Rohan Tandon
CS 446

Report: Programming Assignment 1

The entire source code for part A and B is under code/Webcrawler.

1. Design Tradeoffs: I implemented a single-threaded webcrawler. Queue was used for the frontier and an additional visited queue was used to check if the link had already been visited. Also, robots.txt was parsed for each time a new domain was found and forbidden urls were added to a queue for reference. Also, politeness was maintained by creating a hashmap of the link and time of visit. This hashmap was used to check if the webserver was visited in the last 5 seconds or not. In general data structures were used to avoid parsing the same links/robots.txt files. The time complexity can be improved by using multithreading. One master thread can assign the multiple links to be parsed to several slave threads which will write the new links to the frontier. Only the master thread is given read access to the frontier.

2. The only external library used was JSoup. This was used to parse the documents which were in html format. By looking for the "a href" html tag, outgoing links were accumulated and added to the frontier. Other Java libraries have been used for parsing data, performing I/O, creating data structures etc.

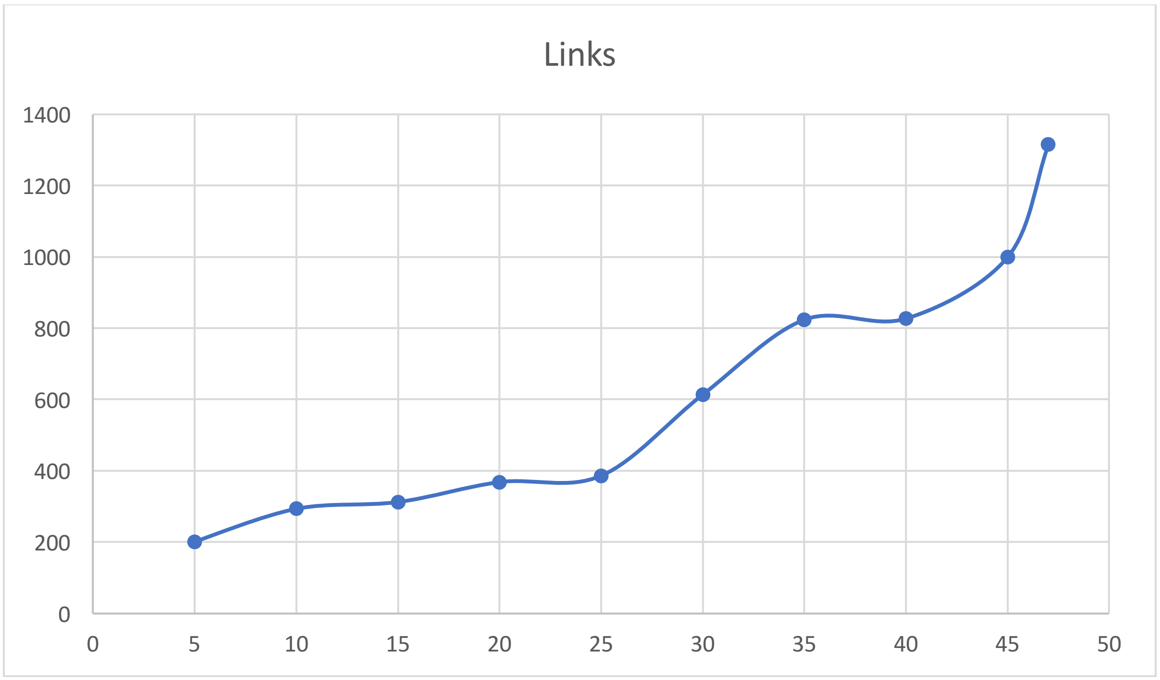
3. I initially checked if the domain had a robots.txt file. If so, I parsed the file line-by-line and using the BufferedReader library. Split the line on observing a "Disallow:" which would give me the url not to visit. I kept a record of the domain and only added the path of the file to the domain to keep a list of urls not to be visited.

   For part A I found 2 robots.txt files and for part B: 12.
   Yes, the robots.txt file in part B caused problems because many paths were unavailable for crawling due to the preferences in robots.txt.

4. Another page which might break my assumptions is a page which only has links to the deep web. These links would be a part of the deep web and hence a different protocol (protocol followed in tor browser) must be used to locate these such pages and links.

5.

6. The shape of the graph looks like:



X axis: Documents
Y axis: Links

The shape of the graph looks like this because the number of links increases rapidly even though few documents from the frontier are actually parsed i.e. increase exponentially.