Google Search Engine Simulation

By: Calvin Luong

Design and Implementation:

To begin with, I had to create classes that accessed the google search, hold the PageRank score and website url, perform the heap sort algorithm, and the simulator.

For the class that has to access the internet, it was provided by Dr. Wu from San Jose State University. This class was able to take a keyword and search for results relating to the keyword. The results would be stored to be later used.

The heap sort algorithm class was based off of the pseudo code that is provided by the textbook used in the CS 146 class. This class was based on the Max-Heap property, allowed more nodes to be added into the heap, take out the root, increase a node's key, sort a specific node in the heap, and sort an Array using the heap sorting algorithm.

The PageRank class had to contain the four factors (frequency of the keyword that appears on the website, age of the site, number of links that point to the site, and the amount the user paid for the site to appear on the search engine) and the four factors then had to sum up to create the PageRank score. The class contained two different constructor, one that generates random variables between 1 to 100 for each of the four factors and a second one that allows custom inputs for each of the four factors. Both of these constructors would then calculate the sum and that resulting sum would be the PageRank score. The class also contained a sum method to allow recalculations of the PageRank score if the user wanted to pay more money for their website.

The Website class had to be able to hold both the website url and the PageRank score. A constructor was created to do that so an ArrayList of the Website class could created to link the url and PageRank score together. The class would be able to return the PageRank's four factors

and website url using getter methods. The class contains a method that would be able to increase the amount of money paid for the website. This method would cap out at the value of 100 no matter how much money is added in.

The simulator class would take all these other classes and put them together. The user interface will prompt the user to look for a keyword. Once a keyword is entered, the class that accesses the internet would look up the keyword and place it into a set. The 30 urls in the set would be placed into a Website ArrayList to be printed out to show the user the list of urls and its PageRank score.

The next step would be sorting the websites based off of their PageRank score, however this step would not execute until the user inputs the right command. I would have to create a copy of just the PageRank scores to be sorted using the heap sorting algorithm. The copy of PageRank scores need to be inverted since the heap sort algorithm sorts the elements from smallest to largest. Once sorted, I would take the first element in the sorted copy of PageRank scores and look at the Website ArrayList for the matching url with the same PageRank score. That Website gets removed from the Website ArrayList and added into the final sorted Website ArrayList. The process would repeat all again with the next element in the sorted copy of PageRank scores.

The next step would be showing the user a website's information from the sorted list, this would be done by letting the user enter the number that the Website is on the list. For example: 1. www.google.com PageRank score: 400. The user would enter "1" to see the information of that website. After entering the website the user would want to see, the name of the website and its PageRank score would be printed out. The four factors would be printed after using the getter

methods created in the Website class. If a user enters a number that is greater than 30 will be capped at 30 and any number less than 1 will jump to the next step. This process would keep looping until the user enters "-1" to exit and perform the process of putting the websites into Heap. When the websites are being placed into Heap, only the first top 20 will be placed in. From then on, the user will be given a series of different commands that each do a specific thing.

Entering "a" would allow a user to create their own Website with custom inputs for each factor. That is where the custom input PageRank constructor would come into play. A new Website and PageRank object would be created and added into the Heap using the Max Heap Insert method in the heapSort class.

Entering "top" would show the user the Website with the highest PageRank score and remove it from the Heap using Heap Extract Maximum. All of the website's information will be shown one last time before it is removed.

Entering "p" would allow the user to pick a Website and increase its PageRank through the money factor. The user would be asked the which position their Website is at. Once the Website is selected, the code will show the remaining amount of money they can add until the max capacity of 100 is reached. If the user adds more than than the capacity, it will automatically cap out at 100. The PageRank score would then be calculated using the PageRank class's sum method.

Entering "quit" would exit out of the entire simulator all together.

Any changes to the Heap, like adding, removing, and paying more, will rebuild to Heap to ensure it keeps Max Heap property.

List of classes/subroutines/function calls:

1) **WebCrawler**: this class was provided by Dr. Wu from San Jose State University. The class allows key words to be searched on the internet by using Google's search engine. The results would be stored into a set to be later accessed.
    a) WebCrawler(String aKeyword): the constructor for the WebCrawler class that enters the aKeyword into Google's search engine.
    b) search(): does the actual searching of the key word. Places the resulting urls into a set.
    c) getDomainName(String url): returns the website url.
    d) Set<String> getUrls(): returns a set containing the urls after the keyword is searched.
    e) crawl(String url): loops the resulting links and places it into a set.
    f) searchForWord(String searchWord): checks if the website contains the keyword.
2) **heapSort**: this class performed the heap sort algorithm for an integer Array.
    a) getParent(**int** i): gets the target node's parent.
    b) getLeftChild(**int** i): gets the target node's left child.
    c) getRightChild(**int** i): gets the target node's right child.
    d) maxHeapify(**int** A[], **int** i): maintains the max heap property for the specific node.
    e) buildMaxHeap(**int** A[]): converts an Array A into a max heap property.
    f) heapSort(**int** A[]): sorts the heap to maintain max heap property using the heap sort algorithm.
    g) maxHeapInsert(**int** A[], **int** key): inserts element key into the Array.

h) heapExtractMaximum(**int** A[]): removes the largest element in the Array.

i) heapIncreaseKey(**int** A[], **int** i, **int** key): picks a specific element and increases its value.

j) heapMaximum(**int** A[]): returns the element at the zero index in the Array.

3) **Website**: this class acted as an object for a single website. The object holds the main website's url, the website's PageRank score, and the four factors that determines the website's PageRank score.

a) Website(String website, pageRank pageRank): Creates a Website constructor for Website class that has the website's url and its PageRank score.

b) getPageRank(): returns the PageRank.

c) getScore(): returns the PageRank's score.

d) getFrequency(): returns the number of times the key word appears on the website.

e) getAge(): returns the age of the website.

f) getNumOfLinks(): returns the number of links pointing to the website.

g) getMoney(): returns the amount the creator paid for the website to be shown on the search engine.

h) getWebsite(): returns the website's url.

i) addMoney(**int** money): Adds to the amount paid for the website. The total amount of money paid will cap out at 100. The website's PageRank score will be recalculated.

4) **pageRank**: this class acted as an object for the PageRank score and the four factors that add up to the PageRank score.

a) pageRank(): creates a pageRank constructor that assigns random numbers between 1 to 100 for each of the four factors (frequency of key word, age, number of links pointing to the website, and money paid for the site to appear on the search engine).

b) pageRank(**int** frequencyOfWord, **int** age, **int** numOfLinks, **int** money): creates a pageRank constructor that allows custom inputs for each factor.

c) getSum(): Recalculates the PageRank score.

5) **GoogleSearchEngine**: this class performed the Google Search Engine simulation in Eclipse.

a) createPageRanks(): creates a list of 30 PageRank scores.

b) createWebsiteList(): searches using Google search engine to look for sites relating to the user's keyword. Then creates a list of 30 urls that each have a unique PageRank score.

c) invertArray(**int** A[]): inverts the order of the Array.

d) sort(): sorts the PageRank score of each website by storing the PageRank scores into pageRankList and then sorting it using heap sort. The end result will be a sorted Array that goes from biggest value to lowest value.

e) getSortedWebsites(): links the PageRank score with the websites by looking for the same PageRanks in pageRankList and websiteList and storing the website into sortedURL.

f) getInfo(**int** i): prints out the website url, the PageRank score, and the four factors that make the score.

g) priorityQueue(): adds the first 20 sorted websites into a heap.

h) createWebsite(String URL, **int** frequencyOfWord, **int** age, **int** numOfLinks, **int** money): creates a Website object with custom factors.

i) insertInHeap(Website w): inserts a Website into the heap.

j) rankOneSite(): takes out the highest PageRank score website out of the heap and shows the user.

k) setRank(**int** i, **int** money): allows the user to pick a Website and increase the PageRank score by adding money.

l) getHeap(): Prints out the heap.

Self-testing Screenshots

Part 1:

```
----------------------------------
What would you like to search up?
----------------------------------
Input: San Jose
**Visiting** Received web page at https://google.com/search?q=san&num=80
Found (285) links
Searching for the word san...
**Success** Word san found at https://google.com/search?q=san&num=80
**Done** Visited 94 web page(s)

Here are the first 30 URL links:

1. money.cnn.com, PageRank Score: 111
2. www.san.org, PageRank Score: 132
3. radar.weather.gov, PageRank Score: 193
4. sanfrancisco.cbslocal.com, PageRank Score: 258
5. therealdeal.com, PageRank Score: 160
6. www.axs.com, PageRank Score: 230
7. www1.ticketmaster.com, PageRank Score: 301
8. www.stubhub.com, PageRank Score: 244
9. en.wikipedia.org, PageRank Score: 270
10. www.sdhumane.org, PageRank Score: 358
11. en.wikipedia.org, PageRank Score: 199
12. sfgov.org, PageRank Score: 130
13. www.comic-con.org, PageRank Score: 295
14. www.cnn.com, PageRank Score: 135
15. www.google.com, PageRank Score: 236
16. www.webopedia.com, PageRank Score: 85
17. www.sdzsafaripark.org, PageRank Score: 202
18. www.sjsu.edu, PageRank Score: 176
19. en.wikipedia.org, PageRank Score: 241
20. hoodline.com, PageRank Score: 203
21. www.flysfo.com, PageRank Score: 251
22. www.ticketfly.com, PageRank Score: 242
23. www.nfl.com, PageRank Score: 238
24. en.wikipedia.org, PageRank Score: 150
25. www.sfsymphony.org, PageRank Score: 233
26. www.nbcsandiego.com, PageRank Score: 304
27. www.pcmag.com, PageRank Score: 256
28. www.nbcbayarea.com, PageRank Score: 198
29. finance.yahoo.com, PageRank Score: 178
30. www.sandiegouniontribune.com, PageRank Score: 125
--------------------------
Please enter 's' to sort.
--------------------------
Input:
```

```
24. en.wikipedia.org, PageRank Score: 150
25. www.sfsymphony.org, PageRank Score: 233
26. www.nbcsandiego.com, PageRank Score: 304
27. www.pcmag.com, PageRank Score: 256
28. www.nbcbayarea.com, PageRank Score: 198
29. finance.yahoo.com, PageRank Score: 178
30. www.sandiegouniontribune.com, PageRank Score: 125
-------------------------
Please enter 's' to sort.
-------------------------
Input: s
1. www.sdhumane.org, PageRank Score: 358
2. www.nbcsandiego.com, PageRank Score: 304
3. www1.ticketmaster.com, PageRank Score: 301
4. www.comic-con.org, PageRank Score: 295
5. en.wikipedia.org, PageRank Score: 270
6. sanfrancisco.cbslocal.com, PageRank Score: 258
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. www.ticketfly.com, PageRank Score: 242
11. en.wikipedia.org, PageRank Score: 241
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
21. finance.yahoo.com, PageRank Score: 178
22. www.sjsu.edu, PageRank Score: 176
23. therealdeal.com, PageRank Score: 160
24. en.wikipedia.org, PageRank Score: 150
25. www.cnn.com, PageRank Score: 135
26. www.san.org, PageRank Score: 132
27. sfgov.org, PageRank Score: 130
28. www.sandiegouniontribune.com, PageRank Score: 125
29. money.cnn.com, PageRank Score: 111
30. www.webopedia.com, PageRank Score: 85
------------------------------------------------------------------------
To get info of a specific site, enter the number of the site .
Any number greater than 30 will be capped at 30.
Any number less than 1 will skip the step.
If you would like to skip this step and perform heap sort, enter '-1'
------------------------------------------------------------------------
Input:
```

```
25. www.cnn.com, PageRank Score: 133
26. www.san.org, PageRank Score: 132
27. sfgov.org, PageRank Score: 130
28. www.sandiegouniontribune.com, PageRank Score: 125
29. money.cnn.com, PageRank Score: 111
30. www.webopedia.com, PageRank Score: 85
--------------------------------------------------------------------------
To get info of a specific site, enter the number of the site .
Any number greater than 30 will be capped at 30.
Any number less than 1 will skip the step.
If you would like to skip this step and perform heap sort, enter '-1'
--------------------------------------------------------------------------
Input: 1
Page is: www.sdhumane.org. With a rank score of: 358
Frequency of key word on the page: 84
Age of the page: 100
Number of link pointing to this page: 82
Money: 92
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input: 10
Page is: www.ticketfly.com. With a rank score of: 242
Frequency of key word on the page: 45
Age of the page: 32
Number of link pointing to this page: 77
Money: 88
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input: 20
Page is: radar.weather.gov. With a rank score of: 193
Frequency of key word on the page: 52
Age of the page: 43
Number of link pointing to this page: 88
Money: 10
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input: 30
Page is: www.webopedia.com. With a rank score of: 85
Frequency of key word on the page: 23
Age of the page: 46
Number of link pointing to this page: 14
Money: 2
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input:
```

Part 2:

```
Page is: radar.weather.gov. With a rank score of: 193
Frequency of key word on the page: 52
Age of the page: 43
Number of link pointing to this page: 88
Money: 10
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input: 30
Page is: www.webopedia.com. With a rank score of: 85
Frequency of key word on the page: 23
Age of the page: 46
Number of link pointing to this page: 14
Money: 2
--------------------------------------------------
Next site or perform heap sort by entering in '-1'?
--------------------------------------------------
Input: -1
------------------------------------------
Here is the first 20 links put into a heap:
------------------------------------------
1. www.sdhumane.org, PageRank Score: 358
2. www.nbcsandiego.com, PageRank Score: 304
3. www1.ticketmaster.com, PageRank Score: 301
4. www.comic-con.org, PageRank Score: 295
5. en.wikipedia.org, PageRank Score: 270
6. sanfrancisco.cbslocal.com, PageRank Score: 258
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. www.ticketfly.com, PageRank Score: 242
11. en.wikipedia.org, PageRank Score: 241
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
--------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
--------------------------------------------------
Input:
```

```
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
--------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
--------------------------------------------------------
Input: a
You will now be asked to input your website url.
Please enter the a website: www.calvinluong.com
You will now be asked to input four factors for you websites, the values range from 1 - 100 for each factor.
Inputs out of bounds will automatically be capped at the lowest or highest value.
Please enter the number of times the main key word for your site will show up: 100
Please enter how long your site has been active since creation: 100
Please enter the number of links that currently refer to your site: 100
Please enter the amount you would like to pay: 100
1. www.calvinluong.com, PageRank Score: 400
2. www.sdhumane.org, PageRank Score: 358
3. www.nbcsandiego.com, PageRank Score: 304
4. www.comic-con.org, PageRank Score: 295
5. en.wikipedia.org, PageRank Score: 270
6. www1.ticketmaster.com, PageRank Score: 301
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. www.ticketfly.com, PageRank Score: 242
11. sanfrancisco.cbslocal.com, PageRank Score: 258
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
21. en.wikipedia.org, PageRank Score: 241
--------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
--------------------------------------------------------
Input:
```

```
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
21. en.wikipedia.org, PageRank Score: 241
--------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
--------------------------------------------------------
Input: top
**REMOVING**
Page extracted is: www.calvinluong.com. With a rank score of: 400
Frequency of key word on the page: 100
Age of the page: 100
Number of link pointing to this page: 100
Money: 100
1. www.sdhumane.org, PageRank Score: 358
2. www.nbcsandiego.com, PageRank Score: 304
3. www1.ticketmaster.com, PageRank Score: 301
4. www.comic-con.org, PageRank Score: 295
5. en.wikipedia.org, PageRank Score: 270
6. sanfrancisco.cbslocal.com, PageRank Score: 258
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. www.ticketfly.com, PageRank Score: 242
11. en.wikipedia.org, PageRank Score: 241
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
--------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
--------------------------------------------------------
Input:
```

```
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
----------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
----------------------------------------------------------
Input: p
Please enter the list number the website is at:
20
The amount remaining that you can add is: 90
The amount you want to add:
90
1. www.sdhumane.org, PageRank Score: 358
2. www.nbcsandiego.com, PageRank Score: 304
3. www1.ticketmaster.com, PageRank Score: 301
4. www.comic-con.org, PageRank Score: 295
5. radar.weather.gov, PageRank Score: 283
6. sanfrancisco.cbslocal.com, PageRank Score: 258
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. en.wikipedia.org, PageRank Score: 270
11. en.wikipedia.org, PageRank Score: 241
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. www.ticketfly.com, PageRank Score: 242
----------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
----------------------------------------------------------
Input:
```

```
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. radar.weather.gov, PageRank Score: 193
----------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
----------------------------------------------------------
Input: p
Please enter the list number the website is at:
20
The amount remaining that you can add is: 90
The amount you want to add:
90
1. www.sdhumane.org, PageRank Score: 358
2. www.nbcsandiego.com, PageRank Score: 304
3. www1.ticketmaster.com, PageRank Score: 301
4. www.comic-con.org, PageRank Score: 295
5. radar.weather.gov, PageRank Score: 283
6. sanfrancisco.cbslocal.com, PageRank Score: 258
7. www.pcmag.com, PageRank Score: 256
8. www.flysfo.com, PageRank Score: 251
9. www.stubhub.com, PageRank Score: 244
10. en.wikipedia.org, PageRank Score: 270
11. en.wikipedia.org, PageRank Score: 241
12. www.nfl.com, PageRank Score: 238
13. www.google.com, PageRank Score: 236
14. www.sfsymphony.org, PageRank Score: 233
15. www.axs.com, PageRank Score: 230
16. hoodline.com, PageRank Score: 203
17. www.sdzsafaripark.org, PageRank Score: 202
18. en.wikipedia.org, PageRank Score: 199
19. www.nbcbayarea.com, PageRank Score: 198
20. www.ticketfly.com, PageRank Score: 242
----------------------------------------------------------
Enter 'a' to add a website.
Enter 'top' to see the number one site and remove it.
Enter 'p' to pay more money for a website.
Enter 'quit' to leave.
----------------------------------------------------------
Input: quit
Thank you for using!
```

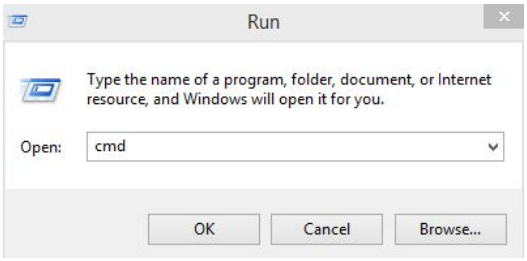
Step by step installation

1) Download the GoogleSearchEngine.jar.

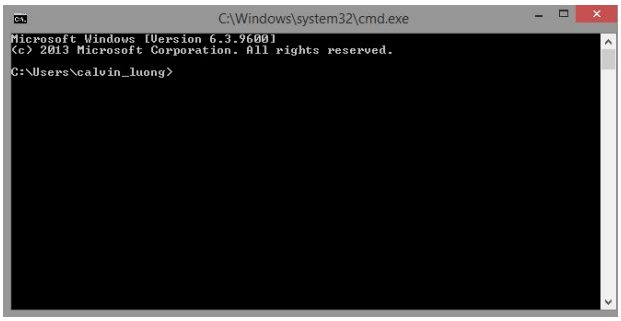
2) For Windows users:

   a) Right click the Windows icon on the lower left corner, and click on run.

   b) Type in “cmd” and click Ok.



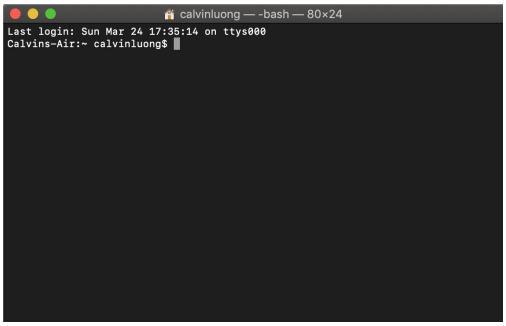
   c)



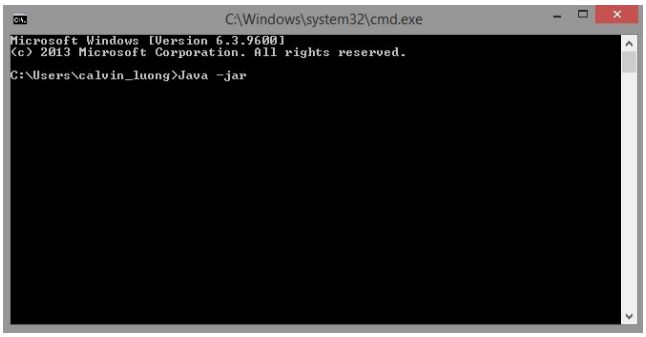
3) For MacOS users:

    a) Press Command and Spacebar together.

    b) In the Spotlight Search, look up “terminal”

    c) 

4) Type in “Java -jar ”. Make sure it is the exact format.



5) Drag that file into Command Prompt or Terminal.

6) Press Enter and the program will start running.

Problems encountered during the implementation

During the implementation process, I came across several problems. The first issue was creating the heap sort algorithm that was based off of the textbook's pseudo code. The pseudo code starts the index at 1 while Java starts at 0. With this in mind, I had to constantly make sure that my code started at index at 0. There were several times when I went to test my heap sort and it would not work due to an out of bound error caused by when the index started at 1.

The second issue was dealing with duplicate PageRank scores. I did not realize that there were four factors to creating a PageRank score. Therefore, instead of creating a random integer for the PageRank score, I created random integers for each of the factors. This will keep the score unique as the score can be the same but the factors that make it up will be different.

The third issue was figuring out how to combine a website's url and PageRank score together during the sorting process. The idea was to create an Array that would have each element be both the url and the PageRank score. I would make a copy of just the PageRank score and sort it. Finally, I had to connect the sorted PageRank scores with the main Array of urls and PageRank score. I did this by taking taking the first PageRank score from the sorted Array and finding the corresponding PageRank score in the initial list of urls and PageRank scores. Once found, I would add it into a new final Array of sorted urls and their PageRank scores. The issue was that I had no way to deal with urls with the same exact PageRank score. To solve this, I removed the website from the Array so two urls of the same PageRank scores can be found. Another issue was that null elements in the empty index of the removed website would appear. I had to change the Arrays into ArrayList. Therefore, every time I removed the website, all the websites after it would shift down one index to fill in the null space.

The fifth issue was taking an input when asking the user to enter the command to sort the list of websites. It would always repeat the question twice before allowing the user to input the command. To solve this, I did a quick fix of creating a new Scanner object and using that to intake a user input.

The sixth issue was that the WebCrawler class would stop working. This was a product of constantly searching the same keyword repeatedly. The actual Google search engine would shut off the searching function for my ip address for a limited amount of time. To combat this, I had to either use a VPN or search up a different keyword every time I want to test my code. This would prevent the error from occurring.

Lessons Learned

In this programming assignment, I have learned about how Google's search engine works on a small scale. There are a variety of factors that goes into determining what to show the user. Creators are able to pay Google to have their site on a higher priority. I was able to create a simple user interface that was able to keep allowing the user to make changes to the list of website. In this interface, I was able to implement mediocre error detections. I also learned to create a semi-professional report that details my code. This is important as I have to create an extremely detailed report that explains the entirety of my code. It has to be detailed enough for a person to look at it and instantly understand what my code does.

The most important lesson I learned was that it is best to start the assignment as early as possible. The reason being that creating the design for the code can already take a while. There is no exact solution to this coding assignment. Also, since the code is based off of our own creativity, it is bound to have errors every step of the way. It took me a total of 5 days to

complete the code as I spent a couple hours every day. This drastically lowered the workload as I did not have to try as hard when it approached the deadline.