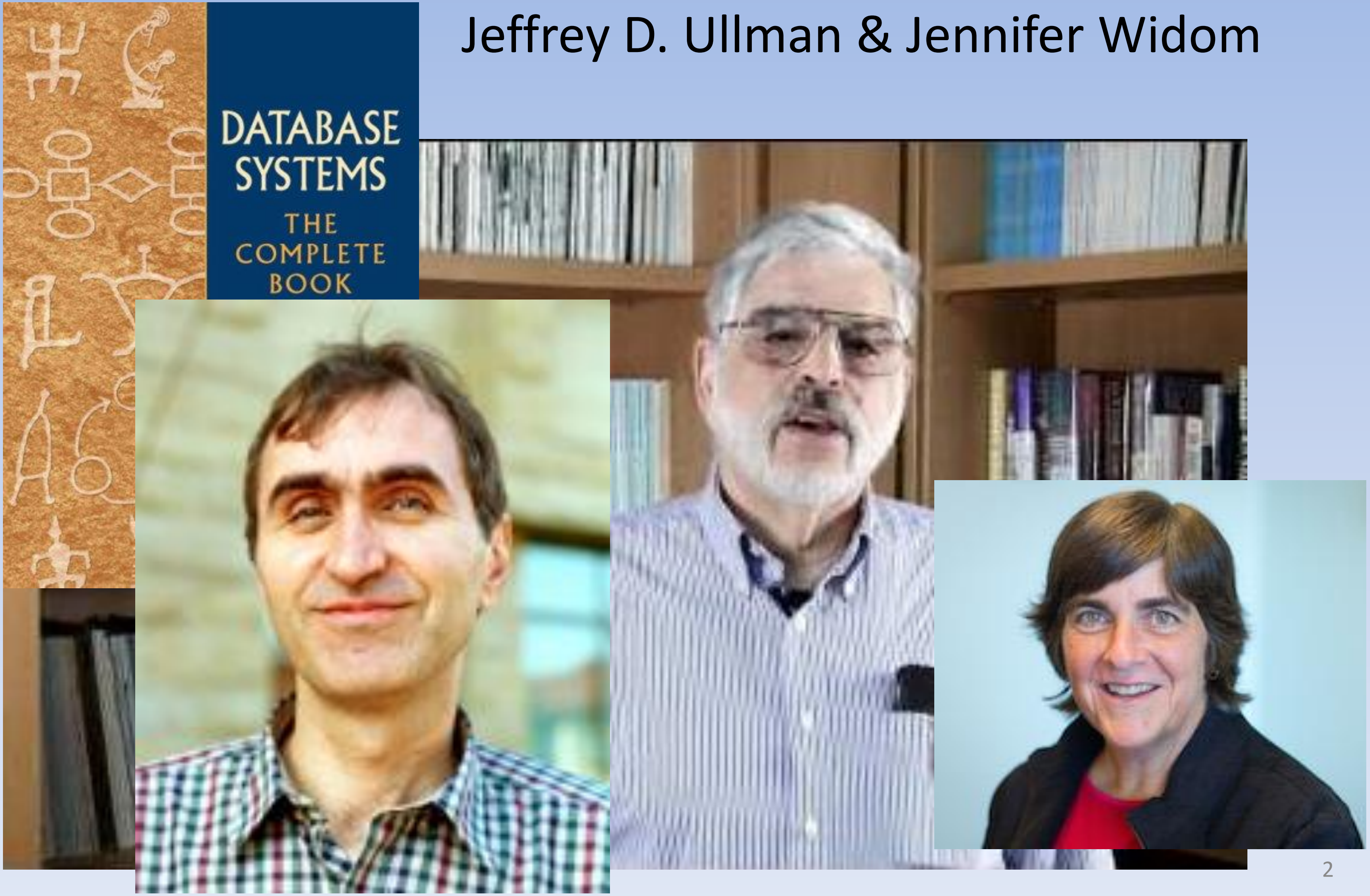
Computer Science 226, Advanced Database Systems (3 units)

Class Instructor:	David Ruby
Class Hours:	TTh 5:00 – 6:15
Office:	Science II – 273
Email:	druby@csufresno.edu

Database Systems: The Complete Book(3rd)

by Hector Garcia-Molina,

Jeffrey D. Ullman & Jennifer Widom



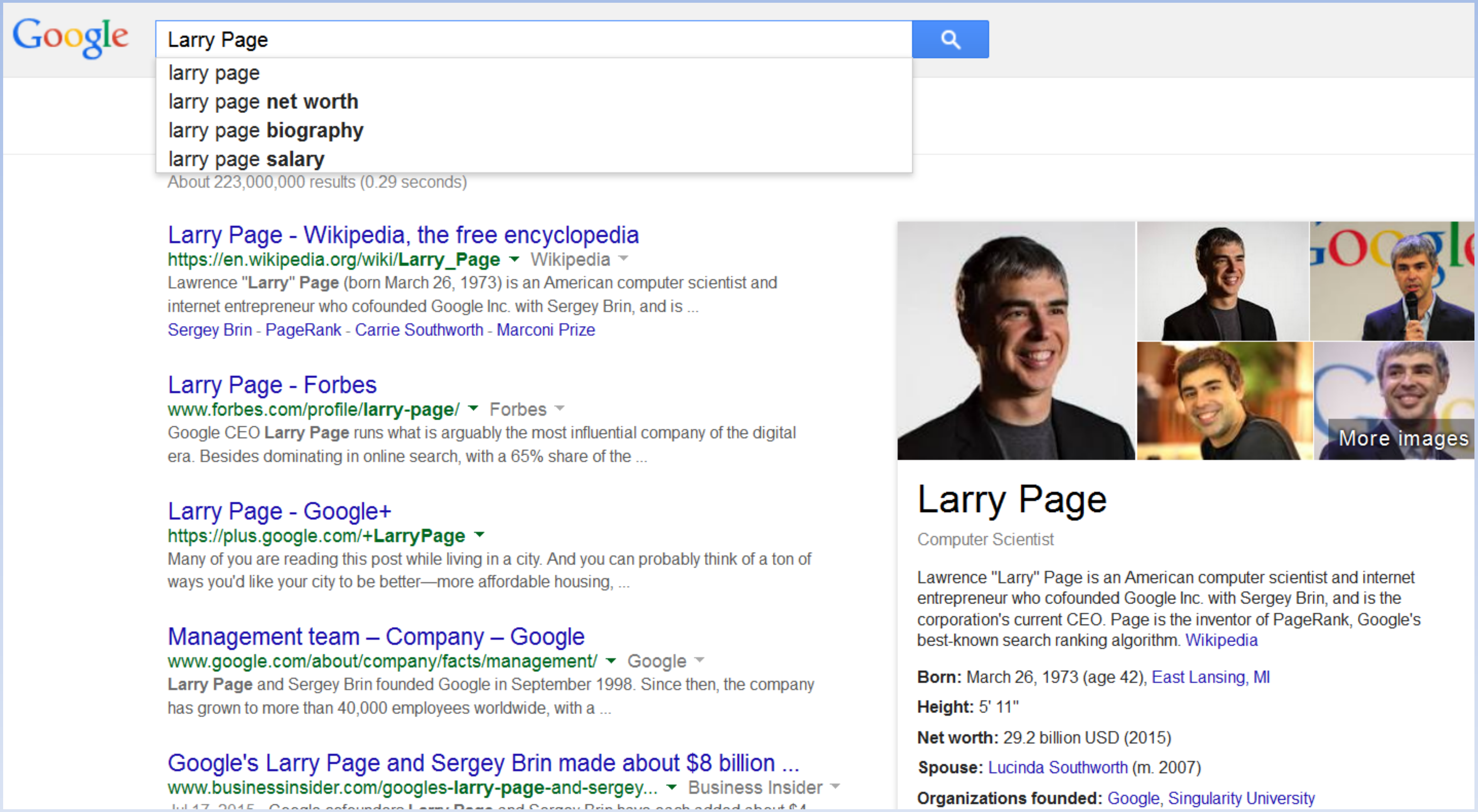
Database Systems and the Internet Chapter 23

- PART V:
- Other Issues in Management of Massive Data
  - 21 Information Integration
  - 22 Data Mining
  - 23 Database Systems and the Internet

Database Systems and the Internet Chapter 23

- 23.1 The Architecture of a Search Engine
- 23.2 PageRank for Identifying Important Pages
- 23.3 Topic-Specific PageRank
- 23.4 Data Streams
- 23.5 Data Mining of Streams

Searching w/ Internet



About 223,000,000 results (0.29 seconds)

Larry Page - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Larry_Page** Wikipedia
Lawrence "**Larry**" **Page** (born March 26, 1973) is an American computer scientist and internet entrepreneur who cofounded Google Inc. with Sergey Brin, and is ...
Sergey Brin - PageRank - Carrie Southworth - Marconi Prize

Larry Page - Forbes
www.forbes.com/profile/**larry-page**/ Forbes
Google CEO **Larry Page** runs what is arguably the most influential company of the digital era. Besides dominating in online search, with a 65% share of the ...

Larry Page - Google+
https://plus.google.com/+**LarryPage**
Many of you are reading this post while living in a city. And you can probably think of a ton of ways you'd like your city to be better—more affordable housing, ...

Management team – Company – Google
www.google.com/about/company/facts/management/ Google
Larry Page and Sergey Brin founded Google in September 1998. Since then, the company has grown to more than 40,000 employees worldwide, with a ...

Google's Larry Page and Sergey Brin made about $8 billion ...
www.businessinsider.com/googles-**larry-page**-and-sergey... Business Insider

Larry Page
Computer Scientist

Lawrence "Larry" Page is an American computer scientist and internet entrepreneur who cofounded Google Inc. with Sergey Brin, and is the corporation's current CEO. Page is the inventor of PageRank, Google's best-known search ranking algorithm. Wikipedia

Born: March 26, 1973 (age 42), East Lansing, MI

Height: 5' 11"

Net worth: 29.2 billion USD (2015)

Spouse: Lucinda Southworth (m. 2007)

Organizations founded: Google, Singularity University

23.1: The Architecture of a Search Engine

- The Search Engine has become of the most important tools of the 21st century!

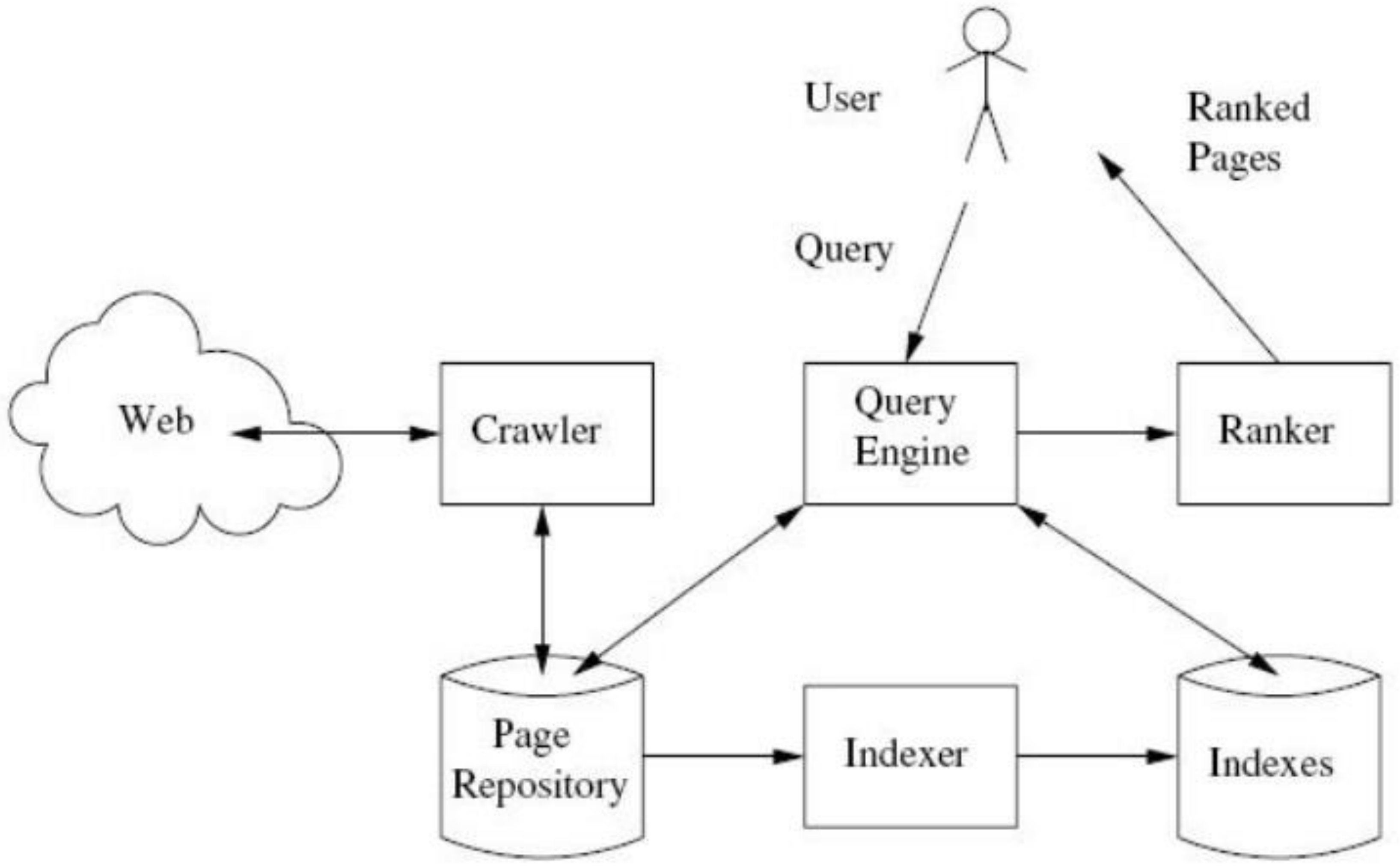
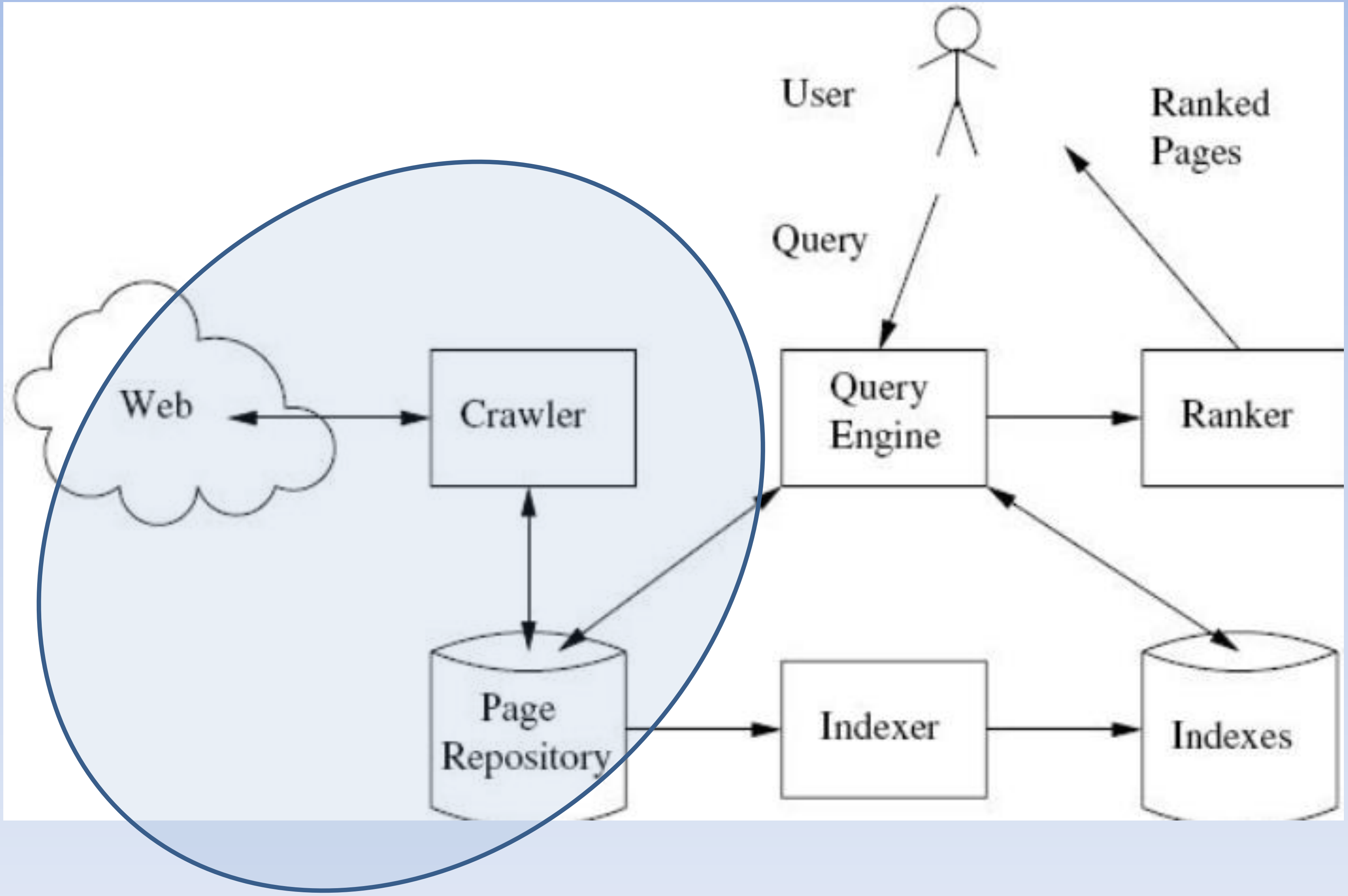


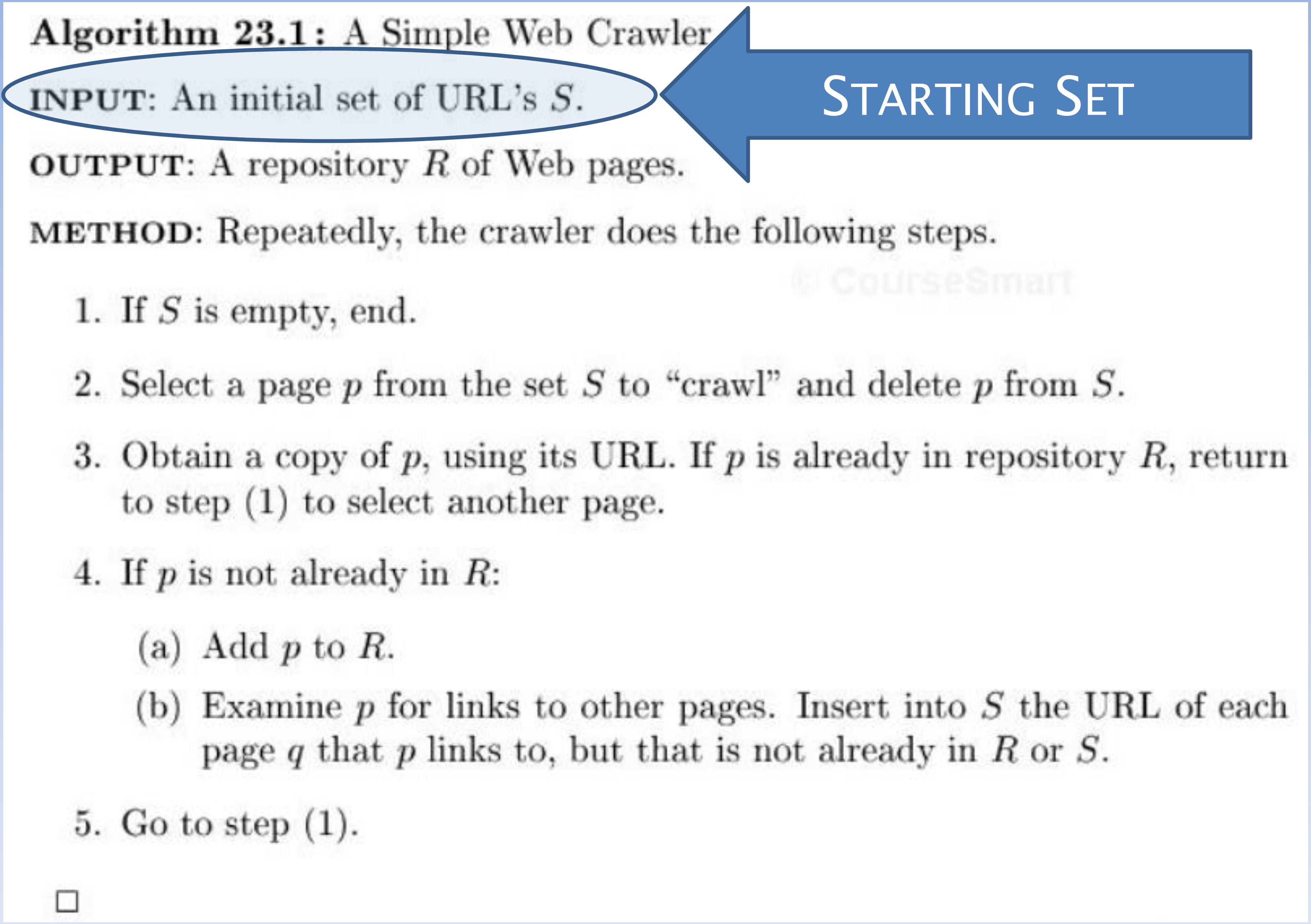
Figure 23.1: The components of a search engine

Components of a Search Engine

- Web Crawler
  - Web pages from throughout the internet must be accessed and processed
- Queries
  - Based on results of web crawler, queries must be answered.
  - queries in form of word or words describing pages desired.

Web Crawler


User
Ranked Pages
Query
Web
Crawler
Query Engine
Ranker
Page Repository
Indexer
Indexes



Algorithm 23.1: A Simple Web Crawler

INPUT: An initial set of URL's S.

STARTING SET

OUTPUT: A repository R of Web pages.

METHOD: Repeatedly, the crawler does the following steps.

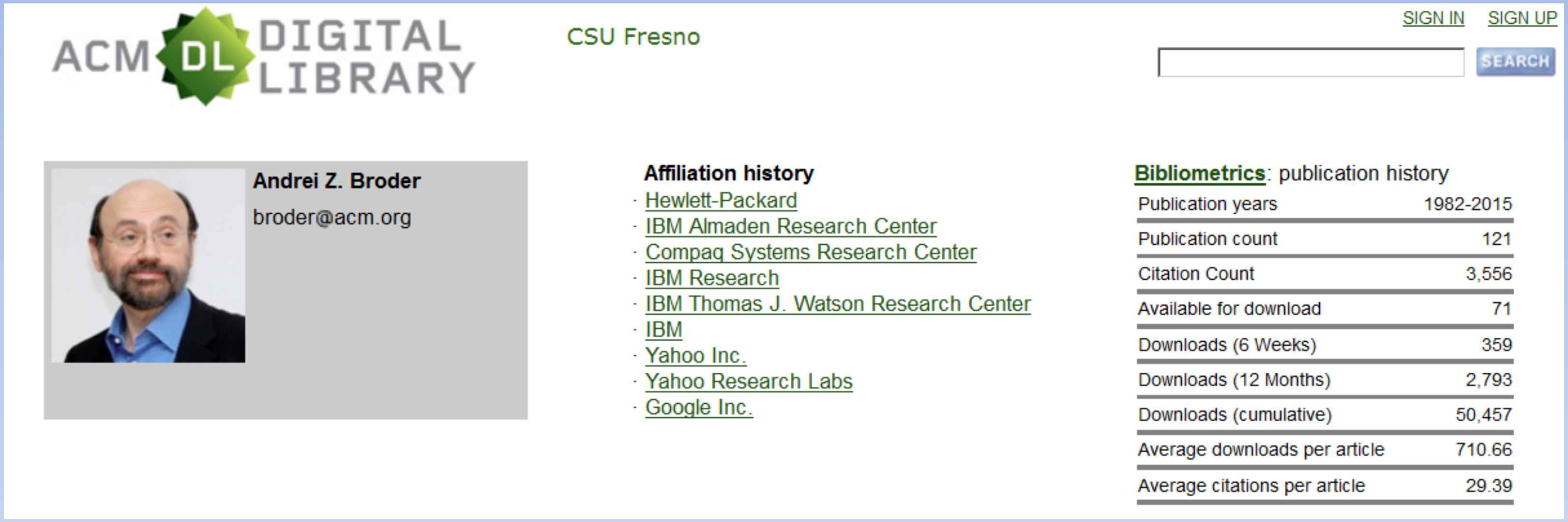
1. If S is empty, end.
2. Select a page p from the set S to "crawl" and delete p from S.
3. Obtain a copy of p, using its URL. If p is already in repository R, return to step (1) to select another page.
4. If p is not already in R:
   (a) Add p to R.
   (b) Examine p for links to other pages. Insert into S the URL of each page q that p links to, but that is not already in R or S.
5. Go to step (1).

□

Web Graph

- Vertices: Web Pages
- Edges: Hyperlinks

Web Graph Structure



CSU Fresno

SIGN IN SIGN UP

SEARCH

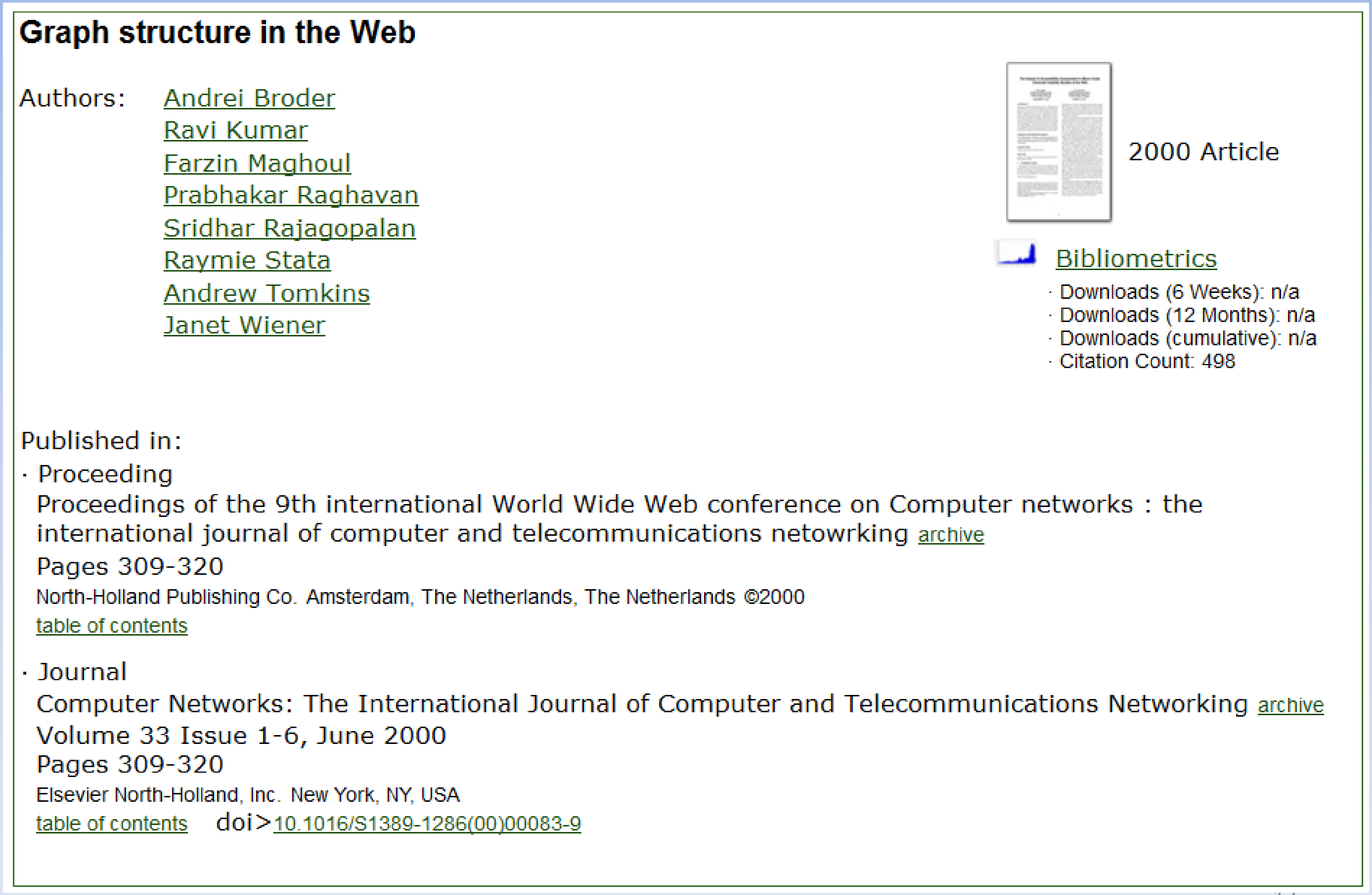
Andrei Z. Broder
broder@acm.org

Affiliation history
- Hewlett-Packard
- IBM Almaden Research Center
- Compaq Systems Research Center
- IBM Research
- IBM Thomas J. Watson Research Center
- IBM
- Yahoo Inc.
- Yahoo Research Labs
- Google Inc.

Bibliometrics: publication history

Publication years	1982-2015
Publication count	121
Citation Count	3,556
Available for download	71
Downloads (6 Weeks)	359
Downloads (12 Months)	2,793
Downloads (cumulative)	50,457
Average downloads per article	710.66
Average citations per article	29.39

- Andrei Broder



Graph structure in the Web

Authors: Andrei Broder
Ravi Kumar
Farzin Maghoul
Prabhakar Raghavan
Sridhar Rajagopalan
Raymie Stata
Andrew Tomkins
Janet Wiener

2000 Article

Bibliometrics

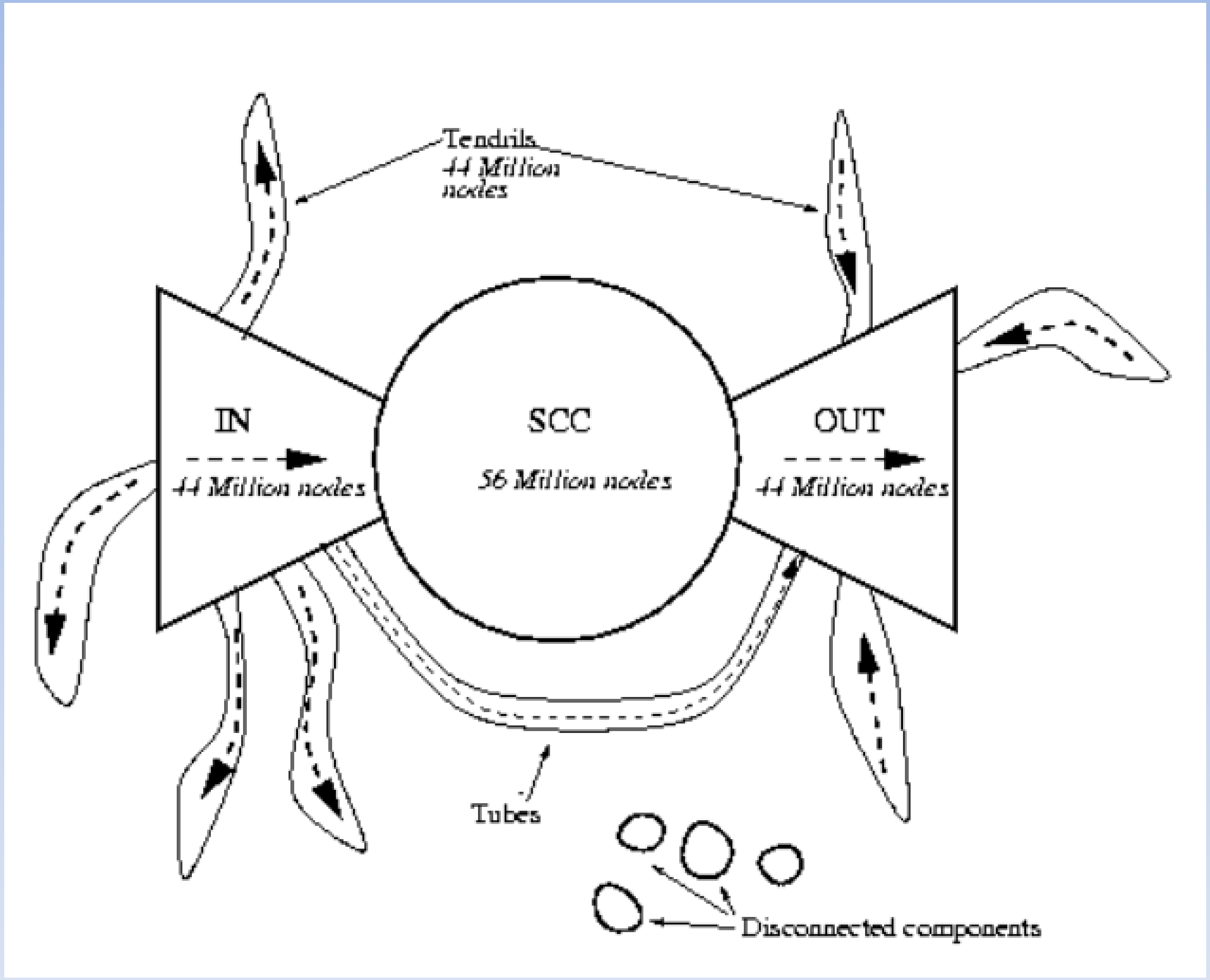
- Downloads (6 Weeks): n/a
- Downloads (12 Months): n/a
- Downloads (cumulative): n/a
- Citation Count: 498

Published in:

- Proceeding
  Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking archive
  Pages 309-320
  North-Holland Publishing Co. Amsterdam, The Netherlands, The Netherlands ©2000
  table of contents

- Journal
  Computer Networks: The International Journal of Computer and Telecommunications Networking archive
  Volume 33 Issue 1-6, June 2000
  Pages 309-320
  Elsevier North-Holland, Inc. New York, NY, USA
  table of contents doi>10.1016/S1389-1286(00)00083-9

Web Graph Structure



- 6/2000

Web Crawling Issues

- Terminating Search
- Efficiently checking if page is in repository
- Selecting next page to search
- Speeding up search w/ Parallelism

Terminating Search

- Maybe limit total number of pages crawled.
- Limit depth of search (Depth-Limited DFS)
  - Some pages generated dynamically.

Managing the Repository

- S: Set of or URL's to Search
- R: Repository of Web Pages
- Efficient Index Structure Required!

- Hashing used to detect duplicates
- Hashing can be used to detect similar pages!

Selecting the Next Page

- Maybe Random?
- Breadth-First is probably better
  - Costly
- Heuristic Search using PageRank!
  - Computing PageRank too costly while searching.
  - Estimating PageRank possible, i.e. looking at number of known in-links to page.
    - Always pick page w/ Highest Number of Known In-Links
      - HEAP

Speeding Up Crawl

- Parallelism! Parallelism!
- Big Data problem
- Map-Reduce developed by Google for this purpose!
- Now a common technique
  - Hadoop
  - Spark

Queries!

- Given a set of words, rank pages from the web!
- Requires an Inverted Index for all words on web.
  - Hundreds of Millions of Words
  - Includes Misspellings, errors codes, jargon, etc…
- Search engines give responses in Fractions of a Second!

Queries

- Once we have the set of pages that match query, they need to be ranked!
- The exact formula is closed guarded secret (Secret Formula)!
- One Components is PageRank!
  - Named for Larry Page / Co-Founder of Google

Ranking Pages

- Does the page have ALL query words?
- Does the page have Query words in Important Positions?
- Does the page have Query Words near each other?
- Do pages link to the page with anchor text that include Query Words!
  - IMPORTANT insight from Google.
  - You may lie about your self,,, But it is harder to get others to confirm it!

PageRank versus SPAM

- Early search engines had problems recognizing SPAM!
- Spammers would put bogus content on their pages (often in font w/ same color as background).
- PageRank tries to measure the likelihood you'll end up on a page if you clicked links randomly!
  - Web page is not very important if people don't link to it.
  - If many people link to a page, the probability of reaching it randomly goes up!

PageRank w/ Markov Model

- State Values = {p_1, p_2, p_3, … p_n}
  - Possible web pages on the internet
- Transition Matrix w/ $m_{ij} = \frac{1}{\mathrm{r}}$ IF:
  - page j links to page i
  - page j links to r different pages
- $m_{ij} = 0$ if no link from page j to page i
- This markov model will converge to a Stationary Distribtution,
  - The probabilities for each state value is the PageRank for the web page!

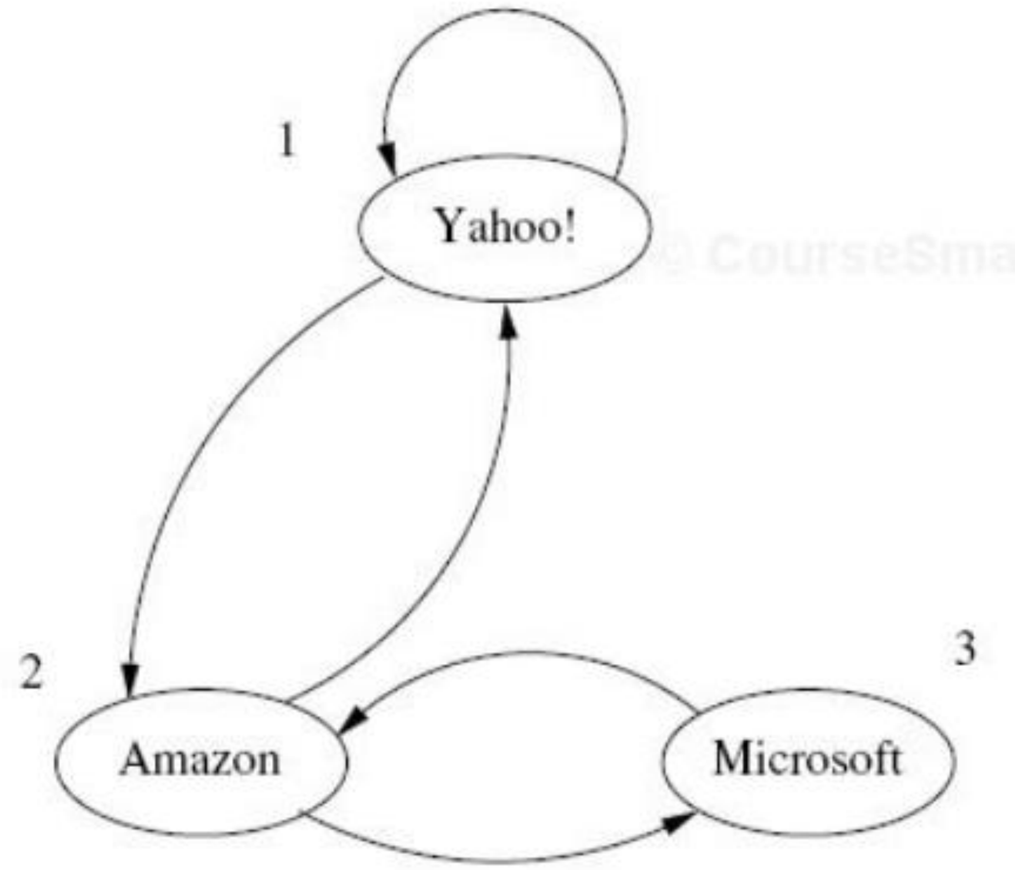
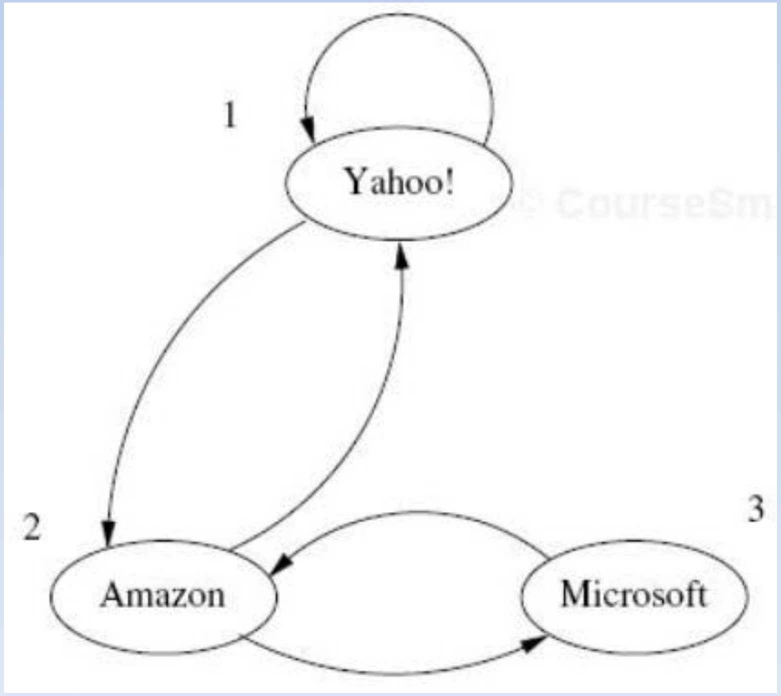


Figure 23.2: The Web in 1839

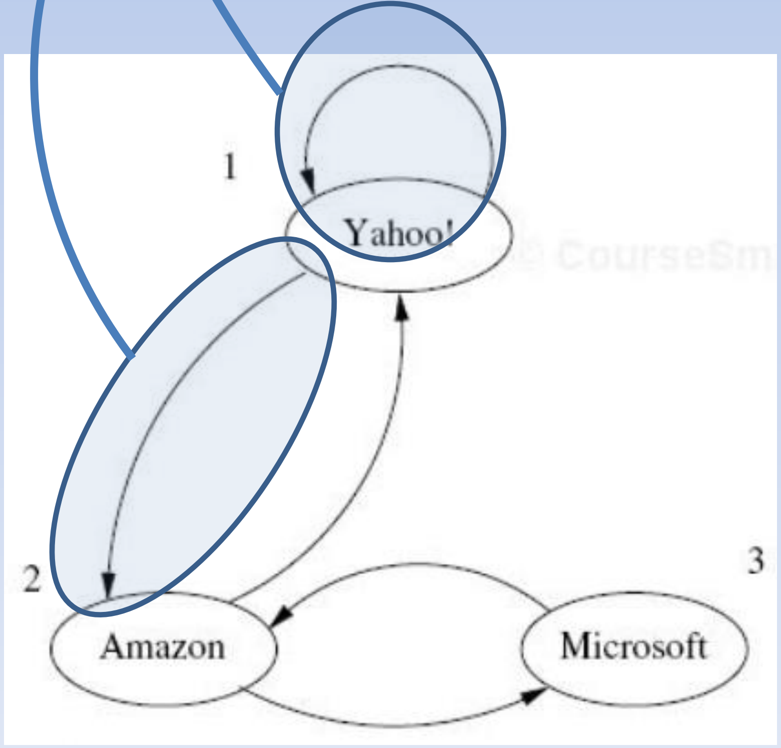
1839 Web Transition Model

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

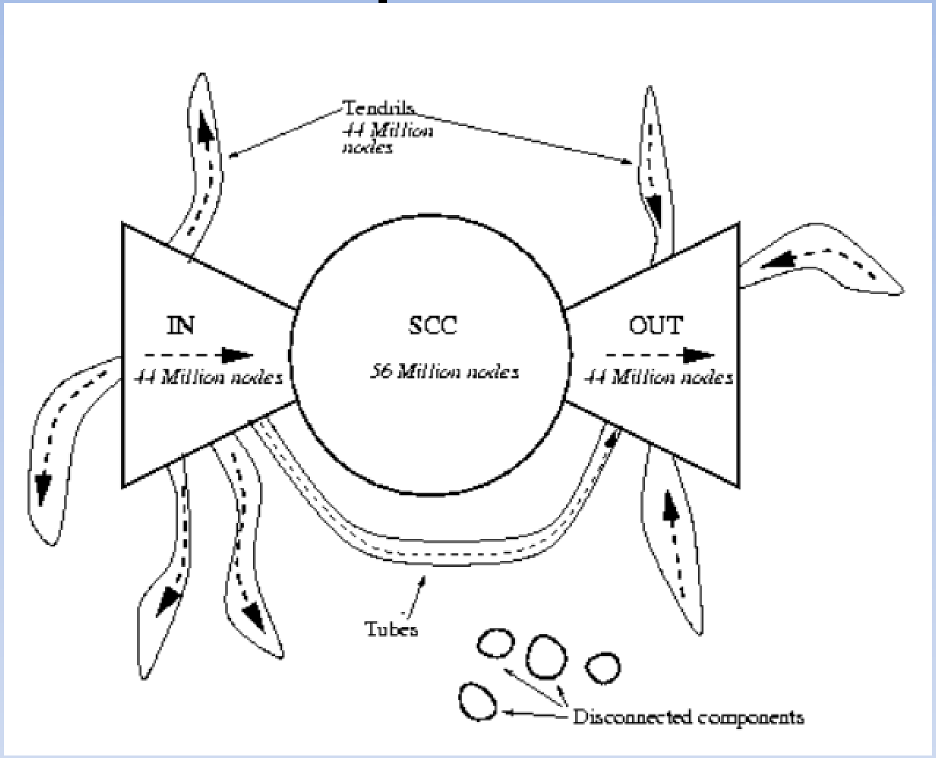


1839 Web Transition Matrix

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$



Remember: Web Graph Structure



- Some web pages have no outlinks!
  – Dead Ends!
- Some sets of web pages have no links outside set.
  – Spider Traps!

Markov Model w/ Spider Trap

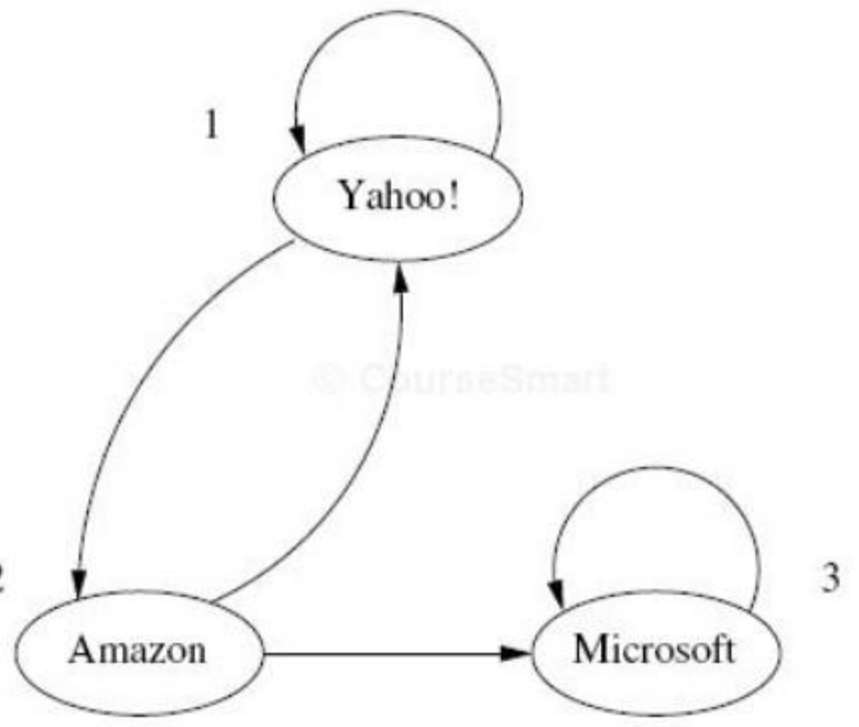
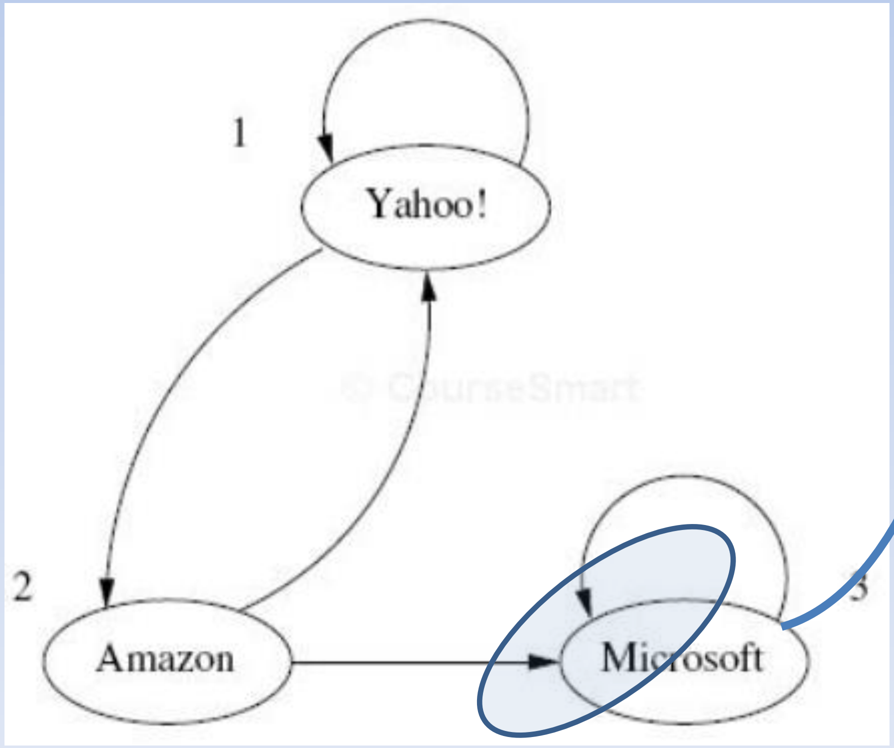


Figure 23.3: The Web, if Microsoft becomes a spider trap

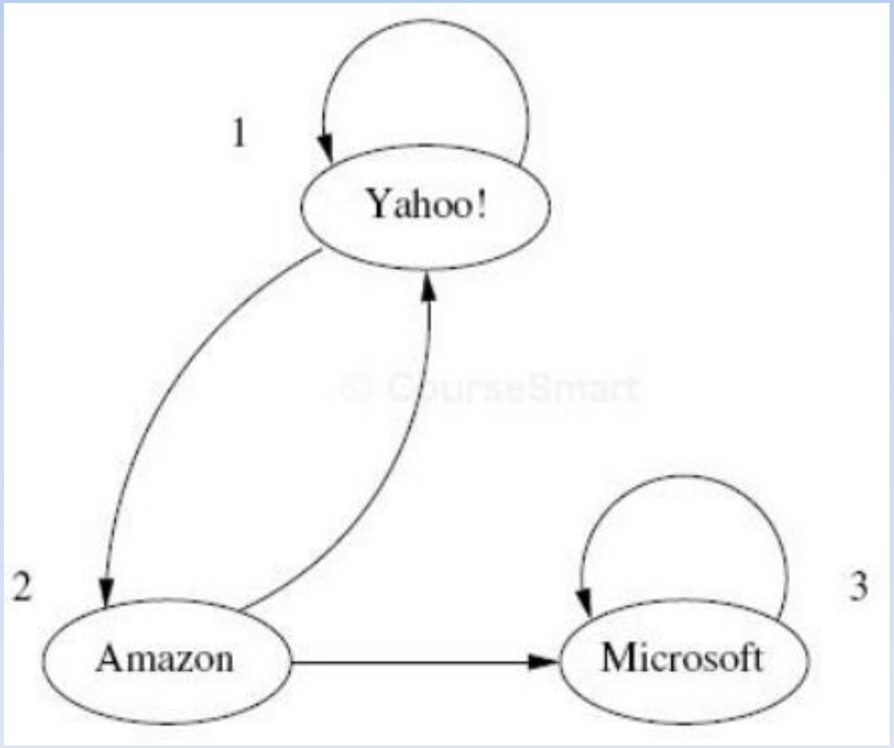
Microsoft w/ Spider Trap!

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$



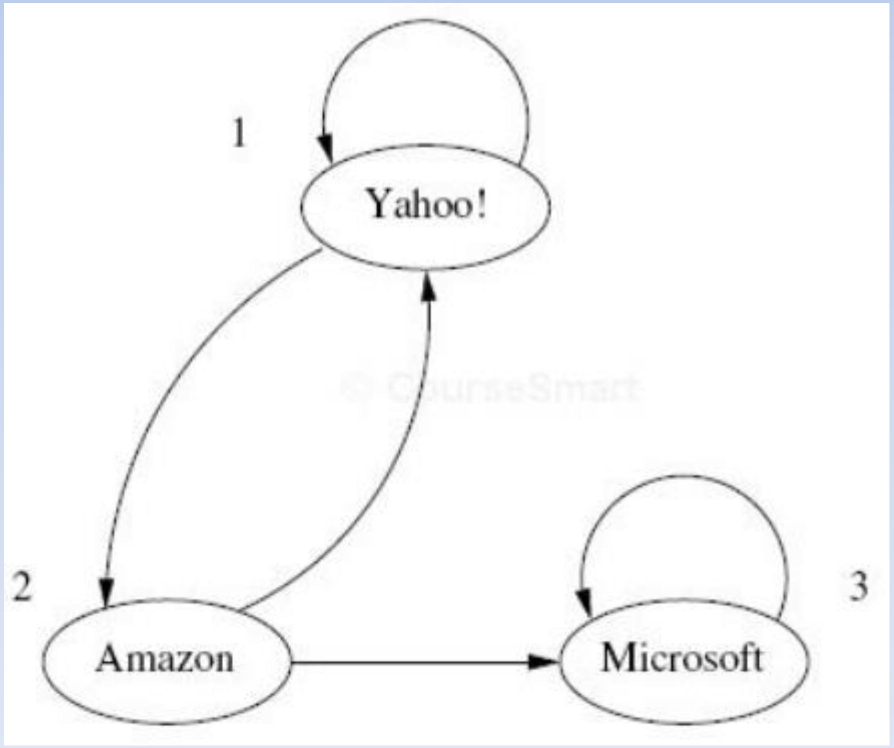
Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_1 \\ a_1 \\ m_1 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$



Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_\infty \\ a_\infty \\ m_\infty \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Markov Model w/ Dead End

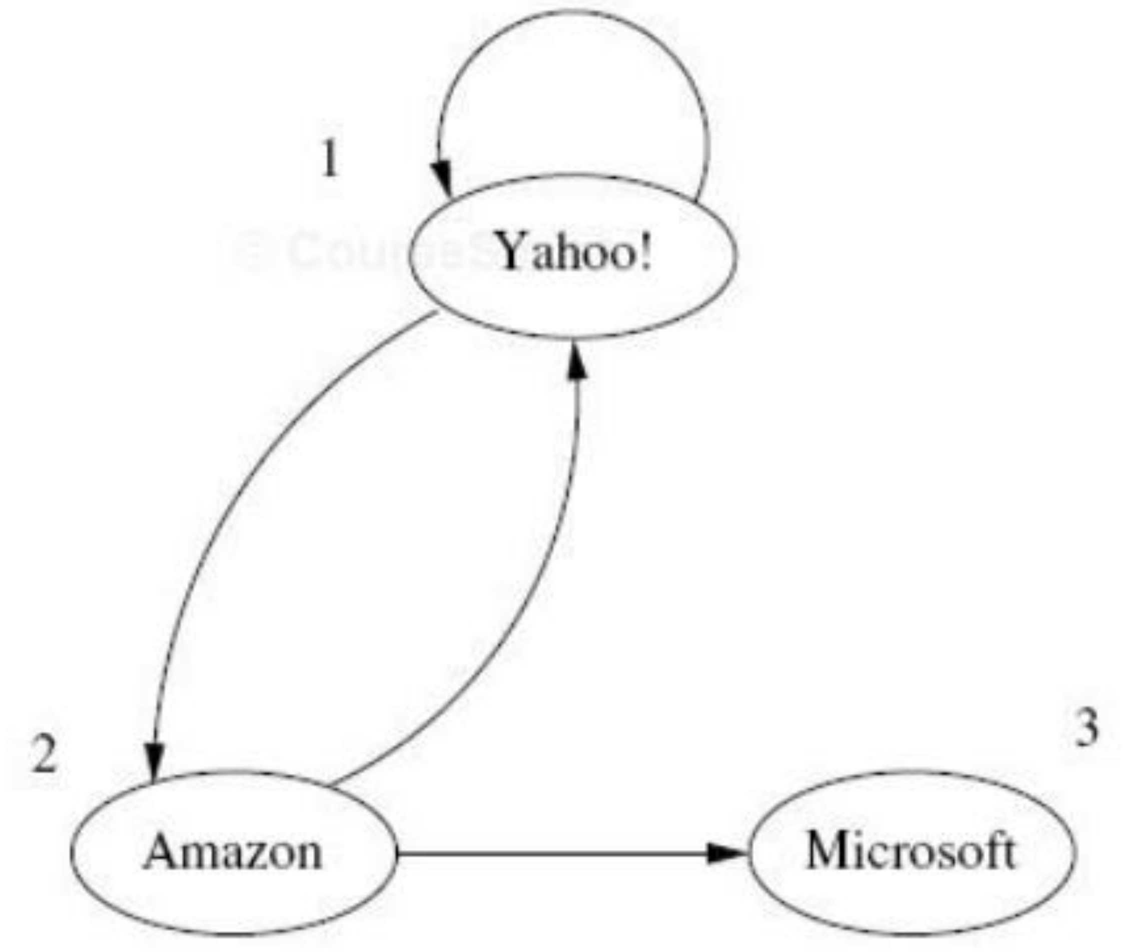
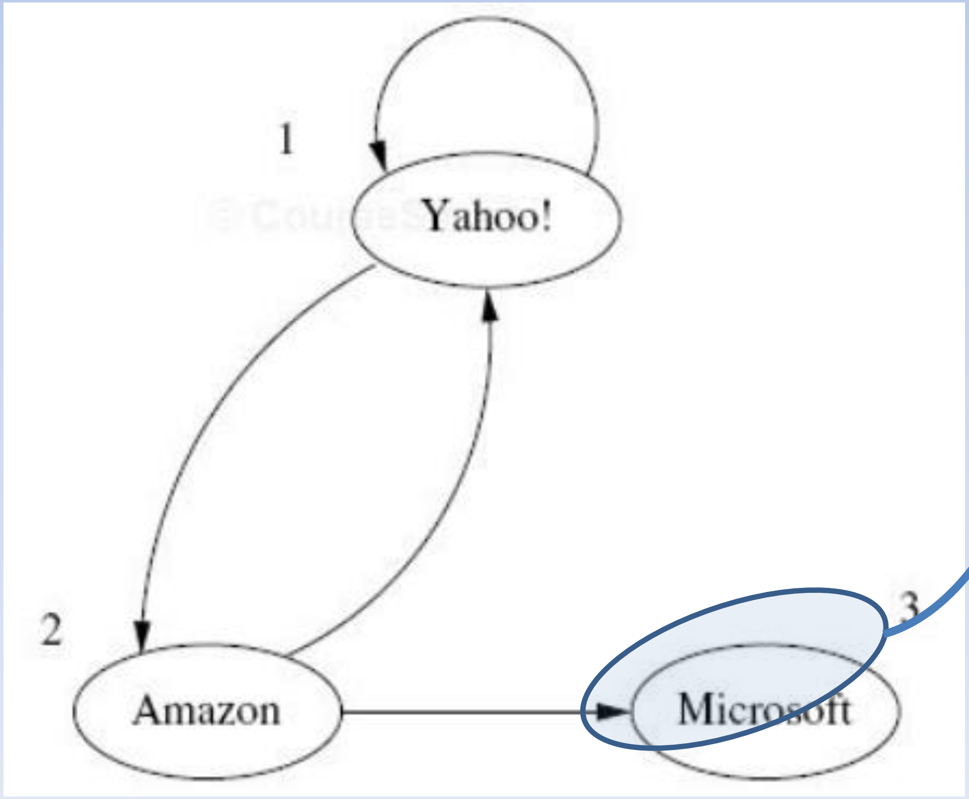


Figure 23.4: Microsoft becomes a dead end

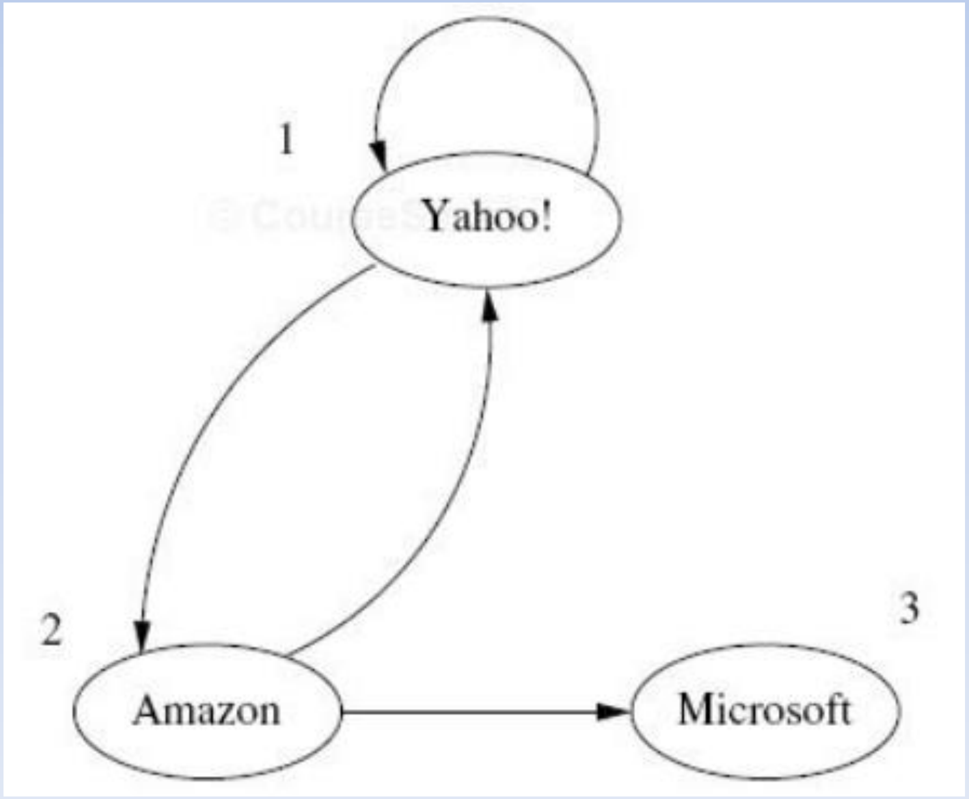
Microsoft w/ Spider Trap!

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix}$$



Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_\infty \\ a_\infty \\ m_\infty \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

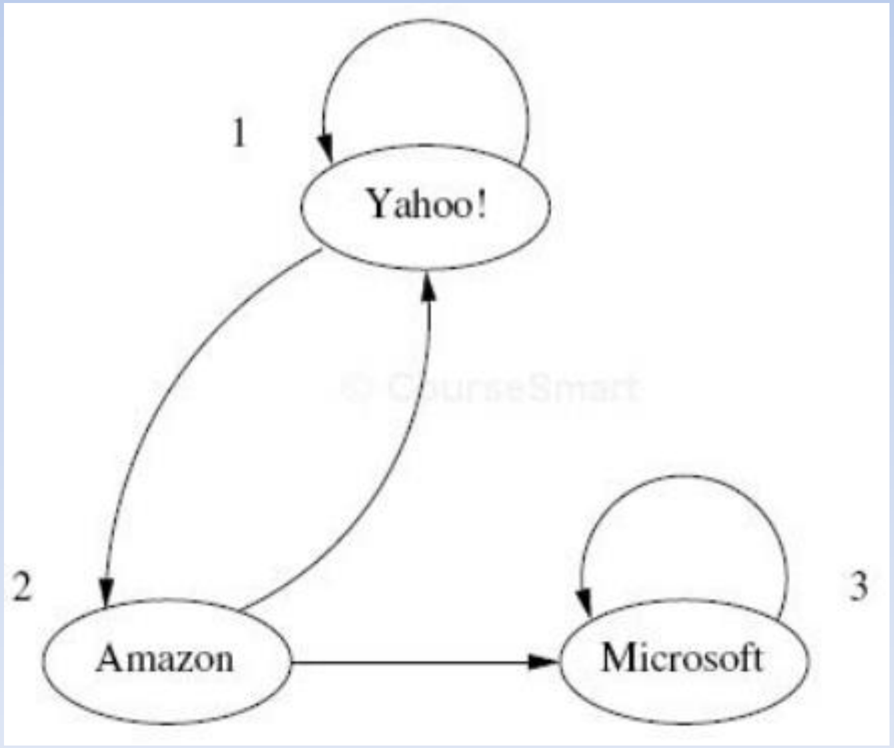


Teleportation! w/ Spider Traps and Dead Ends

- We'll pick some constant $\beta < 1$
- At each step, our walker follows a random out-link (if there is one) w/ probability β.
- With probability $(1 - \beta)$ remove current walker and deposit a new walker at a randomly chosen Web page.

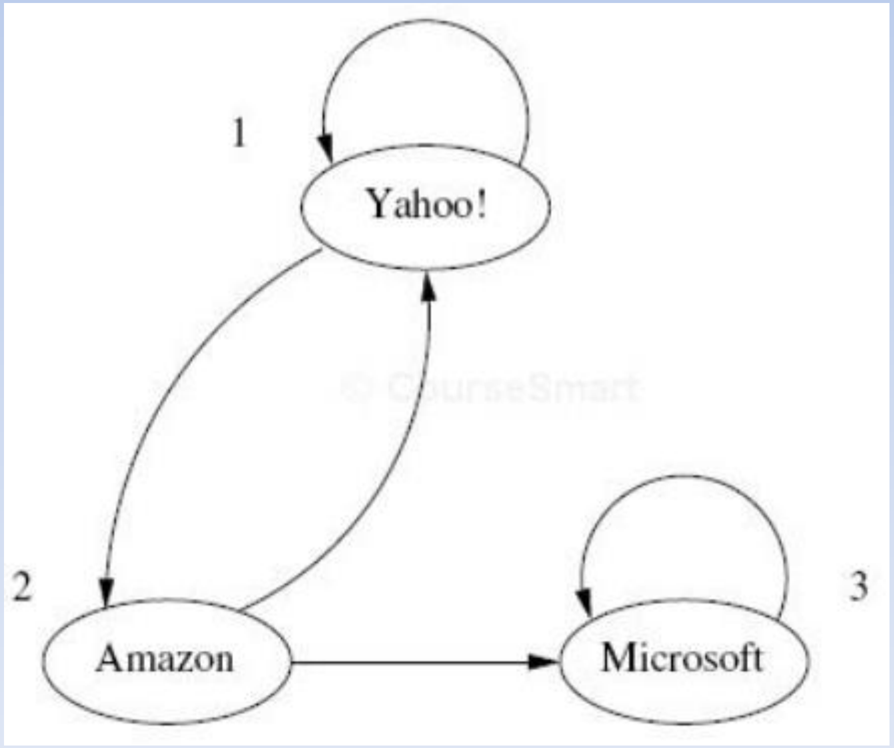
$\beta = 0.8$

$$\begin{bmatrix} y_{i+1} \\ a_{i+1} \\ m_{i+1} \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} y_i \\ a_i \\ m_i \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$



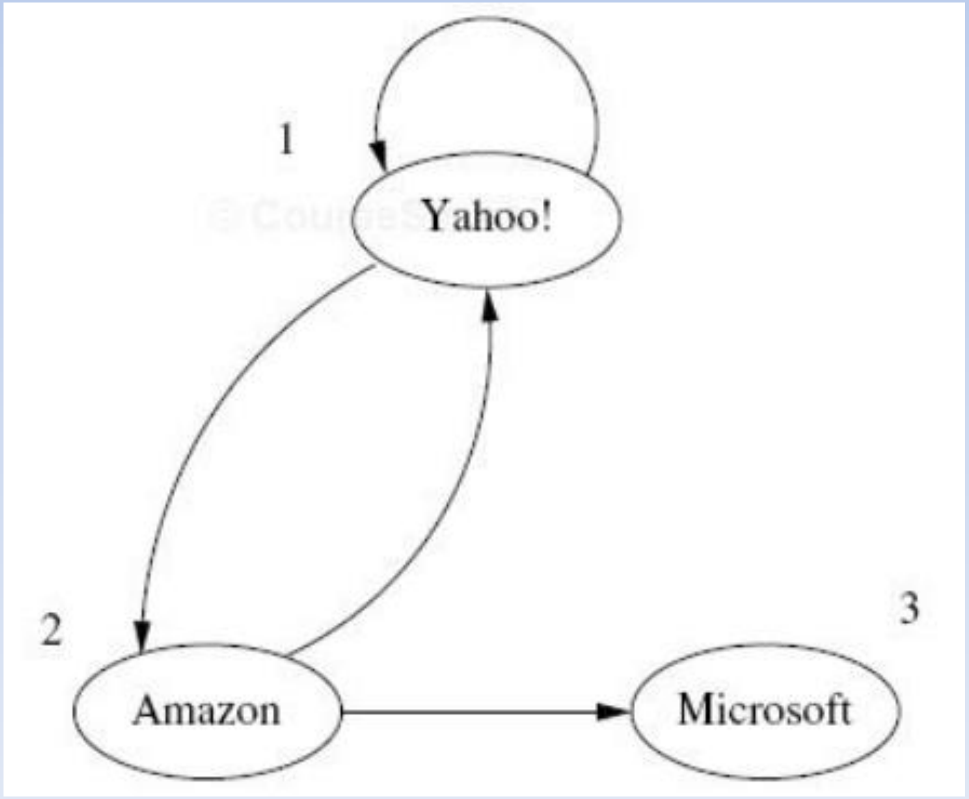
Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_\infty \\ a_\infty \\ m_\infty \end{bmatrix} = \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix}$$



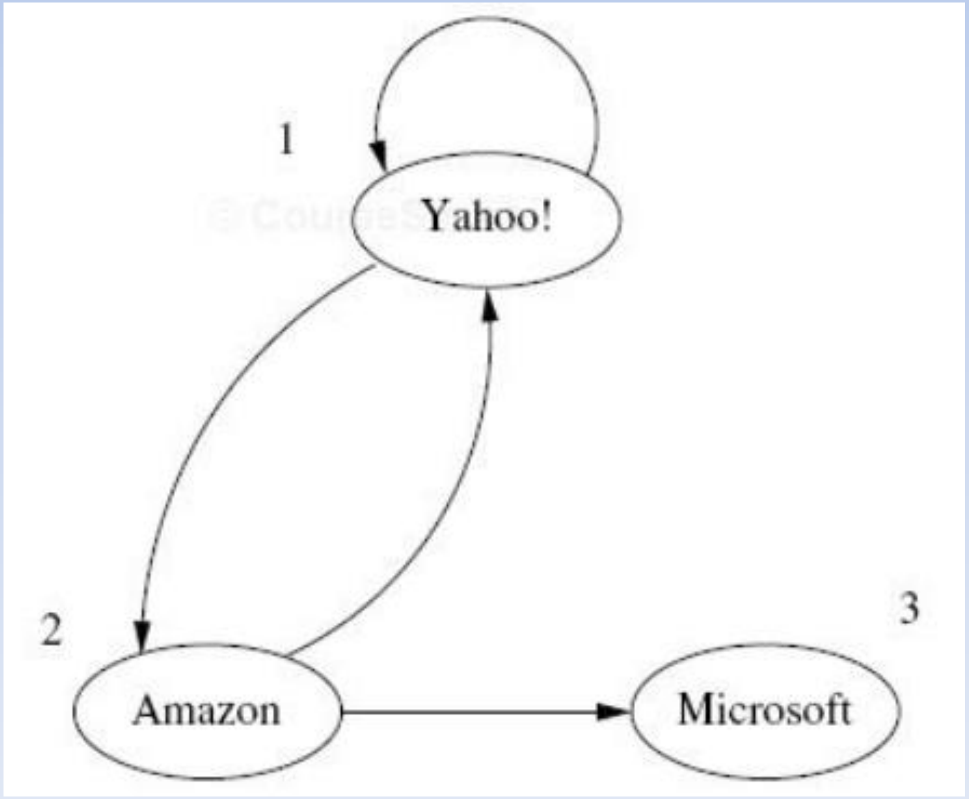
Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_{i+1} \\ a_{i+1} \\ m_{i+1} \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y_i \\ a_i \\ m_i \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$



Microsoft w/ Spider Trap!

$$\begin{bmatrix} y_\infty \\ a_\infty \\ m_\infty \end{bmatrix} = \begin{bmatrix} 35/165 \\ 25/165 \\ 21/165 \end{bmatrix}$$



Compute PageRank

- No Taxation
- 10% Taxation
- 20% Taxation

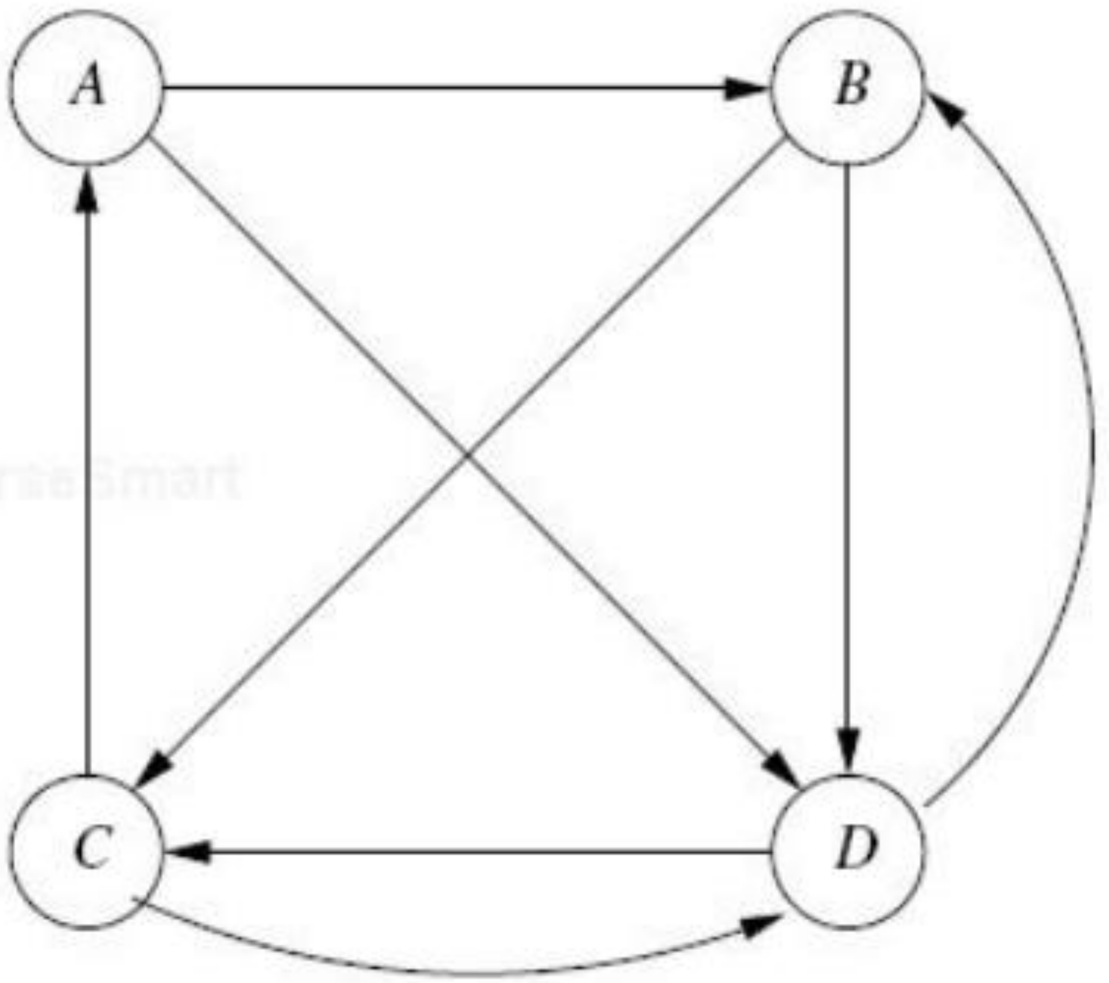


Figure 23.5: A Web graph with no dead-ends or spider traps

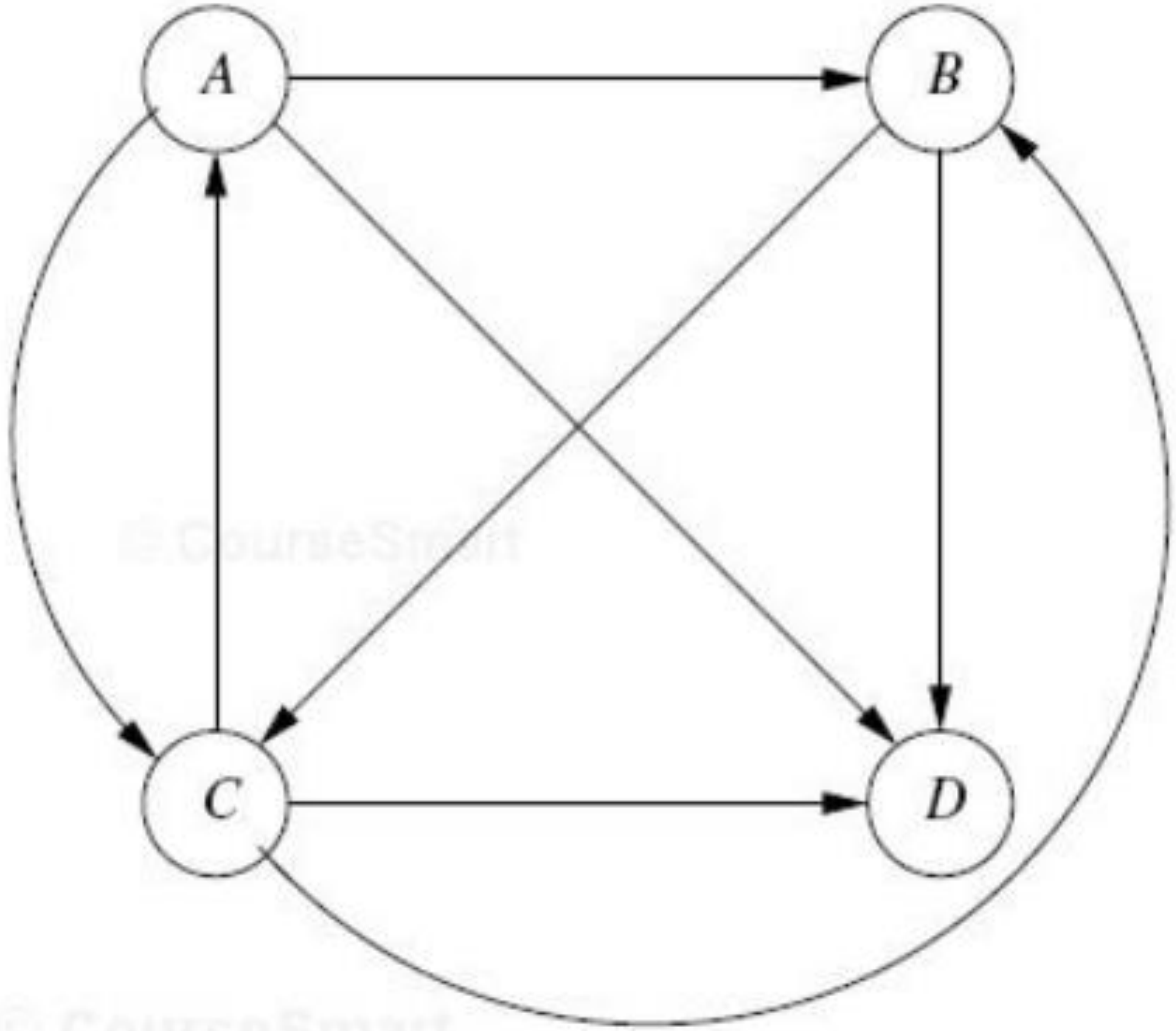


Figure 23.6: A Web graph with a dead end

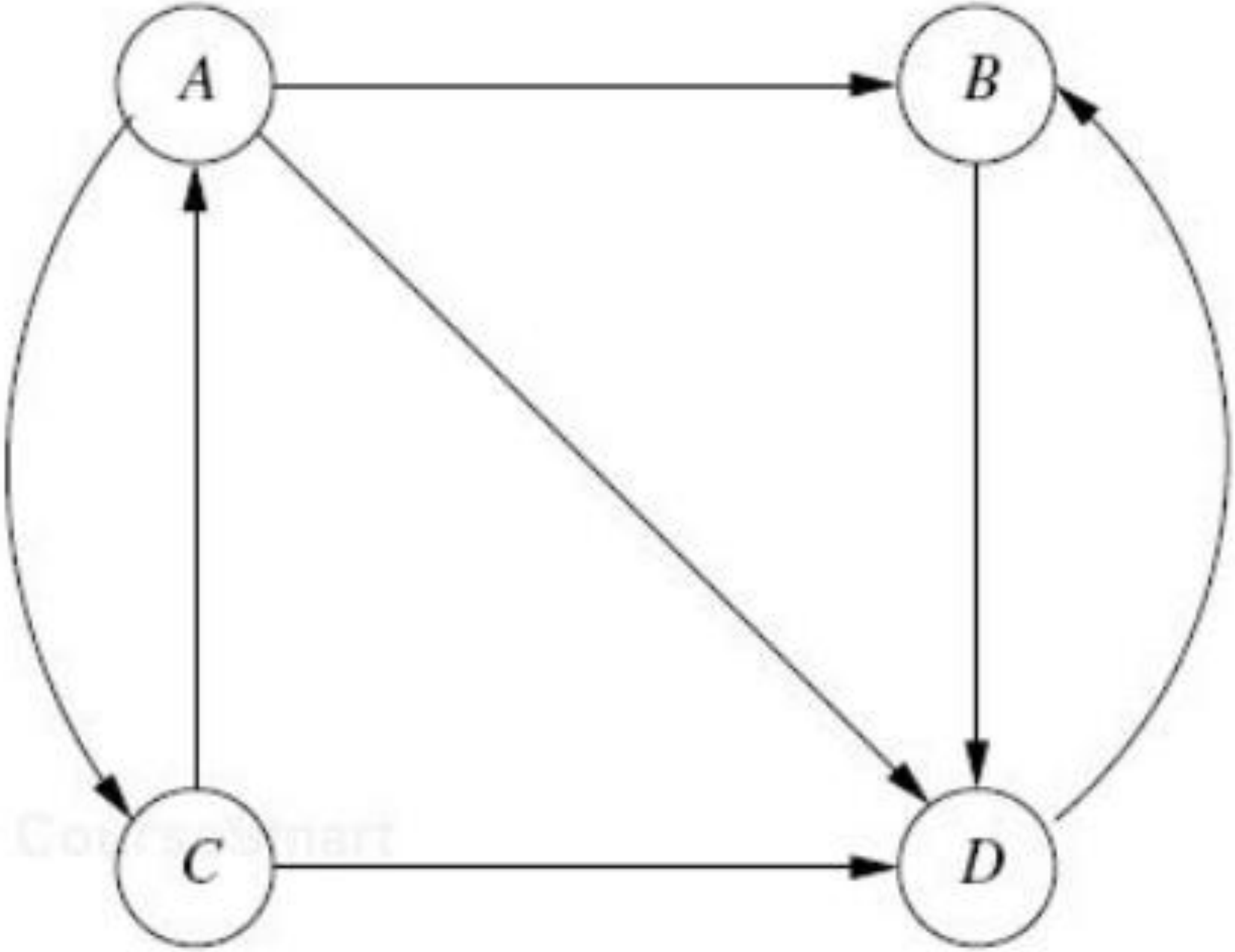


Figure 23.7: A Web graph with a spider trap

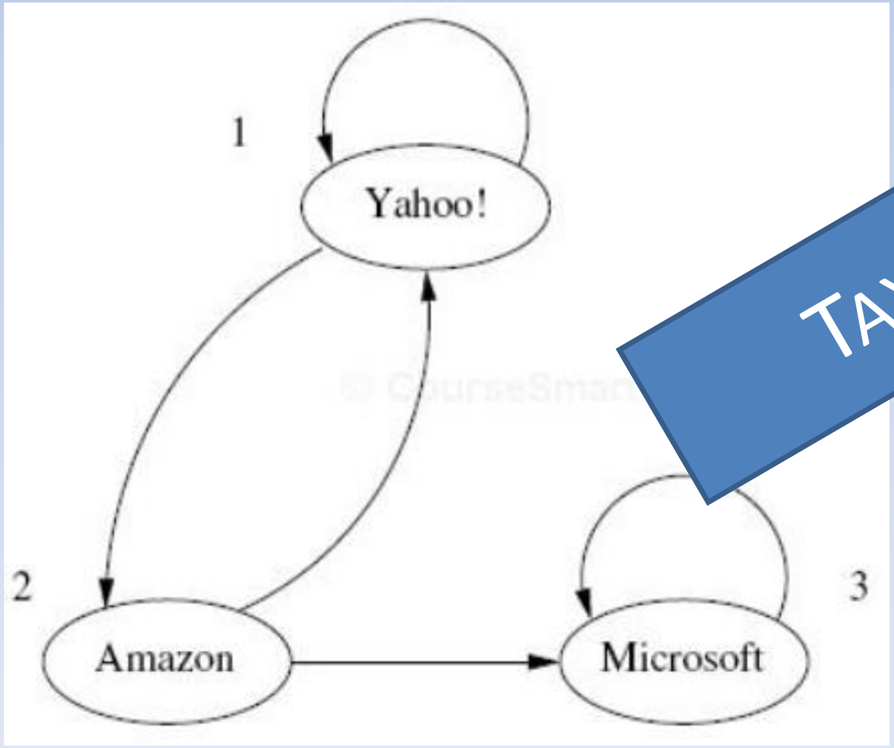
PageRank w/ Topics

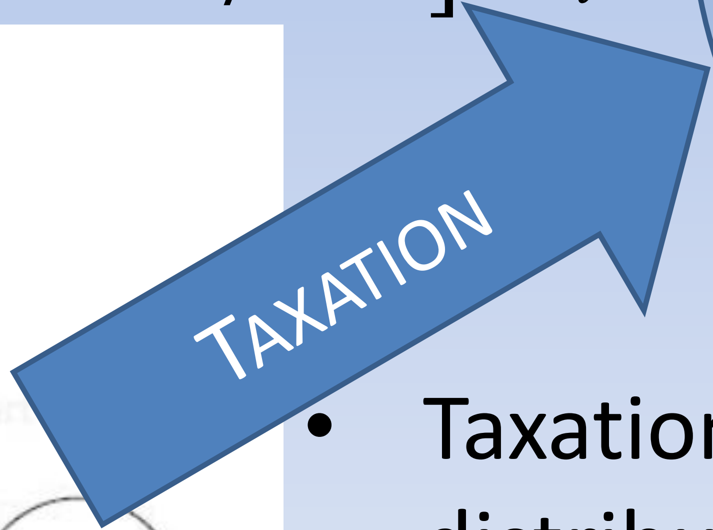
- Our current calculation of PageRank does not incorporate users query!
- Might also need to target SPAMmers!

- We want to extend PageRank to favor types of pages.
- Approach developed also helps combat SPAMmers!

$$\beta = 0.8$$

$$\begin{bmatrix} y_{i+1} \\ a_{i+1} \\ m_{i+1} \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} y_i \\ a_i \\ m_i \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

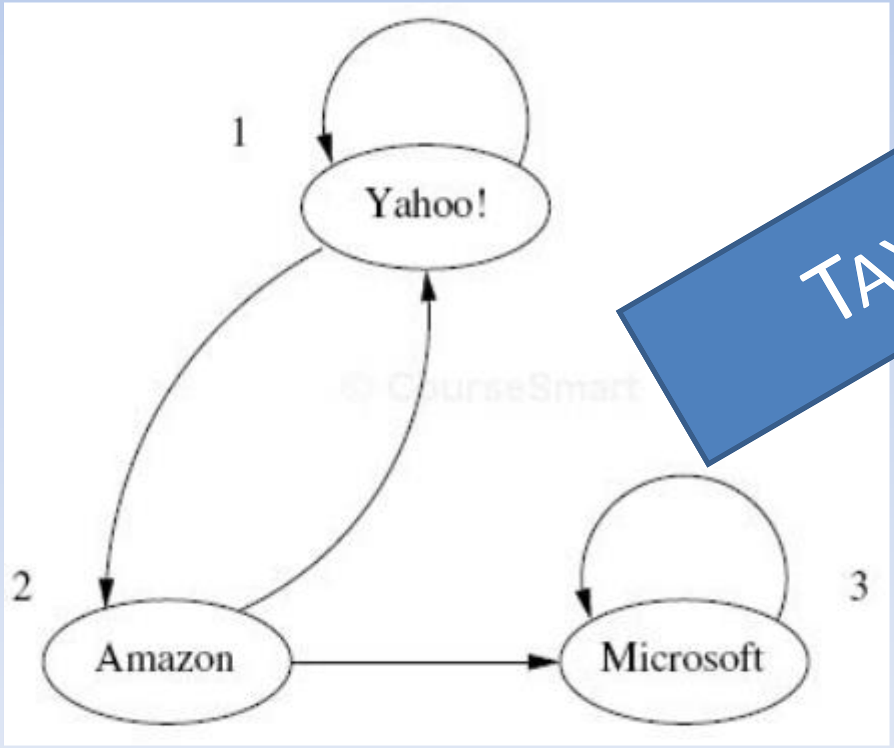




- Taxation distributed equally among all pages!

Teleport Sets!

$$\begin{bmatrix} y_{i+1} \\ a_{i+1} \\ m_{i+1} \end{bmatrix} = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} y_i \\ a_i \\ m_i \end{bmatrix} + 0.2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$



- Taxation distributed equally among SELECTED SET of pages!

Figuring out Teleport Set

- Curated selection of pages:
  - www.dmoz.org
- Learn keywords

Determine Topic from Query

- User Selected Topic
- Predict topic from keywords
- Pages of user interest:
  - Pages bookmarked by user
  - Pages recently searched by user.

Spam Farms

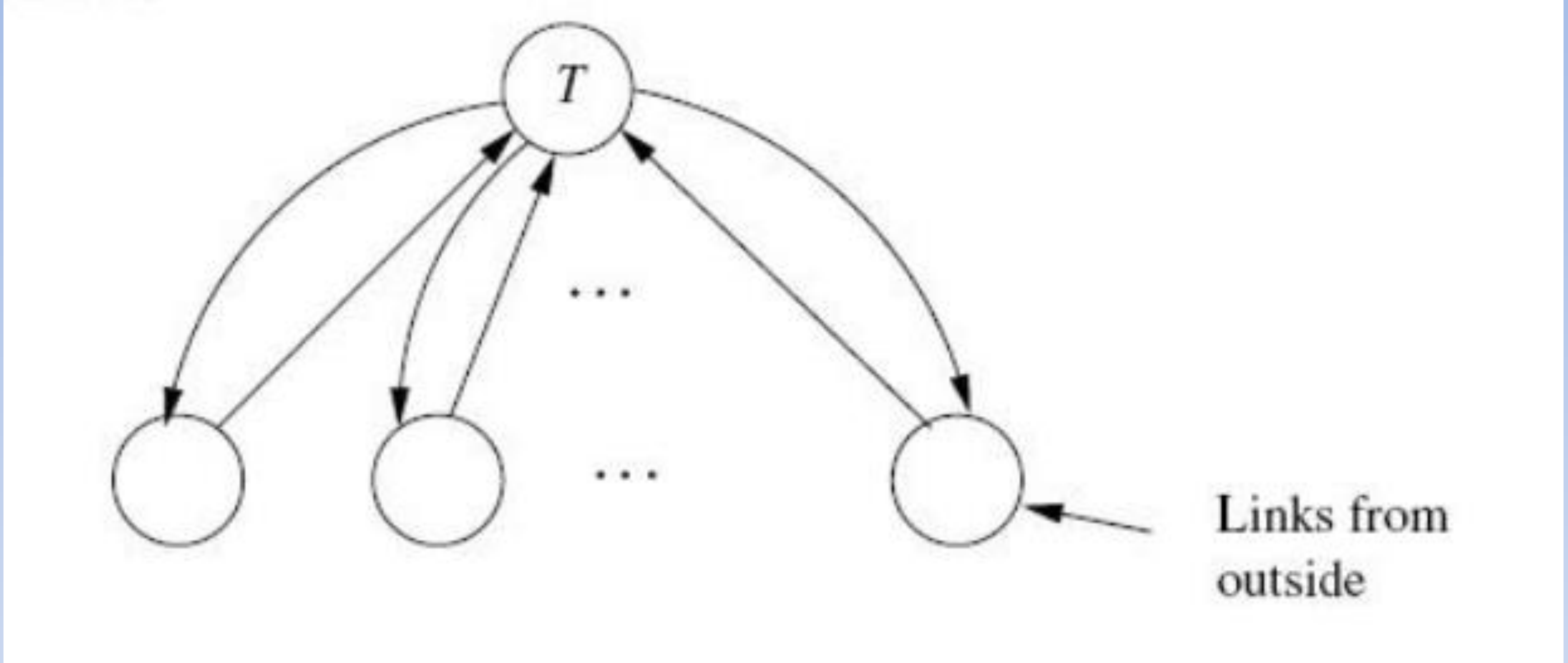


Figure 23.9: A spam farm concentrates PageRank in page T

- Links from outside gained from items like publicly available blogs:
  - "I agree with you!" –Spammer (www.mySite.com)

Trust Rank

- TrustRank is a topic-specific PageRank.
- TrustRank Teleport Set consists of specially constructed set of “trusted” pages:
  - Pages examined by hand.
  - Start with a teleport set likely to contain relatively little spam :
    - UNIVERSITY HOME PAGES!

Spam Mass

- Compute PageRank = p
- Computer TrustRank = t
- Compute difference (p-t) as negative TrustRank = w
- Spam Mass is the ratio of its negative TrustRank to it PageRank: (p-t)/p
  - Fraction of PageRank untrusted (spammy)!

Internet Communication w/ Data Streams

- Internet Communication has enabled data transmissions between machines to stress traditional DB.
- Yahoo! might want to record every “click” made by any user anywhere!
  - Query this Data Also!
- Data of this type is generated constantly, creating a STREAM of Data!

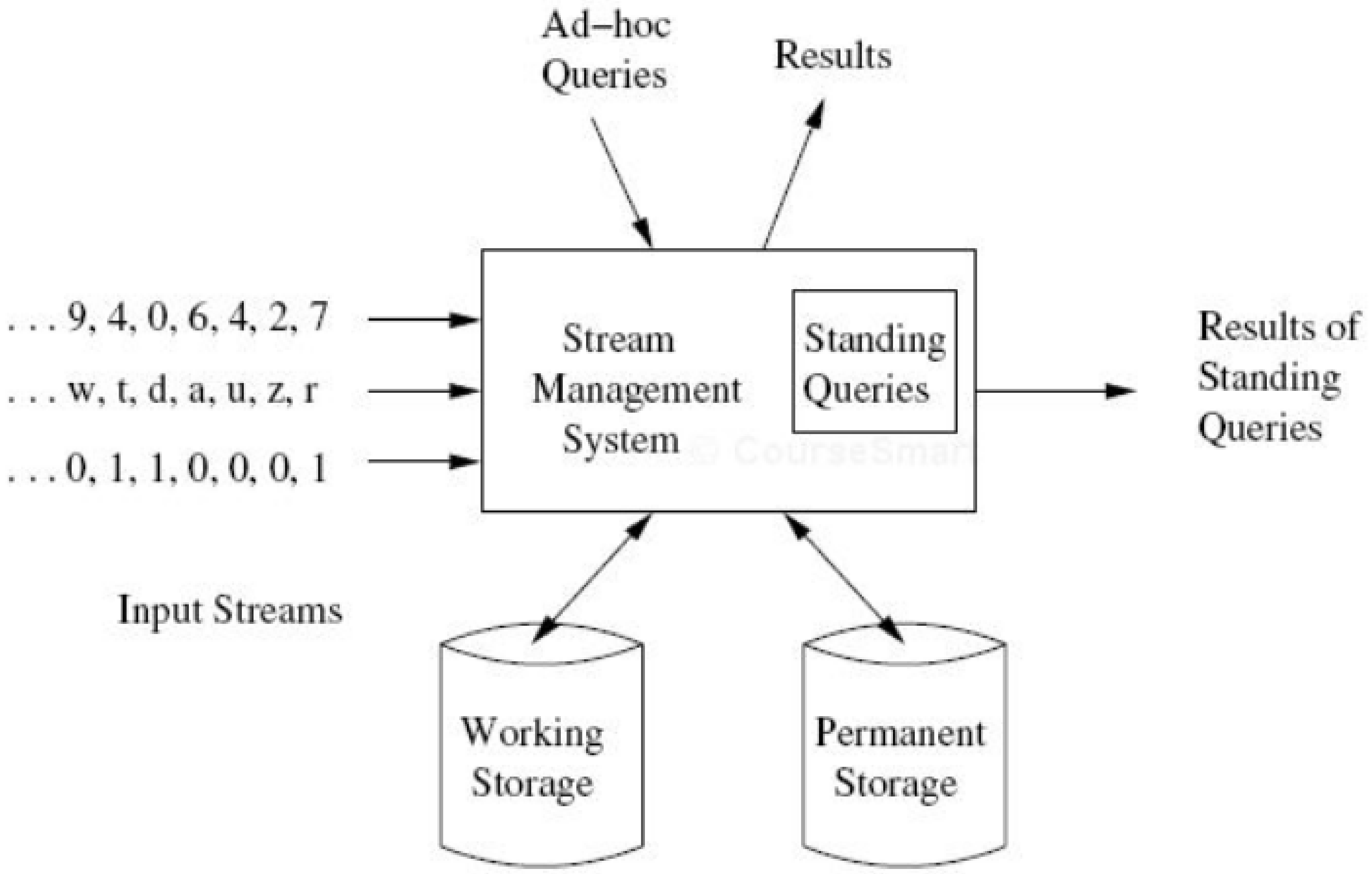
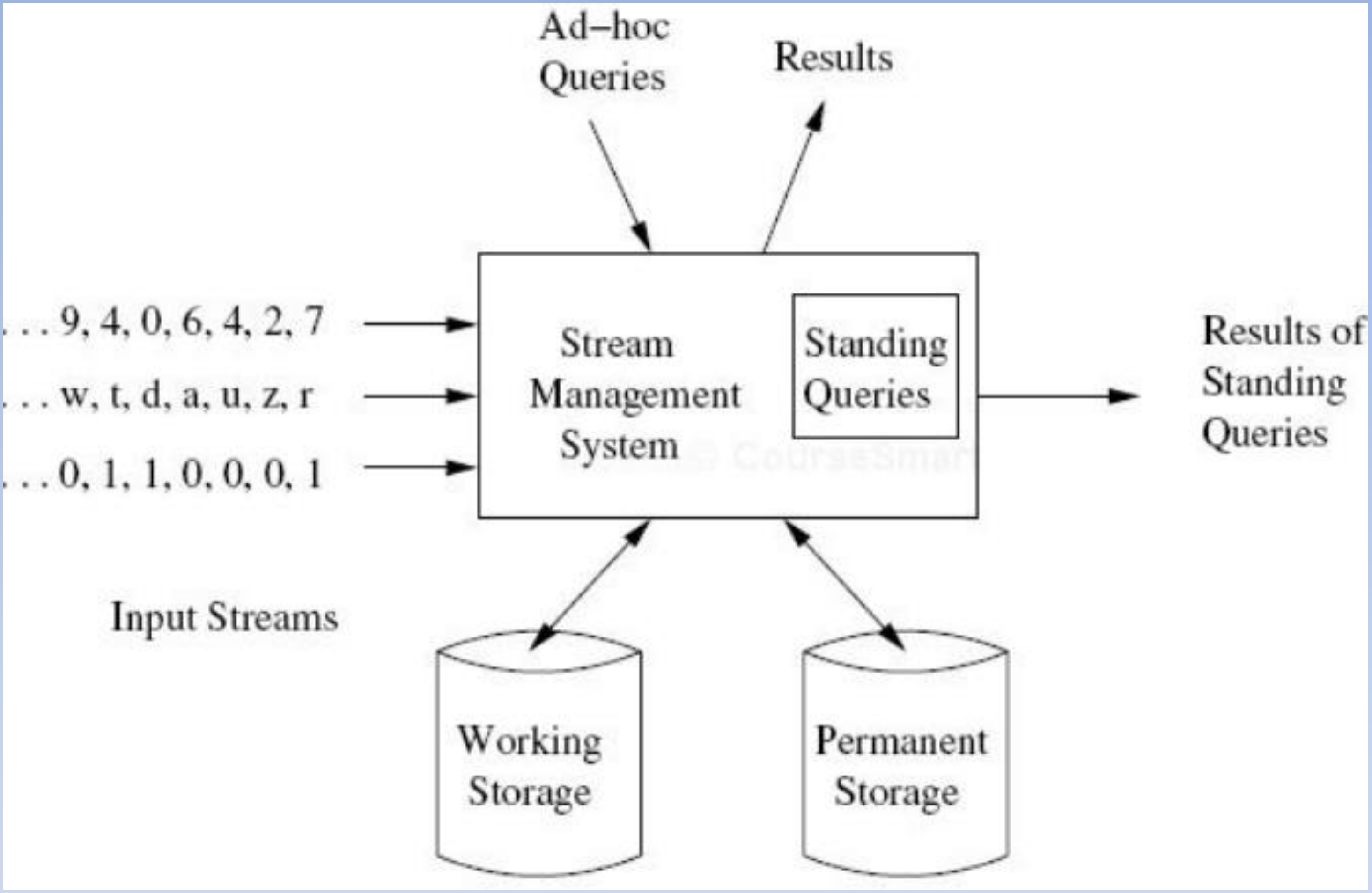


Figure 23.10: A data-stream-management system



- Sliding Window of each input stream stored.
  - Working Storage
- Standing Queries run against Sliding Window
- Ad Hoc Queries can run against Sliding Window

Data Stream Examples

- Click Streams:
  - Clicks by users of a large Web site!
- Packet Streams:
  - IP packets passing through a network switch!
- Sensor Data:
  - Various types of data generated by sensor streams where outputs need to be read and considered collectively.
    - Tsunami Warnings, Security Cameras
- Satellite Data:
  - Often petabytes per day!
- Financial Data:
  - Stock Market prices, Commodities, Various Financial Instruments.

Data-Stream Data Model

- Streams consist of sequence of tuples
  - Tuples have fixed schemas
  - Sequence of tuples may be unbounded!
- Each tuple has an **Arrival Time**
- **Sliding Windows:** Given stream S, S[W]
  - W is rows
  - W is time range

Example 23.10

- Sensor Stream: Sensors(sensID, temp, time)
- Sensors [Rows 1000]
  - Represents the last 1000 tuples
- Sensors [Range 10 Seconds]
  - Represents the last 10 seconds of data

Sensor Stream Windows w/ Queries

- Want to know, for each of our sensor locations, what’s the highest temperature in the last hour??

```
SELECT sensID, MAX(temp)
FROM Sensors [Range 1 Hour]
GROUP BY sensID;
```

Example 23.13

- Streams can be queried using joins and subqueries
- Suppose we want the maximum temperature in the last hour like before, but also the latest time for temperature?

Example 23.13

- Streams can be queried using joins and subqueries
- Suppose we want the maximum temperature in the last hour like before, but also the latest time for temperature?

```
SELECT s1.sensID, s1.temp, s1.time
FROM Sensors [Range 1 Hour] s1
WHERE NOT EXISTS (
  SELECT * FROM Sensor [Range 1 Hour] s2
  WHERE s2.sensID = s1.sensID AND (
    s2.temp > s1.temp OR
      (s2.temp = s1.temp AND s2.time > s1.time)
  )
);
```

- NO Tuple in window from same sensor with higher temperature.

Standing Queries

- When running standing queries, windows will change constantly.
- Maintaining relations requires insertions and deletions of relations never looked at.
- An alternative, is to convert query back into stream!

Queries to Streams

- Given a query with resulting relation R:

```
SELECT s1.sensID, s1.temp, s1.time
FROM Sensors [Range 1 Hour] s1
WHERE NOT EXISTS (
  SELECT * FROM Sensor [Range 1 Hour] s2
  WHERE s2.sensID = s1.sensID AND (
    s2.temp > s1.temp OR
      (s2.temp = s1.temp AND s2.time > s1.time)
  )
);
```

- Convert R back into a stream: Istream(R)
  - Istream(R) has a tuple for every event where a new tuple is added to R.

Queries to Streams

- Convert R back into a stream: Istream(R)
  - Istream(R) has a tuple for every event where a new tuple is added to R.
- Two cases add tuples to our query result R:
  - A sensor reads a temperature at least as high as its seen in the last hour.
  - The current max temperature is over an hour old!

Dstream(R)

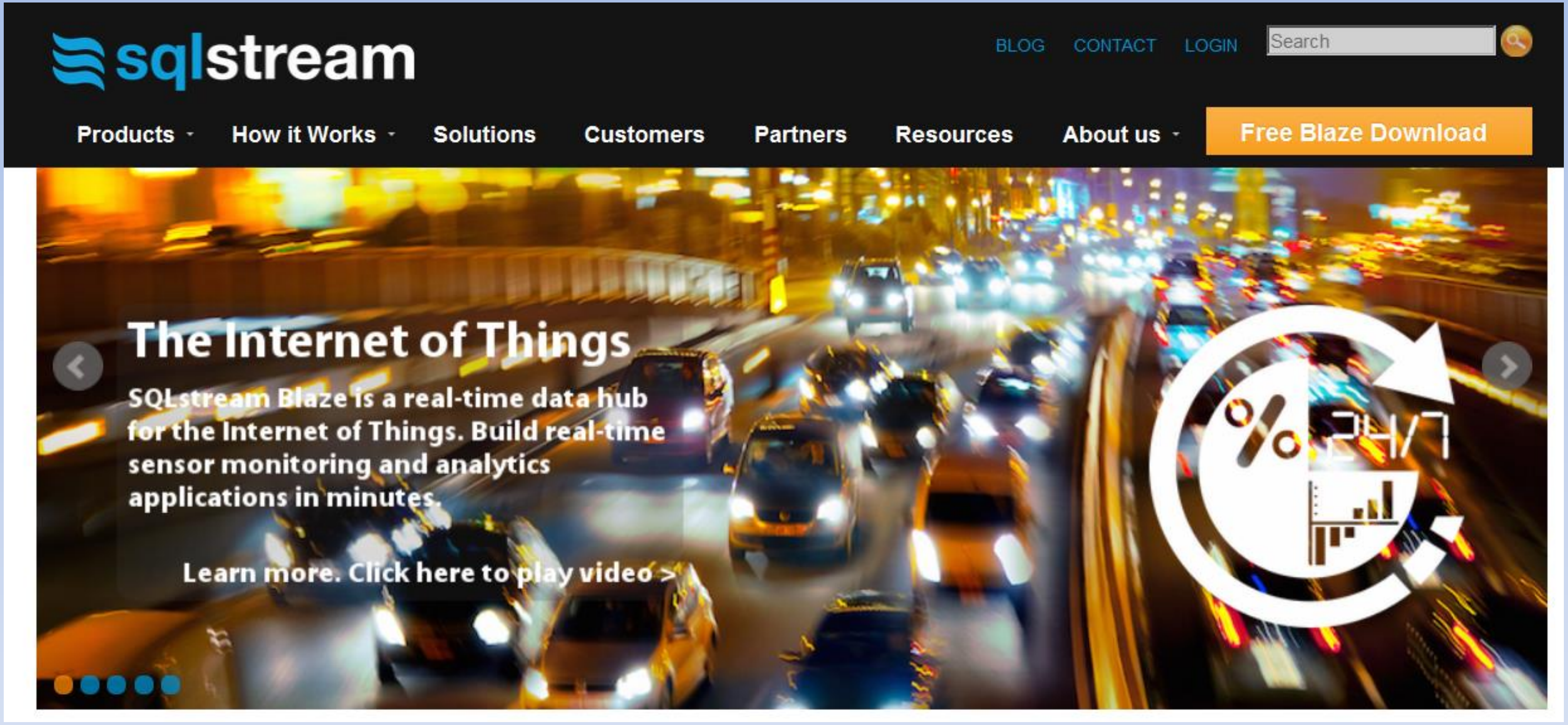
- Dstream(R) is the stream of tuples deleted from R.
- Each tuple deleted from R is added to Dstream(R) when it is deleted:
  - Max value is over an hour old
  - New max value arrived, requiring old max deleted.

Quering Streams

- Find the Max temperature from sensor 100 from the past hour:

```
( SELECT * FROM I [ Range 1 Hour ]
WHERE sensID = 100
  AND time > = [ Now - 1 Hour ] )
EXCEPT
( SELECT * FROM D [ Range 1 Hour ]
WHERE sensID = 100 ) ;
```

SQLStream

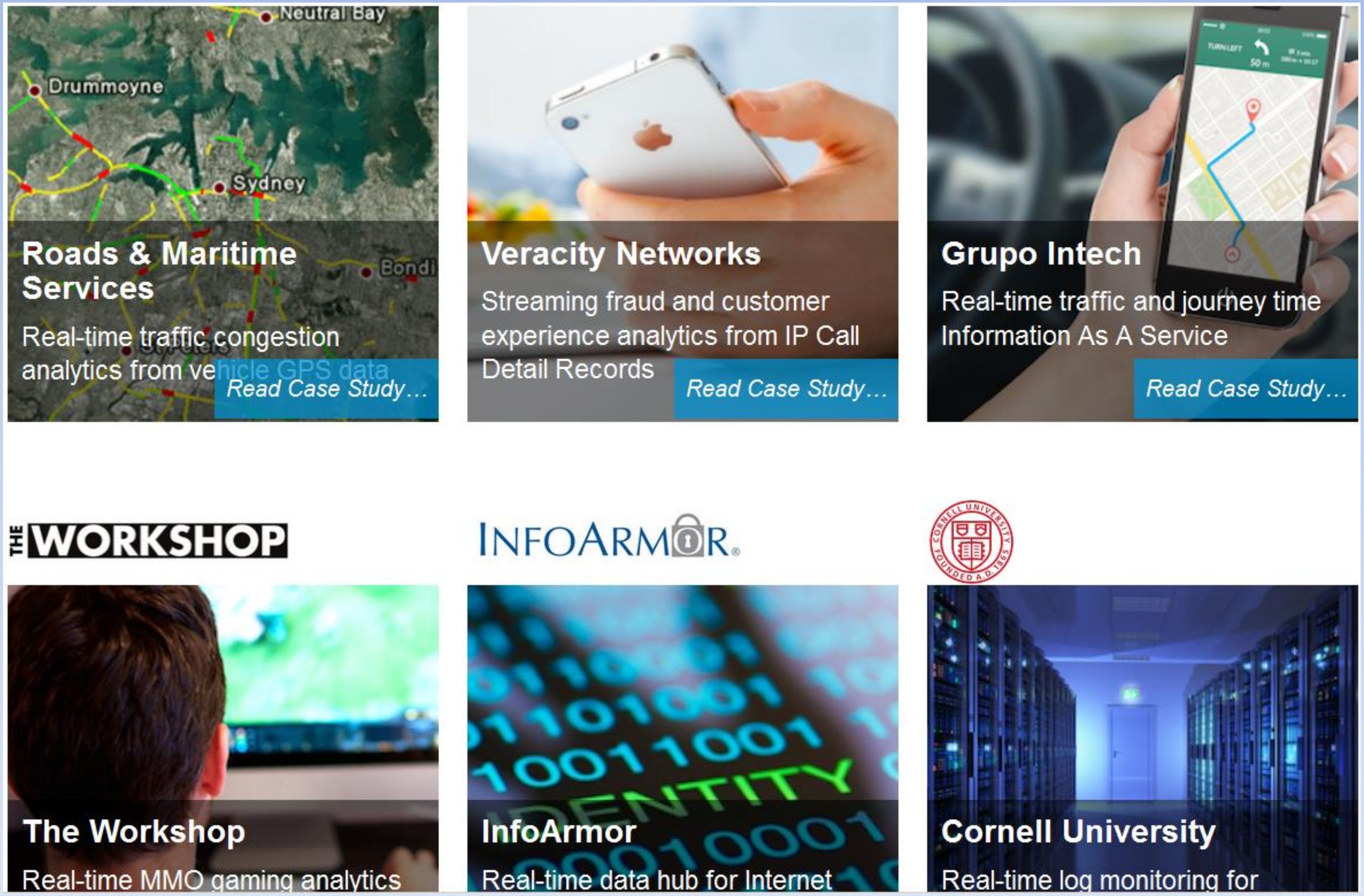


The Internet of Things

SQLstream Blaze is a real-time data hub for the Internet of Things. Build real-time sensor monitoring and analytics applications in minutes.

Learn more. Click here to play video >

SQLStream



Roads & Maritime Services

Real-time traffic congestion analytics from vehicle GPS data

Read Case Study…

Veracity Networks

Streaming fraud and customer experience analytics from IP Call Detail Records

Read Case Study…

Grupo Intech

Real-time traffic and journey time Information As A Service

Read Case Study…

The Workshop

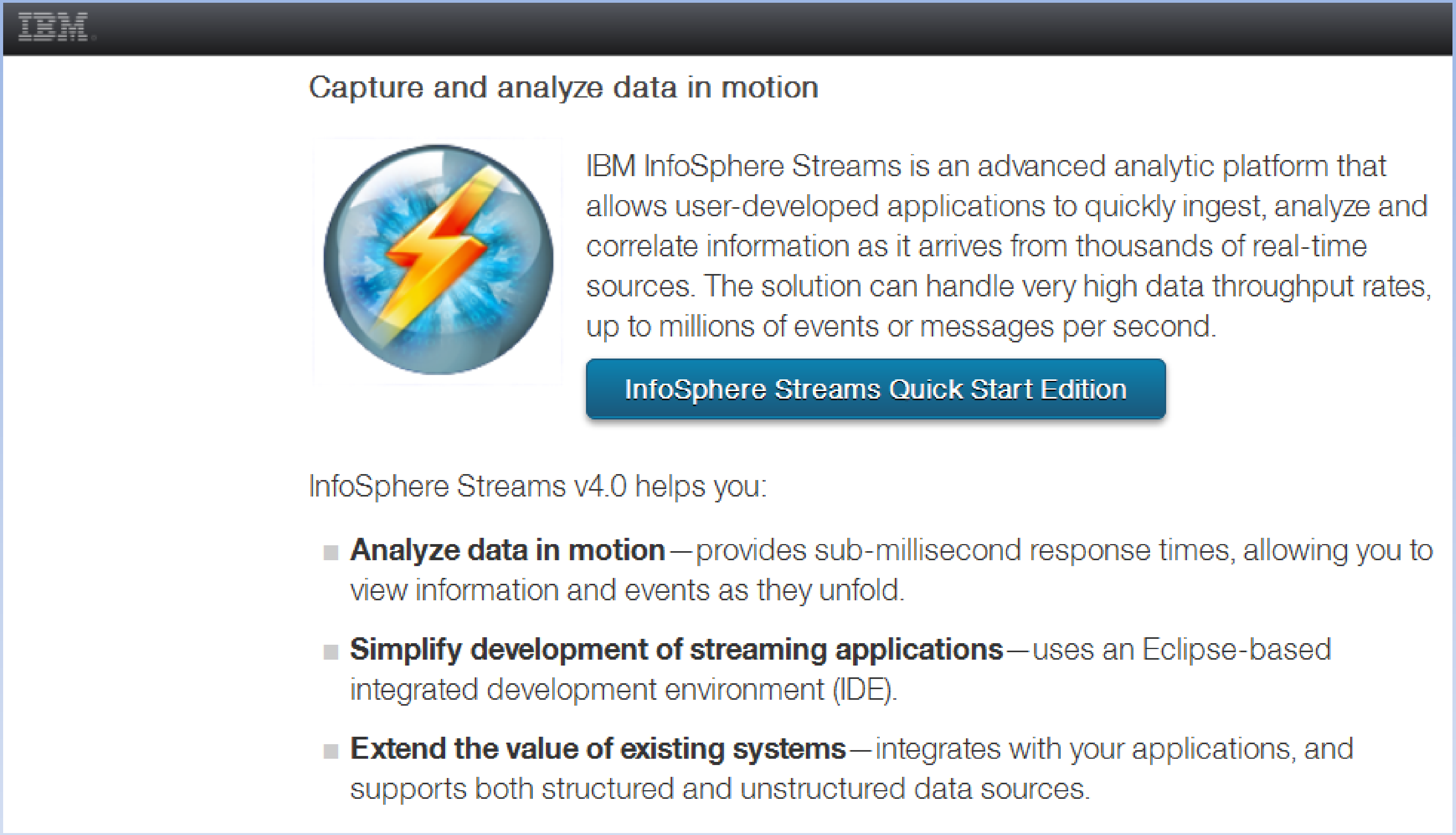
Real-time MMO gaming analytics

InfoArmor

Real-time data hub for Internet

Cornell University

Real-time log monitoring for



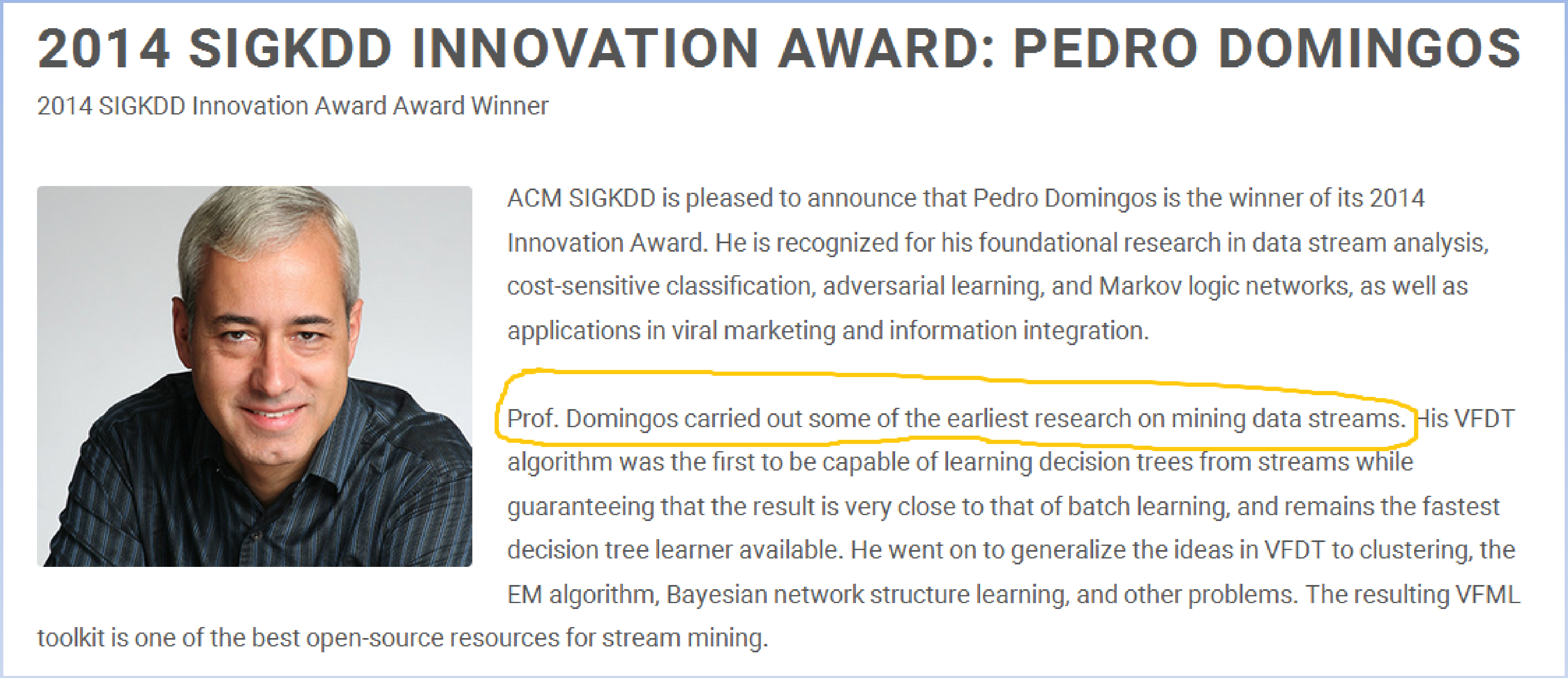
IBM

Capture and analyze data in motion

IBM InfoSphere Streams is an advanced analytic platform that allows user-developed applications to quickly ingest, analyze and correlate information as it arrives from thousands of real-time sources. The solution can handle very high data throughput rates, up to millions of events or messages per second.

InfoSphere Streams v4.0 helps you:

- **Analyze data in motion**—provides sub-millisecond response times, allowing you to view information and events as they unfold.
- **Simplify development of streaming applications**—uses an Eclipse-based integrated development environment (IDE).
- **Extend the value of existing systems**—integrates with your applications, and supports both structured and unstructured data sources.



2014 SIGKDD INNOVATION AWARD: PEDRO DOMINGOS

2014 SIGKDD Innovation Award Award Winner

ACM SIGKDD is pleased to announce that Pedro Domingos is the winner of its 2014 Innovation Award. He is recognized for his foundational research in data stream analysis, cost-sensitive classification, adversarial learning, and Markov logic networks, as well as applications in viral marketing and information integration.

Prof. Domingos carried out some of the earliest research on mining data streams. His VFDT algorithm was the first to be capable of learning decision trees from streams while guaranteeing that the result is very close to that of batch learning, and remains the fastest decision tree learner available. He went on to generalize the ideas in VFDT to clustering, the EM algorithm, Bayesian network structure learning, and other problems. The resulting VFML toolkit is one of the best open-source resources for stream mining.