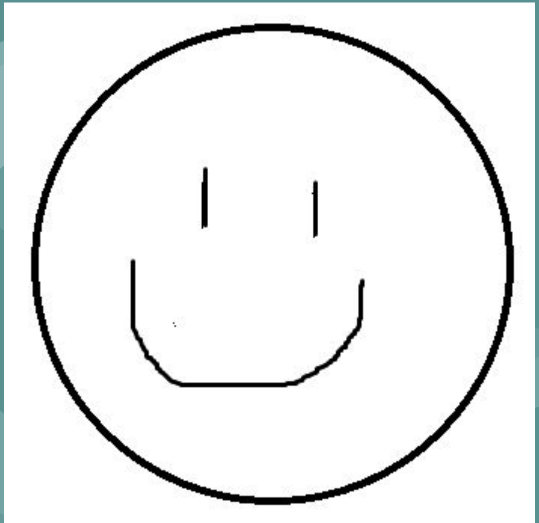
Analysis of: Crawler4j



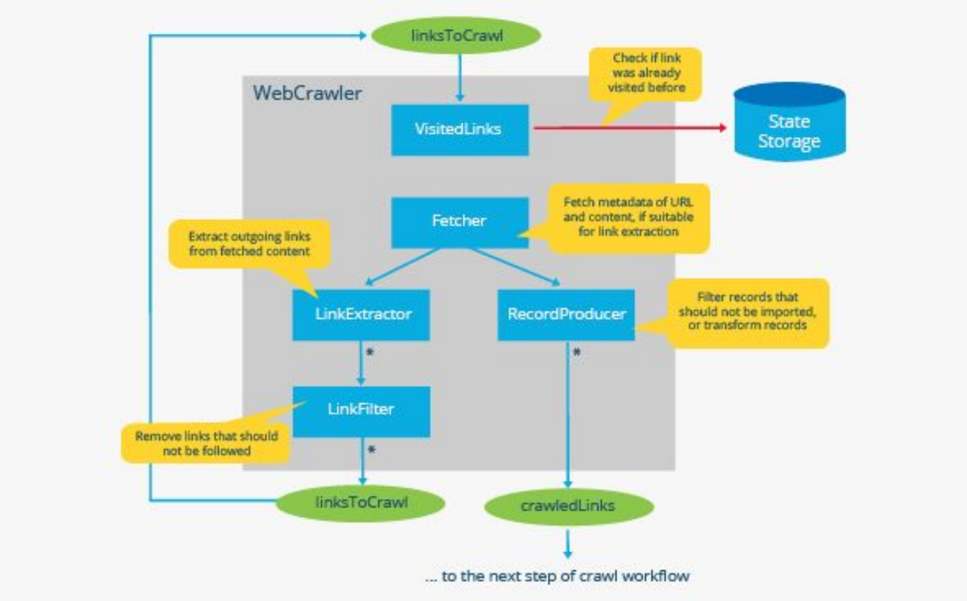
Dajel Bourque - B00795600
Julia Lewandowski - B00732018
Rasheed Khan - B00812361
Tanish Dagar - B00822133

Overview

What it is and how it works?

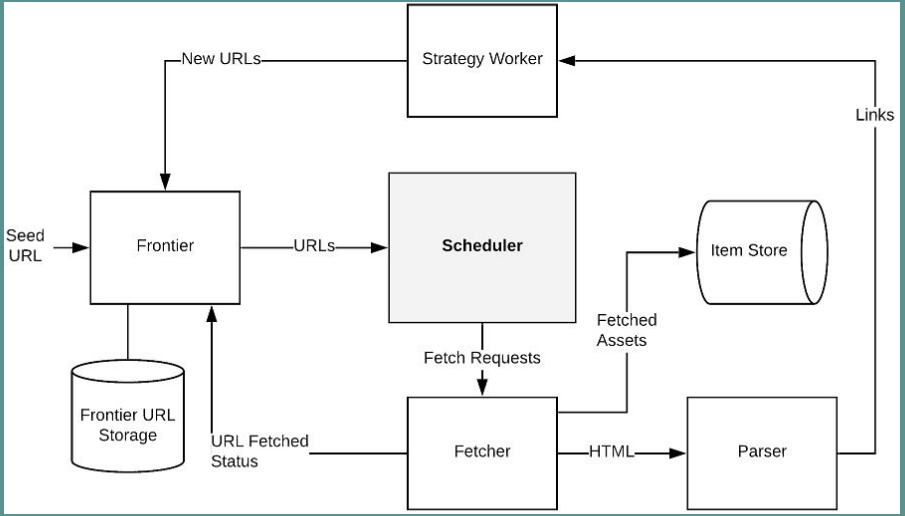
Web Crawler

- Sifts through pages on the web, often to index them
- More specifically, Crawler4j is a data scraper
- Crawling through web pages, but ultimately collecting specified data


linksToCrawl
Check if link was already visited before
WebCrawler
VisitedLinks
State Storage
Fetcher
Fetch metadata of URL and content, if suitable for link extraction
Extract outgoing links from fetched content
LinkExtractor
RecordProducer
Filter records that should not be imported, or transform records
*
*
LinkFilter
Remove links that should not be followed
*
linksToCrawl
crawledLinks
... to the next step of crawl workflow

How is this done?

- Frontier:
    - Priority queue
    - Visited, visiting, to visit
- Fetcher:
    - Collects the page
    - Use credentials
- Parser:
    - Collects elements of page
    - Exceptions



Customizability

- MyCrawler extends WebCrawler
    - boolean shouldVisit
    - void visit

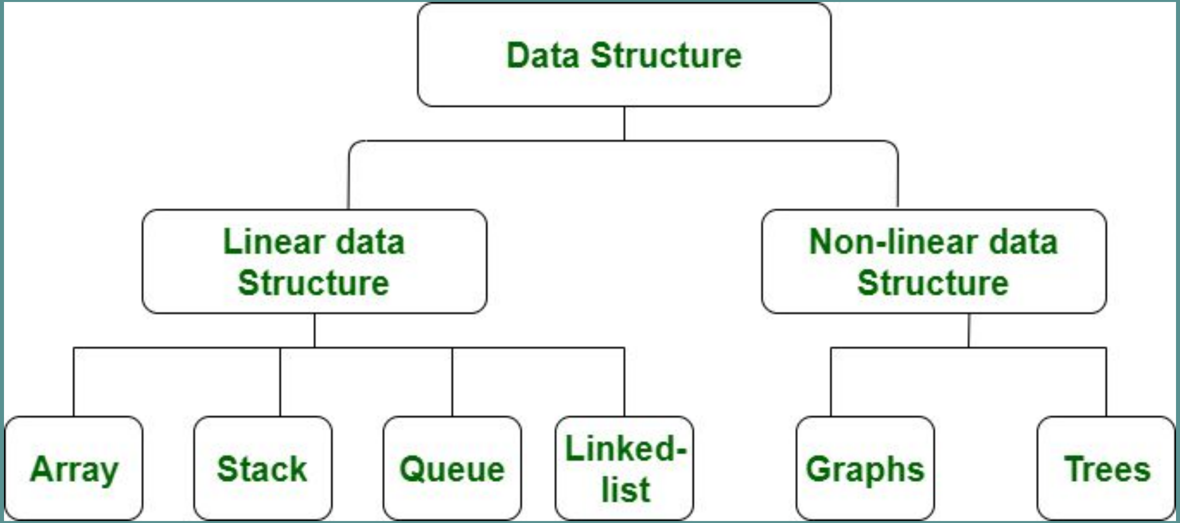
- Controller
    - Seeds
    - Destination Folder
    - Threads

Data Structures

What structures are used and are they efficient?

How is data stored?

- Overview talked about the general idea of the crawler
- This section will be highlighting key data structures used in Crawler4j.



Data structures in Page.java

- This class is used to store data of a fetched or a parsed page.
- It initializes an array giving information about which type of data to be stored?
- For example if it is a .PNG file or a .JPG file

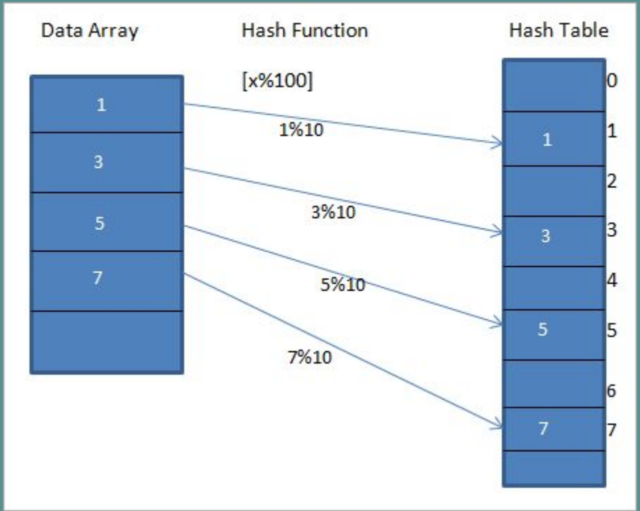
- This class has a method called ***fetchResponseHeader*** which returns the header which were present in the response of the fetch. These headers are stored in an array.

Data structures in WorkQueues.java

- This class appoints a unique key to a URL.
- The key chooses which URL to crawl first.
- The lower the value of the key the higher would be the priority of the URL.
- This is done by making this class as a priority queue class. O(n)

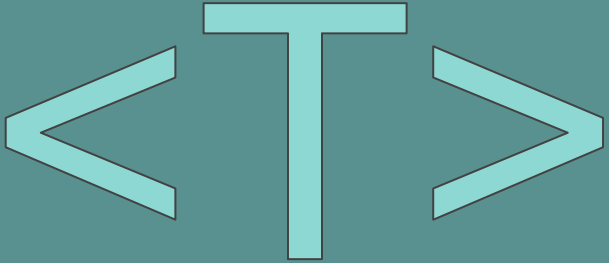
Data structures used in CrawlConfig.java

- In this class hash sets are used for a couple of things for example returning a copy of the default header collection and creating copies of the provided headers.
- It is an unordered collection containing unique items due to which every index in the hash set has a unique value.
- The algorithm complexity of the hash set would be O(1).



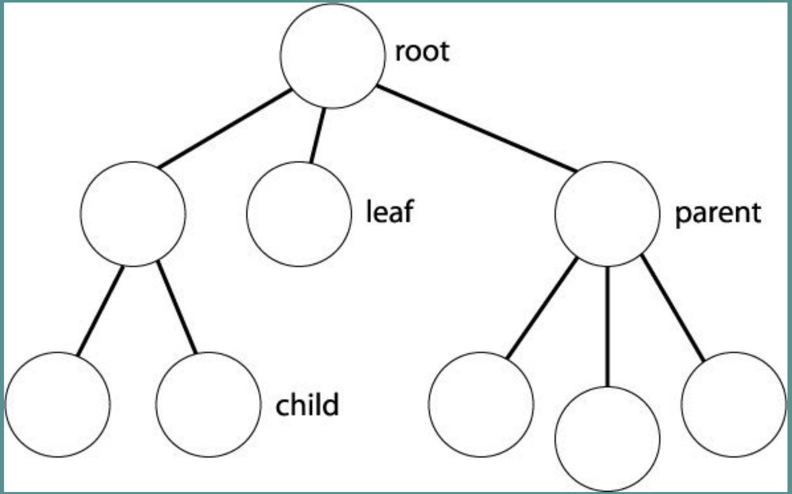
Data structures used in CrawlController.java

- An ArrayList is used in the ***CrawlController*** class of the Crawler folder.
- It essentially collects the local data of the crawler threads and stores them in the list. O(n)
- In this class generics is used too as it extends the webcrawler class.
- Some methods in this class return a generic type value.



Data structures in HostDirectives.java

- This class implements a TreeSet of ***UserAgentDirective*** objects in its void method ***setUserAgent***.
- The TreeSet data structure allows for a set of type Tree to exist so that for each ***UserAgentDirective*** tree node, the relationship between them is preserved.



Algorithms

Uses, Efficiency, and Time Complexity

Some things to keep in mind:

- 2/3rds of the program mostly consists various sections of many set/get, overridden and overloaded code, and customizable code.
- Only certain, less-complex, important algorithms have been highlighted.

Efficiency

- Overall: efficient in terms of time complexity.
- Average: efficient in terms of time complexity.

We are not tackling space complexity.

Algorithms in CrawlConfig.java

Lines 243-258: validate method. This method validates the configuration specified by the instance it is paired with.

Algorithms in CrawlController.java

- **Lines: start method.** Starts crawling session and waits for it to finish.
- **Lines: waitUntilFinish method.** Waits until a crawling session finishes.
- **Lines: sleep method.** This method allows controller to collect local data of crawler threads and stores them.
- **Lines 649-654: shutdown method.** This method sets the current crawling session set to a ‘shutdown’. The Crawler threads (from the Thread objects) will monitor the shutdown flag; if it is set to true, it will call the *shutDown* method from *pageFetcher* and the *finish* method from *frontier*.

Algorithms in WebCrawler.java

- **Lines 313-353: run method**. Algorithm for actual run of the crawler. It initializes a boolean *halt* as False and opens a while loop which continues until *halt* is True. Each visited URL is then passed to the processPage function.

- **Lines 413-585: processPage method.** Fetches page to be parsed. It runs through several condition checks ensuring that: the fetched page is valid, the customized method shouldVisit (see **overview section**) returns boolean True, and then performs several more exception checks.

Algorithms in Parser.java

Lines 64-134: parse method. Performs various conditional checks to determine which parsing objects should be instantiated for the page (e.g. HtmlParser, CssParseData, etc.) and passes the page through these parsers to collect relevant data.

Algorithms in Page.java

Lines 163-190: load method. Loads page content from a fetched HttpEntity and throws an IOException when the load fails. Attributes *contentType* and *contentEncoding* of the Page object to null and then reads in the entity's *contentType* and *contentEncoding* but only if it is NOT null.

Algorithms in HostDirectives.java

Lines 62-64: allows method. Determines if host directives allow path to be visited

Lines 88-90: disallows method. Checks if host directives directly disallow visiting path.

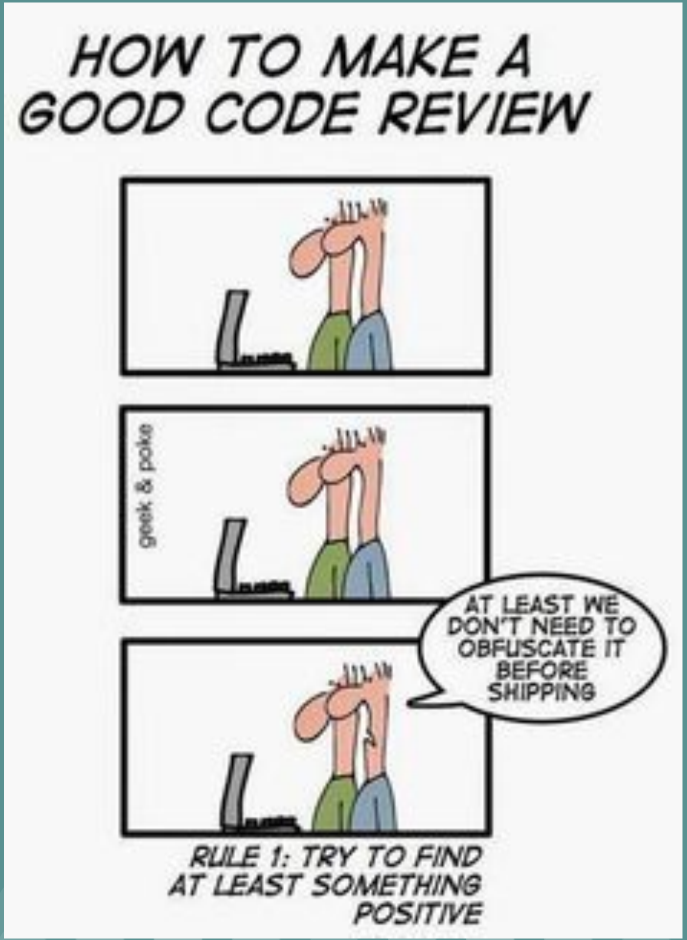
Lines 98-127: checkAccess method. Checks if any rules say anything about the specified path.

Conclusions

Critiques and Suggestions

Positive Factors

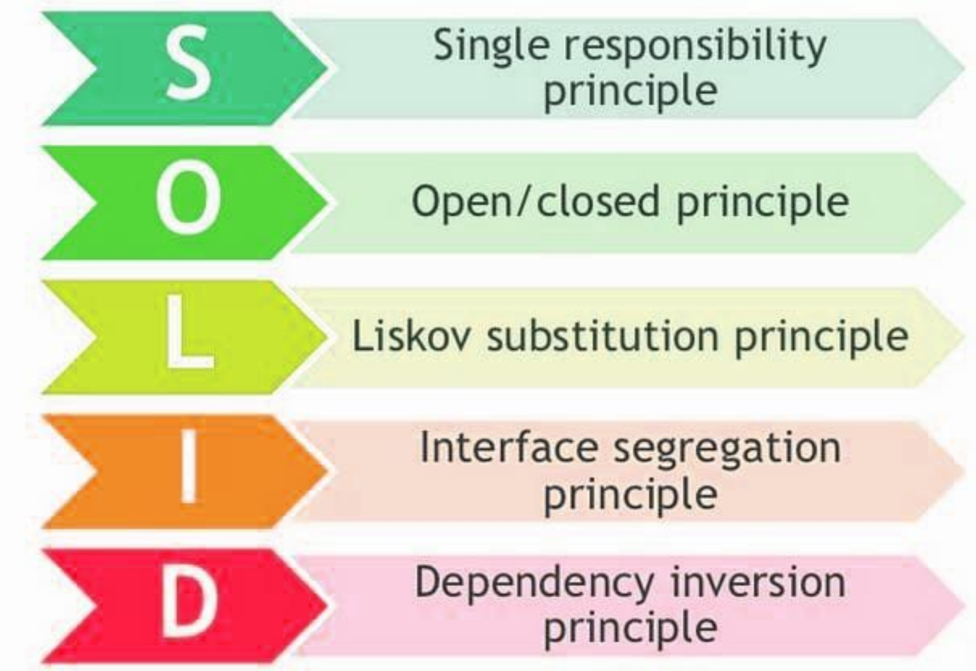
- Speed and efficiency.
- Language neutral scraping.
- Offers scalability options.
- Intelligent error checking.
- Flexible and configurable.


HOW TO MAKE A GOOD CODE REVIEW
geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Areas for Improvement

- Clear commenting is lacking in some larger class files.
- Certain exceptions that are caught or tried may be removable.
- Characteristics of intelligent recrawling are not exhibited (potential area for expansion!).

Robert C. Martin’s SOLID Principles



S Single responsibility principle

O Open/closed principle

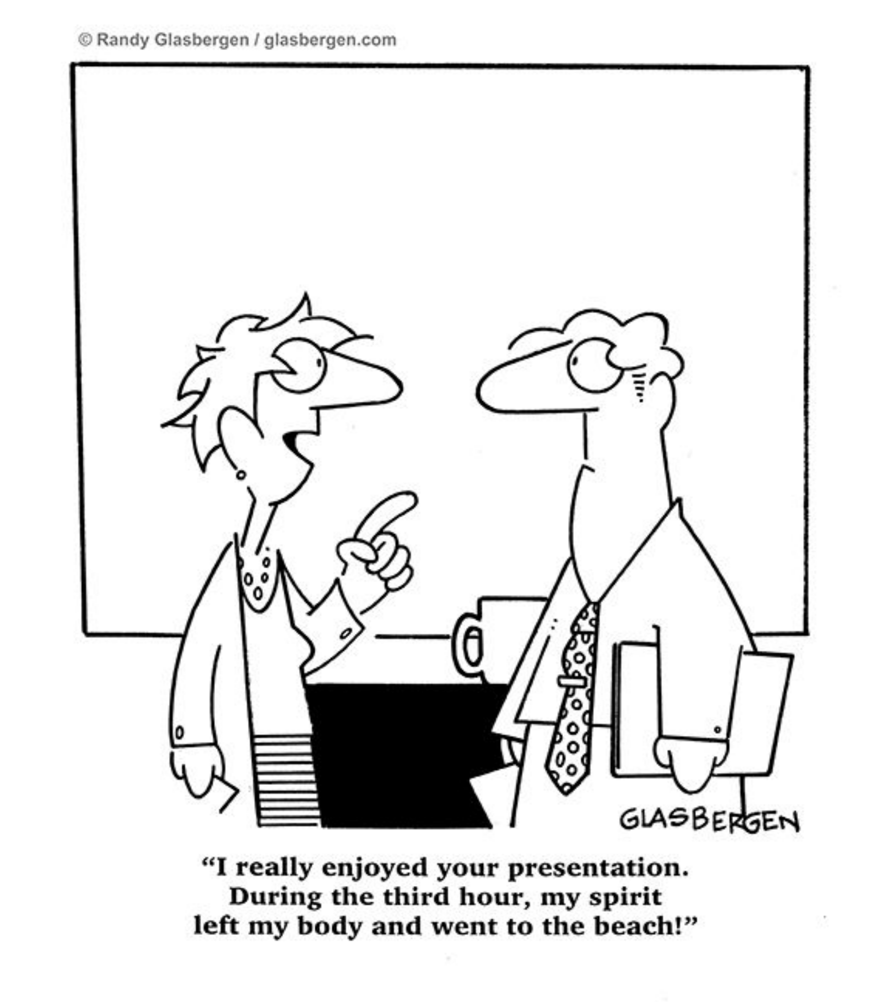
L Liskov substitution principle

I Interface segregation principle

D Dependency inversion principle

Important for evaluating **flexibility**, **maintainability**, and potential for **minimal time complexity** of the software as it grows in size.

Thank you for your attention!



"I really enjoyed your presentation. During the third hour, my spirit left my body and went to the beach!"

Image Sources

Slide 3: https://en.ryte.com/wiki/nsfr_img_auth.php/4/48/Crawler.png

Slide 4: https://livebook.manning.com/book/ai-as-a-service/chapter-7/v-3/

Slide 5: https://www.openeducat.org/page/features/customizable

Slide 10: https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.pinimg.com%2F236x%2Fe2%2F7e%2F30%2Fe27e30d4a5664bc266aff8b03ccc4028--software-development-programming-humor.jpg&f=1&nofb=1\

Slide 26: https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fwww.glasbergen.com%2Fwp-content%2Fgallery%2Fmeetings%2Ftoon-3196.gif&f=1&nofb=1