What should your notes contain?

1. Some questinos

2. Picutres are important too here

And then the code here becomes sth else entirely

1. Used to discover new services here on the internet here

What are the top projects?

1.  Given a set of URLs, download all the web pages addressed by the URLs.

2. What is the main purpose of the crawler? Is it used for search engine indexing, data mining, or something else?

3. What does it discover here or there?

Search engine indexing.

4. Important question such as  Do we need to store HTML pages crawled from the web?

5.

**What's up with google here and there?**

Search engine indexing: This is the most common use case. A crawler collects web pages to create a local index for search engines. For example, Googlebot is the web crawler behind the Google search engine.

As shown above here

What are the steps

1. Propose a high level design here

2. Divide the entire url space into smaller spaces here and there,

**Seed urls?**

1. For example, to crawl all web pages from a university’s website, an intuitive way to select seed URLs is to use the university’s domain name as the starting point here

**ALl web pages from the university website here, and then here**

**How do we choose the seed urls here?**

**1.** Based on topics? or what

2. What's a dns resolver?

- translates url into an ip address ehre

3. Content parser here

**How do we prevent file duplication?**

We can use MD5 or SHA hashing to hash the content of a webpages and then and then genreate a hash value, and then compareing the hash values we can then determined sth

URL seen? Use a set

URL storage

**Why would we be able to use dfs vs bfs in this case?**

1. dfs not a good choice basically, However, DFS is usually not a good choice because the depth of DFS can be very deep.

2. What's wrong with using bfs here?

Standard BFS does not take the priority of a URL into consideration. The web is large and not every page has the same level of quality and importance. Therefore, we may want to prioritize URLs according to their page ranks, web traffic, update frequency, etc.

**How do we deal with the problem of standard of bfs here?**

What's a url frontier?

URL frontier helps to address these problems. A URL frontier is a data structure that stores URLs to be downloaded.

1.

**Why should a web crawler avoid sending too many requesrts to the same host server?**

Can be misconstrued as dos here

2.

**The queue router from before here**

**A diagram of a computer

Description automatically generated**

What's the b1, b2 up here?

basically the b1, b2 are the each queue for each host here

The structure looks like the following:

| **Host** | **Queue** |
| --- | --- |
| wikipedia.com | b1 |
| apple.com | b2 |
| ... | ... |
| nike.com | bn |

**How does the q selecltor know how to prioritize?**

1. Use priority from the website traffic, q selector know where to direct traffic based on the priority of the traffic here

2. Each q has an assigned priority here nad there

A diagram of a diagram

Description automatically generated

**How to design storage for url frotnier here?**

1. Can't keep everything in either memory or the disk not maintable here or there

2. So adopt a hybrid approach if this one doesn't work here

What about the html downloader?

**How to optimize performance?**

**Distributed crawl**

To achieve high performance, crawl jobs are distributed into multiple servers, and each server runs multiple threads. The URL space is partitioned into smaller pieces; so, each downloader is responsible for a subset of the URLs. Figure 9 shows an example of a distributed crawl.

A screenshot of a computer

Description automatically generated

Using the url frontier here

1.

- have a mapping between website hostnames to download (worker) threads

**DOwnloader thread:**

- Each downloader thread has a separate FIFO queue and only downloads URLs obtained from that queue

2. Each host will have a queue

Mapping table

- map each host to a queue

**Why cache DNS resolver here and what is it?**

1. To download a web page, a URL must be translated into an IP address

2. Get the corresponding ip address for the url here URL www.wikipedia.org is converted to IP address 198.35.26.96

3. Once a request to DNS is carried out by a crawler thread, other threads are blocked until the first request is completed. Maintaining our DNS cache to avoid calling DNS frequently is an effective technique for speed optimization.

And now these 2 are done here

What are the 3 steps to a healthy server?

1. Robustness:

consistent hashing:  This helps to distribute loads among downloaders. A new downloader server can be added or removed using consistent hashing. Refer to the "Design consistent hashing" chapter for more details.

3.

Queue router:

It ensures that each queue (b1, b2, … bn) only contains URLs from the same host.

3. Each worker thread is mapped to a FIFO q, t only downloads URLs from that queue.

4. Worker thread 1 to N. A worker thread downloads web pages one by one from the same host. A delay can be added between two download tasks.