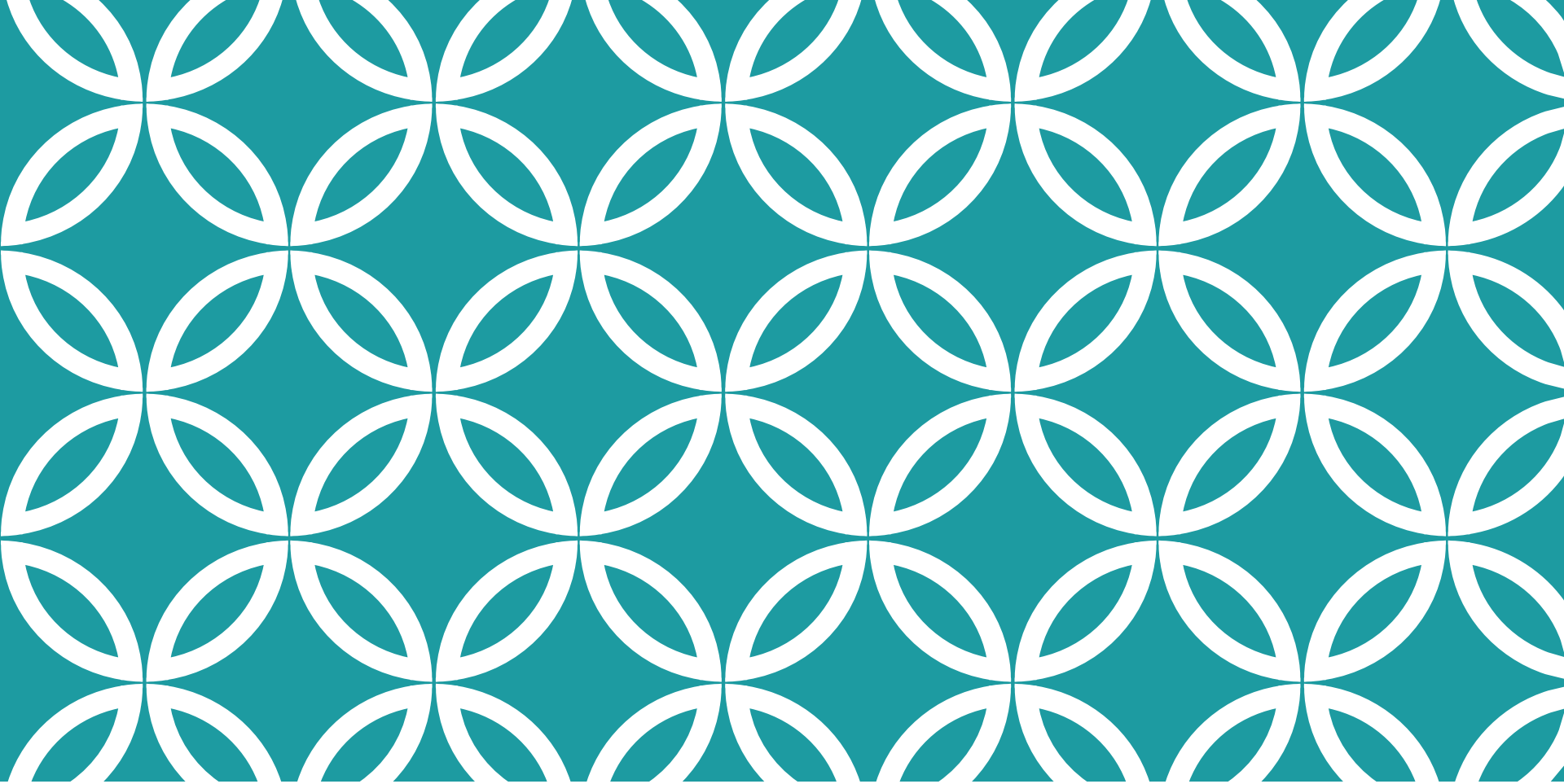


WEB SEARCH:

Web Search Components and Web Crawling

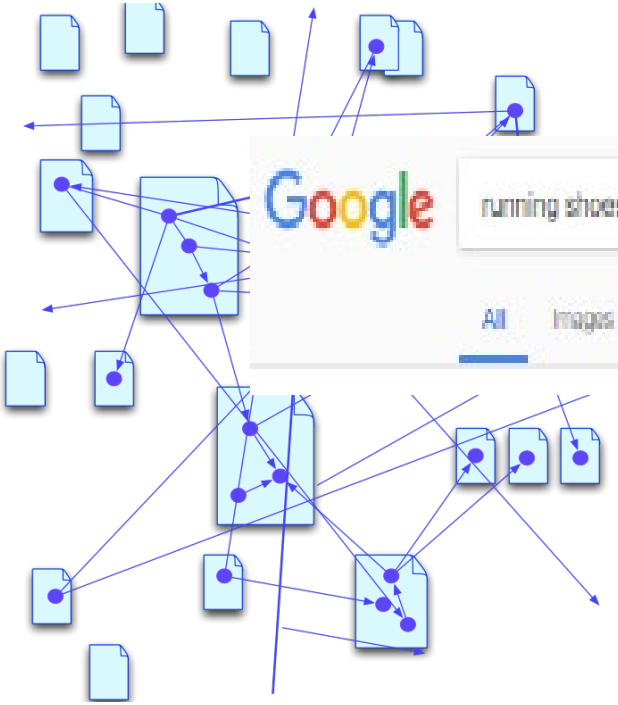
CT102 Information Systems

SEARCH ENGINE OVERVIEW:

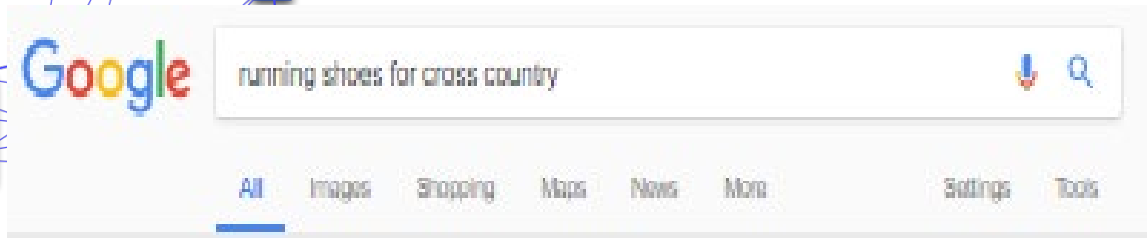
Inputs, processing, outputs

Data ⟶ Information

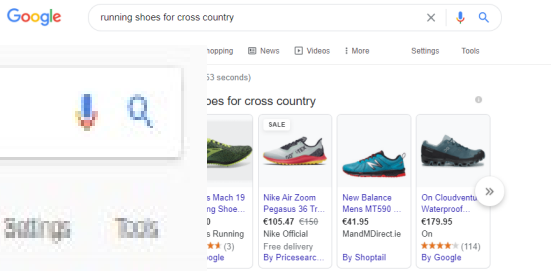
The data?

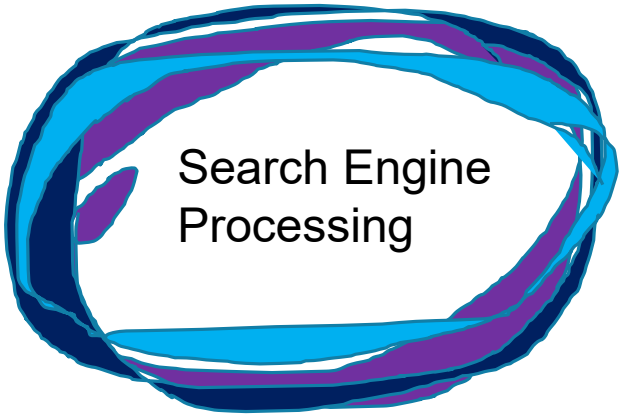


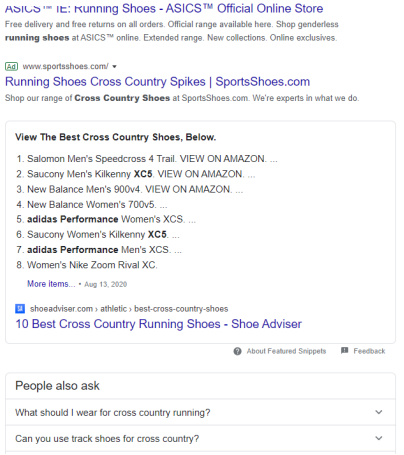
Input: The query



The output: The results







HOW SEARCH ENGINES WORK

A number of different systems are often part of a single search engine (1 of 2):

- Organic unstructured content: a system which matches query terms with terms in the index ("free" content)
- [potentially] Personal data: Personalised System – using personalised data (various forms)
- Organic HTML links: Page ranking System – using the existing HTML links between web pages to infer "importance" (no text/images used)
- Video and image data: Most often uses the text (or tags) associated with videos and images and matches this to query

HOW SEARCH ENGINES WORK
A number of different systems are often part of a single search engine (2 of 2):

- [potentially] Sponsored data: Ad System – matching keywords in paid ads and the user query keywords and using an automated auction
- Organic Structured Content: Semantic Web System – using semantically tagged information from linked open data sources (structured data)
- [potentially] Question Answering Systems, News Sources, Maps, Social Media sources, used if deemed appropriate given the query
- Displaying results: Ranking System – a system that takes all the different outputs from the multiple systems and returns lists of **ranked results** to the user (top to bottom of page, left to right of page)

TYPICAL STAGES IN (ORGANIC) WEB SEARCH (1 OF 3)

Prior to search time the following happens for organic content:

1. Crawl: Navigate the (unstructured) web by *crawling*
2. Parse: parse content and extract meaningful terms, links and other information from some portion of current web page
3. Index: Create indexes of web pages
4. Rank: Find how "important" or "authoritative" indexed web pages are by calculating the page rank scores of web pages

TYPICAL STAGES IN (ORGANIC) WEB SEARCH (2 OF 3): **AT SEARCH TIME**, GIVEN A USER QUERY

5. Use techniques, such as Euclidean dot product, to find relevant (organic documents) in the index including basic personalisation features (language, location, etc.). Incorporate Page Rank scores

6. Search and incorporate sponsored content (ads) including basic personalisation features (language, location, etc.)

7. Search and incorporate structured documents (if applicable)

8. Search and incorporate news or social media data (if applicable), include basic personalisation features if applicable.

9. Search and incorporate images and videos (if applicable)

TYPICAL STAGES IN (ORGANIC) WEB SEARCH (3 OF 3):
After search, **returning results**, given a user query (SERP: Search Engine Results Page)

10. Order and display all results (if they exist) from steps 5-9 based on a number of aspects (Ranking algorithm):

- Winning bids (Ads)
- Highest scores (organic content)
- Diversity across organic content and modalities (i.e., include images, videos, etc.)
- Revenue/business model (e.g., ads first, but not all ads)
- Query classification … e.g., if query "looked like" a question, display structured content first – the "answer".
- And more!

11. Based on previous, similar queries to current query, suggest alternative queries

PUTTING SOME OF THE STEPS TOGETHER FOR ORGANIC SEARCH?

https://www.youtube.com/watch?v=LVV_93mBfSU

Start at 0:44 … slightly simplistic overview but some important points

NOTE . . .

Personalisation mentioned a lot for different processes

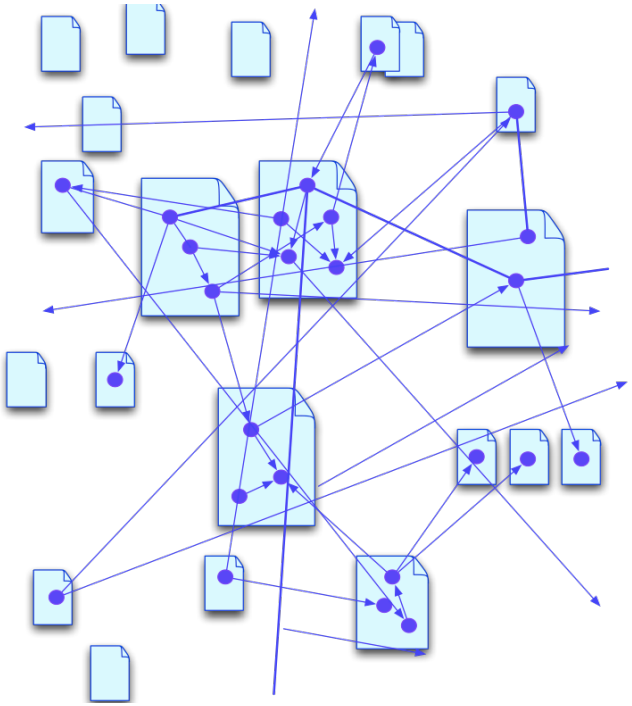
. . . what does this mean?

. . . do we want it?

. . . can we control it?

Now looking at organic search in more detail ….

Starting with Step 1: Web Crawling



WEB CRAWLING

- Web crawlers find content (web pages) on the web (independent of any query).
- It is the index that results from crawling websites and parsing the content that is used in live searches.
- You can submit your site to search engines but for larger sites (which have many links to them), web crawlers will often find and index a site automatically.

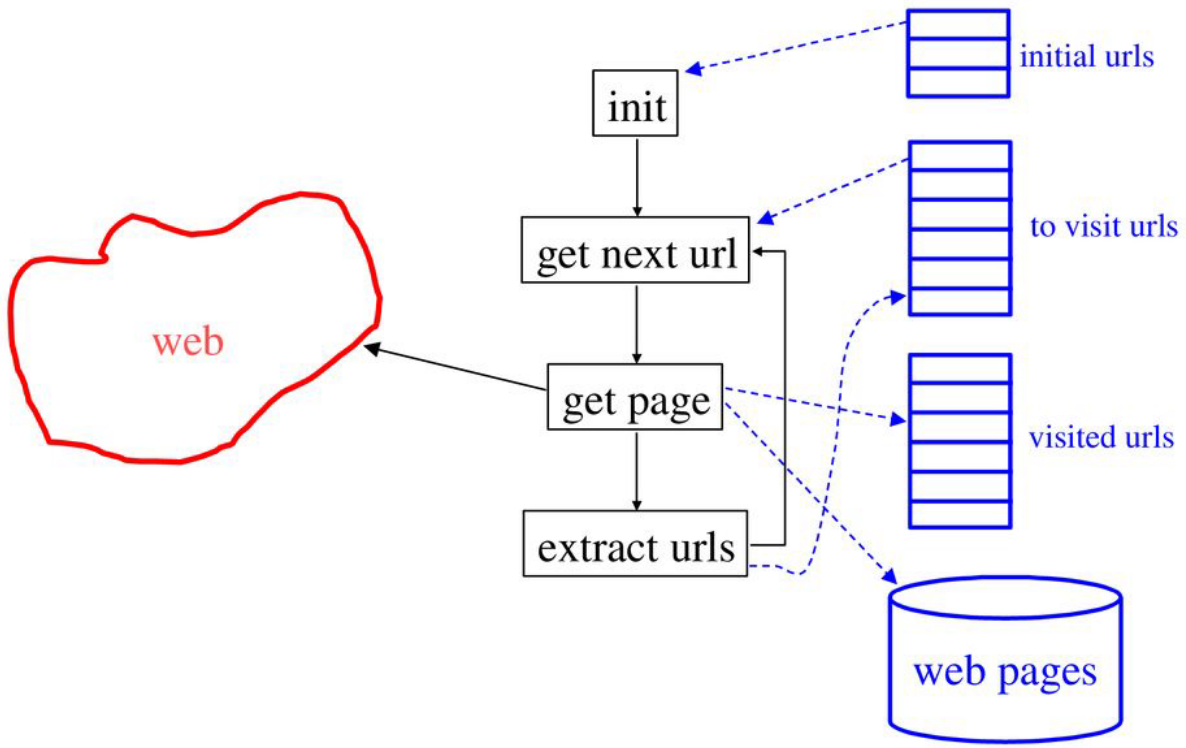
OVERVIEW OF CRAWLING:

There exists no central repository or “directory” of all websites so each search engine builds its own index.

The process of creating and updating this directory is done by “crawlers” (or bots or spiders).

Examples include “Googlebot”, “Slurp” (Yahoo), “bingbot”.

WEB CRAWLING OVERVIEW (diagram)



REP: ROBOTS EXCLUSION PROTOCOL

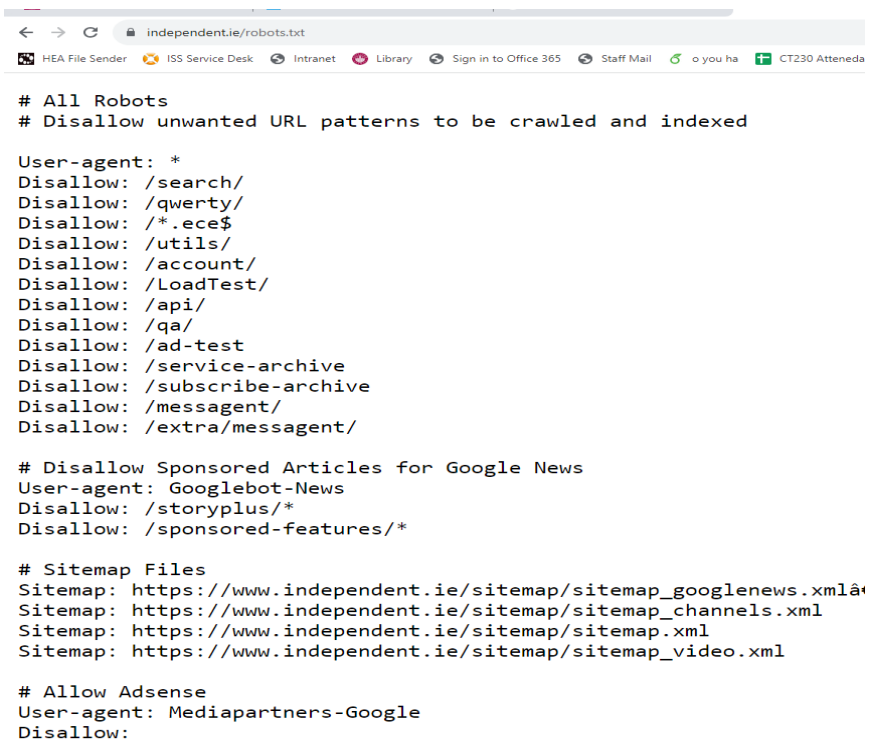
`robots.txt` is a text file webmasters create to instruct crawlers how to crawl pages on their website.

The file must be placed in root domain to be found by the crawler.

When a crawler which is following REP visits a webpage it will look first for a `robots.txt` file at the root domain.

If one exists, the crawler will follow the instructions in that file before crawling the site and downloading any content.

robots.txt



Basic format:

```
User-agent: [user-agent name]
Disallow: [folder or URL not to be crawled]
```

- Might also want to specify a delay for bots that crawl frequently (not supported by all crawlers), e.g.,

```
Crawl-delay: 10
```

- And can also indicate where the sitemap is stored (not supported by all crawlers)

EXAMPLES

Blocking all web crawlers from all content

```
User-agent: *
Disallow: /
```

Blocking one web crawler from a folder:

```
# Disallow Sponsored Articles for Google News
User-agent: Googlebot-News
Disallow: /storyplus/*
Disallow: /sponsored-features/*
```

SAMPLE (*high level*) ALGORITHM FOR CRAWLING WEB (1 of 5)

The crawler begins with one or more URLs that constitute a *seed set* and adds these to the **frontier set**.

The frontier set is a "to do" list of web pages to fetch … or can view it as an open list of unvisited nodes in the web graph.

The crawler **picks** a URL from the frontier set in some order, e.g. FIFO (first in first out) queue or priority queue … often want to re-visit some web pages more frequently than others – that is, those where content is updated frequently.

SAMPLE ALGORITHM FOR CRAWLING WEB (2 of 5)

For the URL picked:

- looks up DNS (find IP address of a domain name)
- connects to host
- sends request
- receives response
- Based on response … wait/retry/redirect/proceed

* Crawlers need to have timeouts so that an unnecessary long amount of time is not spent waiting for a response or reading a web page

SAMPLE ALGORITHM FOR CRAWLING WEB (3 of 5) … FETCHING WEB PAGES

If proceeding, get `robots.txt` and using instructions there, proceed to fetch contents and new urls from the web page ….

The HTTP protocol is used to fetch the web page given the URL picked from the frontier set

Generally do not download an entire web page - only the first portion of a web page – this is seen (and has been proven to be) as *representative enough* of the content of the page

SAMPLE ALGORITHM FOR CRAWLING WEB (4 of 5) … FETCHING WEB PAGES

The fetched portion of the page is written to a temporary store

The page is parsed, to extract both the text, images, video and the links (urls) from the page (each of which points to another web page) and any other information.

Tests are done to see if a web page with the same content has already been seen at another URL or if the page is spam or has been compromised and the index (and/or blacklist) is updated based on these tests

SAMPLE ALGORITHM FOR CRAWLING WEB (5 of 5) … FETCHING WEB PAGES

If the source is valid and should be stored then:

- The extracted text, links and other information are fed to a text indexer
- The extracted links (URLs) are then added to the *URL frontier set*, if it is not already in the list to be visited later

DISTRIBUTED CRAWLING

In reality crawling is often distributed:

- A single URL server sends lists of URLs to a number of crawlers
- Each crawler keeps a number of connections open at once. This is necessary to retrieve web pages at a fast enough pace.
- A major performance stress is DNS lookup so each crawler often maintains its own DNS cache so it does not need to do a DNS lookup before crawling each document.

DISTRIBUTED CRAWLING

Each of the connections open by a single crawler may be in a number of different states at any one time:

- looking up DNS
- connecting to host
- sending request
- receiving response

Each crawler uses a number of queues to keep track of the status of each connection state

CRAWLER BEHAVIOUR

A crawler requires:

A *selection policy*: to know what order to choose URLs from the frontier list and where to add new URLs

A *re-visit policy*: *t*o indicate when a page already in the index should be revisited to check for additions/deletions and new URLs

A *politeness policy*: to avoid overloading websites by visiting them too often or having too many requests in a short time period

A *parallelization policy:* to coordinate distributed web crawlers

SUMMARY: CRAWLER ALGORITHM

1. Pick URL from the list using some selection policy and re-visit policy, and before connecting check politeness policy

2. Try to Connect to URL (start timer for connection): DNS lookup; Connect to host; Send request.

3. If connection made:
   - 3.1 check robots.txt file and note restrictions (if any)
   - 3.2 fetch content (timed)
   - 3.3 write content to temporary store
   - 3.4 update URL list with timestamp of visit to this URL
   - 3.5 go back to step 1 (proceed with new URL from list)

4. If timeout reached and connection not made, based on selection and re-visit policy: wait, re-try or update URL list to indicate unsuccessful connection and go back to step 1 (proceed with new URL from list)

INDEXING

Indexes are very common storage structures in Computing and allow quick look-up of data

For web search, the indexing process organises and stores (a subset of) the data gathered by the crawlers so that it can be searched in a quick and efficient manner

Typically stores some version of

`<url, term, weight>`

Typically some type of compression is used

Two aspects:

- Weighting of terms to represent their importance
- Storage for fast retrieval (indexing and hashing structures) (and also reduces the space used)

Google index approx. 60 trillion web pages?

SUMMARY:
OWN REFLECTIONS AND QUESTIONS

What did you learn?

What are the important things to remember?

What are your questions?

SUMMARY … after this lecture you should be able to answer the following:

1. What is “organic” web search?
2. How does web search work?/ What are the different stages in web search?
3. What is web crawling and why is it used?
4. What are the typical stages in web crawling?
5. What are the typical crawler “rules” (i.e. politeness, etc.)
6. What is distributed web crawling?
7. What is a web search index?