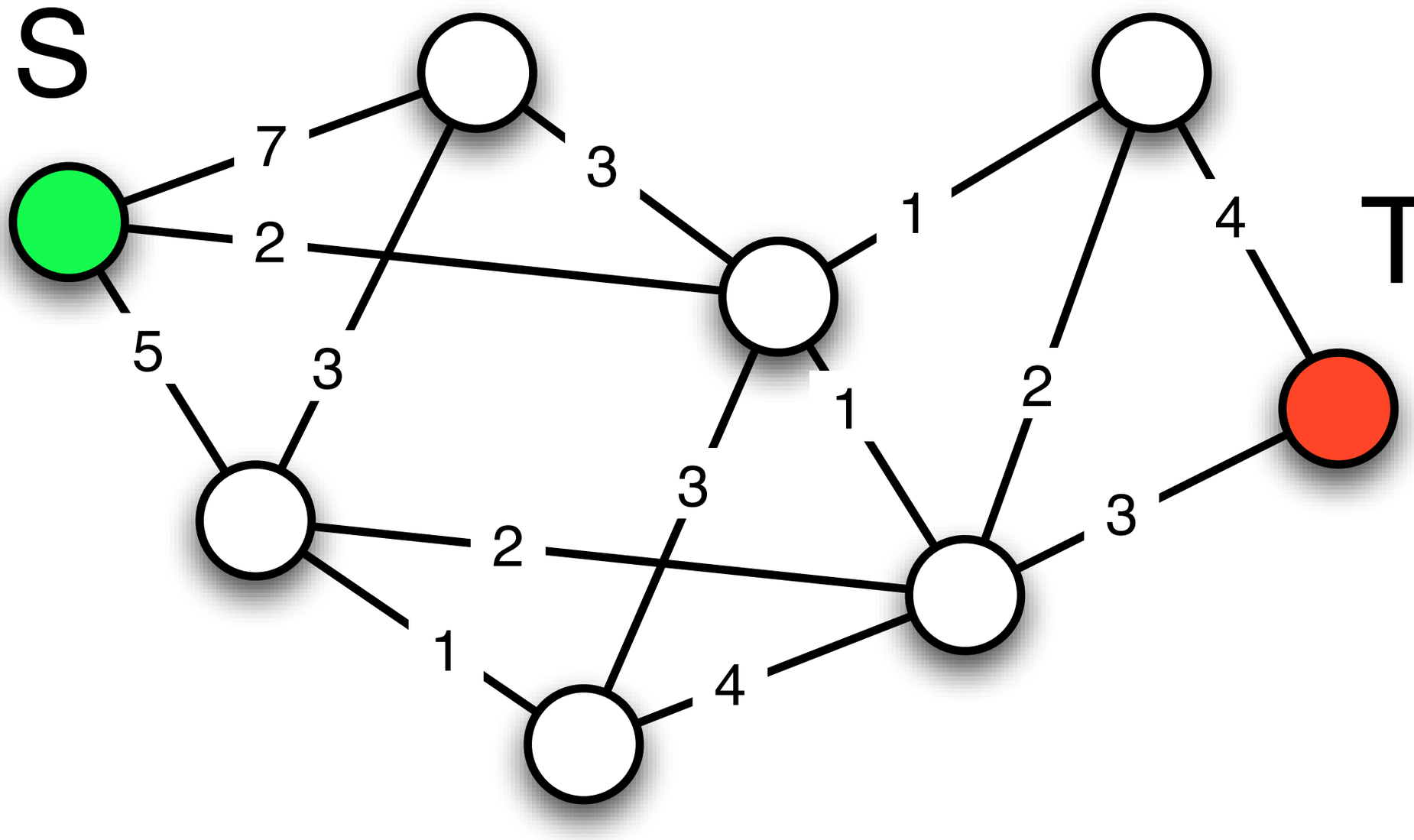
Edmunds-Karp

- **Search path for Ford-Fulkerson algorithm**
- **Choose the shortest augmenting path**
  - Computation by breadth-first-search
- **leads to run-time $O(|V| |E|^2)$**
  - whereas Ford-Fulkerson could have exponential run-time

UNI FREIBURG

CoNe
Freiburg

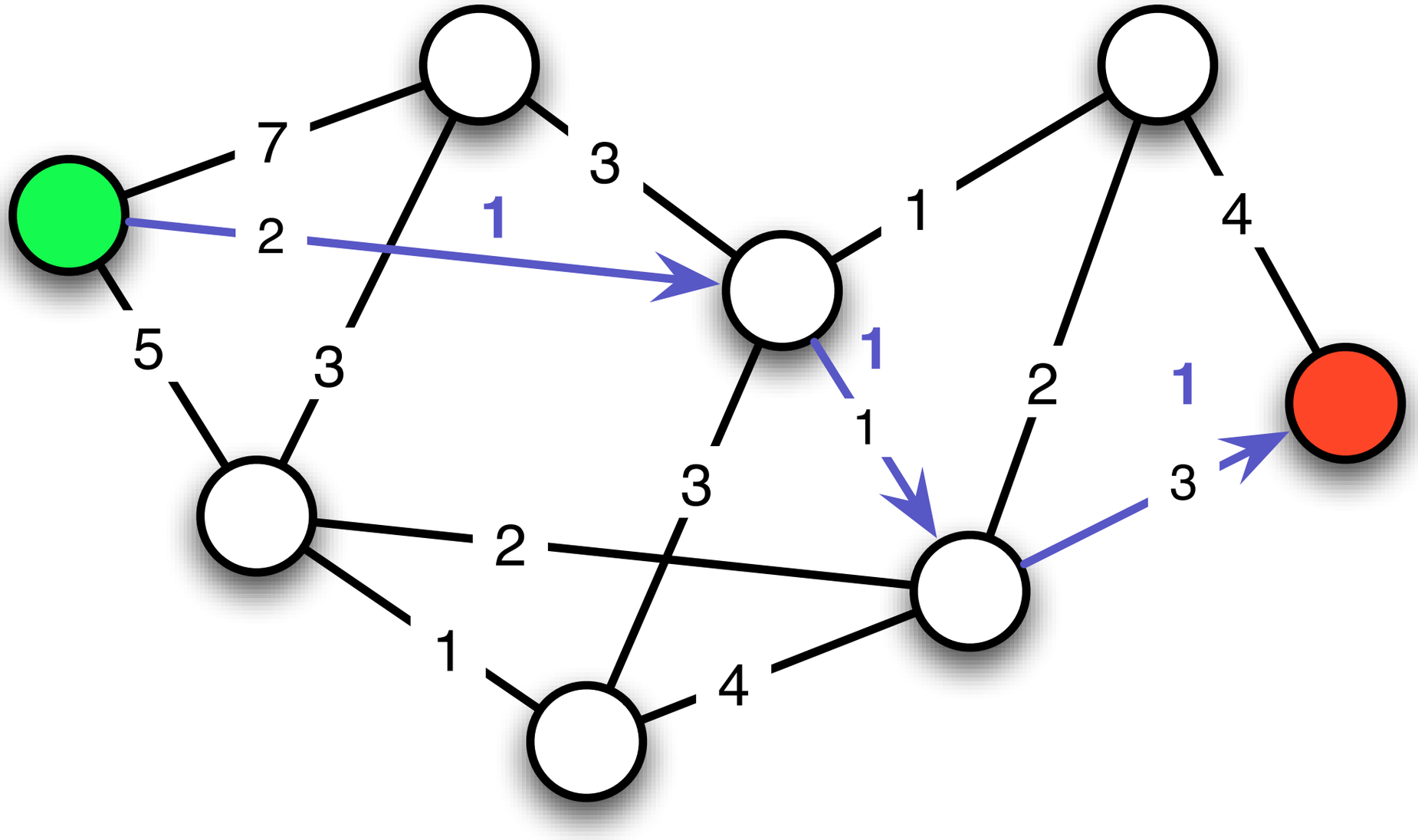
Example



UNI FREIBURG

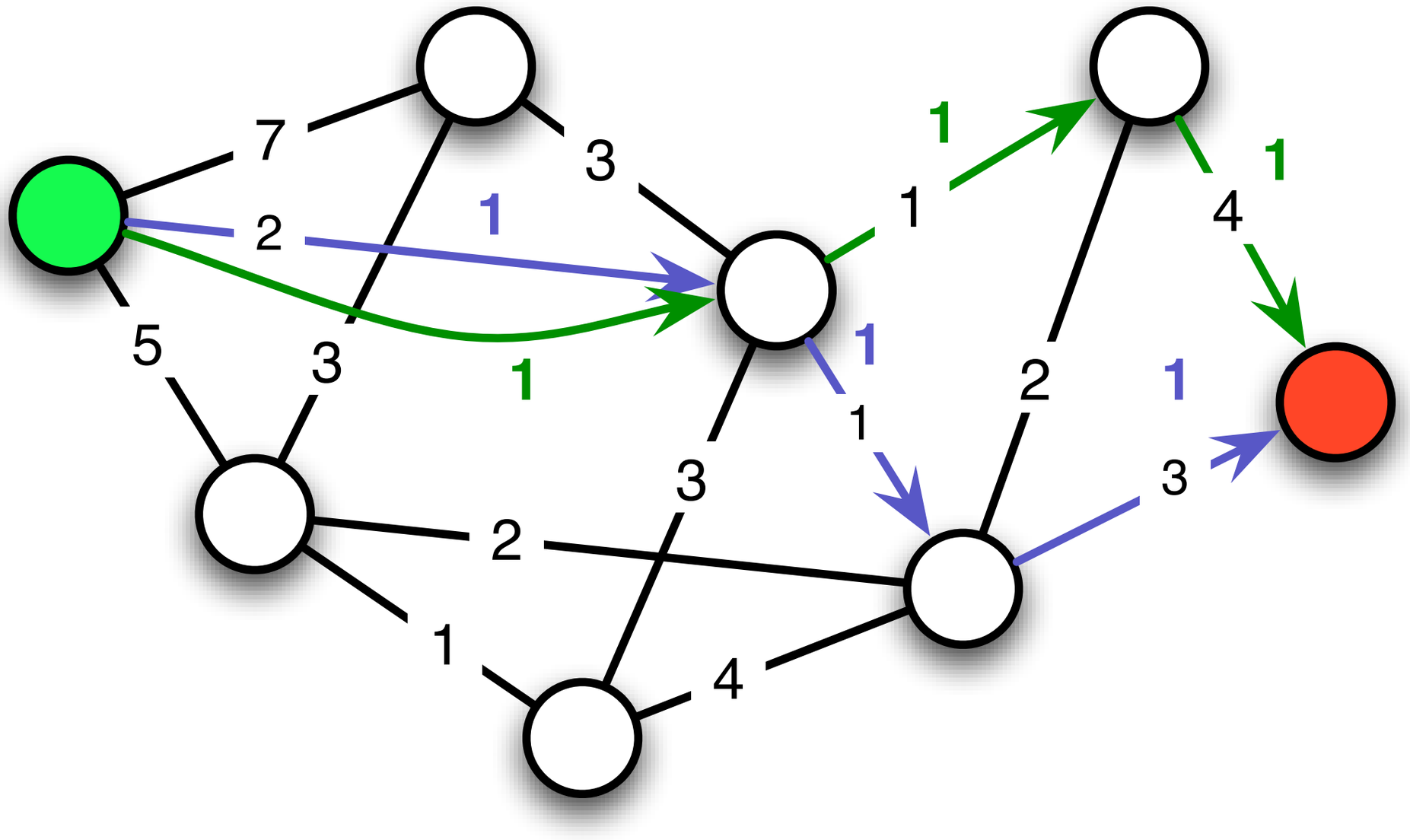
CoNe
Freiburg

Example



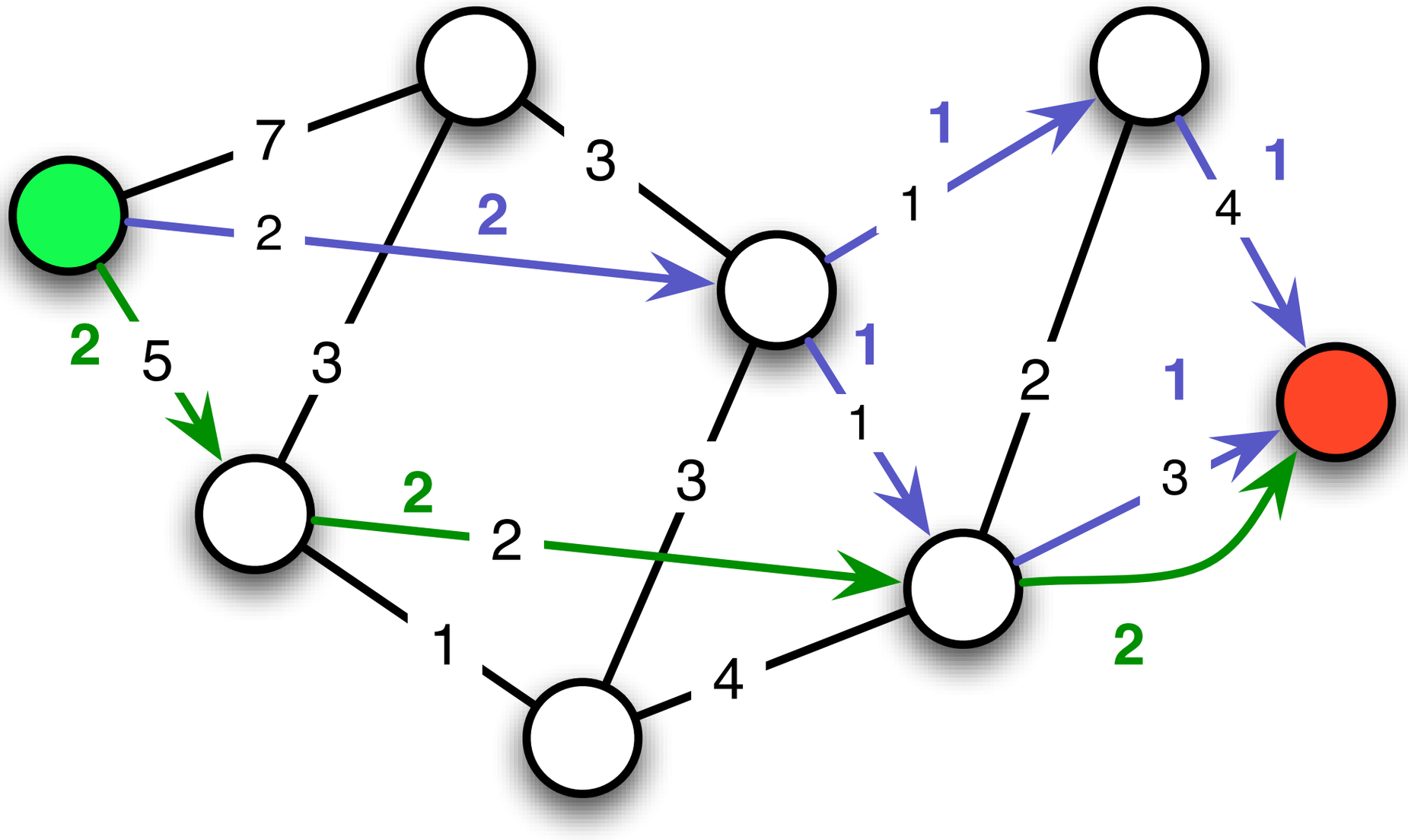
CoNe
Freiburg

Example



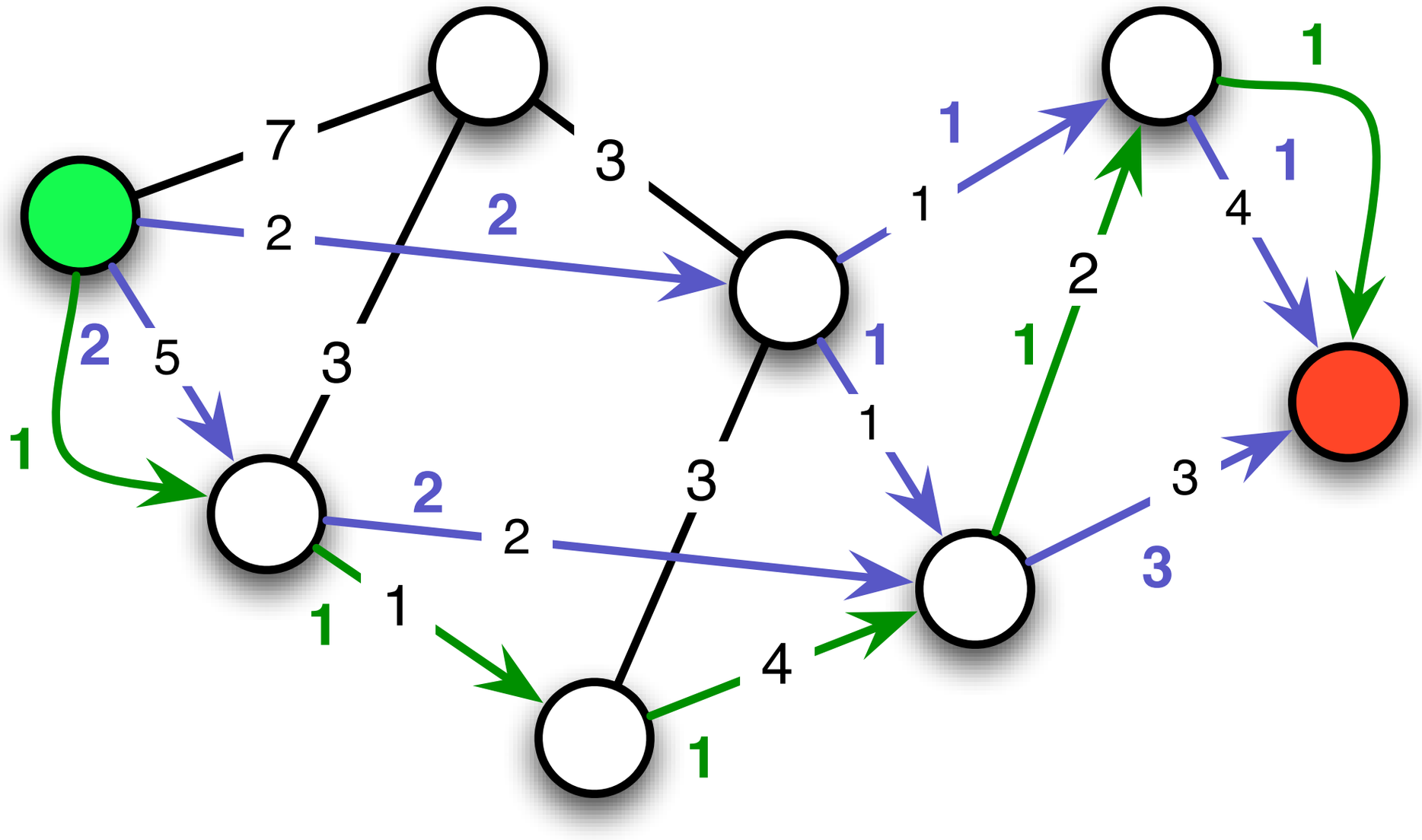
CoNe
Freiburg

Example



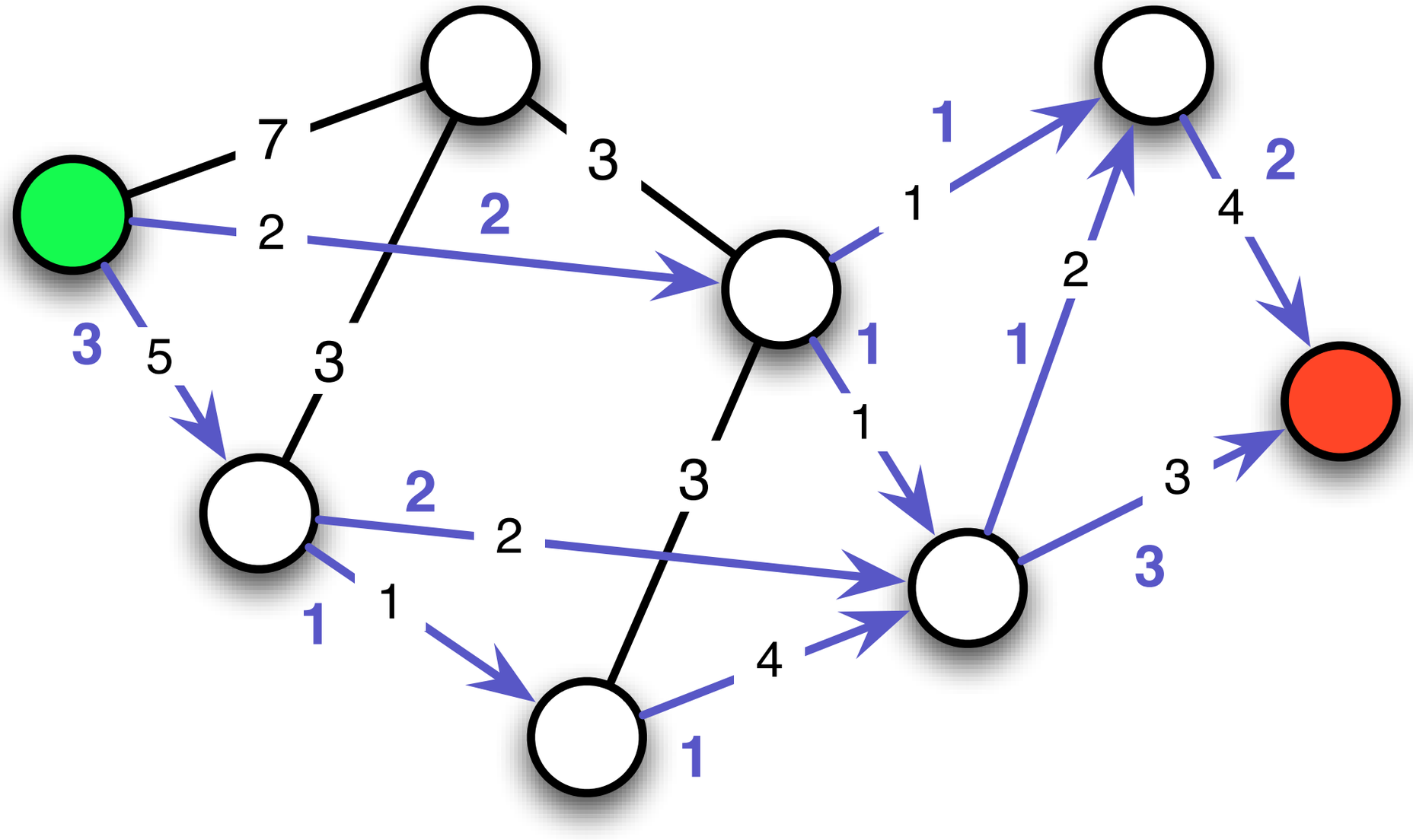
CoNe Freiburg

Example



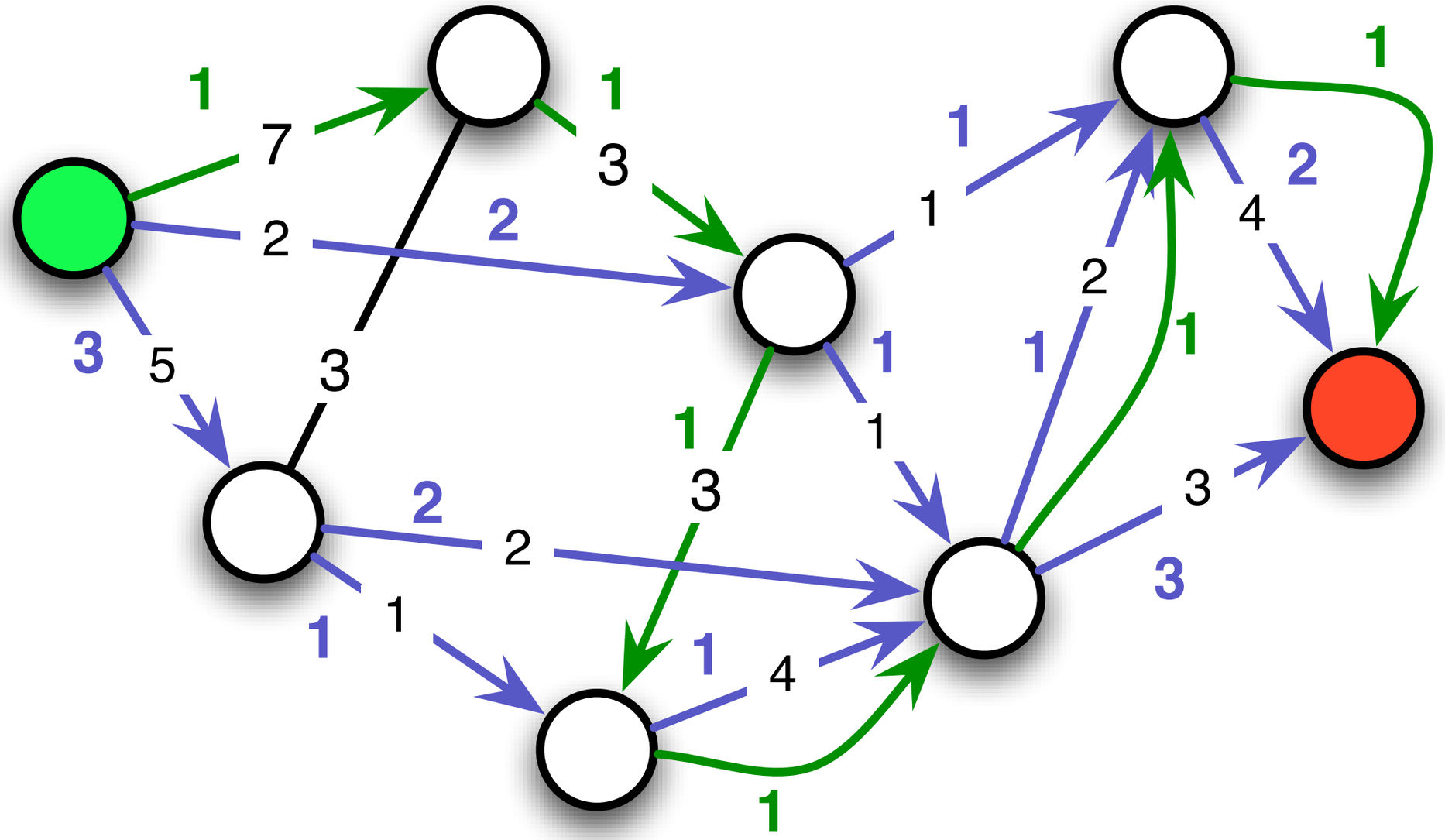
CoNe Freiburg

Example



CoNe Freiburg

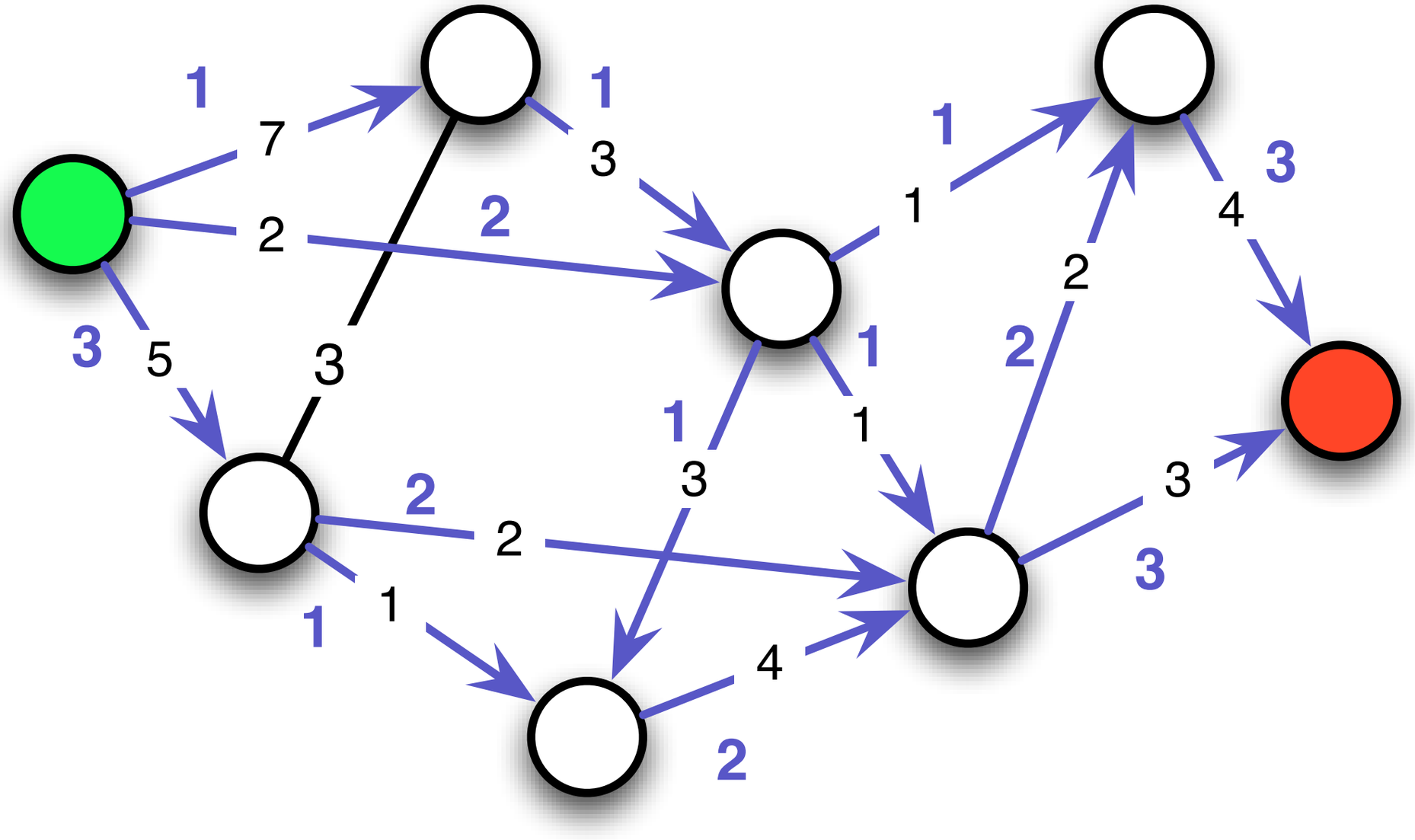
Example



UNI FREIBURG

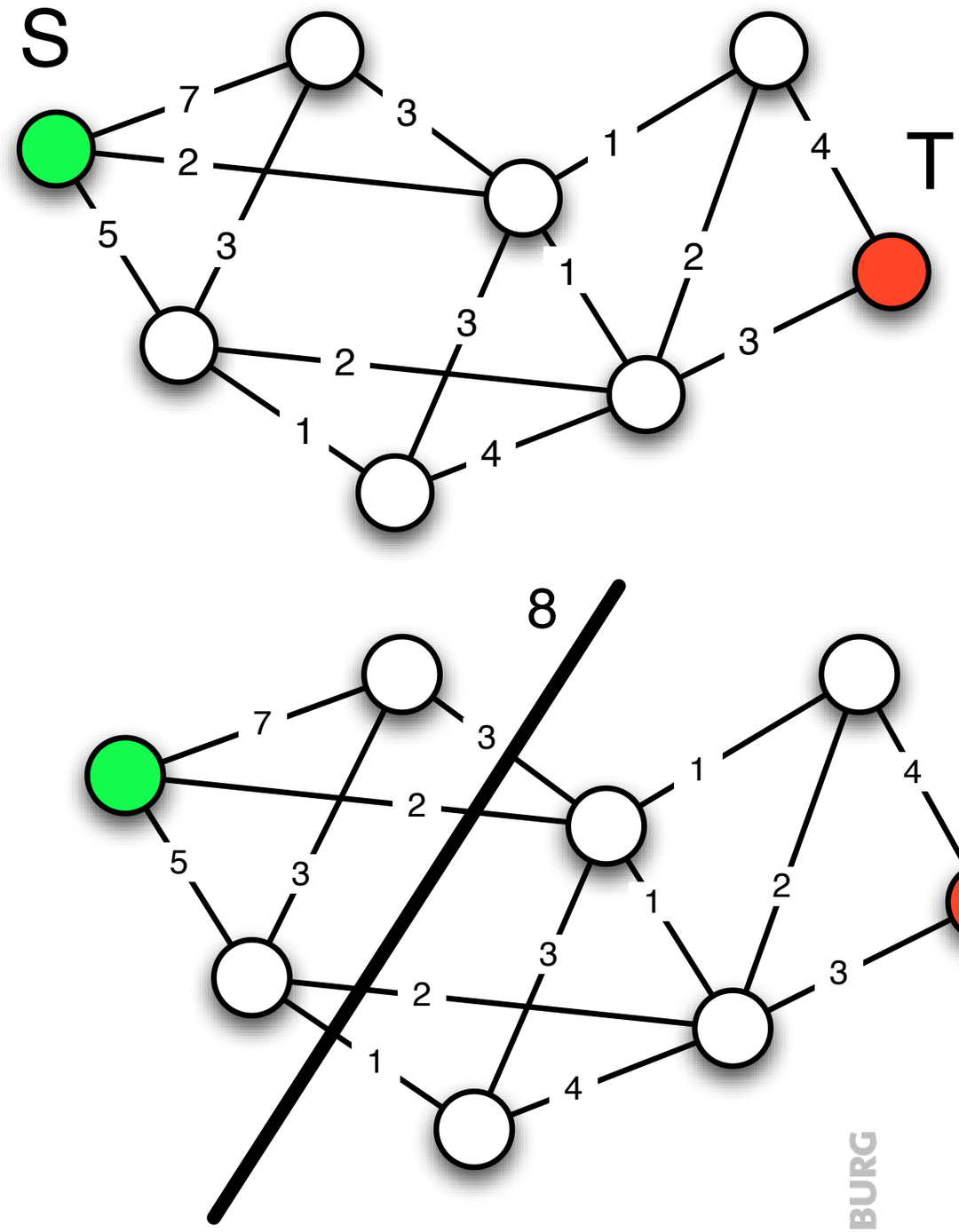
CoNe
Freiburg

Example



Minimum Cut in Networks

- Motivation
  - Find bottleneck in networks
- Definition
  - Min Cut problem
  - Given
    - graph G=(V,E)
    - capacity function w: E → R+0,
    - sources S and targets T
  - Find minimum cut between S and T
- A cut C is a set of edges
  - such that every path from a node of S to a node of T, contains an edge of C
- The size of a cut is $\sum_{e \in C} w(e)$

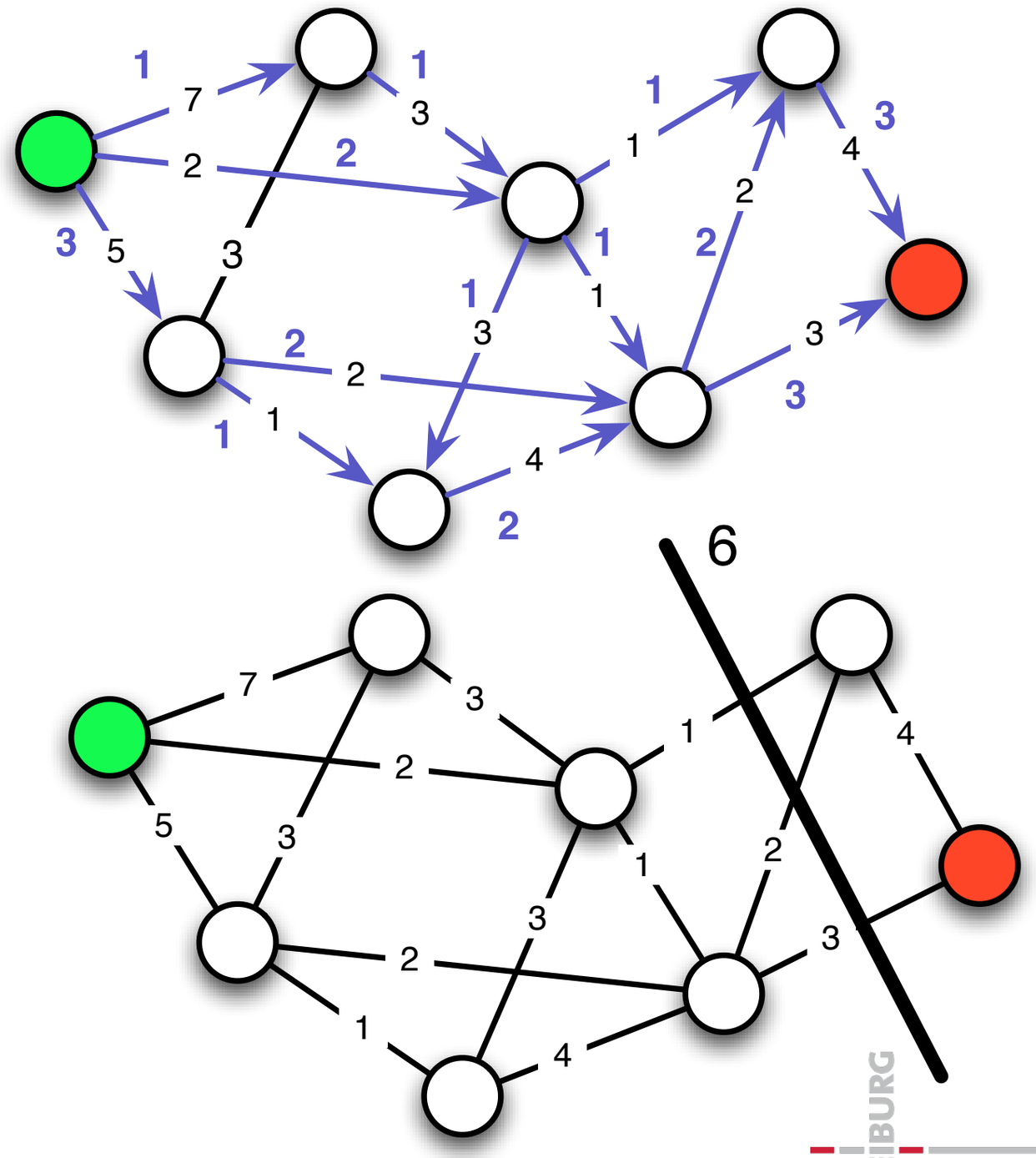


CoNe Freiburg

Min-Cut-Max-Flow Theorem

- Theorem
  - The minimum cut equals the maximum flow

- Algorithms for minimum cut
  - can be obtained from the maximum flow algorithms



UNI FREIBURG

CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Energy Informatics

03 Network Algorithms

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg

UNI FREIBURG