universität koblenz
weiter:denken

Big Data

Session 7: Link Analysis at Scale

Frank Hopfgartner
Institute for Web Science and Technologies

Previous week

- Network Theory (Briefly)
- Data representation
- Graph Processing Examples
- Distributed Systems for Graph Processing

Intended Learning Outcomes

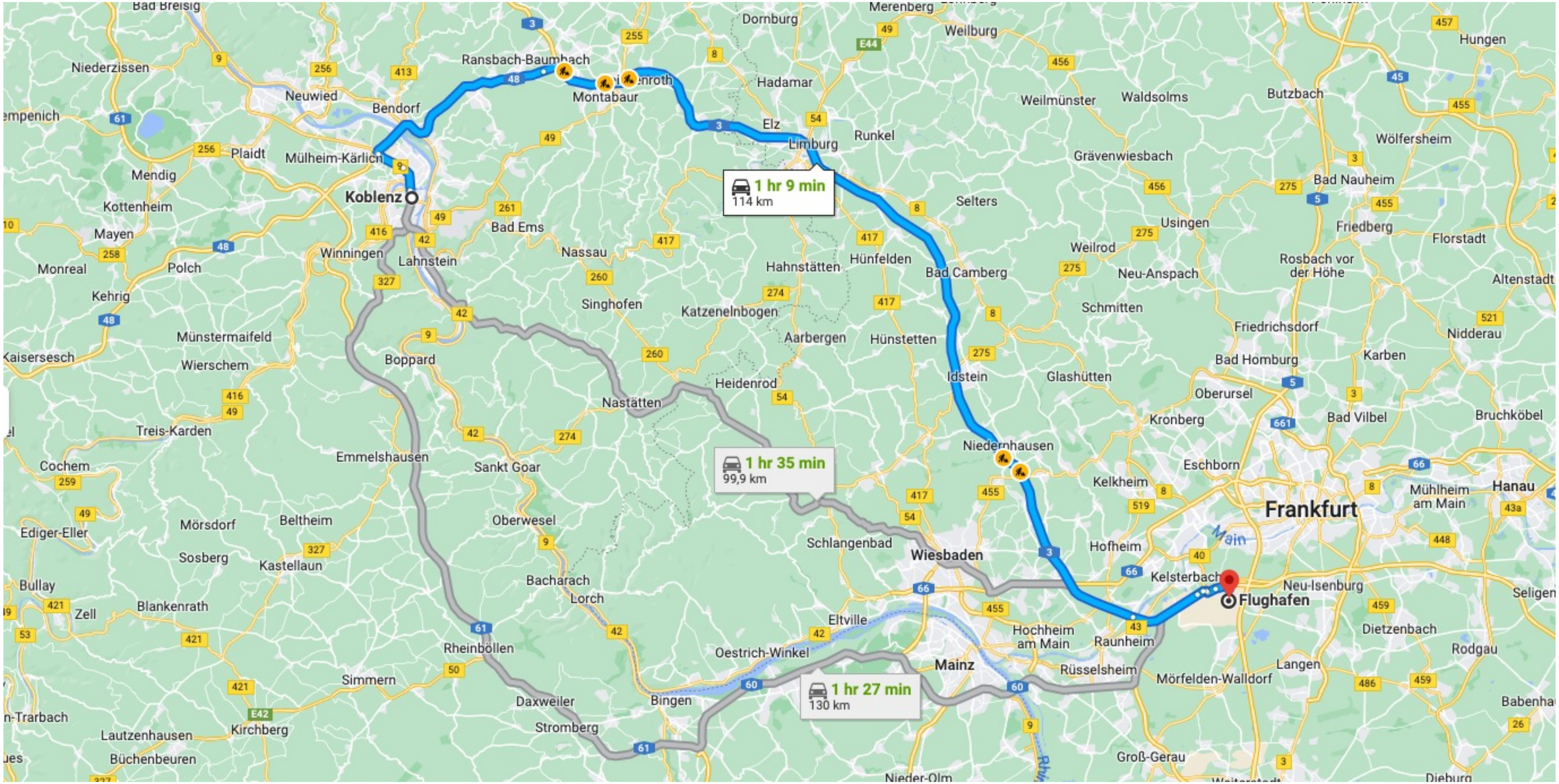
At the end of this lecture, you will be able to:

- Describe common distributed graph algorithms, namely
  - Single-source Shortest Path
  - PageRank
  - Community detection

Outline

- Single-source Shortest Path
- PageRank
- Community detection

How would you determine the shortest path from Koblenz to Frankfurt Airport?



Routing

- Based on Dijkstra’s shortest path algorithm
- Many improvements to deal with huge networks
- Improvements use preprocessing

Netherlands
Belgium
Germany
Czechia
London
Amsterdam
The Hague
Rotterdam
Antwerp
Ghent
Bruges
Brussels
Lille
Calais
Dunkirk
Paris
Versailles
Reims
Amiens
Rouen
Le Havre
Caen
Beauvais
Metz
Nancy
Luxembourg
Liège
Mons
Aachen
Cologne
Bonn
Düsseldorf
Essen
Dortmund
Münster
Bielefeld
Paderborn
Kassel
Göttingen
Koblenz
Frankfurt
Mainz
Trier
Saarbrücken
Mannheim
Heidelberg
Heilbronn
Karlsruhe
Stuttgart
Würzburg
Fulda
Erfurt
Jena
Nuremberg
Bamberg
Bayreuth
Regensburg
Ingolstadt
Passau
Hanover
Bremen
Bremerhaven
Hamburg
Oldenburg
Osnabrück
Groningen
Leeuwarden
Zwolle
Arnhem
Eindhoven
Breda
Alkmaar
Wilhelmshaven
Lüneburg
Schwerin
Wolfsburg
Brunswick
Magdeburg
Brandenburg
Berlin
Oranienburg
Frankfurt an der Oder
Cottbus
Halle (Saale)
Leipzig
Dresden
Chemnitz
Zwickau
Ústí nad Labem
Teplice
Liberec
Karlovy Vary
Plzeň
Prague
Pardubice
Hradec Kralove
Ceske Budejovice
Brno
Wrocław
Legnica
Jelenia Góra
Zielona Góra
Leszno
Poznań
Gorzów Wielkopolski
Piła
Sheffield
Nottingham
Peterborough
Norwich
Cambridge
Ipswich
Colchester
Luton
Canterbury
Brighton

Routing

- Based on Dijkstra’s shortest path algorithm
- Many improvements to deal with huge networks
- Improvements use preprocessing

Bremerhaven
Hamburg
Schwerin
Nottingham
Groningen
Oldenburg
Bremen
Lüneburg
Peterborough
Norwich
Berlin
Poznań
Alkmaar
Zwolle
Cambridge
Amsterdam
Hanover
Wolfsburg
Brunswick
Osnabrück
Magdeburg
Ipswich
Colchester
The Hague
Netherlands
Bielefeld
Münster
Luton
London
Rotterdam
Arnhem
Paderborn
Cottbus
Breda
Göttingen
Halle (Saale)
Leipzig
Eindhoven
Dortmund
Essen
Canterbury
Antwerp
Düsseldorf
Kassel
Dresden
Wrocław
Bruges
Ghent
Brighton
Calais
Dunkirk
Germany
Cologne
Brussels
Bonn
Aachen
Erfurt
Jena
Chemnitz
Zwickau
Lille
Liège
Mons
Belgium
Koblenz
Fulda
Prague
Frankfurt
Mainz
Würzburg
Bamberg
Bayreuth
Amiens
Pilsen
Czechia
Le Havre
Trier
Luxembourg
Rouen
Caen
Beauvais
Mannheim
Nuremberg
Brno
Heidelberg
Saarbrücken
Heilbronn
Reims
Metz
Regensburg
Paris
Karlsruhe
Versailles
Stuttgart
Ingolstadt
Nancy
Passau

Single source shortest path (SSSP)

- Problem
  - Find shortest path from a source node to all target nodes
- Solution
  - **Single processor machine: iterative algorithm developed by Edsger W. Dijkstra, now used by Google Maps**

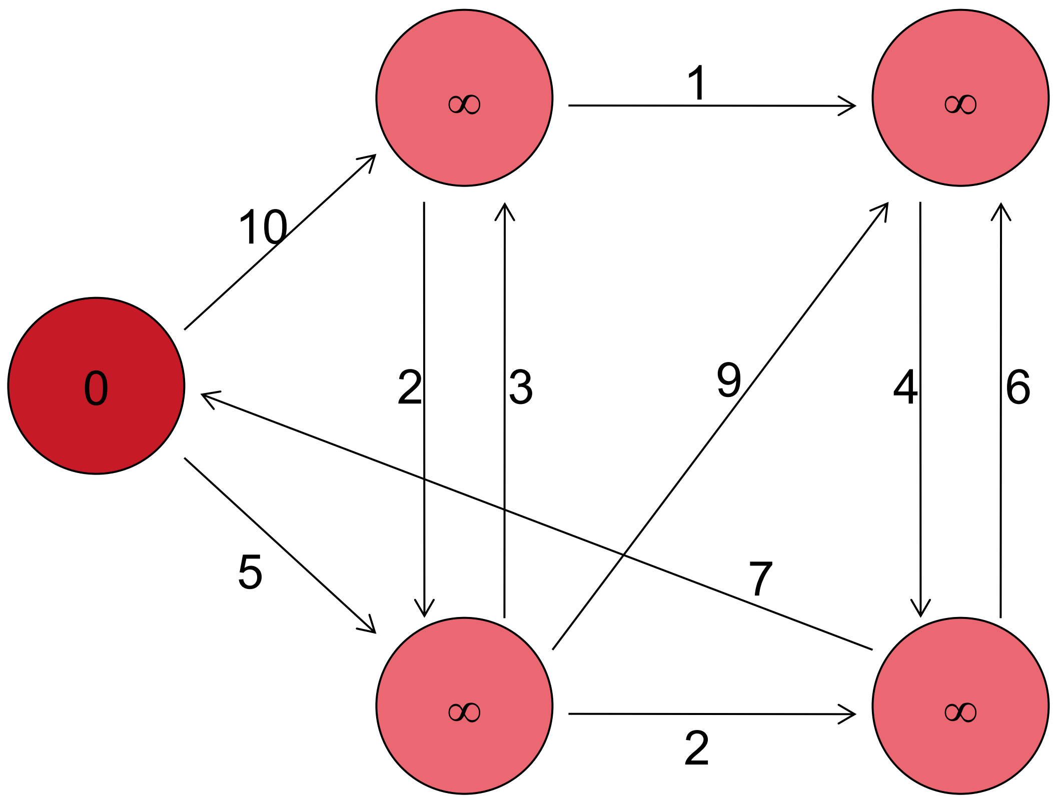
Lin and Dyer. *Data-Intensive Text Processing with MapReduce*, Chapter 5.2

Dijkstra‘s algorithm

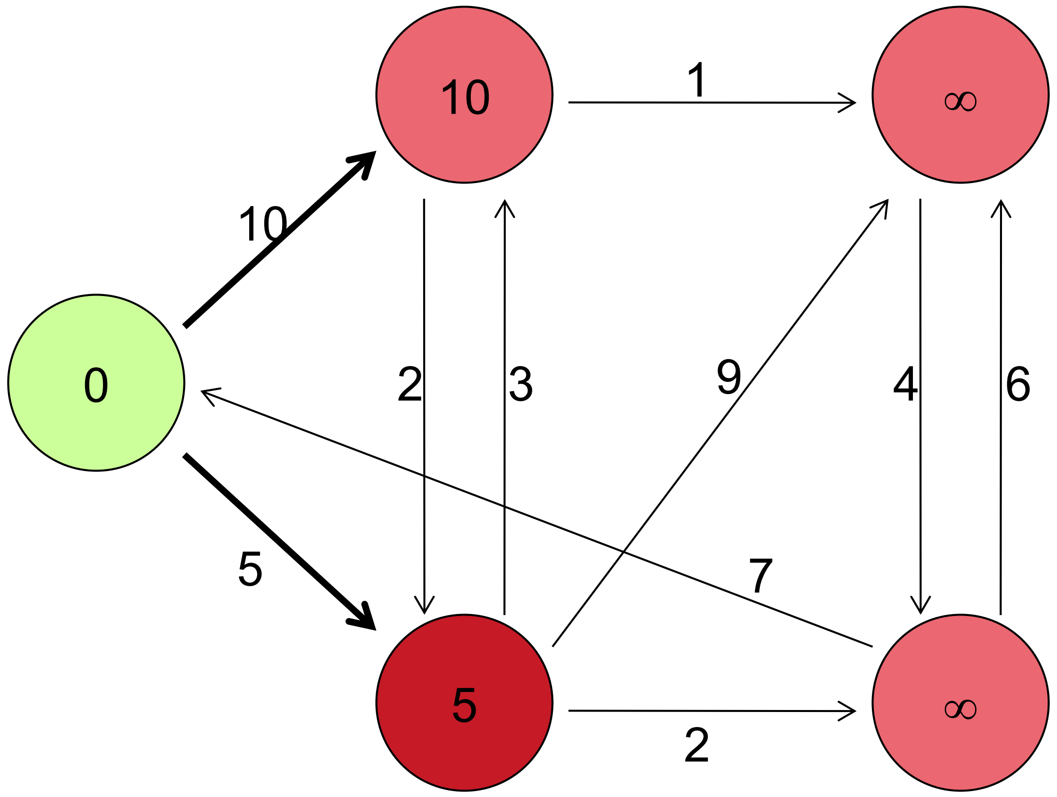
- **Input:** A weighted directed graph *G*=(*V*, *E*, *w*), and a source vertex *s*.

- **Output:** Shortest-path weight from *s* to each vertex *v* in *V*, and a shortest path from *s* to each vertex *v* in *V* if *v* is reachable from *s*. In each step, find the minimum edge of a node not yet visited.

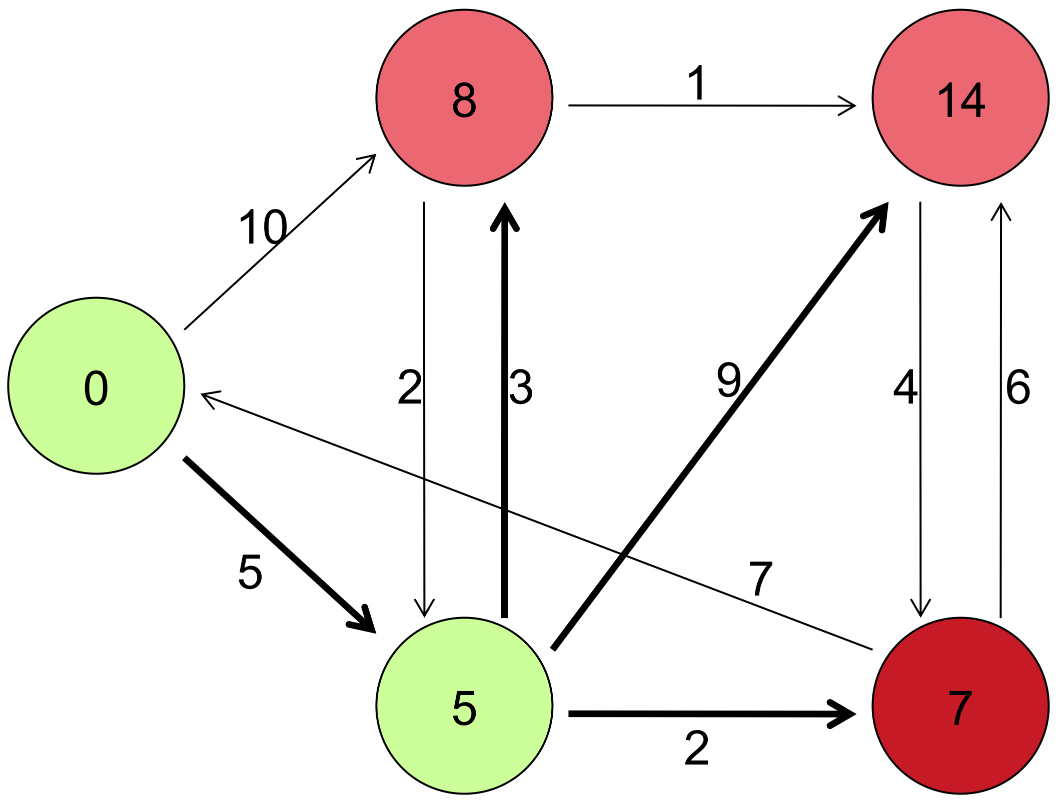
Example: SSSP – Dijkstra‘s algorithm



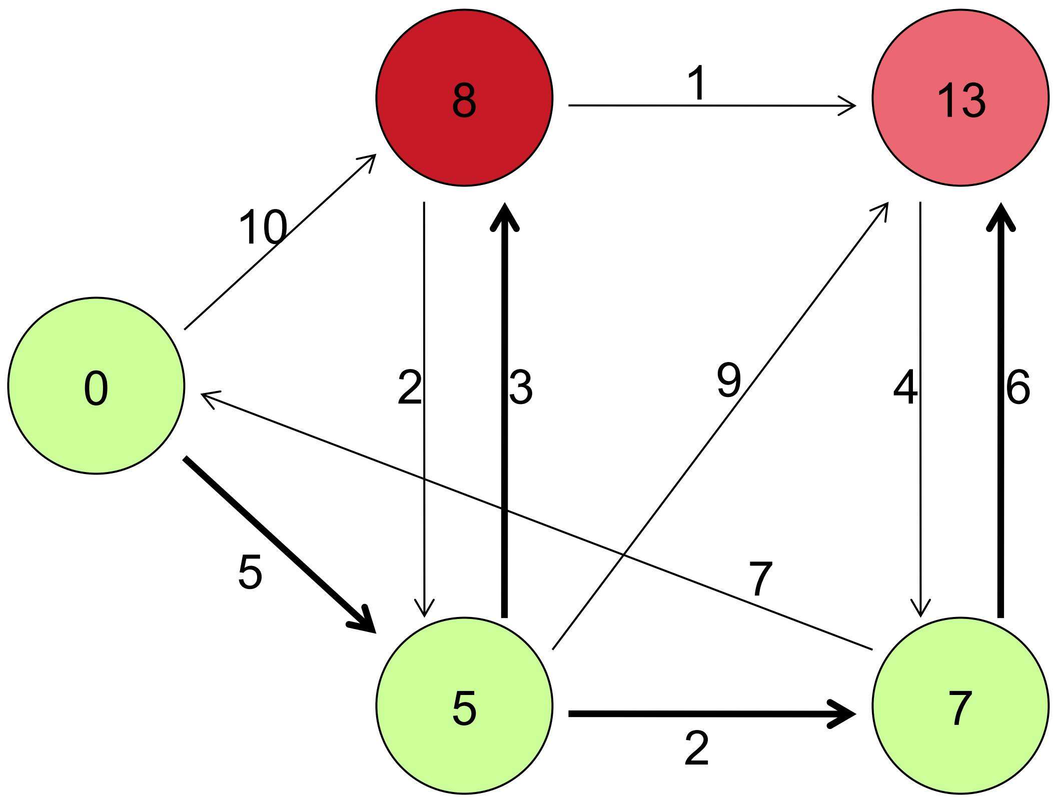
Example: SSSP – Dijkstra‘s algorithm



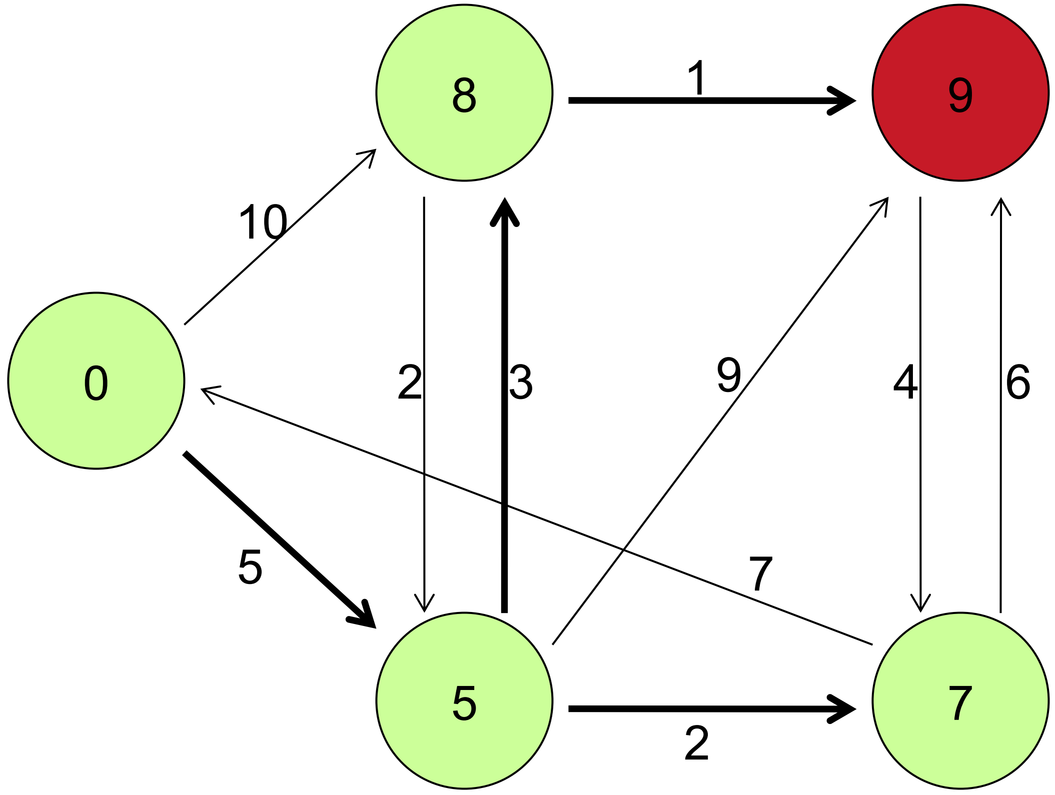
Example: SSSP – Dijkstra‘s algorithm



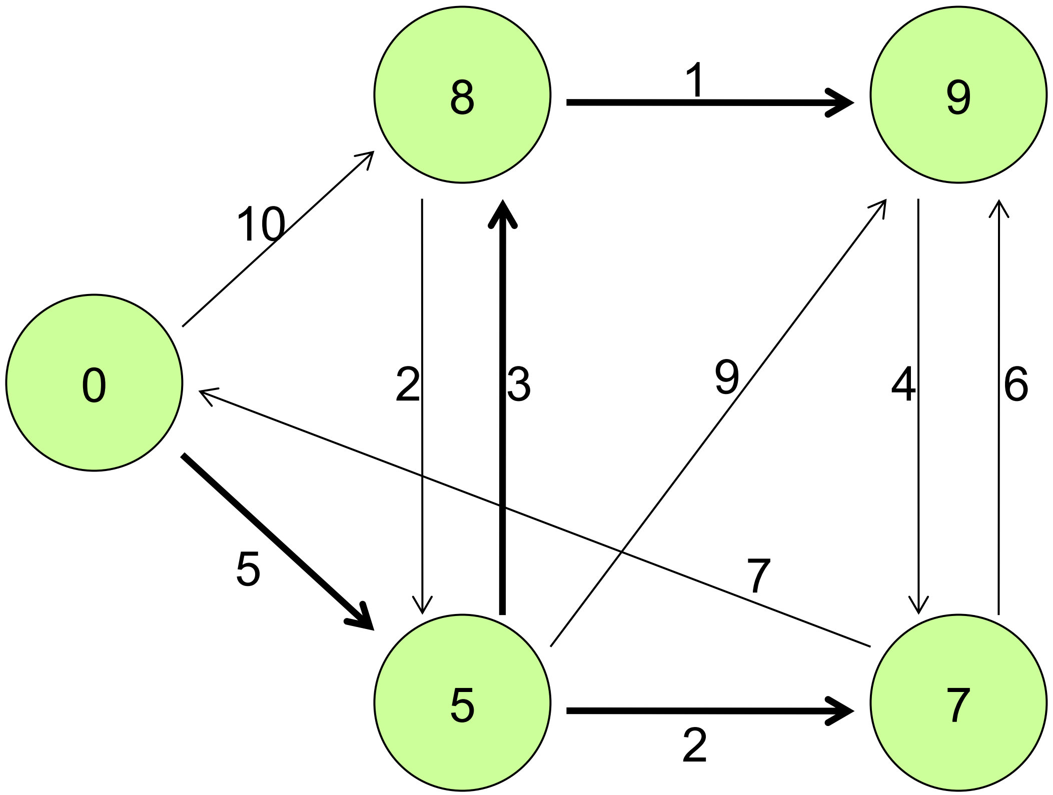
Example: SSSP – Dijkstra‘s algorithm



Example: SSSP – Dijkstra‘s algorithm



Example: SSSP – Dijkstra‘s algorithm



SSSP Distributed computing

- How to distribute it?

- Brute-force (if edge weights are all 1)
  - Distance of all nodes N directly connected to the source is one
  - Distance of all nodes directly connected to nodes connected to N is two …
  - Multiple paths from the source to a node x exist
    - the shortest path must go through one of the nodes having an outgoing edge to x; use the minimum

- Approach: Parallel breadth-first search (BFS) using MapReduce/Pregel

Single-Source Shortest Path in M/R

- How to distributedly-compute it?
  - MAP: emit all distances for all reachable nodes, and the graph structure
  - REDUCE: recover graph structure, find shortest distance and update scores

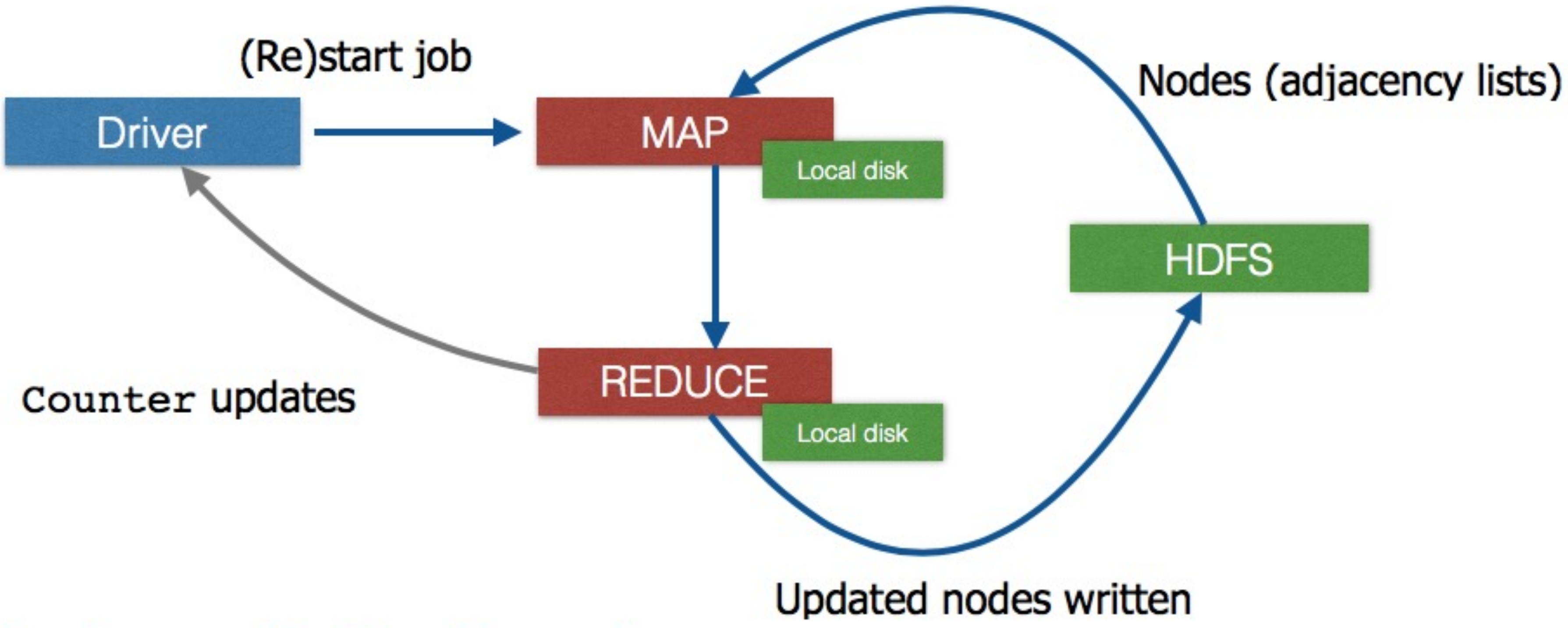
- Each **iteration** of the algorithm is **one Map/Reduce job**
  - A **map** phase to compute the distances
  - A **reduce** phase to find the current minimum distance

Single-Source Shortest Path in M/R

- Iterations in MR
  1. All nodes connected to the source are discovered
  2. All nodes connected to those discovered in 1. are found
  3. …
  - Reducer output is input for the next iteration (job)
  - Graph structure is read/written from/to HDFS at each iteration

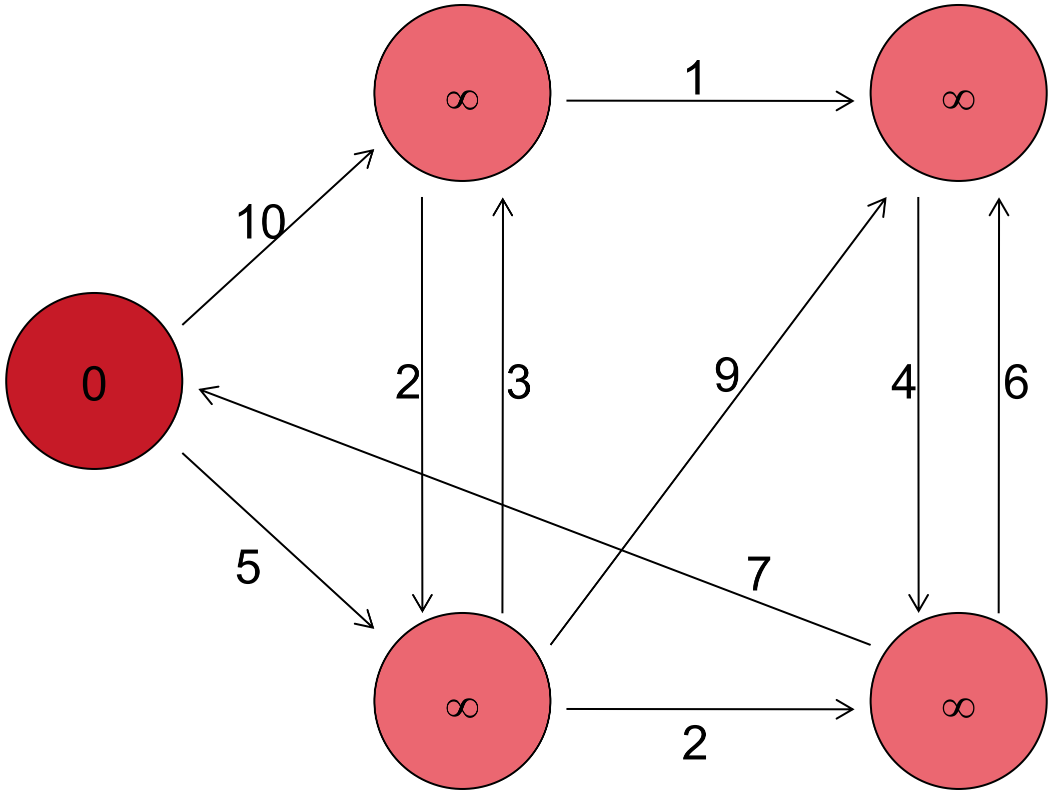
How many iterations should we expect?

Single-Source Shortest Path in M/R

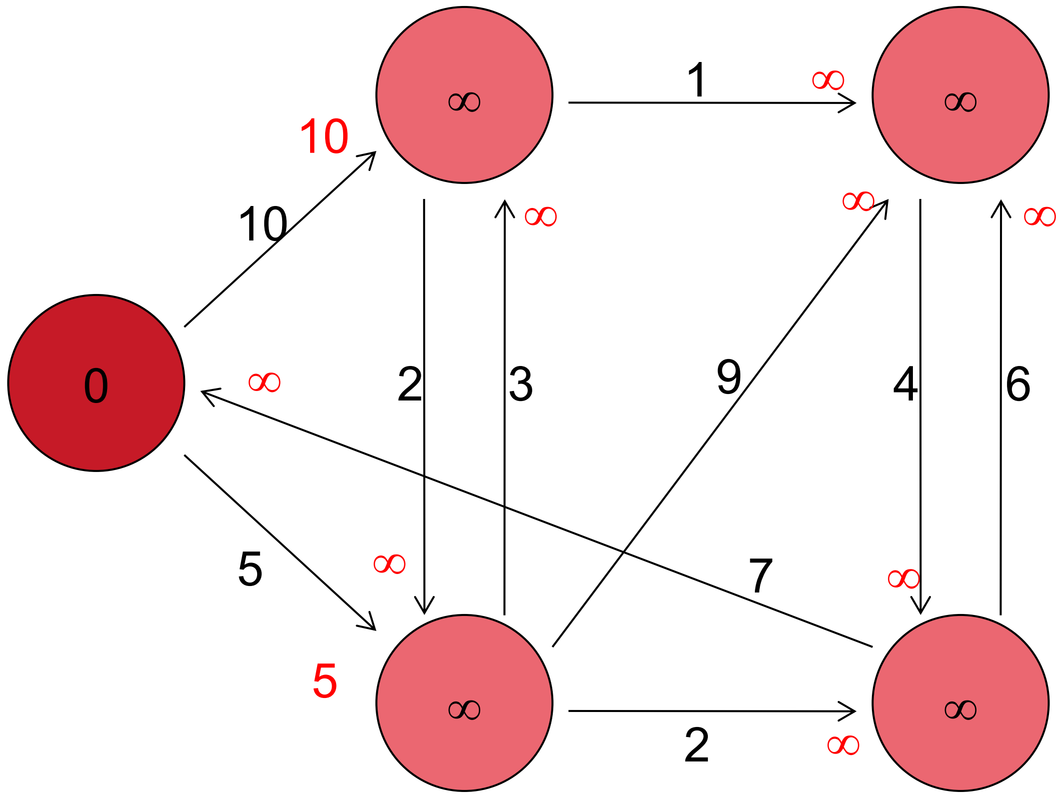


Disadvantage: a lot of reading and writing to/from HDFS

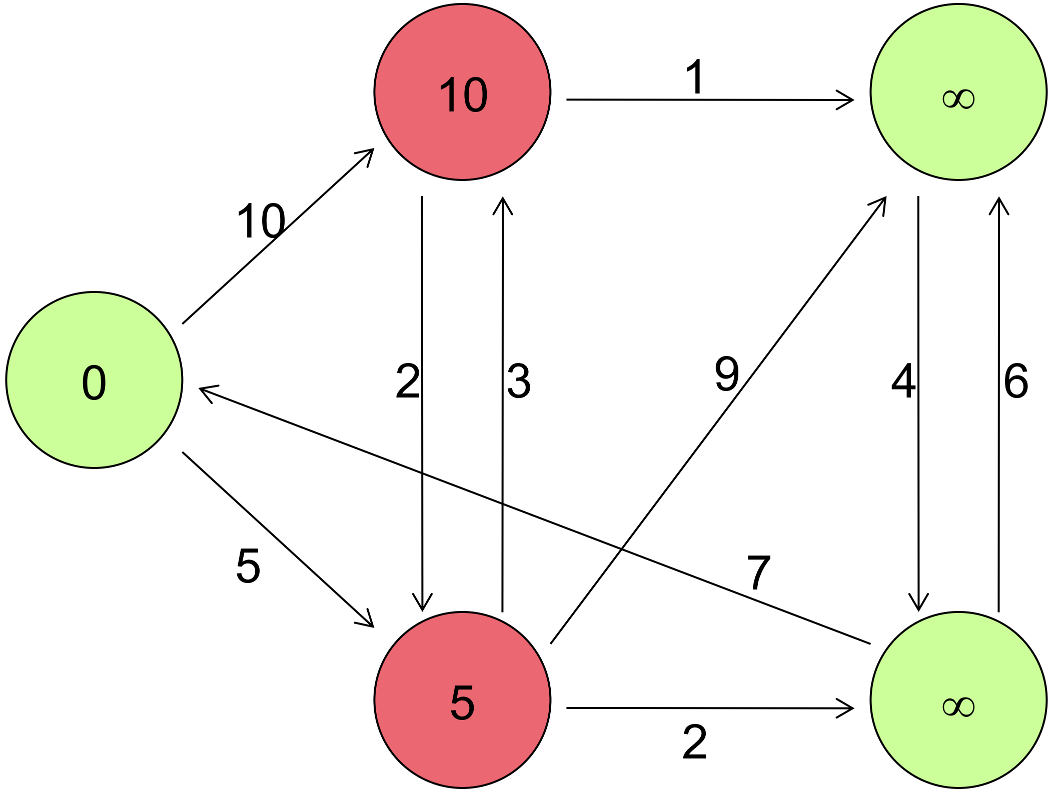
SSSP – Parallel BFS in Pregel



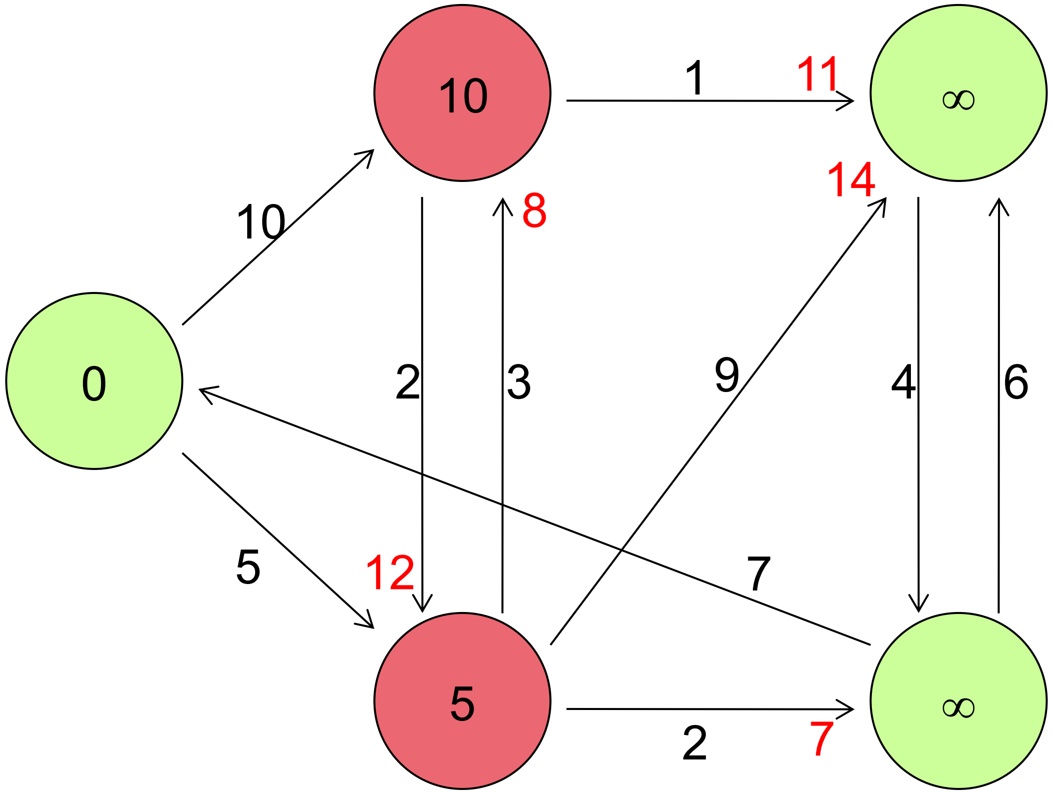
SSSP – Parallel BFS in Pregel



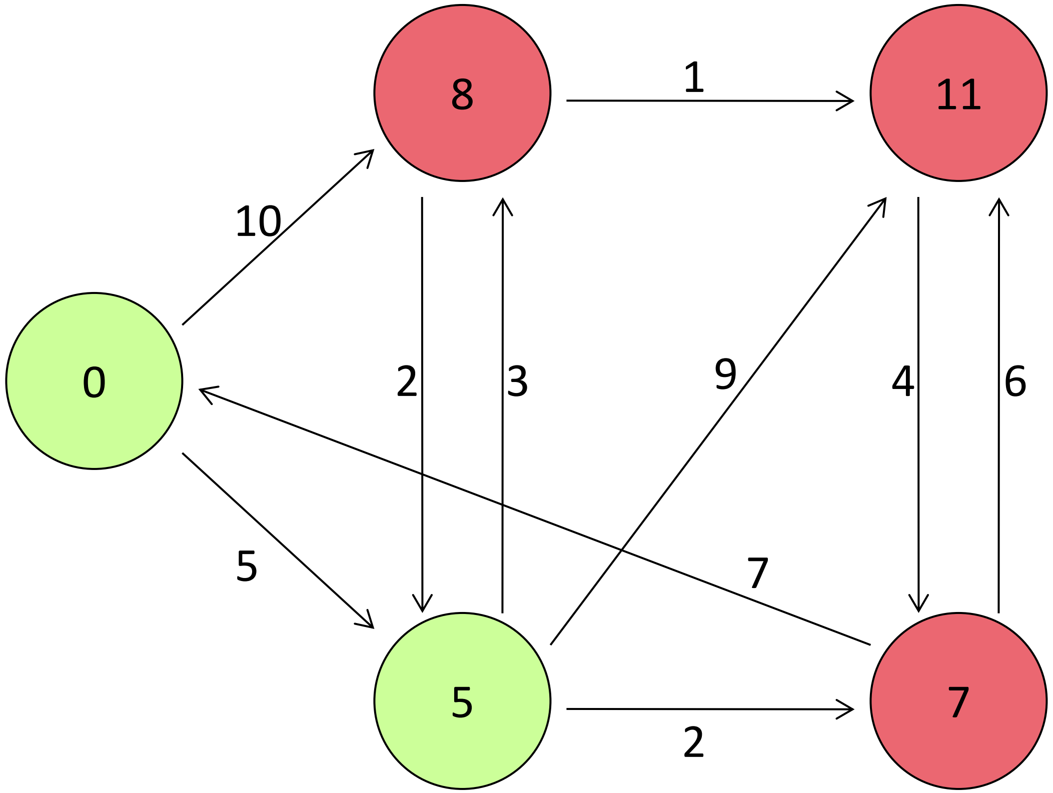
SSSP – Parallel BFS in Pregel



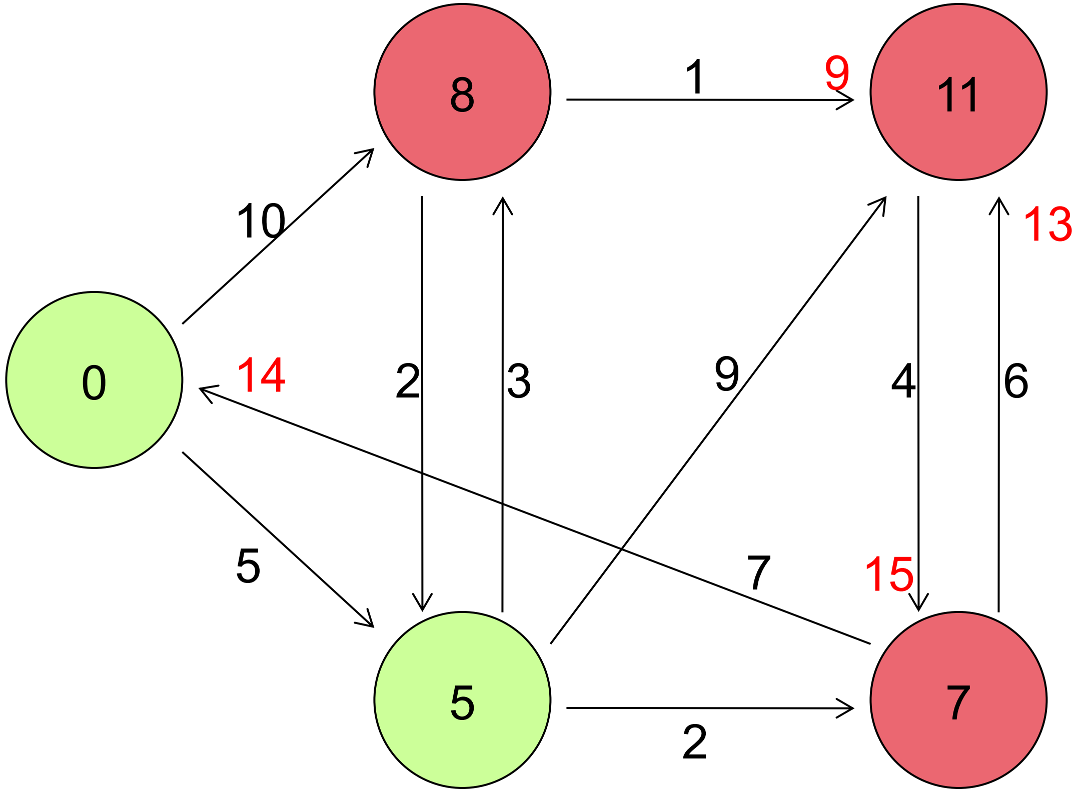
SSSP – Parallel BFS in Pregel



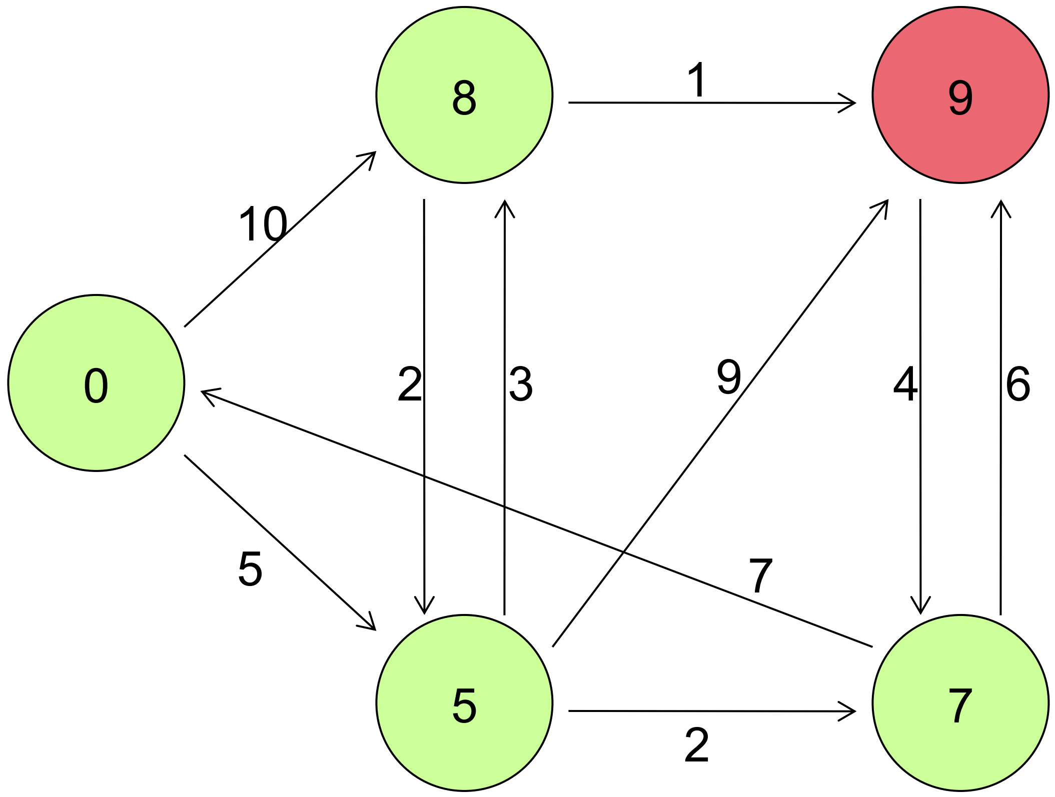
SSSP – Parallel BFS in Pregel



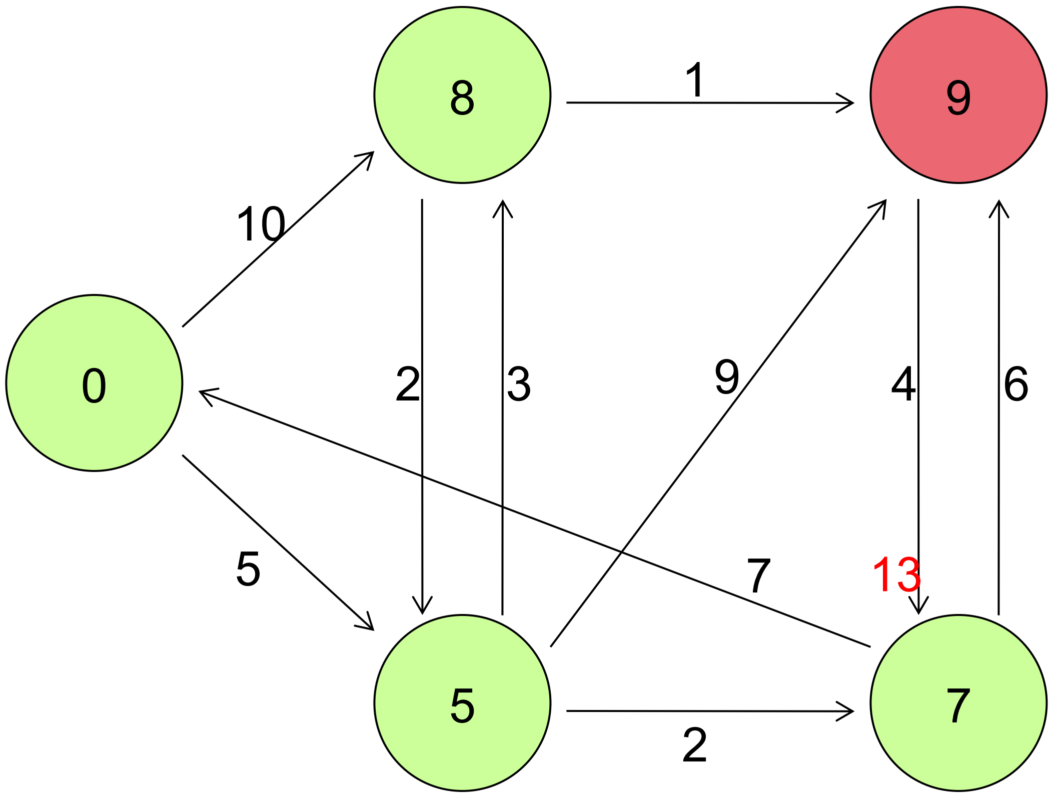
SSSP – Parallel BFS in Pregel



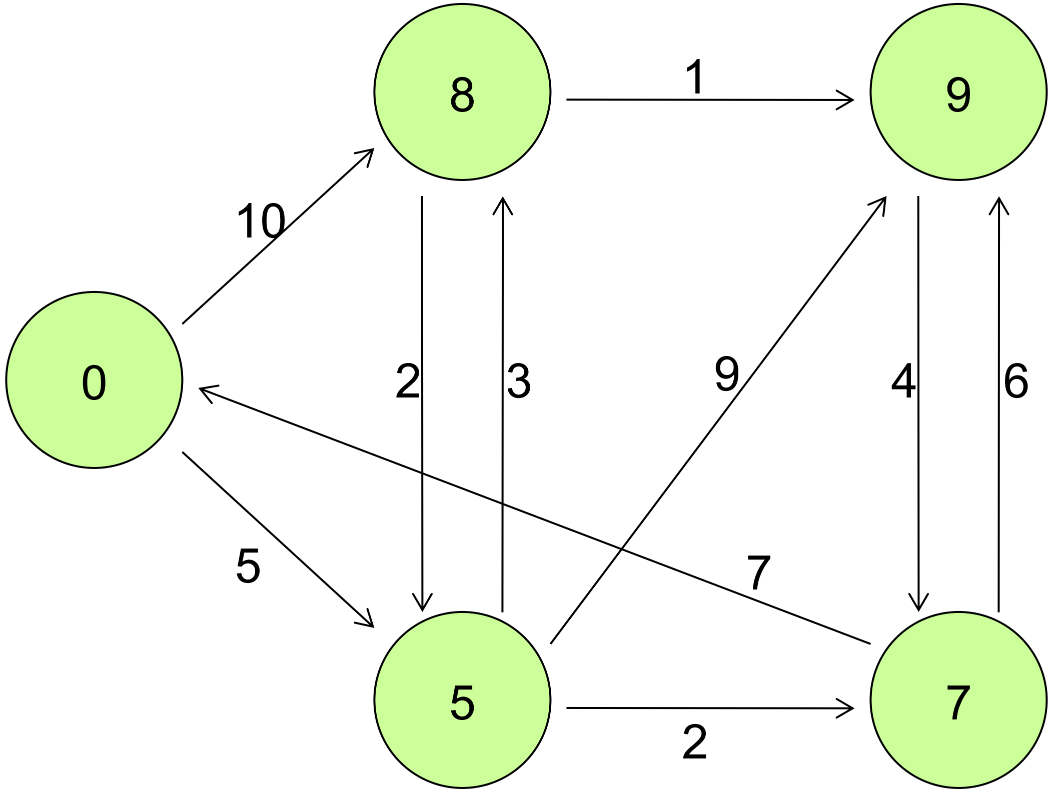
SSSP – Parallel BFS in Pregel



SSSP – Parallel BFS in Pregel



SSSP – Parallel BFS in Pregel



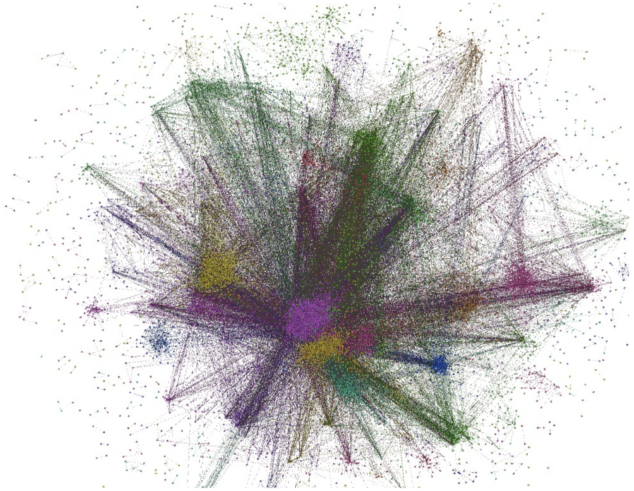
Outline

- Single-source Shortest Path
- **PageRank**
- Community detection

The Web is a Graph

universität koblenz
weiter:denken

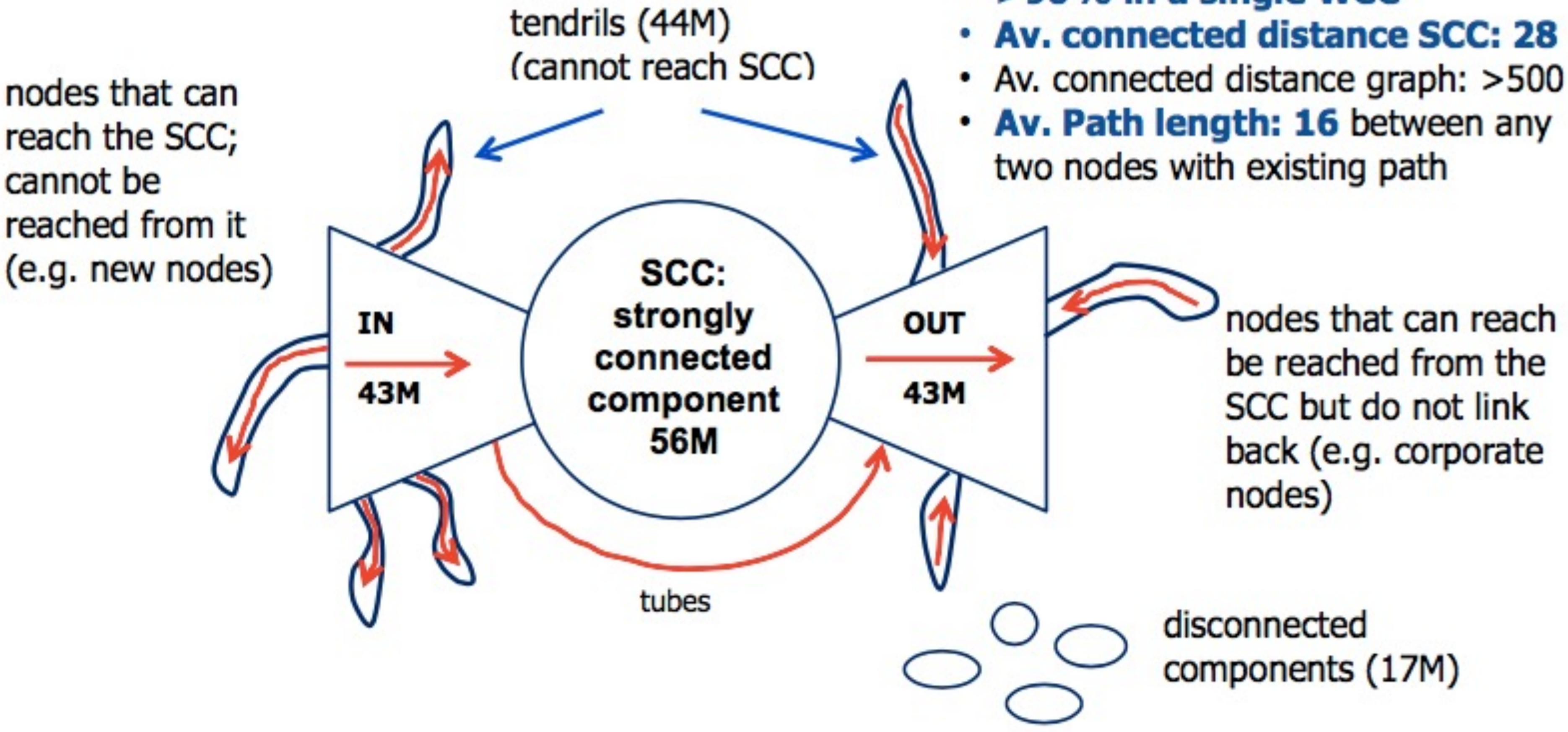
- The World Wide Web was formed in the early 1990’s by Tim Berners-Lee at CERN
- Early years
  - Full-text search engines
  - Taxonomies with pages in categories
- Then
  - Users view the Web through the lenses of the search engine
- Now
  - Social media as entry point to Web content



Web Graph Structure

Broder et al., 1999

- ~200M nodes in total
- **>90% in a single WCC**
- **Av. connected distance SCC: 28**
- Av. connected distance graph: >500
- **Av. Path length: 16** between any two nodes with existing path


nodes that can reach the SCC; cannot be reached from it (e.g. new nodes)
tendrils (44M) (cannot reach SCC)
IN
43M
SCC: strongly connected component 56M
OUT
43M
nodes that can reach be reached from the SCC but do not link back (e.g. corporate nodes)
tubes
disconnected components (17M)

Linkage Analysis

A method of ranking web sites which is based on the exploitation of latent human judgments mined from the hyperlinks that exist between documents on the WWW.

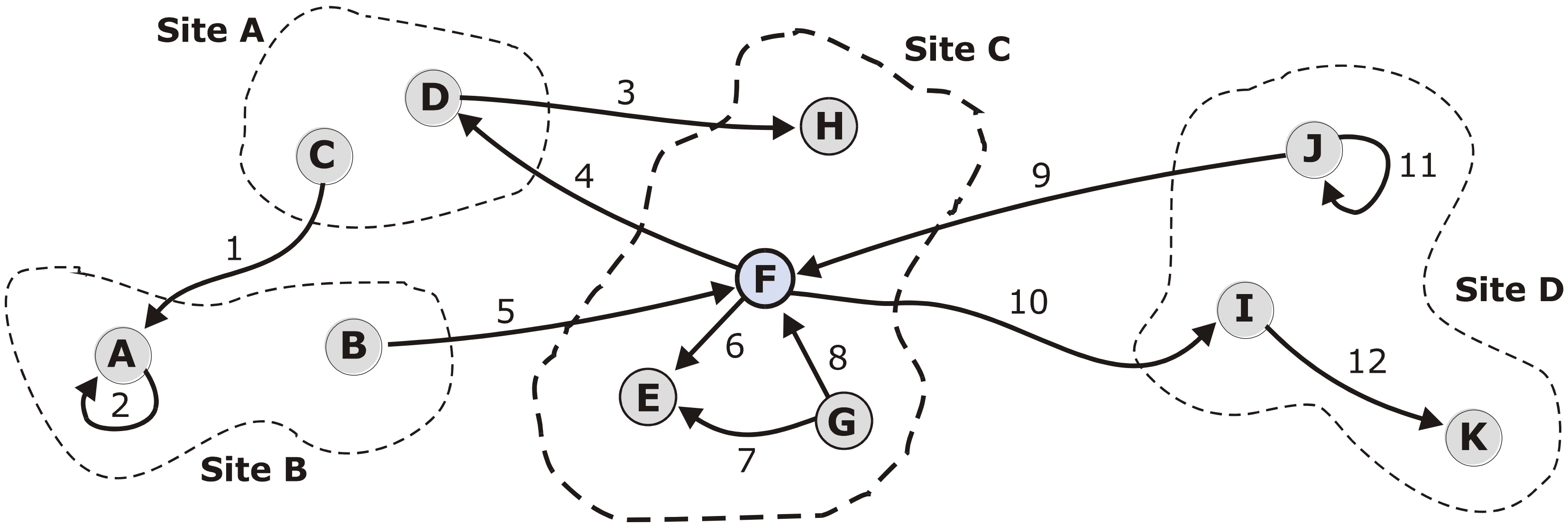
Assumed Properties of Links

When extracting information for linkage analysis from hyperlinks on the Web, three core properties can be assumed:

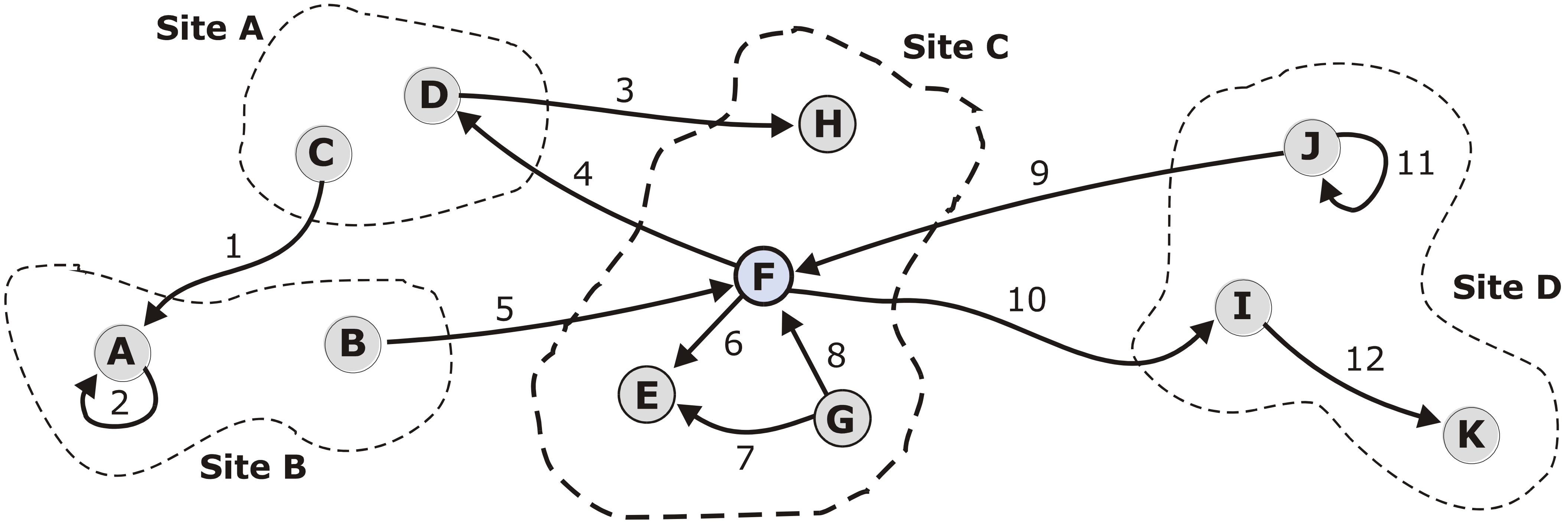
- A link takes cognitive effort to create, so people do it for a reason
- A link between two documents on the web carries the implication of related content.
- If different people authored the documents (different domains, therefore off-site links), then the first author found the second document valuable.
  - If I like to another person's WWW page, then I am saying that this page is important in the context of my page.

Link Types

in-link to doc F :	5,8,9
out-link from doc F :	4,6,10
self-links:	2,11
on-site links:	6,8,12
off-site links:	1,3,4,5,9,10
on-site in-links to doc F:	?
off-site out-links of doc F:	?

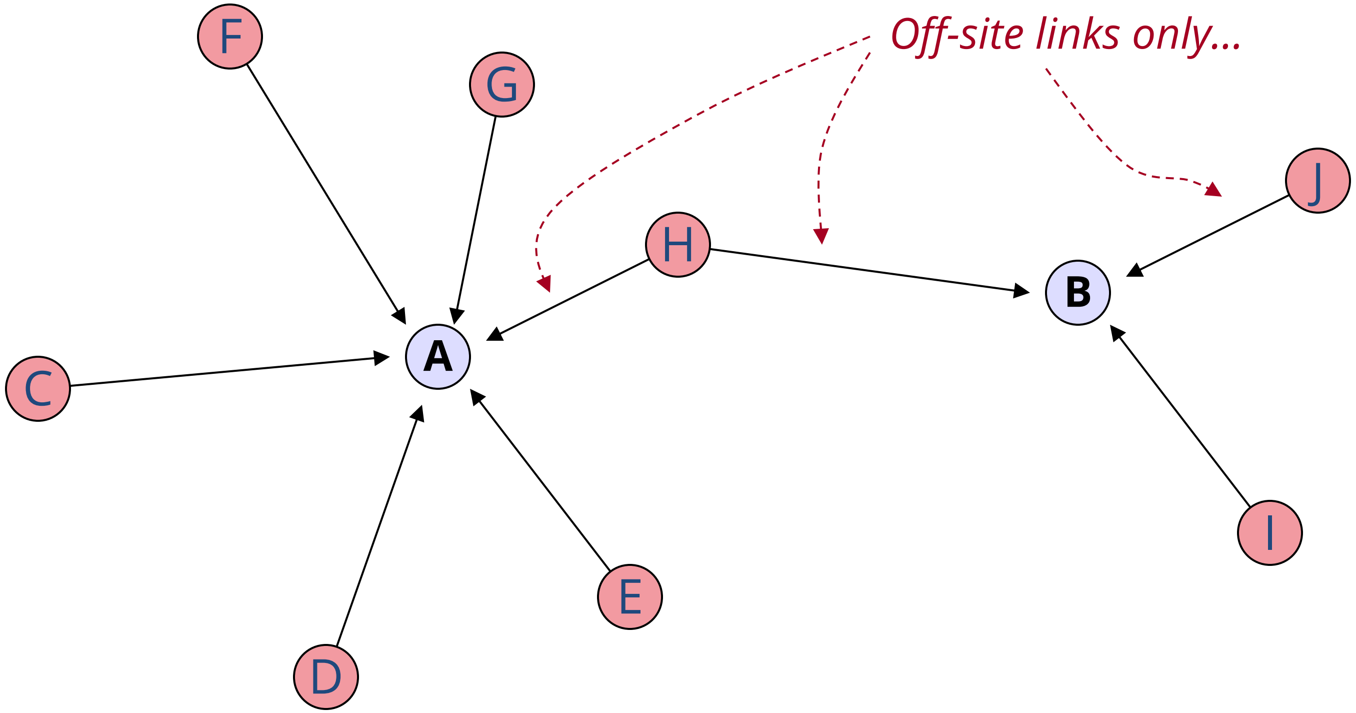


So which are most important?



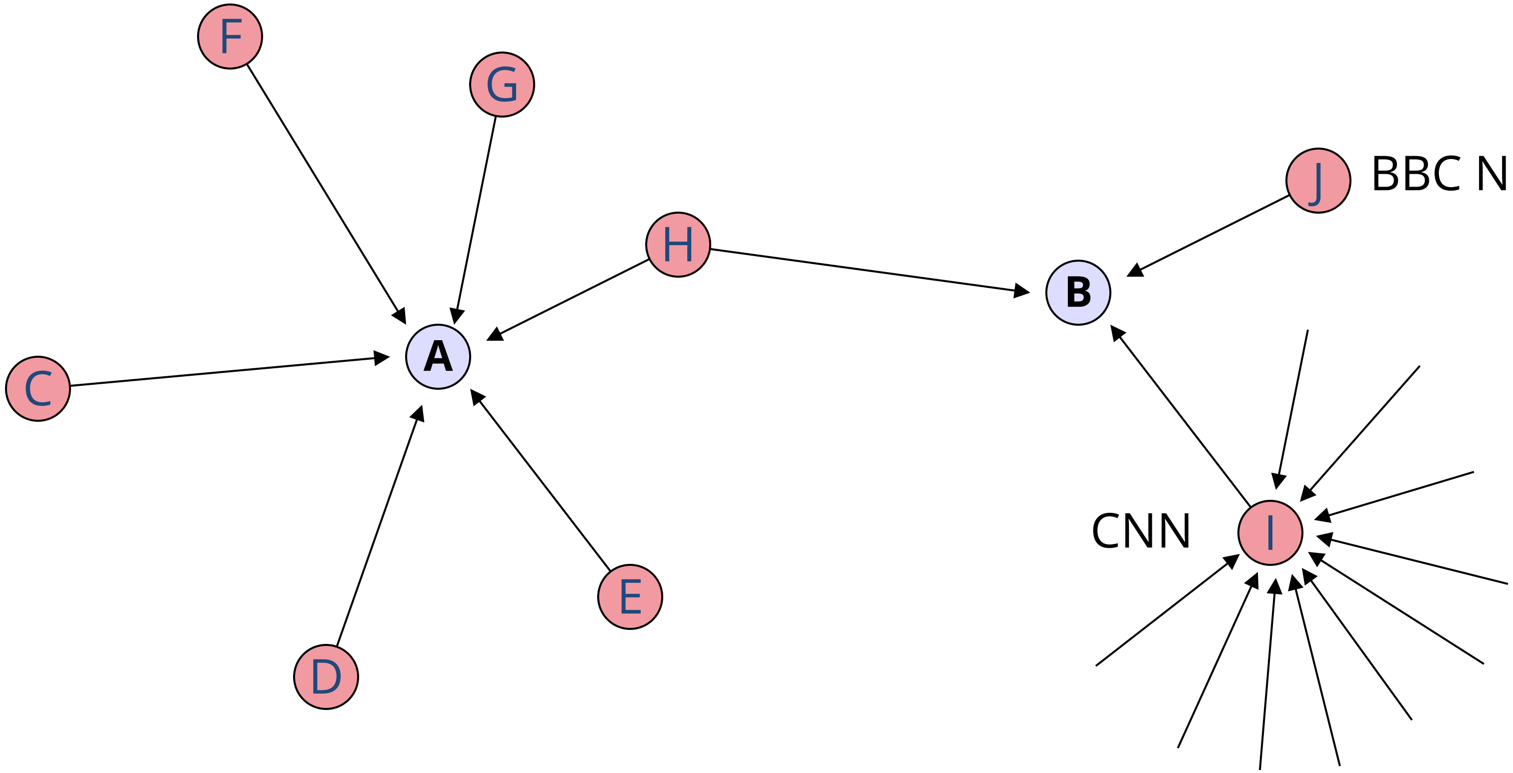
Basic Linkage Analysis

Given a linkage graph (below), Page A is a better page than B because...



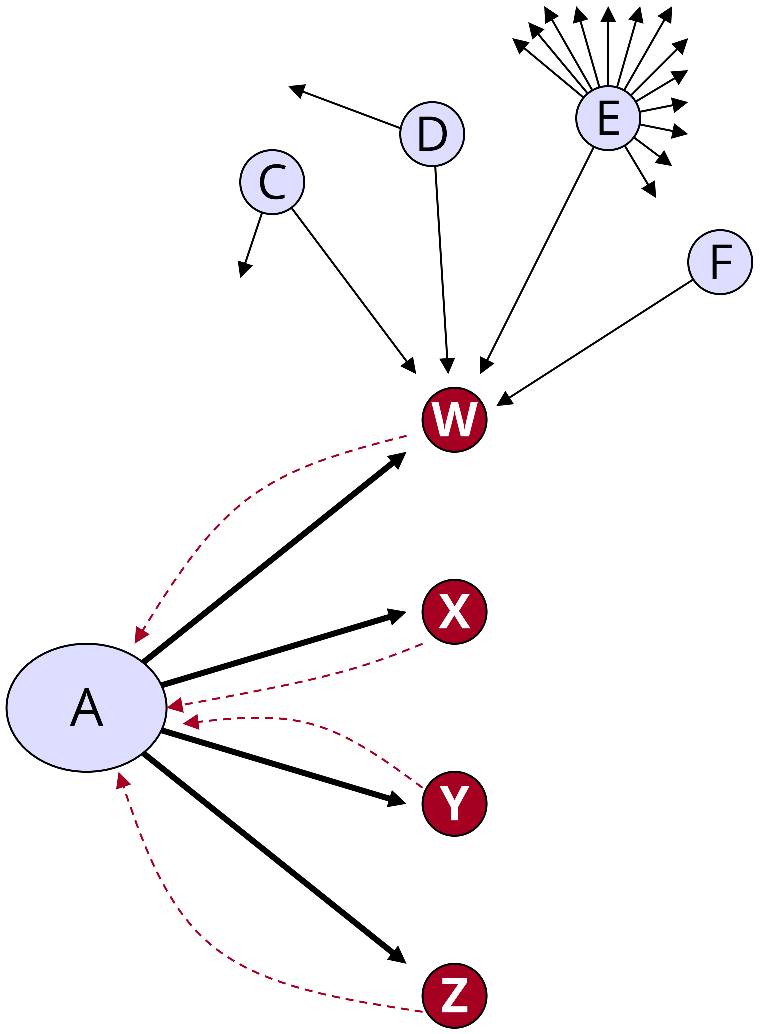
Expanding on this...

However, page B may actually be better...



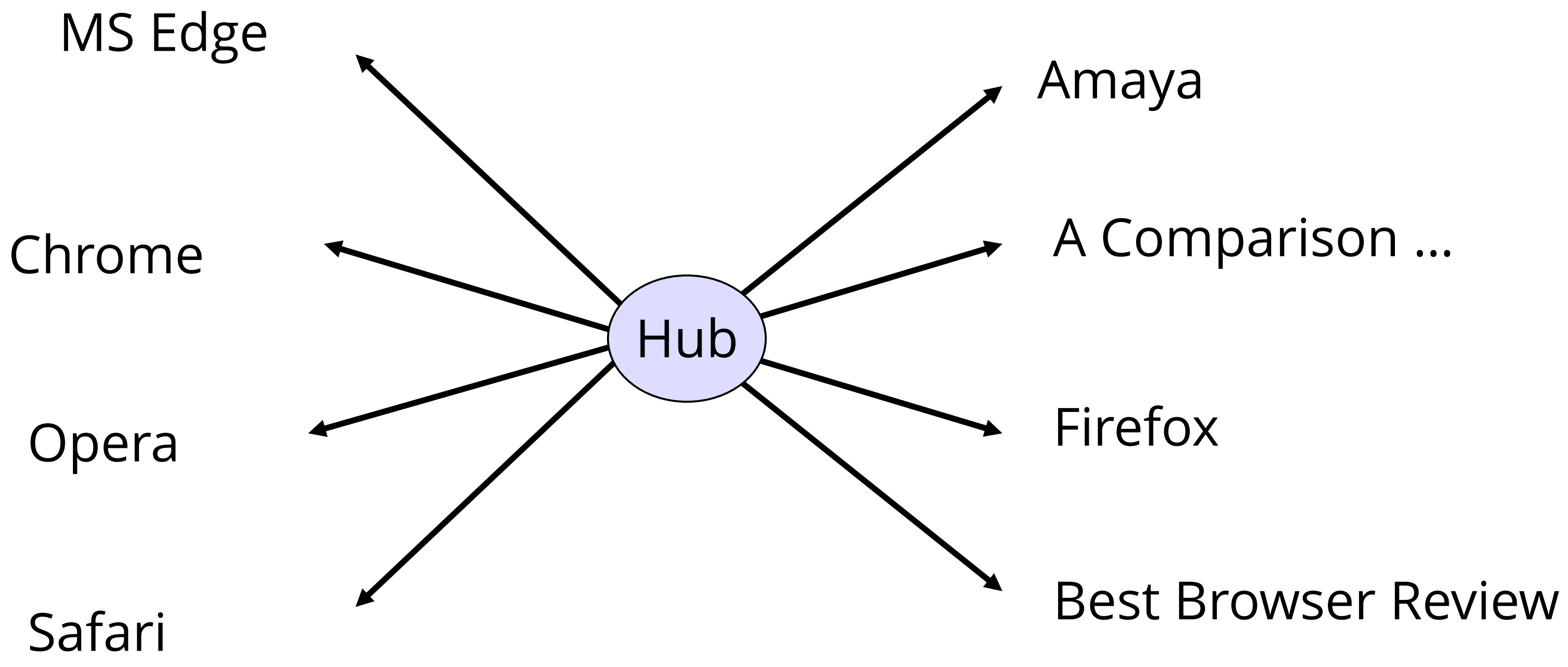
Hubs & Authorities

- A **Hub** is a document that contains links to many other documents
- An **Authority** is a document that many documents link to
- A good Hub links to good Authorities
- A good Authority links to good Hubs



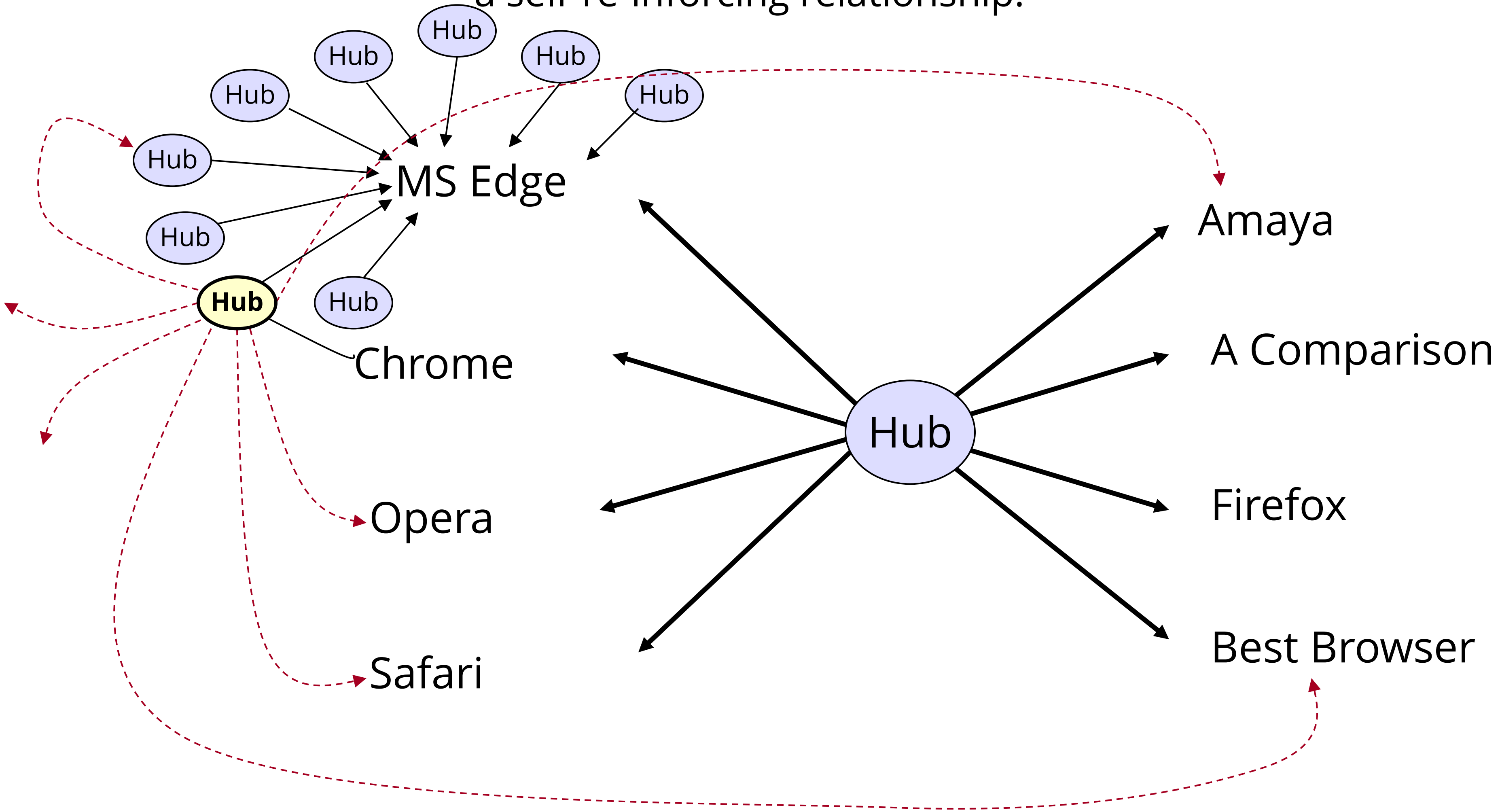
What makes a good Hub… ?

What makes a good hub for the query “web browsers“?



What makes these authorities good?

Good hubs that themselves link into good authorities…
a self-re-inforcing relationship!



So… we need to capture this somehow… we do this by having the calculation done many times (iterations) … e.g. PageRank.

What is PageRank?

- A method for rating the importance of web pages objectively and mechanically using the link structure of the web.
- It was developed by Larry Page (hence the name Page-Rank) and Sergey Brin.
- It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.
- Shortly after, Page and Brin founded Google.

The PageRank Citation Ranking: Bringing Order to the Web. Page et al. 1999.

PageRank

- Query INDEPENDENT score for every document..

- It allocates a PageRank score to every document in an index, and this score is used (with a text score – e.g. TFIDF) when ranking documents.

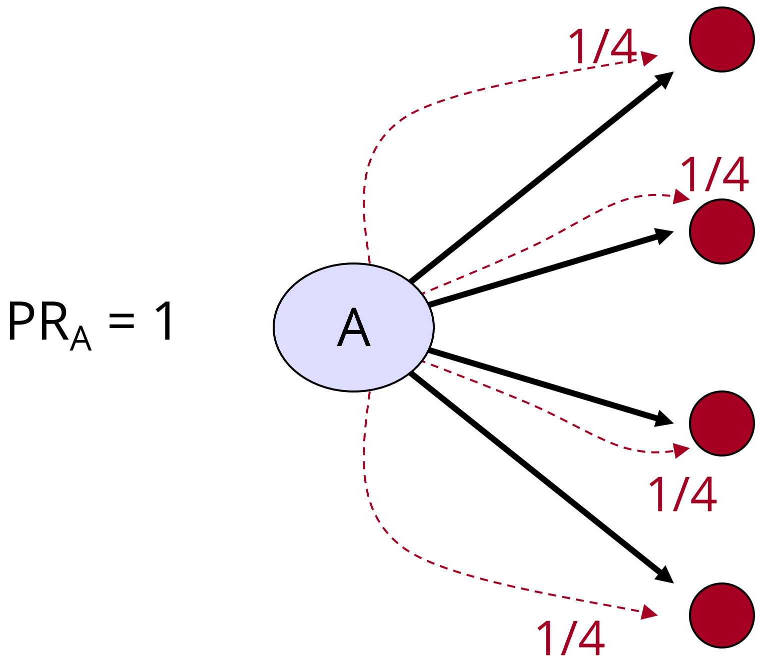
- Simple Iterative Algorithm
  - Iterate many times..

- A simulation of a random user's behaviour when browsing the web.
  - Equivalent to a user randomly following links, or getting bored and randomly jumping to a random page anywhere on the WWW. In effect it is based on the probability of a user landing on any given page.

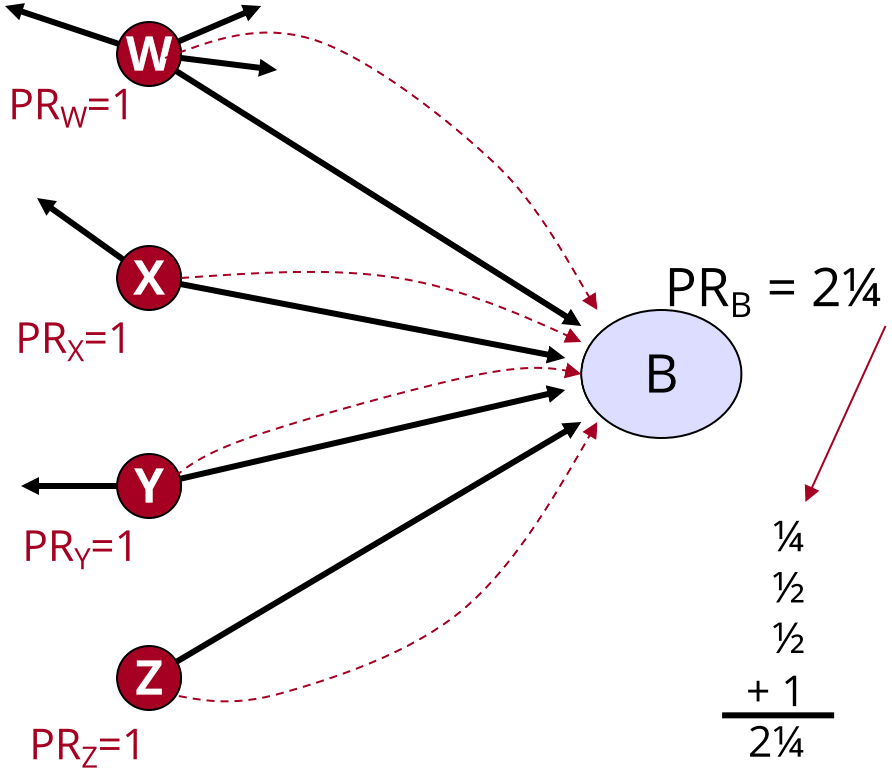
- This can be applied to other graphs than the WWW graph… social networks, blog comments?

Key points

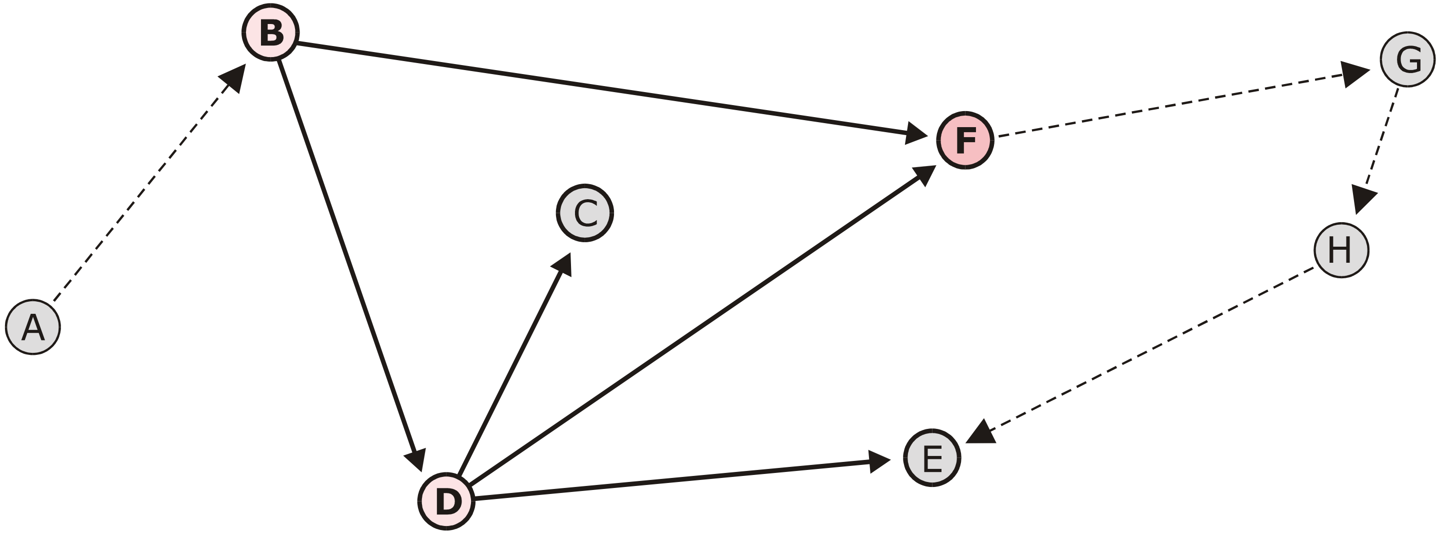
The PR of A is divided equally among its out-links



The PR of B is equal to the sum of the transferable PR of all its in-links



For example



the PageRank PR_F of document F is equal to PR_B divided the out-degree of B summed with PR_D divided by the out-degree of D.

$$PR_F = \frac{PR_B}{2} + \frac{PR_D}{3}$$

The iterative PageRank technique

1, Calculate a pre-iteration PageRank score for each document

$$for\,all\;\; n\;\; in\, N, \quad PR_n = \frac{1}{N}$$

2, Calculate PageRank score for each document

$$PR'_n = c \cdot \sum_{m \in S_n} \frac{PR_m}{outd\; egree_m}$$

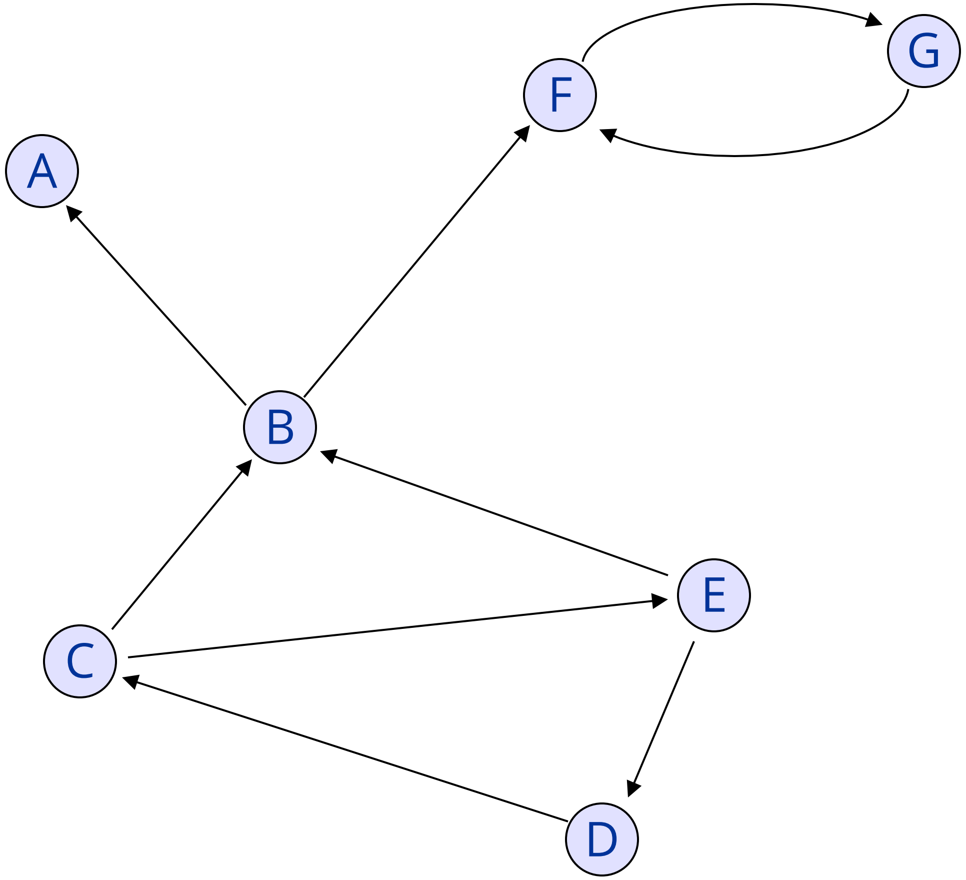
...assume c = 1

3, Store new PageRank scores

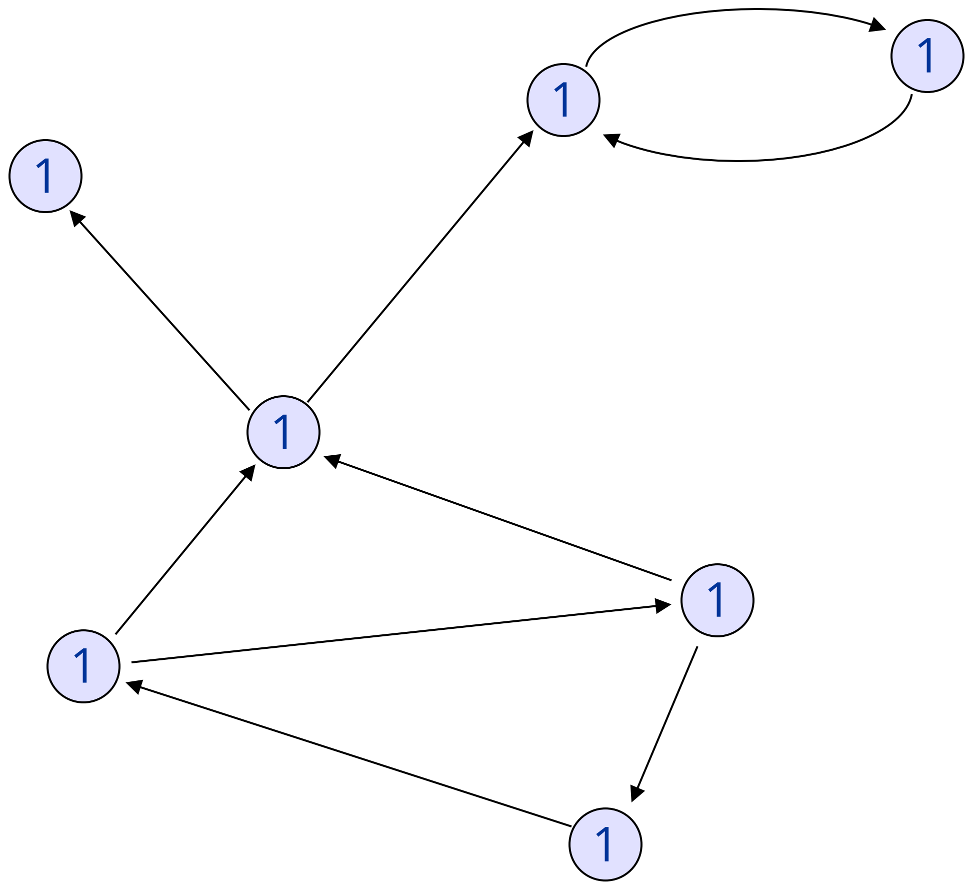
$$for\,all\;\; n\;\; in\, N, \quad PR_n = PR'_n$$

4, If not stop, then goto 2

A simple Web Graph

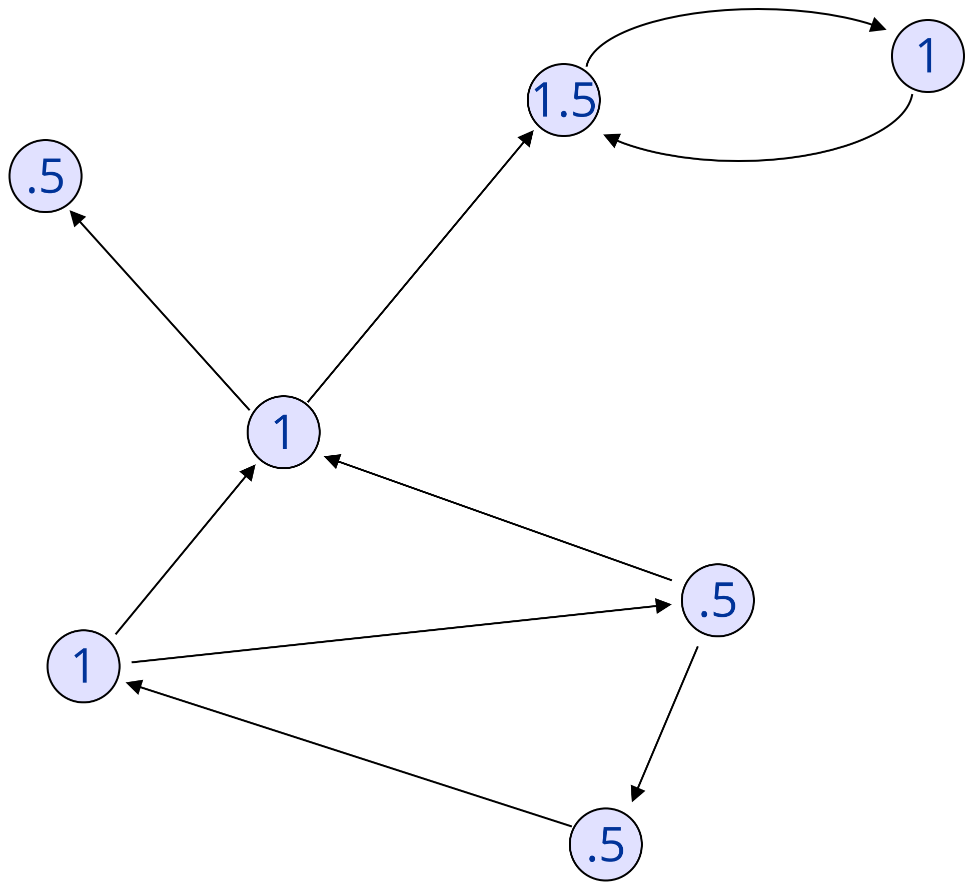


A simple Web Graph



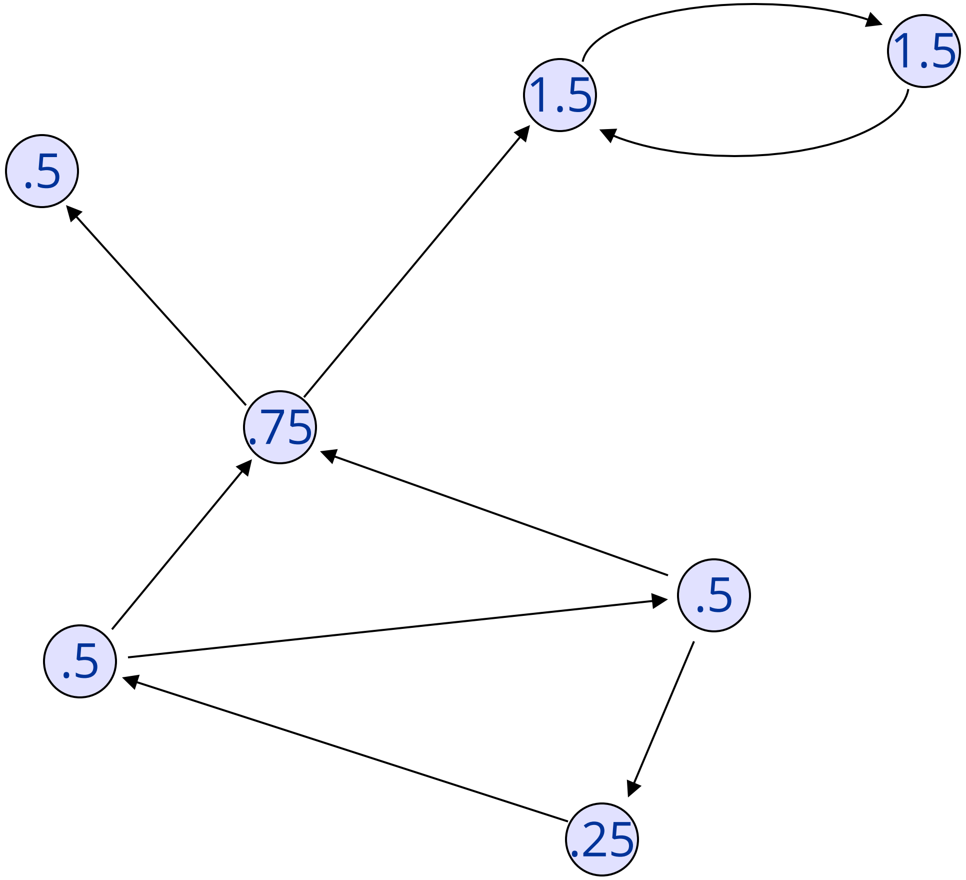
Total = 7.0

A simple Web Graph – after Iteration 1



Total = 6.0

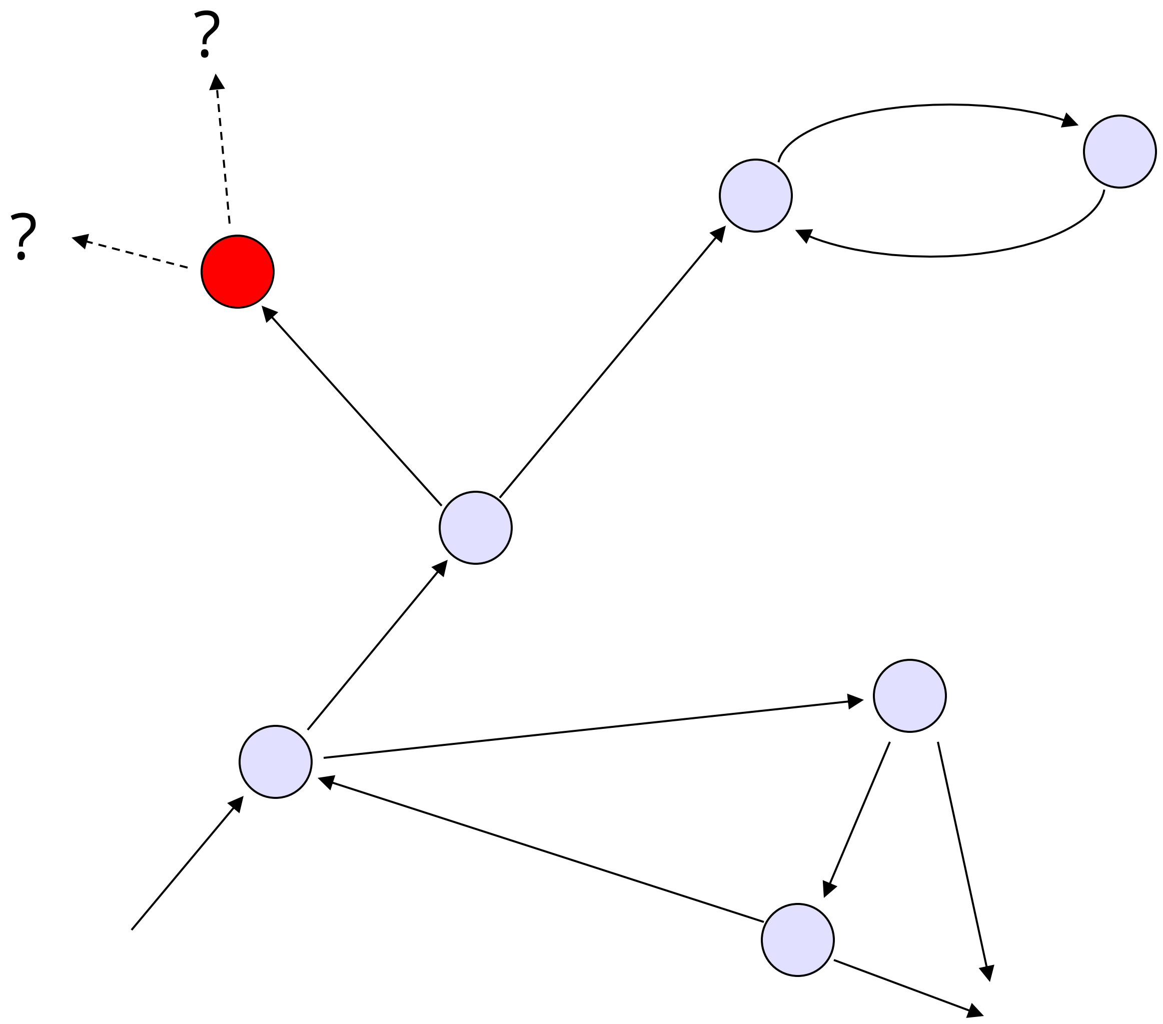
A simple Web Graph – after Iteration 2



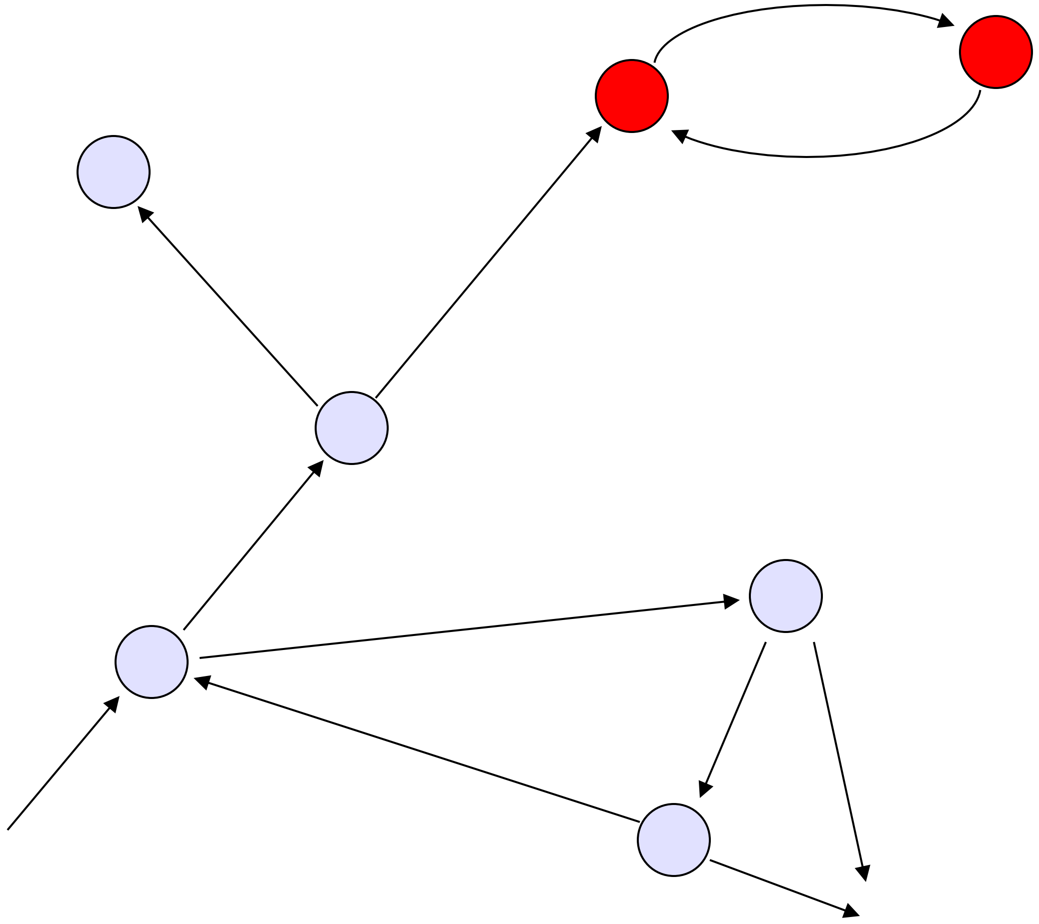
Total = 5.5

What are the problems of PageRank?

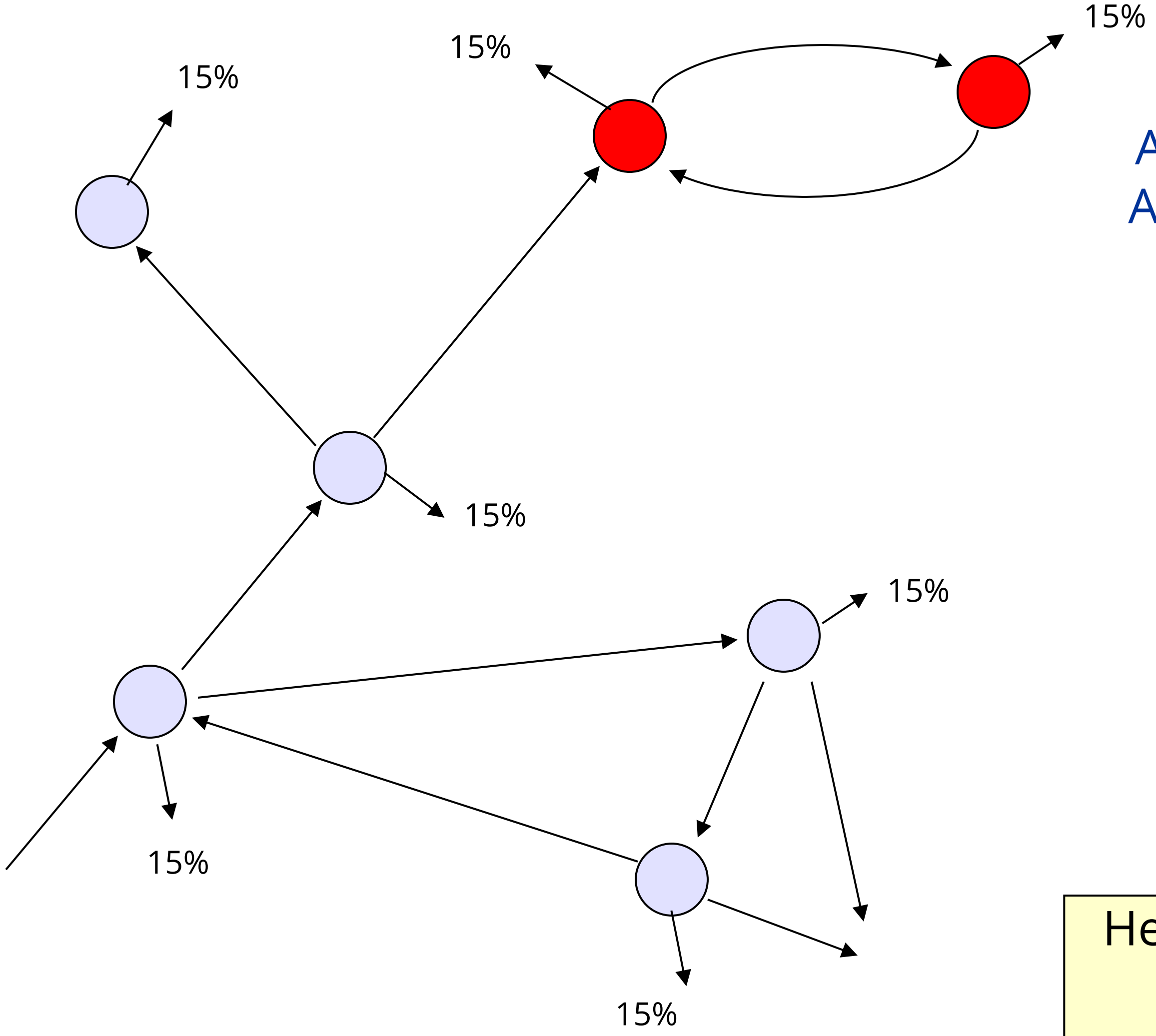
PageRank – Problem 1 (Dangling Links)



Page Rank – Problem 2 (Cycles)



Page Rank – Problem 2 (Cycles)



A Vector over
All Web Pages

0.14	Doc 1
0.14	Doc 2
0.14	Doc 3
0.14	Doc 4
0.14	Doc 5
0.14	Doc 6
0.14	Doc 7

Hence if all PageRanks sum to 1.0, then ||E|| = 0.15

PageRank

- Iterative algorithm
  - Update scores of nodes at each iteration
  - Until convergence: no (or very small) score changes

- Pregel / Giraph / GraphX can all do this by design
  - Local computation + message passing to other nodes

PageRank in M/R

- MAP
  - A node passes its PageRank score contribution to nodes it points to
- REDUCE
  - Each node sums up all contributions received
- Each iteration is a M/R job
  - Or two if you redistribute score of nodes with no outlinks
- Termination
  - Scores don’t change
  - Fixed number of iterations

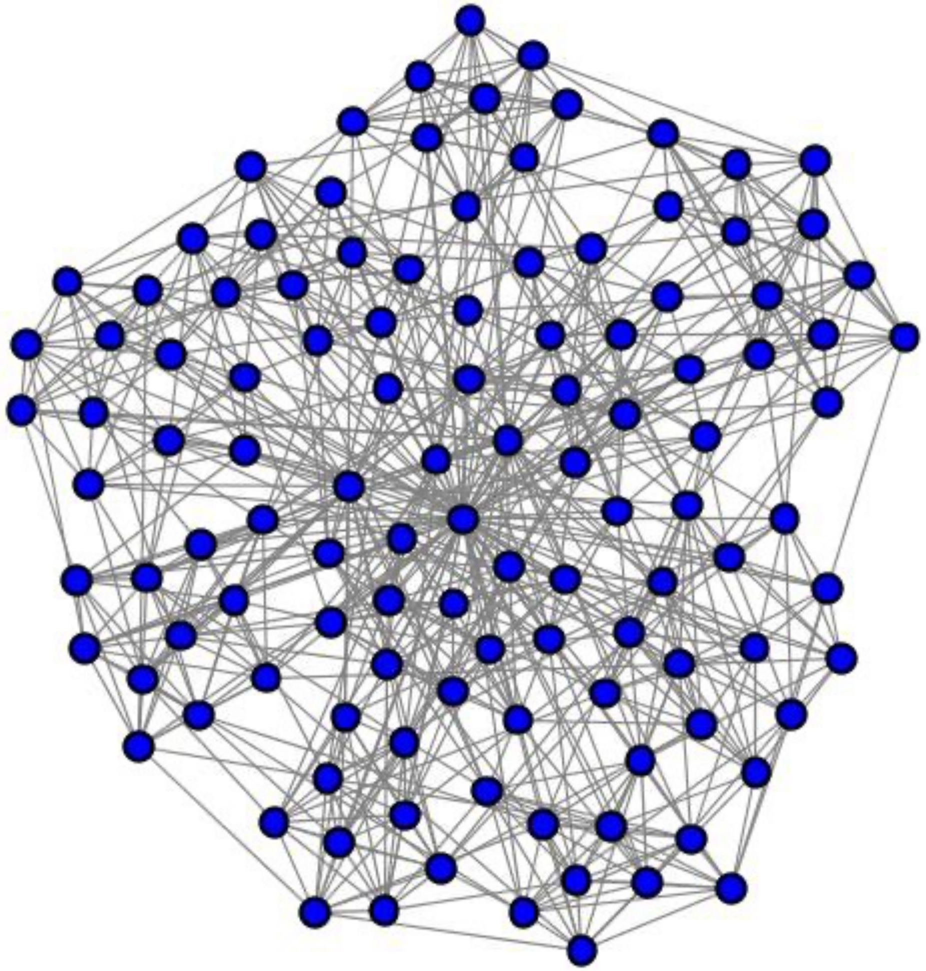
PageRank in M/R

- For dense graph MR jobs are dominated by sending data across the machines in the cluster
  - Use combiners to aggregate results at the MAP level
  - Heuristics: e.g. web pages from the same domain to the same MAP

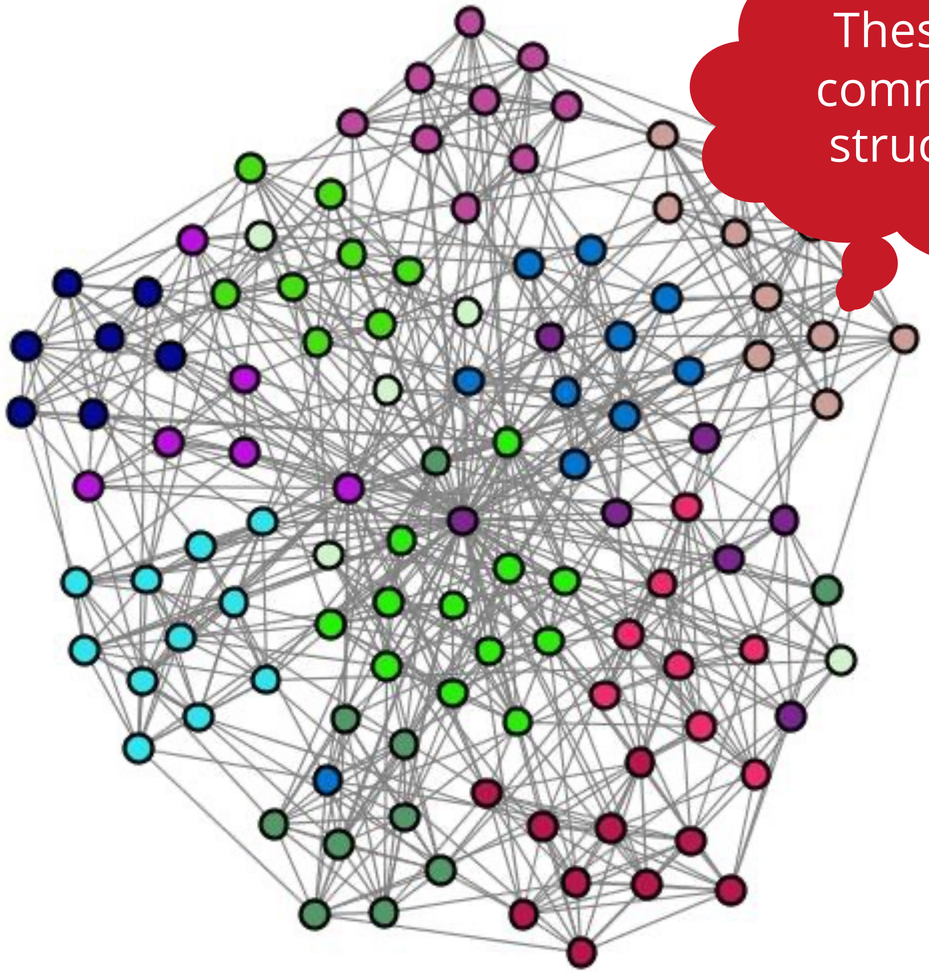
Outline

- Single-source Shortest Path
- PageRank
- **Community detection**

Challenge: Given a graph like this, what can we learn from it?



Probably not much, but what about this one here?



These are community structures

Communities

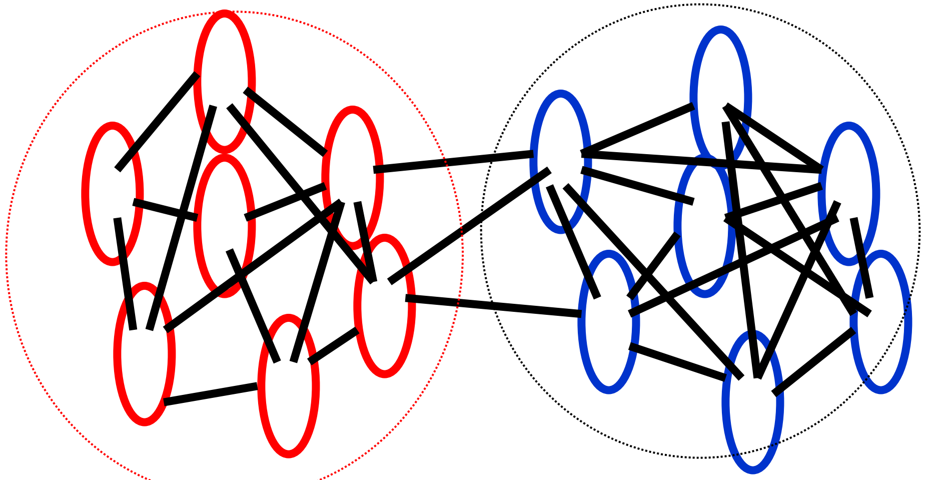
- Formed by individuals such that those within a group interact with each other more frequently than with those outside the group

  a.k.a. **group, cluster, cohesive subgroup, module in different contexts**

Graph-based community structure

- Groups of vertices within which connections are dense but between which they are sparser.

  - Within-group (intra-group) edges.
    - High density
  - Between-group (inter-group) edges.
    - Low density.



Community structure

- Communities are of interest in many cases:
  - World Wide Web
  - Citation networks
  - Social networks
  - Metabolic networks
- Properties of communities may be quite different from average properties of a network

Examples

- Real-world network: World Wide Web
  - Nodes : web pages
  - Edges : hyper-references
- Communities: Nodes on related topics

- Real-world network: Metabolic networks
  - Nodes : metabolites
  - Edges : participation in a chemical reaction
- Communities: Functional modules

Communities in social media

- Human beings are social
- Easy-to-use social media allows people to extend their social life in unprecedented ways
- Difficult to meet friends in the physical world, but much easier to find friend online with similar interests
- Interactions between nodes can help determine communities

Two types of groups in social media

- **Explicit Groups:** formed by user subscriptions
- **Implicit Groups:** implicitly formed by social *interactions*

Community detecion in social media data

- Some social media sites allow people to join groups, is it necessary to extract groups based on network topology?
  - Not all sites provide community platform
  - Not all people want to make effort to join groups
  - Groups can change dynamically

- Network interaction provides rich information about the relationship between users
  - Can complement other kinds of information, e.g. user profile
  - Help network visualization and navigation
  - Provide basic information for other tasks, e.g. recommendation
  - Note that each of the above three points can be a research topic.

Challenge – Community detection

- Discovering groups in a network where individuals’ group memberships are not explicitly given
- Not quite the same as graph partitioning – the number K of modules is not known a priori.
- It’s an unsupervised learning task; “ground truth” not available.

Finding communities in a graph

- Hierarchical clustering
  - Distance measure between two nodes (u,v)
  - Use adjacency matrix with, e.g., **cosine similarity**

- Modularity
  - Find the best grouping of nodes based on modularity score
    - Girvan-Newman method: Starts with the full graph and breaks it up to find communities

Betweenness centrality

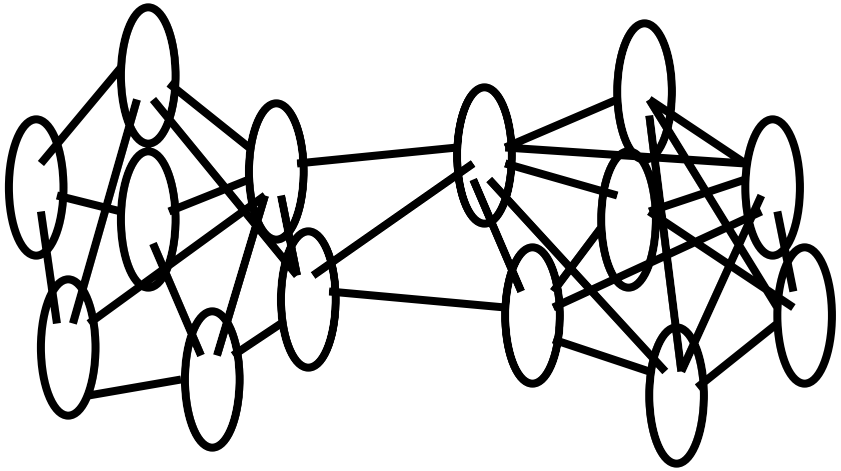
- Centrality measure: most important nodes
  - Based on shortest paths

- A score for a each node *v* that indicates how many shortest paths between all possible pairs of nodes in *G* pass through *v*

- Indicates nodes that serve as connector in the graph/network
  - Influencers / hubs

Girvan-Newman algorithm

- Based on edge betweenness centrality
- "How often does an edge form part of a shortest path?"



Girvan-Newman algorithm

1. Calculate the betweenness centrality of all edges

2. Remove the edge having the highest betweenness centrality

3. Repeat S1 and S2 until a certain threshold is achieved

Modularity clustering

- A measure of the network structure
  - Edges in a group vs edges distributed at random
- Identify best communities in social networks
- Used as clustering approach for nodes in the network
- Impossible to compute Modularity for all possible grouping of nodes
  - Hierarchical bottom-up: start with one node per community and aggregate to check if it increases modularity

Cuijuan Wang, Wenzhong Tang, Bo Sun, Jing Fang and Yanyang Wang, "Review on community detection algorithms in social networks," *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, Nanjing, 2015, pp. 551-555, doi: 10.1109/PIC.2015.7489908.

Summary

- Single-source Shortest Path
- PageRank
- Community detection

Next week – Recommender systems

- Basics
- Collaborative Filtering
- Content-based Recommendations
- Distributed Recommender Systems