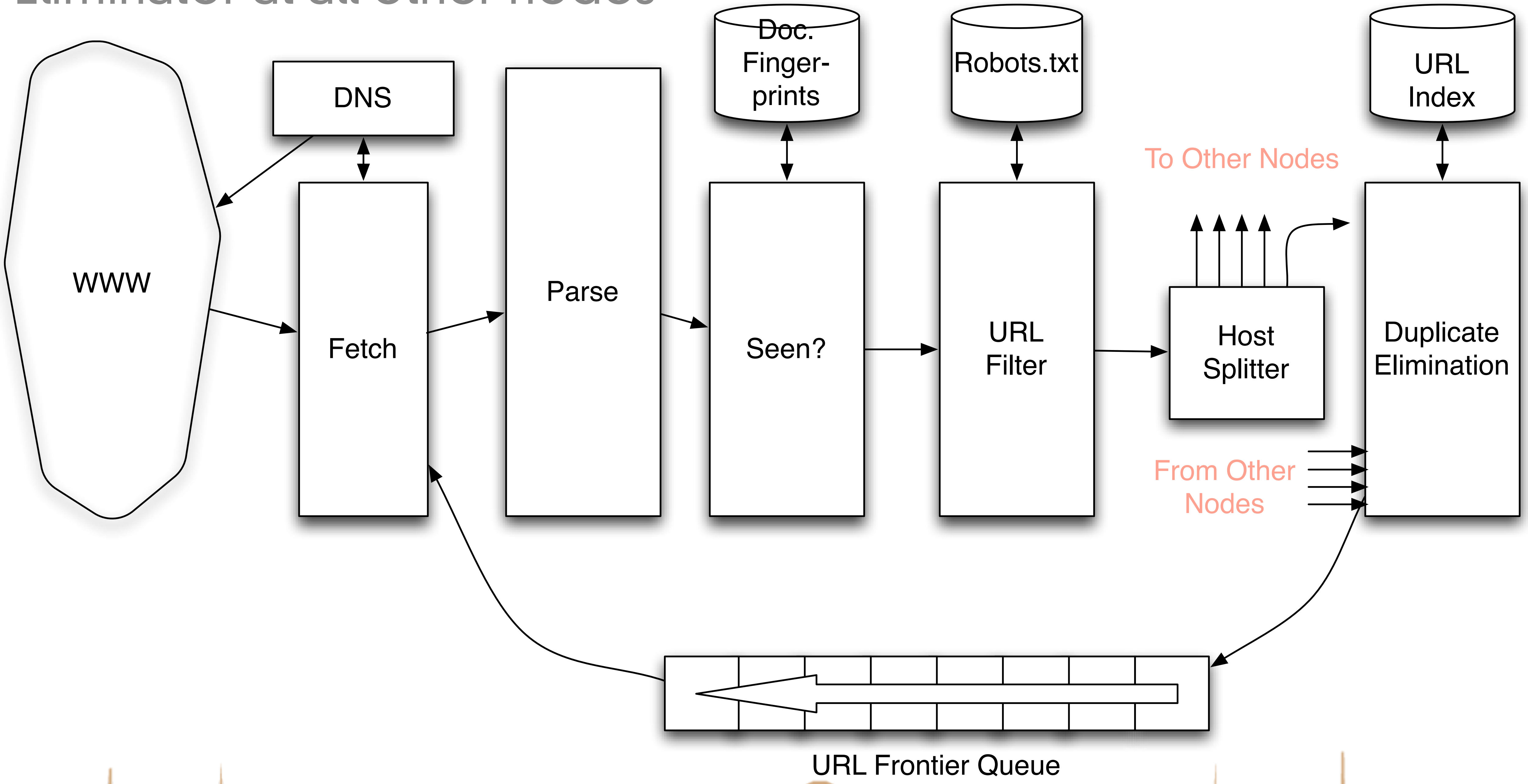
Web Crawling

Introduction to Information Retrieval
INF 141
Donald J. Patterson

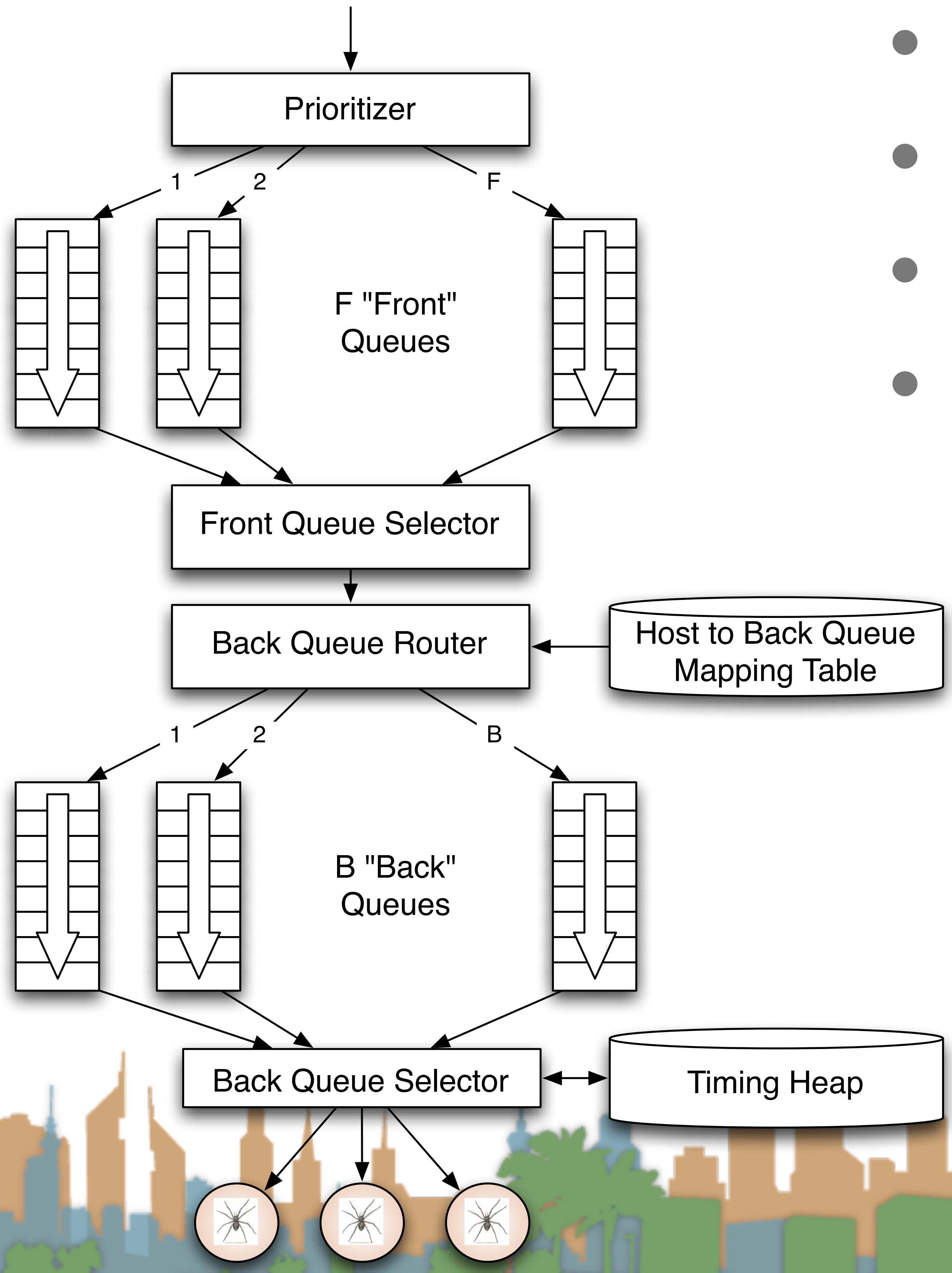
Content adapted from Hinrich Schütze
http://www.informationretrieval.org

Robust Crawling

The output of the URL Filter at each node is sent to the Duplicate Eliminator at all other nodes



URL Frontier Implementation - Mercator



- URLs flow from top to bottom
- Front queues manage priority
- Back queue manage politeness
- Each queue is FIFO

http://research.microsoft.com/~najork/mercator.pdf

Web Crawling Outline

Overview

- Introduction
- URL Frontier
- Robust Crawling
  - DNS
  - Various parts of architecture
  - URL Frontier
- Index
  - Distributed Indices
  - Connectivity Servers

Indices

The index

Indices

The index

- Why does the crawling architecture exists?

The index

- Why does the crawling architecture exists?
    - To gather information from web pages (aka documents).

The index

- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?

The index

- Why does the crawling architecture exists?
    - To gather information from web pages (aka documents).
- What information are we collecting?
    - Keywords

The index

- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?
  - Keywords
    - Mapping documents to a “bags of words” (aka vector space model)

The index

- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?
  - Keywords
    - Mapping documents to a “bags of words” (aka vector space model)
  - Links

The index

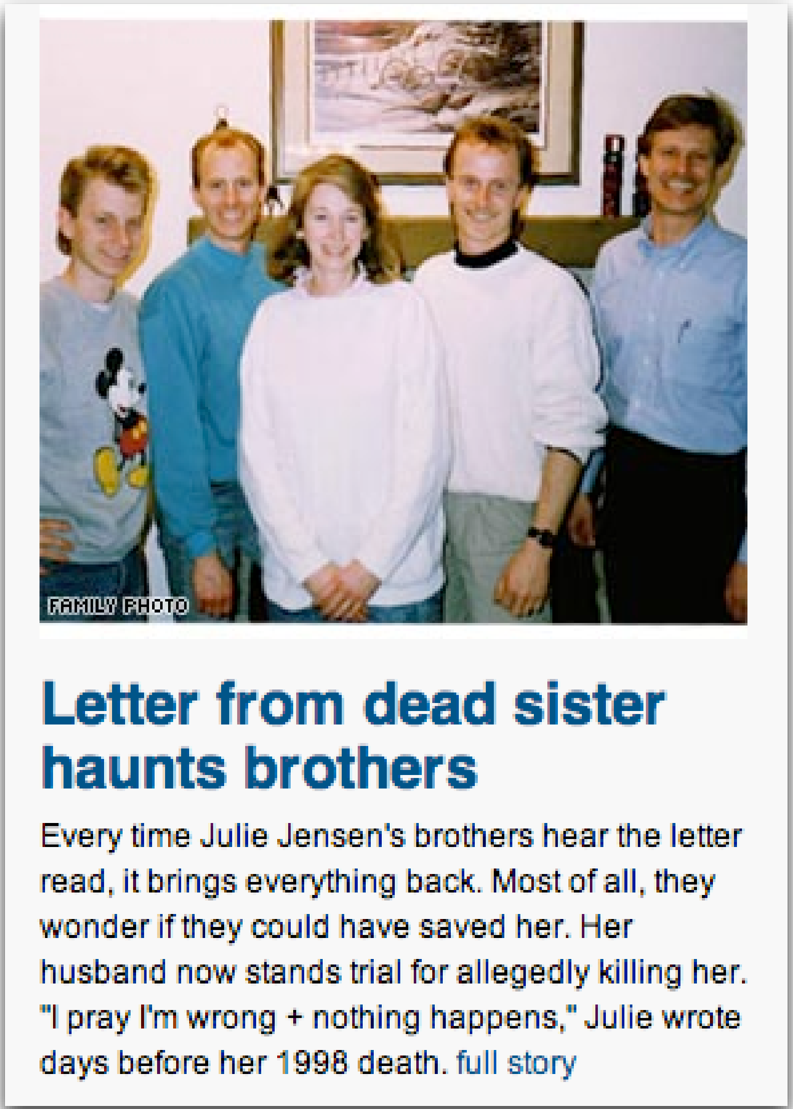
- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?
  - Keywords
    - Mapping documents to a “bags of words” (aka vector space model)
  - Links
    - Where does a document link to?

The index

- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?
  - Keywords
    - Mapping documents to a “bags of words” (aka vector space model)
  - Links
    - Where does a document link to?
    - Who links to a document?

Indices

The index has a list of vector space models


FAMILY PHOTO

Letter from dead sister haunts brothers

Every time Julie Jensen's brothers hear the letter read, it brings everything back. Most of all, they wonder if they could have saved her. Her husband now stands trial for allegedly killing her. "I pray I'm wrong + nothing happens," Julie wrote days before her 1998 death. full story

1	1998		
1	Every	1	have
1	Her	1	hear
1	I	3	her
1	I'm	1	husband
1	Jensen's	1	if
2	Julie	1	it
1	Letter	1	killing
1	Most	1	letter
1	all	1	nothing
1	allegedly	1	now
1	back	1	of
1	before	1	pray
1	brings	1	read,
2	brothers	1	saved
1	could	1	sister
1	days	1	stands
1	dead	1	story
1	death	1	the
1	everything	2	they
1	for	1	time
1	from	1	trial
1	full	1	wonder
1	happens	1	wrong
1	haunts	1	wrote

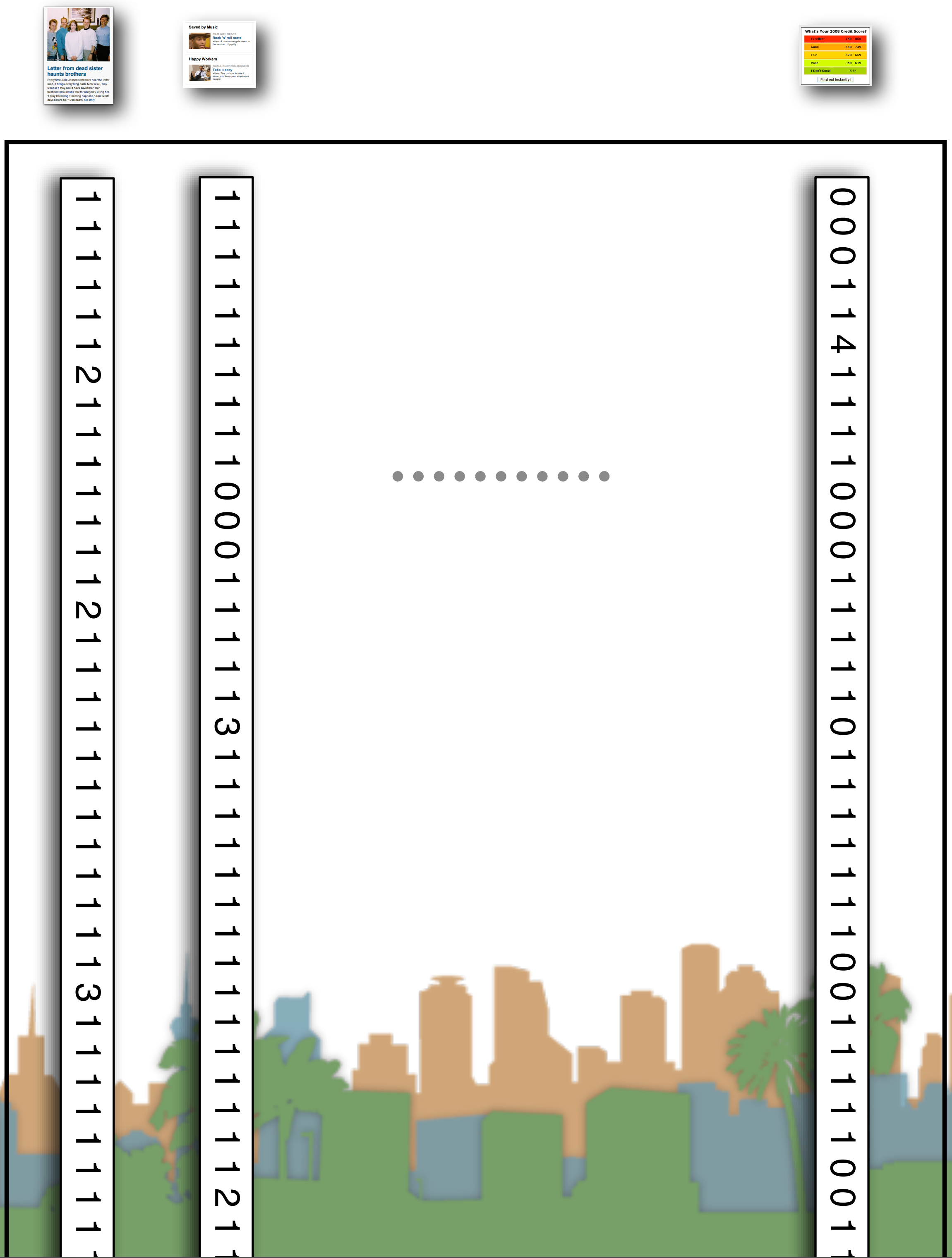
1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1

Indices

Our index is a 2-D array or Matrix

A Column for Each Web Page (or "Document")

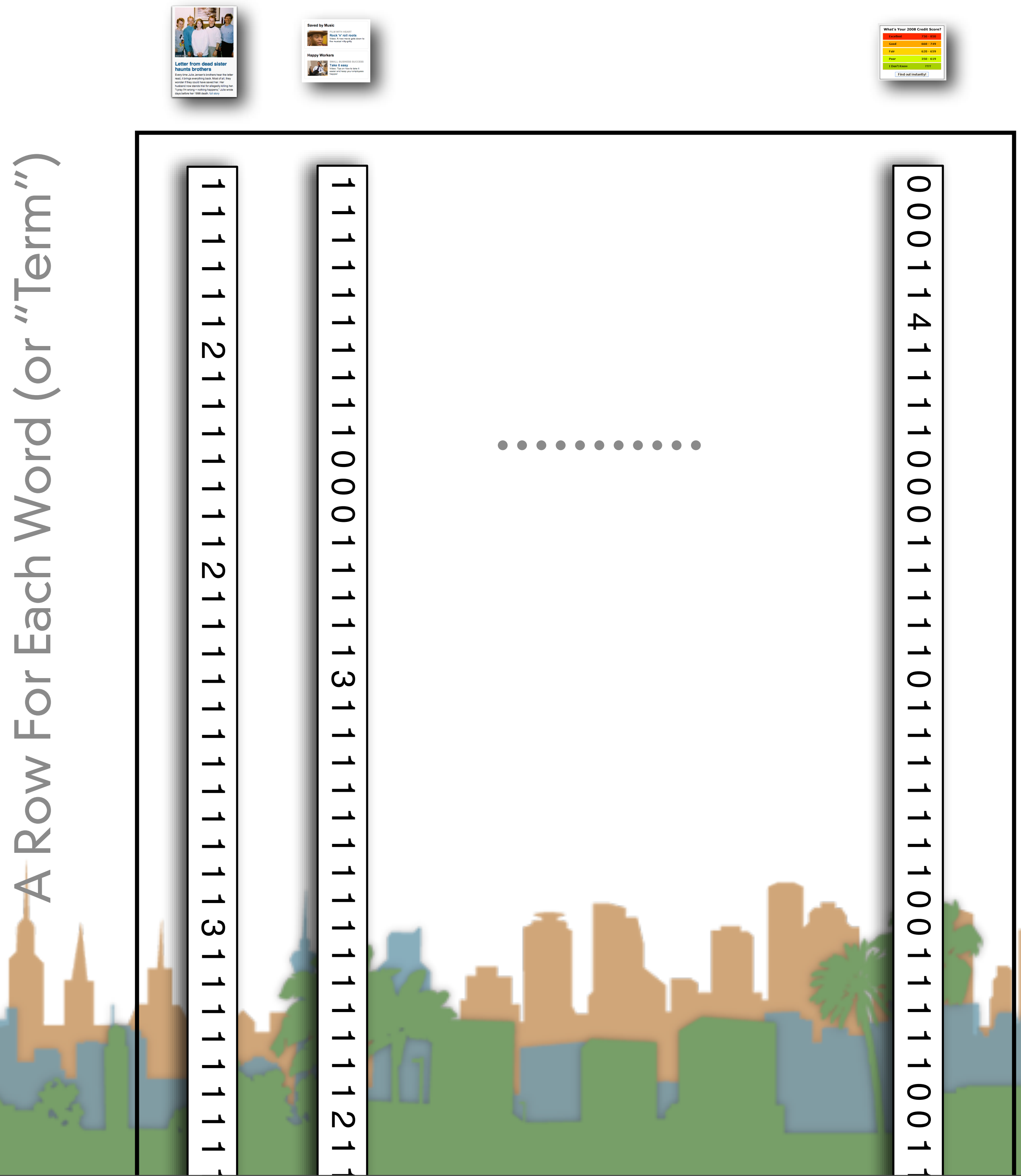
A Row For Each Word (or "Term")



Indices

"Term-Document Matrix" Capture Keywords

A Column for Each Web Page (or "Document")



Indices

The Term-Document Matrix

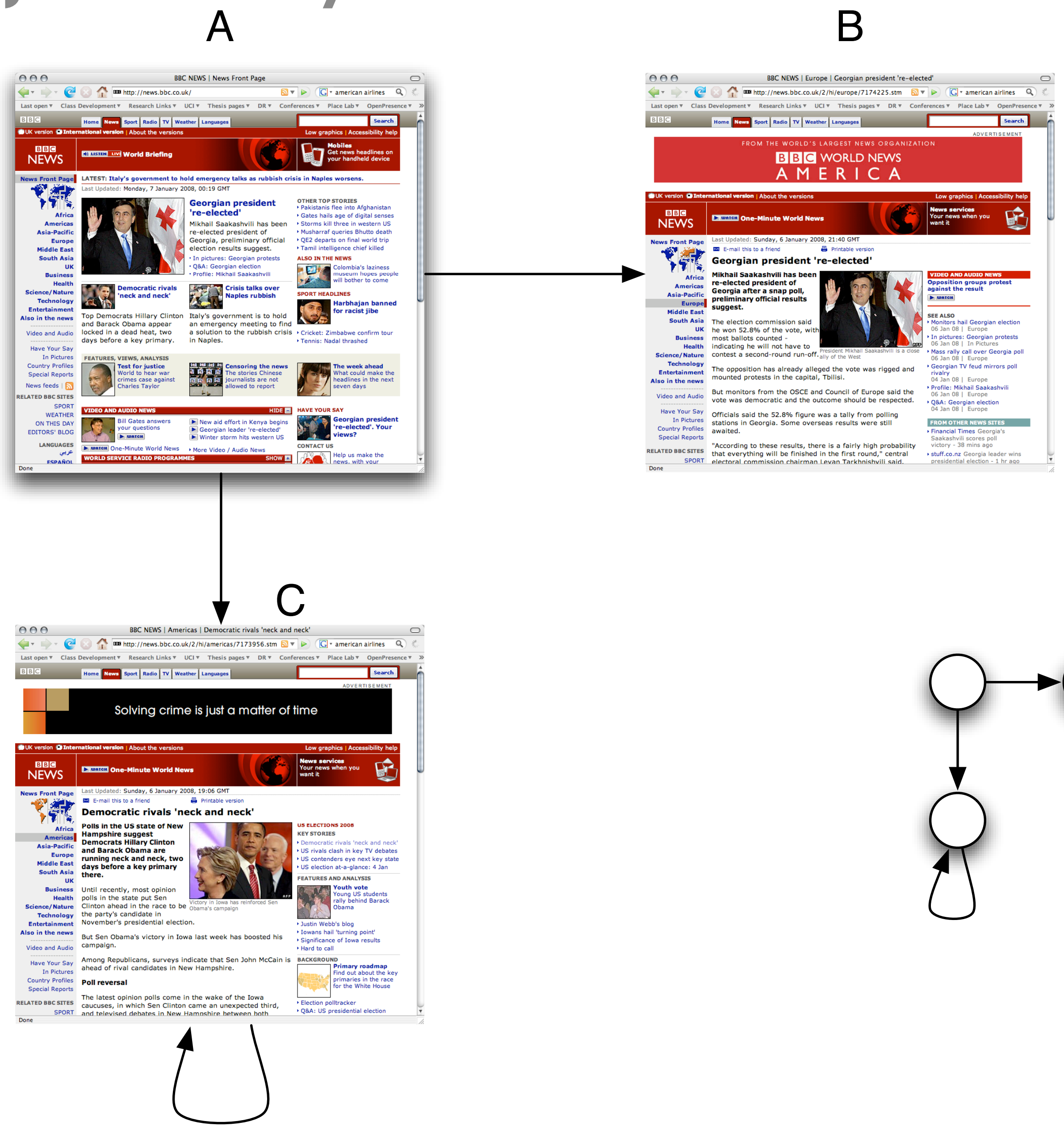
- Is really big at a web scale
- It must be split up into pieces
- An effect way to split it up is to split up the same way as the crawling
  - Equivalent to taking vertical slices of the T-D Matrix
  - Helps with cache hits during crawl
- Later we will see that it needs to be rejoined for calculations across all documents

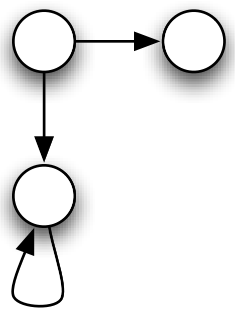
Connectivity Server

- Other part of reason for crawling
- Supports fast queries on the web graph
    - Which URLS point to a given URL (in-links)?
    - Which URLS does a given URL point to (out-links)?
- Applications
    - Crawl control
    - Web Graph Analysis
    - Link Analysis (aka PageRank)
        - Provides input to "quality" for URL frontier

Indices - Connectivity Server

Adjacency Matrix - Conceptual Idea





$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 1 & 1 \\ B & 0 & 0 & 0 \\ C & 0 & 0 & 1 \end{array}$$

Connectivity Server in practice

- What about Adjacency Lists instead?
    - Set of neighbors of a node
    - Assume each URL represented by an integer
    - i.e. 4 billion web pages need 32 bits per URL
    - Naive implementation requires 64 bits per link
        - 32 bits to 32 bits

Connectivity Server in practice

- What about Adjacency Lists instead?
  - Non-naive approach is to exploit compression
    - Similarity between lists of links
    - Locality (many links go to “nearby” links)
    - Use gap encodings in sorted lists
    - Leverage the distribution of gap values

Connectivity Server in practice

- Current state of the art is Boldi and Vigna
  - http://www2004.org/proceedings/docs/1p595.pdf
  - They are able to reduce a URL to URL edge
    - From 64 bits to an average of 3 bits
    - For a 118 million node web graph
  - How?

Connectivity Server in practice

- Consider a lexicographically ordered list of all URLS, e.g:
  - http://www.ics.uci.edu/computerscience/index.php
  - http://www.ics.uci.edu/dept/index.php
  - http://www.ics.uci.edu/index.php
  - http://www.ics.uci.edu/informatics/index.php
  - http://www.ics.uci.edu/statistics/index.php

Connectivity Server in practice

- Each of these URLs has an adjacency list
- Main idea: because of templates, the adjacency list of a node is similar to one of the 7 preceding URLs in the lexicographic ordering.
- So, express adjacency list in terms of a template

Connectivity Server in practice

- Consider these adjacency lists
    - 1, 2, 4, 8, 16, 32, 64
    - 1, 4, 9, 16, 25, 36, 49, 64
    - 1, 2, 3, 5, 6, 13, 21, 34, 55, 89, 144
    - 1, 4, 8, 16, 25, 36, 49, 64
        - Encode this as row(-2), -URL(9), +URL(8)
- Very similar to tricks done in assembly code

Connectivity Server in practice summary

- The web is enormous
- A naive adjacency matrix would be several billion URLS on a side
- Overall goal is to keep the adjacency matrix in memory
- Webgraph is a set of algorithms and a java implementation for examining the web graph
  - It exploits the power law distribution to compress the adjacency matrix very tightly
  - http://webgraph.dsi.unimi.it/

LUCI