Crawling the web

Announcements

• Late A1 submissions accepted until Wednesday 11:59pm
  – With 10% grade penalty
• A2 coming soon!
The web as a graph

- Each vertex of the graph is a webpage
- Edges represent links
  - An edge between A and B means that A links to B
- Crawling the web == traversing this graph
  - Except that we don’t know the structure of the graph ahead of time

Crawler Architecture

- Initialize frontier with seed URLs
- While frontier is not empty
  - Pick URL from frontier
  - If URL has not been visited
    - Add URL’s children to the back of frontier

Two useful data structures

- Queue (First-in-First-out)
  - Add new elements to the end
  - Remove elements from the front
- Stack (Last-in-First-out)
  - Add new elements to the end (or top)
  - Also remove elements from the top

BFS pseudocode:
- Queue Q;
- Add seed nodes (URLs) to end of Q;
- While Q is not empty
  - Remove node n from front of Q
  - If n has not been visited, add n’s children to the back of Q

DFS pseudocode:
- Stack S;
- Add seed nodes (URLs) to front of S;
- While S is not empty
  - Remove node n from front of S
  - If n has not been visited, add n’s children to the front of S
Graph traversal

• Breadth First Search
  — Visits all children of the root, then all children of the children, etc.
  — Finds pages along shortest paths from the seed page
  — Implemented with a Queue (First-in-First-out)

• Depth First Search
  — Visits the root's first child, then the first child of that child, etc.
  — Implemented with a Stack (Last-in-First-out)

Finding and following links

• Crawler needs to parse HTML code to find links to follow
  — look for tags like `<a href="http://site.com/page.html">`

• Also needs to resolve relative URLs to absolute URLs
  — E.g. in the page http://www.cnn.com/linkto/:

Canonical URLs

• Crawler converts URLs to a canonical form:
  — e.g. convert:
    http://www.cnn.com/TECH
    http://WWW.CNN.COM/TECH/
    http://www.cnn.com/bogus/../TECH/ to:
    http://www.cnn.com/TECH/

Document Conversion

• Text is stored in hundreds of incompatible file formats
  — e.g., raw text, RTF, HTML, XML, Microsoft Word, PDF

• Non-text files also important
  — e.g., PowerPoint, Excel

• Crawlers use a conversion tool
  — converts the document content into a tagged text format such as HTML or XML
  — retains some of the important formatting information

Character Encoding

• A character encoding maps between bits and glyphs
  — i.e., getting from bits in a file to characters on a screen
  — Can be a major source of incompatibility

• ASCII is basic character encoding scheme for English
  — encodes 128 letters, numbers, special characters, and control characters in 7 bits

• Other languages can have many more glyphs
  — e.g., Chinese has more than 40,000 characters, with over 3,000 in common use
Unicode

- Unicode is a mapping from numbers to glyphs that attempts to include all glyphs in all known languages
  - does not uniquely specify bits to glyph mapping!
  - e.g., UTF-8, UTF-16, UTF-32
  - UTF-8 uses between 1 and 4 bytes for each character – variable length encoding
  - e.g. in UTF-8 English (ASCII) characters require 1 byte, some Chinese characters require 4 bytes
  - UTF-32 uses 4 bytes for every character

UTF-8

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-127</td>
<td>0-7F</td>
<td>0xxxxxxx</td>
</tr>
<tr>
<td>128-2047</td>
<td>80-1FF</td>
<td>110xxxxx 10xxxxxx</td>
</tr>
<tr>
<td>2048-55295</td>
<td>80-1FFFF</td>
<td>1110xxx 10xxxxxx 10xxxxxx</td>
</tr>
<tr>
<td>55296-65535</td>
<td>D800-DFFF</td>
<td>Undefined</td>
</tr>
<tr>
<td>65536-1114111</td>
<td>E000-EFFFF</td>
<td>11110xxx 10xxxxxx 10xxxxxx 10xxxxxx</td>
</tr>
</tbody>
</table>

Static vs. dynamic pages

- Is it worth trying to eliminate dynamic pages and only index static pages?
  - http://www.census.gov/cgi-bin/gazetteer
  - http://informatics.indiana.edu/research/colloquia.asp
  - http://www.imdb.com/name/nm0578801/
- Why or why not?
- What do Google and other search engines do?

Web Crawling

- Web crawlers waste a lot of time waiting for responses to requests
  - What’s a solution?

Distributed Crawling

- Advantages to using multiple computers for crawling
  - Helps to put the crawler closer to the sites it crawls
  - Reduces the number of sites the crawler has to remember
  - Reduces computing resources required
- Distributed crawler uses a hash function to assign URLs to crawling computers
  - hash function should be computed on the host part of each URL
- Disadvantages of distributed crawling?
Politeness

• Modern crawlers use multiple machines to fetch hundreds of pages at once
  – But this could flood sites with requests for pages
• To avoid this, web crawlers use politeness policies
  – e.g., delay between requests to same web server

Controlling Crawling

• Even slow crawling will anger some web hosts, who object to any copying of their data
• Robots.txt file can be used to control crawlers
  – Websites can include this file in the main directory of their site; (nice) crawlers look for it and follow its directions

  User-agent: *
  Disallow: /private/
  Disallow: /confidential/
  Disallow: /other/
  Allow: /other/public/

  User-agent: FavoredCrawler
  Disallow: 

  Sitemap: http://mysite.com/sitemap.xml.gz