

Crawling the Web

1

Web Crawling

- ❖ Retrieve (for indexing, storage, ...) Web pages by using the links found on a page to locate more pages.

Must have some starting point

2

Type of crawl

- **Web crawl** versus
crawl of more limited network – **web**
 - cs.princeton.edu
 - internal co. network
- **complete crawl** versus
focused crawl by some criteria
 - pages on one topic
- Type of crawl will affect necessity/usability of various techniques

3

Issues?

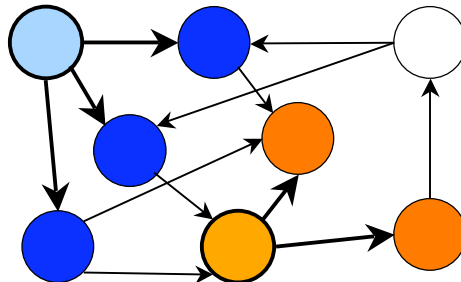
4

Main Issues I

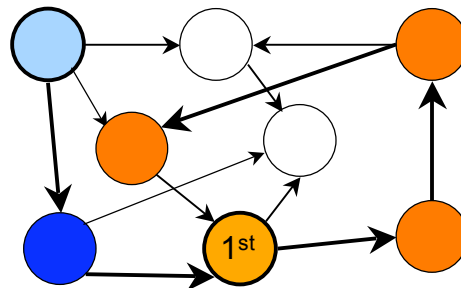
- starting set of pages?
- can visit whole of Web (or web)?
- how determine order to visit links?
 - graph model:
 - breadth first vs depth first
 - what are pros and cons of each?
 - “black holes”
 - other aspects /considerations
 - how deep want to go?
 - associate priority with links

5

- Breadth-first:



- Depth-first:



6

Main Issues II

- Web is dynamic
 - time to crawl “once”
 - how mix crawl and re-crawl
 - priority of pages
- Social behavior
 - robot exclusion protocol
 - not flood servers

7

Technical issues

- maintain one or **more** queues of URLs to be visited
 - order of URLs in queues?
 - FIFO = breadth first
 - LIFO = depth first
 - priority queues
- bottleneck: resolve hostname in URLs to get actual IP addresses – Domain Name Service servers (DNS lookup)
- To do large crawls must have **multiple crawlers with multiple network connections** (sockets) open and probably **multiple queues**

8

DNS lookup

- don't want temporal locality of reference
 - be nice to servers (or else)
- cache DNS map
 - large, local, in memory
 - hold most recently used mappings
- prefetch DNS resolution for URLs on page when it parsed
 - put in cache
 - use when URL gets to head of queue
- How “large” ?
 - Problems?

9

Duplicate URL removal

Has URL been visited already?

- Use:
 - canonical, fully specified URLs
 - canonical hostname provided by DNS
- *Visited?* hash table
 - hash canonical URL to entry
- *Visited?* table may be too large for MM

10

Caching *Visited?* table ?

- not temporal but “spatial” locality:
 - most popular URLs
 - most popular sites
 - *some* temporal locality within
- to exploit site-level locality need hash that brings pages on same site together:
 - two-level hash:
 - hash hostname and port
 - hash path
- can use B+ tree, sorted on i then ii
 - if no entry for URL in tree, not visited

11

Duplicate Page Removal

Has page been visited already?

- mirror sites – different URLs, same page
 - bad: duplicate page in search results
 - worse?: add links from duplicate pages to queues
 - also mirrors?
- table with fingerprint (hash) of each page
 - fit in main memory?
 - if not, costs disk access per page retrieved by crawler
 - bad

12

“almost” duplicates

- mirrored pages may have slight differences
 - indicate which mirror they on?
- need something like shingling
- when apply?
 - while crawling
 - v.s.
 - for search results
- cost?

13

Good and bad behavior

- Crawler **not flood servers**
 - queue for each server of near-term visits
- Crawler **check robot exclusion** for each server
- Sites may be badly behaved
 - dynamically generated pages to create:
 - **infinitely many pages**
 - **infinitely deep paths**
- Need strategies to detect/avoid bad behavior by sites

14

Re-crawling

- When re-crawl what pages?
 - finish crawl and start over
 - finish = have enough?
 - re-crawl high priority pages in middle of crawl
 - how determine priority?
- How integrate re-crawl of high priority pages?
 - One choice – separate cycle for crawl of high priority pages

15

Crawling **large** number pages

- indexing is **not** dynamic and continuous
 - Index all pages collected at certain time (end of crawl?)
 - Provide search half of engine with new index
- crawling is continuous
 - start over
 - in some sense

16