Web Crawling

Introduction to Information Retrieval
Informatics 141 / CS 121
Donald J. Patterson

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

- Introduction
- URL Frontier
- Robust Crawling
  - DNS
The basic crawl algorithm

- Initialize a queue of URLs (“seed” URLs)
- Repeat
  - Remove a URL from the queue
  - Fetch associated page
  - Parse and analyze page
  - Store representation of page
  - Extract URLs from page and add to queue
Crawling the web

Introduction

Seed Pages
Web Spider
Crawled Pages
URL Frontier
The Rest of the Web
Basic Algorithm is not reality...

- Real web crawling requires multiple machines
  - All steps distributed on different computers
- Even Non-Adversarial pages pose problems
  - Latency and bandwidth to remote servers vary
  - Webmasters have opinions about crawling their turf
    - How “deep” in a URL should you go?
- Site mirrors and duplicate pages
- Politeness
  - Don’t hit a server too often
Basic Algorithm is not reality...

- Adversarial Web Pages
- Spam Pages
- Spider Traps
Minimum Characteristics for a Web Crawler

- **Be Polite:**
  - Respect implicit and explicit terms on website
  - Crawl pages you’re allowed to
  - Respect “robots.txt” (more on this coming up)

- **Be Robust**
  - Handle traps and spam gracefully
Introduction

Desired Characteristics for a Web Crawler

- Be a distributed systems
  - Run on multiple machines
- Be scalable
  - Adding more machines allows you to crawl faster
- Be Efficient
  - Fully utilize available processing and bandwidth
- Focus on “Quality” Pages
  - Crawl good information first
Introduction

Desired Characteristics for a Web Crawler

- Support Continuous Operation
  - Fetch fresh copies of previously crawled pages
- Be Extensible
  - Be able to adapt to new data formats, protocols, etc.
  - Today it’s AJAX, tomorrow it’s SilverLight, then....
Updated Crawling picture

URL Frontier

Seed Pages
Spider Thread
Crawled Pages
URL Frontier "Priority Queue"
The Rest of the Web
• Frontier Queue might have multiple pages from the same host
• These need to be load balanced ("politeness")
• All crawl threads should be kept busy
Politeness?

- It is easy enough for a website to block a crawler
- Explicit Politeness
  - “Robots Exclusion Standard”
    - Defined by a “robots.txt” file maintained by a webmaster
    - What portions of the site can be crawled.
      - Irrelevant, private or other data excluded.
    - Voluntary compliance by crawlers.
    - Based on regular expression matching
Politeness?

- Explicit Politeness
  - “Sitemaps”
    - Introduced by Google, but open standard
    - XML based
    - Allows webmasters to specify:
      - Location of pages (URL islands)
      - Importance of pages
      - Update frequency of pages
    - Sitemap location listed in robots.txt
Politeness?

- Implicit Politeness
  - Even without specification avoid hitting any site too often
  - It costs bandwidth and computing resources for host.
# Politeness?

## Statistics for: dpj3.net

<table>
<thead>
<tr>
<th>Summary</th>
<th>When:</th>
<th>Monthly history</th>
<th>Days of month</th>
<th>Days of week</th>
<th>Hours</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Who:</th>
<th>Countries</th>
<th>Full list</th>
<th>Last visit</th>
<th>Unresolved IP Address</th>
<th>Hosts</th>
<th>Full list</th>
<th>Last visit</th>
<th>Unresolved IP Address</th>
</tr>
</thead>
</table>

## Last Update: 14 Jan 2008 - 02:59

<table>
<thead>
<tr>
<th>Reported period:</th>
<th>Year: 2007</th>
<th>OK</th>
</tr>
</thead>
</table>

## Robots/Spiders visitors

<table>
<thead>
<tr>
<th>30 different robots</th>
<th>Hits</th>
<th>Bandwidth</th>
<th>Last visit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Googlebot</td>
<td>1393868+104</td>
<td>5.11 GB</td>
<td>31 Dec 2007 - 23:50</td>
</tr>
<tr>
<td>Inktomi Slurp</td>
<td>36668+221</td>
<td>554.25 MB</td>
<td>31 Dec 2007 - 23:55</td>
</tr>
<tr>
<td>MSNBot</td>
<td>19522+2</td>
<td>699.90 MB</td>
<td>28 Dec 2007 - 08:01</td>
</tr>
<tr>
<td>Unknown robot (identified by 'crawl')</td>
<td>15949+13</td>
<td>89.34 MB</td>
<td>31 Dec 2007 - 22:24</td>
</tr>
<tr>
<td>AskJeeves</td>
<td>7016+1</td>
<td>106.29 MB</td>
<td>31 Dec 2007 - 23:49</td>
</tr>
<tr>
<td>Google AdSense</td>
<td>2701</td>
<td>100.26 MB</td>
<td>31 Dec 2007 - 22:10</td>
</tr>
<tr>
<td>psbot</td>
<td>2268+1</td>
<td>80.48 MB</td>
<td>31 Dec 2007 - 09:59</td>
</tr>
<tr>
<td>Unknown robot (identified by 'robot')</td>
<td>930+1</td>
<td>19.10 MB</td>
<td>31 Dec 2007 - 09:34</td>
</tr>
<tr>
<td>Turn It In</td>
<td>350+1</td>
<td>6.32 MB</td>
<td>03 Sep 2007 - 15:44</td>
</tr>
<tr>
<td>BaiduSpider</td>
<td>300</td>
<td>10.22 MB</td>
<td>26 Nov 2007 - 07:32</td>
</tr>
<tr>
<td>GigaBot</td>
<td>243</td>
<td>5.27 MB</td>
<td>30 Dec 2007 - 05:06</td>
</tr>
<tr>
<td>Scooter</td>
<td>90+3</td>
<td>288.75 KB</td>
<td>27 Nov 2007 - 14:30</td>
</tr>
<tr>
<td>PhpDig</td>
<td>91</td>
<td>2.28 MB</td>
<td>21 Oct 2007 - 09:51</td>
</tr>
<tr>
<td>WISENutbot</td>
<td>76</td>
<td>1.94 MB</td>
<td>13 Jan 2007 - 14:04</td>
</tr>
<tr>
<td>Magpie</td>
<td>25</td>
<td>43.48 MB</td>
<td>24 Dec 2007 - 00:51</td>
</tr>
<tr>
<td>Unknown robot (identified by hit on 'robots.txt')</td>
<td>0+16</td>
<td>4.38 KB</td>
<td>14 Nov 2007 - 03:43</td>
</tr>
<tr>
<td>EchoO!</td>
<td>14</td>
<td>287.09 KB</td>
<td>27 Dec 2007 - 13:56</td>
</tr>
<tr>
<td>Internet Shinchakubin</td>
<td>13</td>
<td>385.03 KB</td>
<td>27 Nov 2007 - 15:23</td>
</tr>
<tr>
<td>BBot</td>
<td>10</td>
<td>146.35 KB</td>
<td>13 Jun 2007 - 15:17</td>
</tr>
<tr>
<td>arks</td>
<td>8</td>
<td>142.24 KB</td>
<td>27 Nov 2007 - 12:25</td>
</tr>
<tr>
<td>MSIECrawler</td>
<td>8</td>
<td>263.02 KB</td>
<td>26 Dec 2007 - 11:16</td>
</tr>
<tr>
<td>The Python Robot</td>
<td>5</td>
<td>423.91 KB</td>
<td>23 Nov 2007 - 08:44</td>
</tr>
</tbody>
</table>

---

Back to main page
### Monthly history

<table>
<thead>
<tr>
<th>Month</th>
<th>Unique visitors</th>
<th>Number of visits</th>
<th>Pages</th>
<th>Hits</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 2007</td>
<td>1221</td>
<td>2946</td>
<td>8938</td>
<td>30536</td>
<td>699.28 MB</td>
</tr>
<tr>
<td>Feb 2007</td>
<td>1179</td>
<td>3099</td>
<td>7852</td>
<td>20475</td>
<td>415.75 MB</td>
</tr>
<tr>
<td>Mar 2007</td>
<td>1120</td>
<td>3063</td>
<td>7099</td>
<td>18978</td>
<td>350.88 MB</td>
</tr>
<tr>
<td>Apr 2007</td>
<td>1362</td>
<td>3067</td>
<td>7175</td>
<td>30320</td>
<td>599.91 MB</td>
</tr>
<tr>
<td>May 2007</td>
<td>1612</td>
<td>3746</td>
<td>7584</td>
<td>25114</td>
<td>469.32 MB</td>
</tr>
<tr>
<td>Jun 2007</td>
<td>1474</td>
<td>3662</td>
<td>7138</td>
<td>22292</td>
<td>370.11 MB</td>
</tr>
<tr>
<td>Jul 2007</td>
<td>1592</td>
<td>4210</td>
<td>9165</td>
<td>24766</td>
<td>430.61 MB</td>
</tr>
<tr>
<td>Aug 2007</td>
<td>1658</td>
<td>4567</td>
<td>10600</td>
<td>24142</td>
<td>336.08 MB</td>
</tr>
<tr>
<td>Sep 2007</td>
<td>1458</td>
<td>4403</td>
<td>11149</td>
<td>24414</td>
<td>356.60 MB</td>
</tr>
<tr>
<td>Oct 2007</td>
<td>2148</td>
<td>5299</td>
<td>12877</td>
<td>36427</td>
<td>783.78 MB</td>
</tr>
<tr>
<td>Nov 2007</td>
<td>2890</td>
<td>6317</td>
<td>15300</td>
<td>40487</td>
<td>833.75 MB</td>
</tr>
<tr>
<td>Dec 2007</td>
<td>2748</td>
<td>6631</td>
<td>17553</td>
<td>42281</td>
<td>1.03 GB</td>
</tr>
<tr>
<td>Total</td>
<td>20462</td>
<td>51010</td>
<td>122430</td>
<td>340232</td>
<td>6.55 GB</td>
</tr>
</tbody>
</table>
Robots.txt - Exclusion

- Protocol for giving spiders ("robots") limited access to a website
  - Source: [http://www.robotstxt.org/wc/norobots.html](http://www.robotstxt.org/wc/norobots.html)
  - Website announces what is okay and not okay to crawl:
    - This file holds the restrictions
Robots.txt Example

- http://www.ics.uci.edu/robots.txt

```plaintext
User-agent: MOMspider
Disallow: /cgi-bin/ # The Multi-Owner Maintenance Spider
Disallow: /Admin/MOM/ # Local MOMspider output
Disallow: ~/fielding/MOM/ # Local MOMspider output
Disallow: /TR/ # Dienst Technical Report Server
Disallow: /MetaServer/ # Dienst Technical Report Server
Disallow: ~/eppstein/pubs/cites/ # Eppstein Database
Disallow: ~/fiorello/pvt/ # Private pages

User-agent: *
Disallow: /cgi-bin/ # All other spiders should avoid
Disallow: /Test/ # Script files
Disallow: /Admin/ # The test area for web experimentation
Disallow: /TR/ # Huge server statistic logs
Disallow: /MetaServer/ # Dienst Technical Report Server
Disallow: ~/fielding/MOM/ # Local MOMspider output
Disallow: ~/kanderso/hidden # Ken Anderson's stuff
Disallow: ~/eppstein/pubs/cites/ # Eppstein Database
Disallow: ~/fiorello/pvt/ # Private pages
Disallow: ~/dean/
Disallow: ~/wwwoffice/
Disallow: ~/ucounsel/ 
Disallow: ~/sao/
Disallow: ~/support/ 
Disallow: ~/icsdb/ 
Disallow: ~/bin/
```
Sitemaps - Inclusion


```xml
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=12&amp;desc=vacation_hawaii</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=73&amp;desc=vacation_new_zealand</loc>
    <lastmod>2004-12-23</lastmod>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=74&amp;desc=vacation_newfoundland</loc>
    <lastmod>2004-12-23T18:00:15+00:00</lastmod>
    <priority>0.3</priority>
  </url>
  <url>
    <loc>http://www.example.com/catalog?item=83&amp;desc=vacation_usa</loc>
    <lastmod>2004-11-23</lastmod>
  </url>
</urlset>
```
Overview

- Introduction
- URL Frontier
- Robust Crawling
  - DNS
A Robust Crawl Architecture

- WWW
- DNS
- Fetch
- Parse
- Doc. Finger-prints
- Seen?
- URL Filter
- Robots.txt
- URL Index
- Duplicate Elimination
- URL Frontier Queue
Robust Crawling

Processing Steps in Crawling

- Pick a URL from the frontier (how to prioritize?)
- Fetch the document (DNS lookup)
- Parse the URL
  - Extract Links
- Check for duplicate content
  - If not add to index
- For each extracted link
  - Make sure it passes filter (robots.txt)
  - Make sure it isn’t in the URL frontier
DNS

Domain Name Server

- A lookup service on the internet
- Given a URL, retrieve its IP address
  - www.djp3.net -> 69.17.116.124
- This service is provided by a distributed set of servers
  - Latency can be high
    - Even seconds
- Common OS implementations of DNS lookup are blocking
  - One request at a time

Solution:
- Caching
- Batch requests
Where is www.djp3.net?

Ask 192.5.6.30

Ask 72.1.140.145

Use 69.17.116.124

Give me a web page

dig +trace www.djp3.net
What really happens

The User
flickr:crankyT

Give me a web page

ISP DNS Cache

ISP DNS Server

OS DNS Cache

OS DNS Resolver

Firefox DNS cache

Where is www.djp3.net?

Ask 192.5.6.30

?.root-servers.net = 198.41.0.4

?.GTLD-SERVERS.net = 192.5.6.30

ns?.speakeasy.net = 72.1.140.145

Use 69.17.116.124

Give me a web page
Class Exercise

- Calculate how long it would take to completely fill a DNS cache.
- How many active hosts are there?
- What is an average lookup time?
- Do the math.
A Robust Crawl Architecture
Parsing: URL normalization

- When a fetched document is parsed
  - some outlink URLs are relative
  - For example:
    - has a link to “/wiki/Special:Statistics”
    - which is the same as
- Parsing involves normalizing (expanding) relative URLs
A Robust Crawl Architecture

- WWW
- DNS
- Fetch
- Parse
- Seen?
- Doc. Fingerprints
- Robots.txt
- URL Filter
- URL Index
- Duplicate Elimination
- URL Frontier Queue
Duplication

Content Seen?

- Duplication is widespread on the web
- If a page just fetched is already in the index, don’t process it any further
- This can be done by using document fingerprints/shingles
  - A type of hashing scheme
A Robust Crawl Architecture

WWW
DNS
Fetch
Parse
Seen?
Doc. Finger-prints
Robots.txt
Duplicate Elimination
URL Index
URL Frontier Queue
Compliance with webmasters wishes...

- Robots.txt
  - Filters is a regular expression for a URL to be excluded
  - How often do you check robots.txt?
    - Cache to avoid using bandwidth and loading web server
- Sitemaps
  - A mechanism to better manage the URL frontier
A Robust Crawl Architecture
Duplicate Elimination

- For a one-time crawl
  - Test to see if an extracted, parsed, filtered URL
    - has already been sent to the frontier.
    - has already been indexed.
- For a continuous crawl
  - See full frontier implementation:
    - Update the URL’s priority
      - Based on staleness
      - Based on quality
      - Based on politeness
Distributing the crawl

- The key goal for the architecture of a distributed crawl is **cache locality**
- We want multiple crawl threads in multiple processes at multiple nodes for robustness
  - Geographically distributed for speed
  - Partition the hosts being crawled across nodes
    - Hash typically used for partition
- How do the nodes communicate?
The output of the URL Filter at each node is sent to the Duplicate Eliminator at all other nodes.
URL Frontier

- Freshness
  - Crawl some pages more often than others
    - Keep track of change rate of sites
    - Incorporate sitemap info
- Quality
  - High quality pages should be prioritized
  - Based on link-analysis, popularity, heuristics on content
- Politeness
  - When was the last time you hit a server?
• Freshness, Quality and Politeness
  • These goals will conflict with each other
  • A simple priority queue will fail because links are bursty
    • Many sites have lots of links pointing to themselves
      creating bursty references
    • Time influences the priority
  • Politeness Challenges
    • Even if only one thread is assigned to hit a particular host it can hit it repeatedly
    • Heuristic: insert a time gap between successive requests
Magnitude of the crawl

- To fetch 1,000,000,000 pages in one month...
- a small fraction of the web
- we need to fetch 400 pages per second!
- Since many fetches will be duplicates, unfetchable, filtered, etc. 400 pages per second isn’t fast enough
Overview

- Introduction
- URL Frontier
- Robust Crawling
  - DNS
  - Various parts of architecture
  - URL Frontier
- Index
  - Distributed Indices
- Connectivity Servers
The output of the URL Filter at each node is sent to the Duplicate Eliminator at all other nodes.
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
URL Frontier Implementation - Mercator

Front queues

- Prioritizer takes URLs and assigns a priority
  - Integer between 1 and F
  - Appends URL to appropriate queue
- Priority
  - Based on rate of change
  - Based on quality (spam)
  - Based on application
Selection from front queues is initiated from back queues.

Pick a front queue, how?
- Round robin
- Randomly
- Monte Carlo
- Biased toward high priority
URL Frontier Implementation - Mercator

Back queues

- Each back queue is non-empty while crawling
- Each back queue has URLs from one host only
- Maintain a table of URL to back queues (mapping) to help
Back queues

- **Timing Heap**
  - One entry per queue
  - Has earliest time that a host can be hit again
- **Earliest time based on**
  - Last access to that host
  - Plus any appropriate heuristic
A crawler thread needs a URL

- It gets the timing heap root
- It gets the next eligible queue based on time, b.
- It gets a URL from b
- If b is empty
- Pull a URL v from front queue
- If back queue for v exists place it in that queue, repeat.
- Else add v to b - update heap.
URL Frontier Implementation - Mercator

Back queues

- How many queues?
- Keep all threads busy
- ~3 times as many back queues as crawler threads
- Web-scale issues
- This won’t fit in memory
- Solution
  - Keep queues on disk and keep a portion in memory.
Overview

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

- Why does the crawling architecture exist?
  - 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?
The index has a list of vector space models

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

A Column for Each Word (or “Term”)

A Row For Each Web Page (or “Document”)
### Term-Document Matrix

A Column for Each Word (or “Term”)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | Column 8 | Column 9 | Column 10 | Column 11 | Column 12 | Column 13 | Column 14 | Column 15 | Column 16 | Column 17 | Column 18 | Column 19 | Column 20 | Column 21 | Column 22 | Column 23 | Column 24 | Column 25 | Column 26 | Column 27 | Column 28 | Column 29 | Column 30 |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1 1 1 1 1 | 2 1 1 1 1 | 1 1 1 1 1 | 2 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 1 1 1 1 | 1 1 1 1 1 | 2 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 1 |

A Row For Each Web Page (or “Document”)

<table>
<thead>
<tr>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 1 1</td>
<td>1 1 1 1 0</td>
<td>0 0 1 1 1</td>
</tr>
</tbody>
</table>

Indices
Indices

The Term-Document Matrix

- Is really big at a web scale
- It must be split up into pieces
- An effective way to split it up is to split up the same way as the crawling
  - Equivalent to taking horizontal 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 (see Assignment #03)
  - Link Analysis (aka PageRank)
    - Provides input to “quality” for URL frontier
Indices - Connectivity Server

Adjacency Matrix - Conceptual Idea

A

B

C

A

B

C

0 1 1

0 0 0

0 0 1

A B C

A

B

C

0 1 1

0 0 0

0 0 1
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
- 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:
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/](http://webgraph.dsi.unimi.it/)
End of Chapter 20