HW2: Web-crawling

Summary

In this homework, you will develop a 'crawler' by incorporating calls from existing APIs (eg. 'crawler4j' in Java) to 'crawl' (visit, fetch contents of) a list of URLs, starting with an empty queue which you would initialize, then recursively shrink/grow it (remove items, possibly add new items), till the queue is empty again [similar to how you'd save a set of web pages (including linked pages) to disk 'by hand'] :)

Description

Here is what you need to do [launch a crawler, compile stats from the crawling results, submit the stats]. The description uses crawler4j [Java] to discuss the steps, BUT you can feel free to use Python, JS, C#, C++, etc! **In addition to what's asked, please also submit your crawler source file(s).**

If you want to use Java, these will help:

- https://www.baeldung.com/crawler4j
- https://github.com/yasserg/crawler4j
- crawler4j installation
- crawler4j crawling loop
- https://replit.com/@satychary/Crawl-Fly-Run-Roll-Slither-Wiggle-BounceOMG (!!)

Here are resources for other (non-Java) langs [feel free to use any others, eg. Ruby, Dart, Go, Rust(!!), C#, Perl...]:

JS: https://www.zenrows.com/blog/javascript-web-crawler-nodejs

Python: https://www.scrapingbee.com/blog/crawling-python

C: https://github.com/ben-muldrow/webcrawler [the recursive parse() function is the heart of a crawler!]

C++: https://www.webscrapingapi.com/c-web-scraping

You can do the coding in pieces (each piece adds to ones before it):

- start with a single url in a queue
- remove queue item (url), fetch its contents
- examine contents, extract links
- add the links to the queue
- recurse the above three steps
- add stats collection to the above
- systematically accumulate results
- write out what you accumulated

Here is how we will evaluate your submission!

Getting help

There is a hw2 'forum' on Piazza, for you to post questions/answers. You can also meet w/ the TAs, CPs, or me.

Ok now go crawl!! Have fun :)
