

Homework 1

Google Search Engine Simulator

James Nguyen

I. Introduction

Usually when one wants to search an aspect across the internet, the first idea that appears in their mind is to search on 'Google'. 'Google' is one of the top visited sites across the country and the company's search engine plays a large role within the worldwide web, and is used by millions of people today. With its efficient algorithm to sort and rearrange links, the search engine can effectively find links that users are looking for in an instance.

This assignment will efficiently show the background of what exactly is occurring within the search engine when a user enters a search onto Google. The objective of this assignment can be divided into two main parts. The first one is to build a smaller scale search engine like of Google's and to allow users to view the thirty links result, and the different aspects of the links. The second one is to create a separate queue which contains the twenty links of the thirty links result in which users may be able to edit and view. The assignment will be running through the Java since it was originally coded through the Java language.

II. Design and Implementation

This section will go over the design and implementation of the Google Simulator, and how the design will effectively be implemented as code.

A. Searching for Links with the Keyword

To start of the Google Simulator, we must first find a way to get links from the internet which contains the keywords that a user searched for. In the Google Search Engine, a user usually first searches for a keyword and then links are given back to them that contains the keyword. In order to implement this feature, the program will be using a Web Crawler. The Crawler will be using a Google Bot to crawl through several links and return the links which contains the keyword the user has searched for.

B. Page Rank

To put the links in order, we must first find an aspect that we can sort them based on. In Google, the links are sorted and placed in the search engine web result based on the Page Rank, a score that is given based on several factors about the website that contains the keyword such as the frequency of the word. In order to implement this feature, we will be calculating the Page Rank on four main factors:

- 1) the frequency and location of the specified keyword within the webpage
- 2) how long the webpage has existed
- 3) the number of other webpages which links toward this webpage
- 4) how much the webpage owner has paid the search Engine for advertisement purposes

Since we are unable to determine the realistic ranks of the four factors for the Page Rank because factors such as the paid factors are unable or difficult to be found, we will be using a random number generator to compute a number between one and a hundred for each of the four factors, and then computing the sum of the four different scores to determine the Page Rank of the specific link. Thus, we will then have something to sort the links based on.

C. Sorting Algorithm

Google has a very fast sorting algorithm to quickly sort the links and order them by their PageRank. In our Google simulation, we will be using the heap sort algorithm to sort the links. Heap Sort is a sorting algorithm in which the algorithm will take a list and create a max heap tree, that matches the Heap Property which states that the parent node must be greater than its child nodes, and then slowly decreases the heap size within the list and moves the largest node into the back of the heap such that the list will have the nodes in least to greatest order. The Heap Sort Algorithm also runs very quick, and sorts in place. It's runtime of $O(n \lg n)$ which means that it will efficiently sort in a small amount of time.

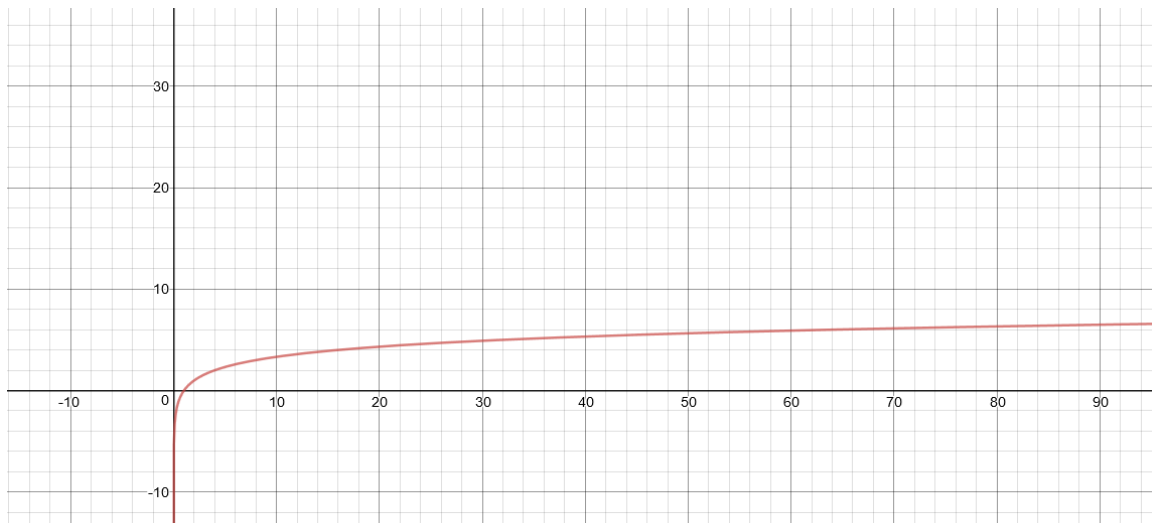


Image: Graph of $n \lg n$ provided by Desmos. It can be noticed how slowly it increases which means there is a small amount of time.

D. Heap Queues

The second objective of the programming assignment is to have a queue in which users are able to edit and essentially use. This is because in the Google Search Engine, users are able to edit their status within a search result page by paying Google. Since we are already using the Heap Sort Algorithm which creates heaps, we can use the methods within the algorithm to just build a queue known as the Heap Priority Queue, in which the largest node will always be the first node coming out of the queue. Furthermore with the Heap Priority Queue, users will be able edit the queue through several methods that will be coded such as buildMaxHeap.

E. Data Structure

Throughout the programming assignment, there will be a data structure that is needed since there requires a data structure to hold the different links for the list and Queue. In this assignment, arrays as the main data structure. There will be two main arrays in which one is for the list of links, and another one for the queue of links.

F. User Interface

The two parts of the objective of the assignment requires for user to be able to view the features within the Link List and edit the Link Queue. As a result, a user interface must be formed in order to allow for users to choose options to either view the Link List and Link Queue or edit the Link Queue. Implementing a pop-up menu in this case would suffice for the requirements with seven main different options:

- 1) View the 30 links
- 2) View the Four Factor Rank Score of each Link
- 3) View the 30 Web Links In Order based on its Page Rank
- 4) View the Links within the Queue
- 5) Increase the Page Rank of a Link within the Queue
- 6) View the Top Link of the Queue
- 7) Insert a Web Link within the Queue

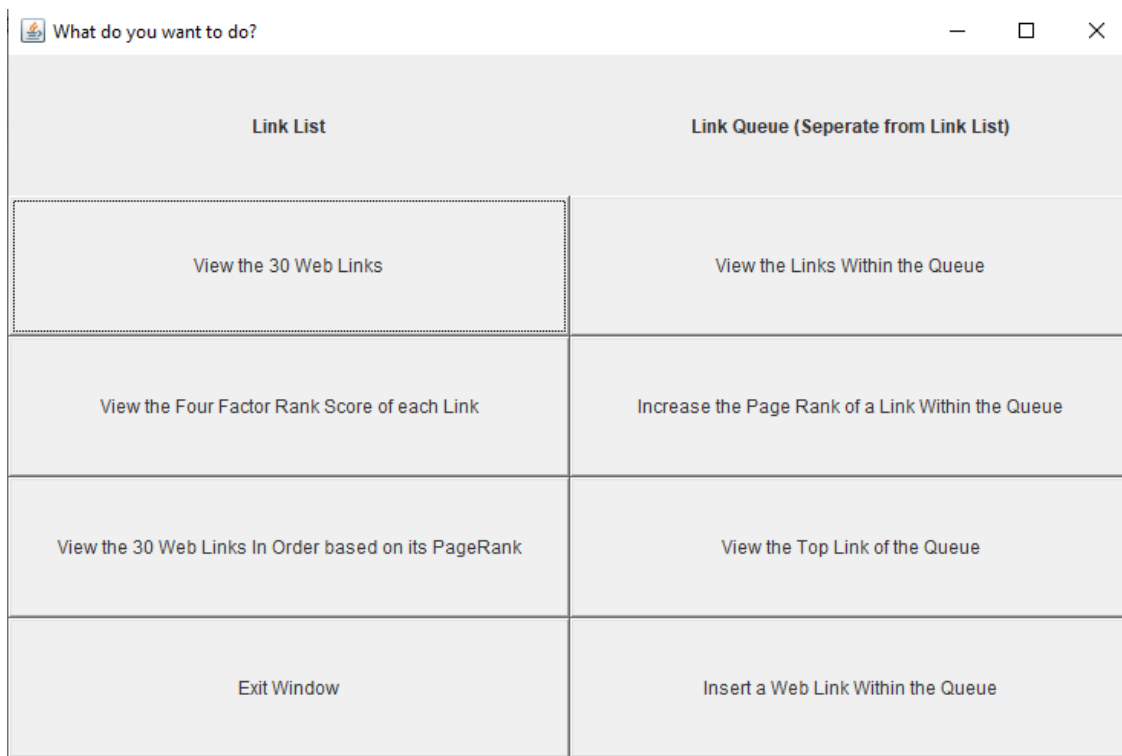


Image: Implementation of menu with different options for users to choose from.

The menu options will efficiently allow users to view the Link List and Link Queue, while being able to edit anything within the Queue which fits exactly with the objective of the programming assignment.

III. List of classes /subroutines/function calls

This section will go over the different classes as well as the different methods that are used within each class.

A. Web Crawler Class

This class will allow one to search the web and crawl across the internet using the Google Bot to find links that match for a specific keyword.

- 1) searches– This method within the class searches the keyword that was passed in the constructor throughout the web and puts the link that the Google Bot finds into the urls list of the class.

B. URL Class

The URL Class allows users to create URL objects that will store the score for each of the Four Factors as well as the Page Rank or total of the Four Factor Scores.

- 1) Constructor – In order to create this object, the user must have each score for the four factors as well as have the string of the link.

```
public Url(String url, int freqNum, int existNum, int linkNum, int adverNum) {  
    this.url = url;  
    this.freqNum = freqNum;  
    this.existNum = existNum;  
    this.linkNum = linkNum;  
    this.adverNum = adverNum;  
    pageRank= freqNum + existNum +linkNum + adverNum;  
}
```

Image: Constructor of the Url Class

- 2) getTotal, getUrl, getFreqNum, getexistNum, getlinkNum, getadverNum – These getter methods allow users to pull information about the Url object such as the Page Rank, Four Factor Score, and the string of the link.
- 3) setPageRank- The setter method allows for users to set the Page Rank of the URL object. This method will be effectively be used when a User wants to insert a URL in the Link Queue.

C. Heap Editor Class

The Heap Editor Class has several methods which build heaps queues, edit the queue, or sort a specific list. If a user were to make a Heap object from this class and creates a heap priority queue in the list using the Heap object, they must use the same heap object to edit the queue within the list.

- 1) maxHeapify – This method takes in a list of URLS and a specified index of a node, in this case a URL, and rearranges that URL node such that it will match the heap property for that specified URL node where the URL's page rank is larger.
- 2) buildMaxHeap- This method takes in a list of URL and creates a max heap by calling maxHeapfiy on every single parent URL node within the heap so that every parent URL node's Page Rank is larger than it's children. This can be effectively used as a Heap Priority Queue.

- 3) `getTrueLength` - This method takes in an array and returns the true length of that array, meaning that it counts the number of spaces in the array that aren't null.
- 4) `heapSorter` – This method will use `buildMaxHeap` and `maxHeapify` to sort any URL list that is passed through the method and place them from least to greatest. The method does this by first building a max heap, and then decrementing the heap size and calling `maxHeapify` towards the first node, as the first node or the greatest node of the heap is swapped with the last node of the heap until the heap size only has one node which would be the lowest node.

```
public void heapSorter(Url[] searchList) {
    buildMaxHeap(searchList); // creates a heap from the array, and sorts the heap
    for (int i = getTrueLength(searchList) - 1; i > 0; i--) { // takes the first number in the heap and adds it to
                                                                // the end
        // of the array until, the array is from least to greatest
        Url correctUrl = searchList[0];
        searchList[0] = searchList[i];
        searchList[i] = correctUrl;
        heapSize--;
        maxHeapify(searchList, 0);
    }
}
```

Image: heapSorter algorithm

- 5) `heapMaximum` – This method can be used on a Heap Priority Queue that is passed into the method and return the first URL in the Queue.
- 6) `heapExtractMax`- This method will extract the first URL within the Heap Priority Queue that is passed into the method and return it to the user. It will then update the Heap Priority Queue by moving the last URL to the front of the heap, decrementing the heap, and then calling the `maxHeapify` method onto the first URL in the heap to make the heap a Max Heap again.
- 7) `heapIncreaseKey` – This method will take an index of a node, a Max Heap, and a key in which the URL node on the index will have. The method will first change the PageRank of the URL to the specified value which needs to be larger than the original PageRank or an Exception will be thrown. Then, the method will rearrange the URL node such that it's parent node's Page Rank is larger than it.
- 8) `heapInsert`- This method will add a URL into the Heap Priority Queue that is passed into the method. The method will first make a URL with a Page Rank of 0, the smallest value possible for a Page Rank, and then add it to the bottom of the Heap. The method will then call `heapIncreaseKey` to increase the URL to the true value of the URL and move it to the correct place in the Queue.

D. Google Simulation

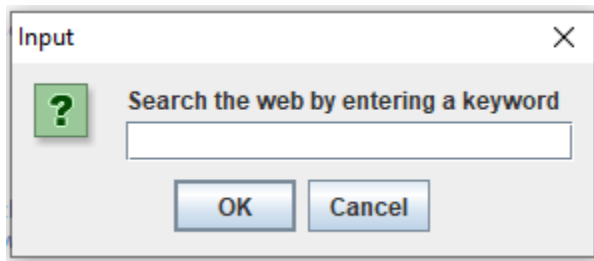
This class creates a Google Simulator in which users are able to search a specific keyword, and the class will create a queue and list in which user are able to view the list, and edit the queue. The class extends to `JFrame` and implements `Action Listener` in order for the buttons in the window that it creates is able to work. This class will effectively also use all the methods of the `Heap Editor`, `URL`, and `Web Crawler Class`.

- 1) Constructor – The Constructor will first ask the user for an input of a keyword. Then it would use the searches method of the Web Crawler class to crawl the keyword. The list of links from the WebCrawler is then transferred into two arrays: one for the link list, and one for the link queue. The link queue is then turned into a max heap since it's a Heap Priority Queue. The user is then into a window with a menu of different buttons for the user to choose from and execute actions to edit the queue, or view aspects of the list and queue.
- 2) actionPerformed – This method causes actions to occur if a specific button is pressed since each button on the method passes an action command which this method will detect. There are many different actions that will occur for each different button:
 - a) “View the 30 links” – When this button is pressed, the command, “viewLink”, is sent out. As a result, the method will cause a window to pop up which will show the user the first thirty links that was found by the Web Crawler.
 - b) “View the Four Factor Rank Score of each Link” – When this button is pressed, the command, “viewScore”, is sent out. As a result of this, the method will open a window of all the 30 links. The method will ask the user to input a corresponding number of one of the links, and then display the four factors score of the link the user has chosen.
 - c) “View the 30 Web Links In Order based on its Page Rank” – When this button is pressed, the command, “viewRank”, is passed out. The method then reads this command and then first creates a copy of the array of links, and then calls heapSorter towards the copied array of links to sort the Urls from least to greatest. A window will then open with the Urls in reversed order in order to print the links from greatest to least. The Page Rank is also listed right next to each link.
 - d) “View the Links within the Queue” – When this button is pressed, the command “viewQueue” is sent out. The method then creates a window which has all the different links in the queue.
 - e) “Increase the Page Rank of a Link within the Queue”- When this button is pressed, the command “increase key” is sent out. The method then opens a window which contains the links within the queue and ask the user to input a number to choose a link. The method then ask the user to input the specified Page Rank. The method then calls heapIncreaseKey onto the queue array to increase the PageRank of the URL and update the Queue.
 - f) “View the Top Link of the Queue” – When this button is pressed , the command “viewTop” is sent out. The method, after receiving, the command, will call heapExtratMax() to get the top link within the queue, and then update the queue in order for it to remain a Max Heap. The return top link is then shown in another window for the user to see.
 - g) “Insert a Web Link within the Queue” – When this button is pressed, the command, “insert” , is sent out. The method will ask the user to input a link, and Page Rank score of the link they would like to add. The method then calls upon heapInsert to insert the link into the correct position within queue.

IV. Self -Testing Screen Shots

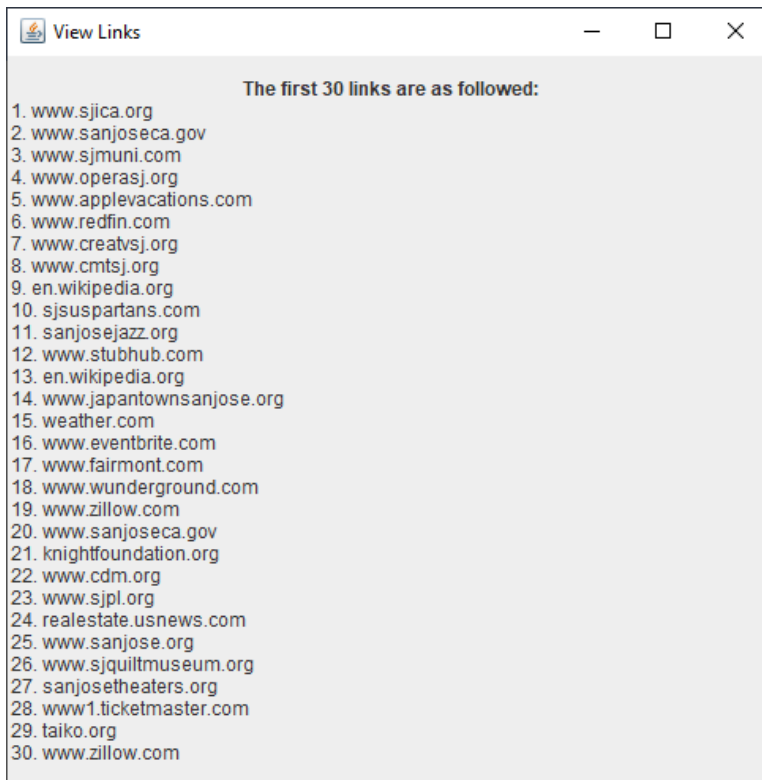
This section will display screenshots of testing the program to show that it has met the Functional Requirements of the programming assignment. In this case, we will be using San Jose as the keyword to test.

A. Allowing User to enter a keyword



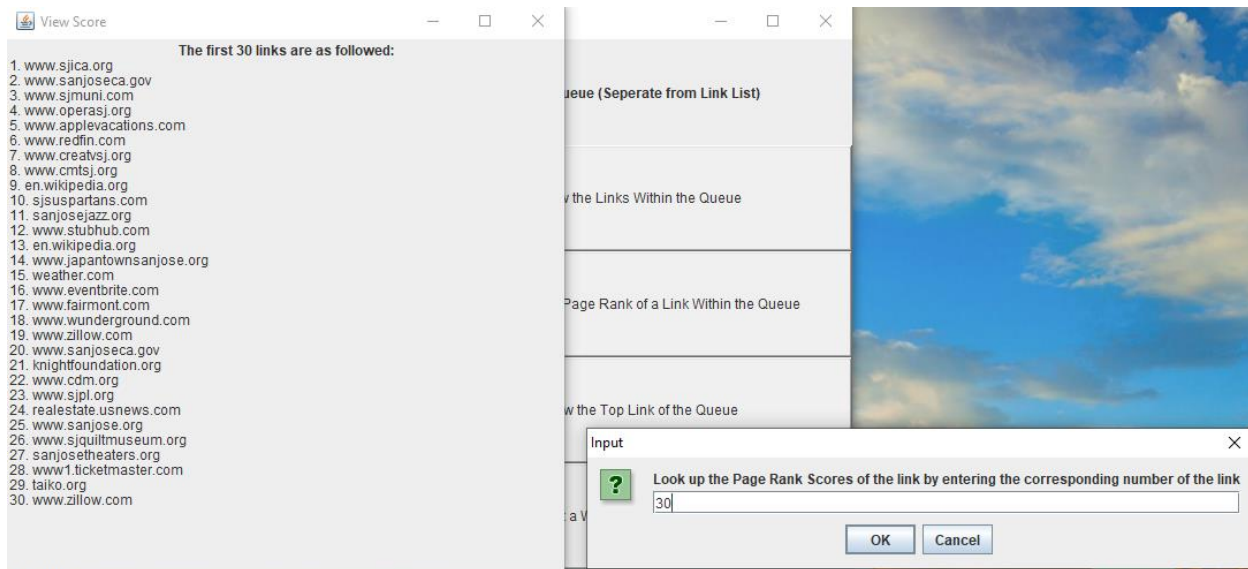
Users are able to enter a keyword into the text box.

B. Allowing users to display the search result containing at least 30 web url links

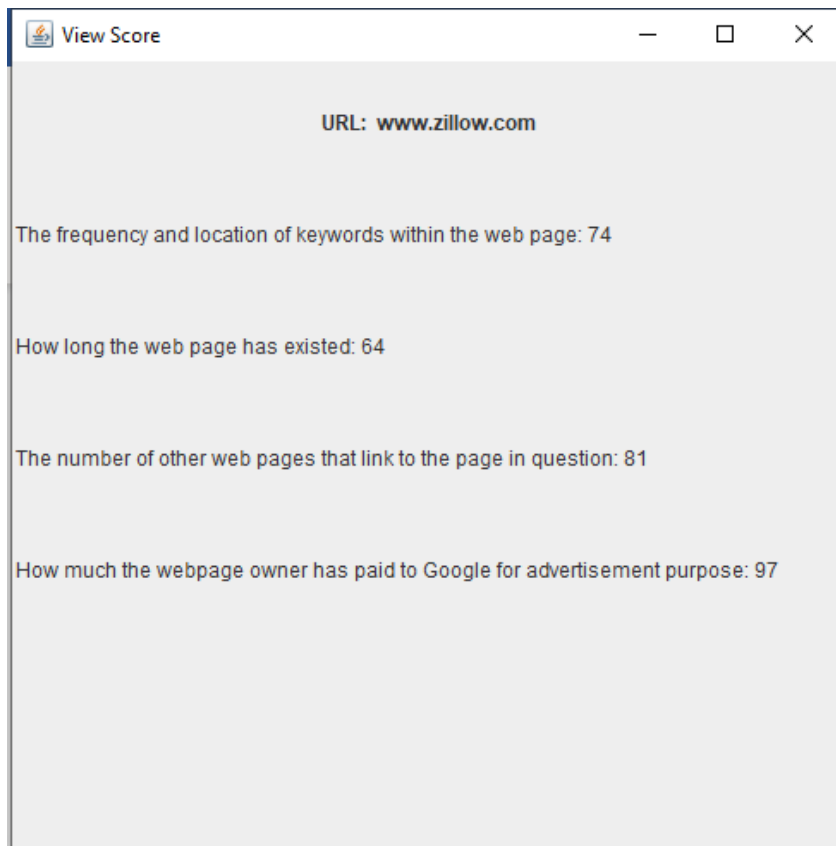


If the user clicks on the “View the 30 Web Links” button, they are able to see the 30 links of San Jose.

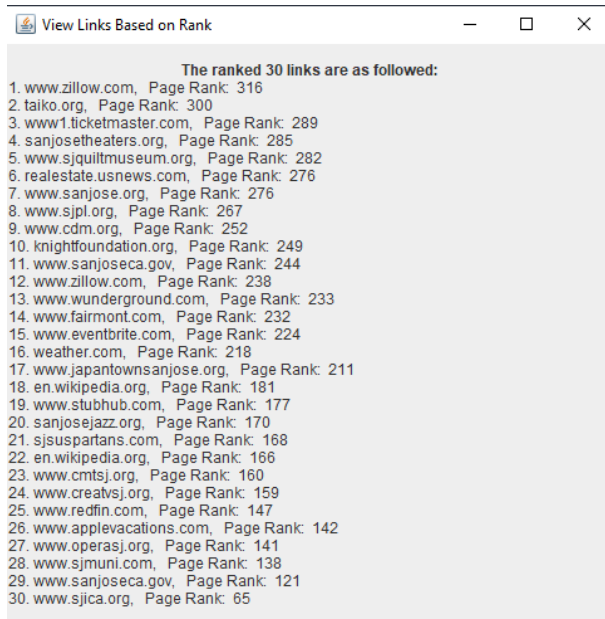
C. Providing users a user interface to view the 4 factor scores of each web url link



If the user clicks on the “View the Four Factor Rank Score of each Link”, they will see a similar window in which they can pick a link and view the 4-factor score. In this case, the user decides to view the 30th link.

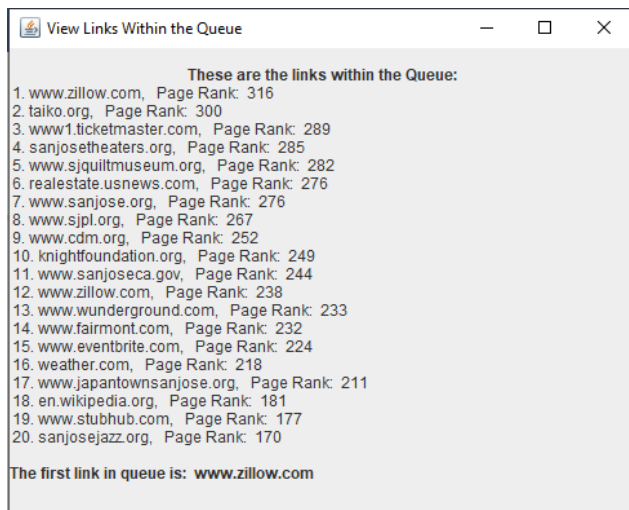


D. Allowing users to print out the sorting result in the sequence order based on the PageRank score



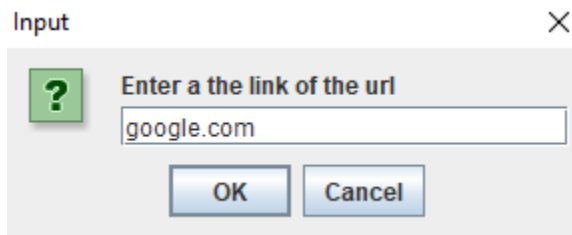
If the user clicks on the “View the 30 Web Links In Order based on its Page Rank” button, this window will pop up.

E. Implementing Heap Priority Queue to store the first sorted 20 of 30 web urls into the heap



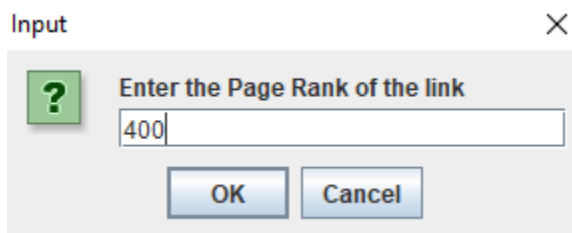
Shows the links within the queue are the first sort 20 of the 30 web urls. The Queue can be seen by pressing the “View the Links within the Queue” button.

F. Allowing users to insert a new web url link into the heap queue based on the Page Rank Score:



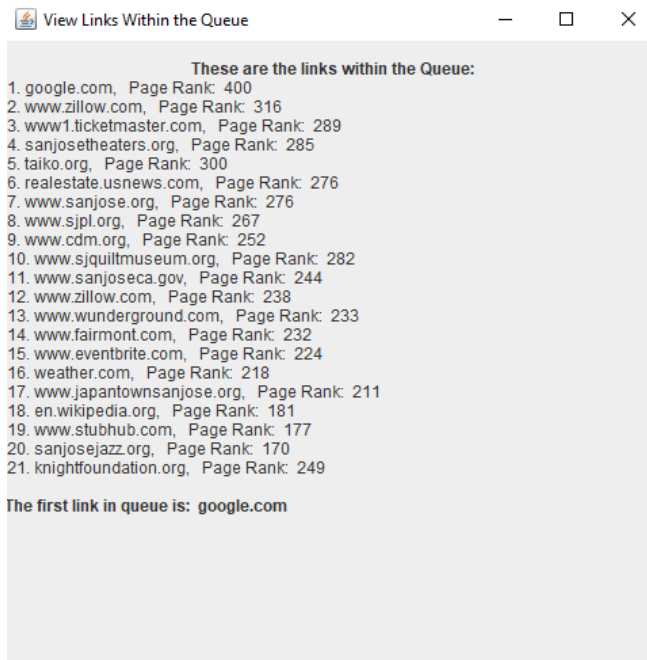
An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a white question mark. To the right of the icon is the text "Enter a the link of the url". Below this text is a text input field containing "google.com". At the bottom of the dialog are two buttons: "OK" and "Cancel".

Ask the user to input the link that they would like to add when they press the “Insert a Web Link within the Queue” button.



An input dialog box titled "Input" with a close button (X) in the top right corner. It contains a green square icon with a white question mark. To the right of the icon is the text "Enter the Page Rank of the link". Below this text is a text input field containing "400". At the bottom of the dialog are two buttons: "OK" and "Cancel".

It then asks the user to input the page rank of the link we would like to add. In this case we will use 400 to see if it will be sent to the front of the queue like it is supposed to.



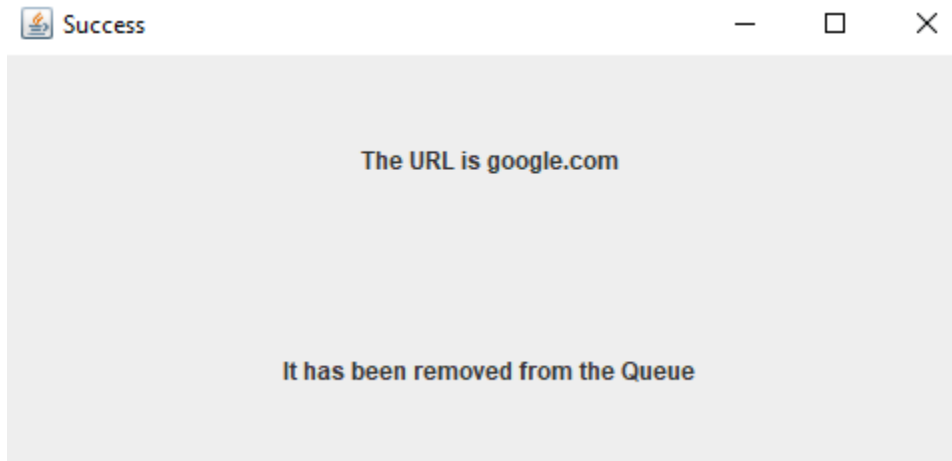
A dialog box titled "View Links Within the Queue" with a close button (X) in the top right corner. It contains a list of 21 links and their Page Ranks. The first link is "1. google.com, Page Rank: 400". Below the list is the text "The first link in queue is: google.com".

Rank	Link	Page Rank
1.	google.com	400
2.	www.zillow.com	316
3.	www1.ticketmaster.com	289
4.	sanjosetheaters.org	285
5.	taiko.org	300
6.	realestate.usnews.com	276
7.	www.sanjose.org	276
8.	www.sjpl.org	267
9.	www.cdm.org	252
10.	www.sjqilmmuseum.org	282
11.	www.sanjoseca.gov	244
12.	www.zillow.com	238
13.	www.wunderground.com	233
14.	www.fairmont.com	232
15.	www.eventbrite.com	224
16.	weather.com	218
17.	www.japantownsanjose.org	211
18.	en.wikipedia.org	181
19.	www.stubhub.com	177
20.	sanjosejazz.org	170
21.	knightfoundation.org	249

The Queue is the updated with google at the top of the list as the function uses maxHeapInsert.

G. Allowing user to view and extract the first ranked web url link from the queue

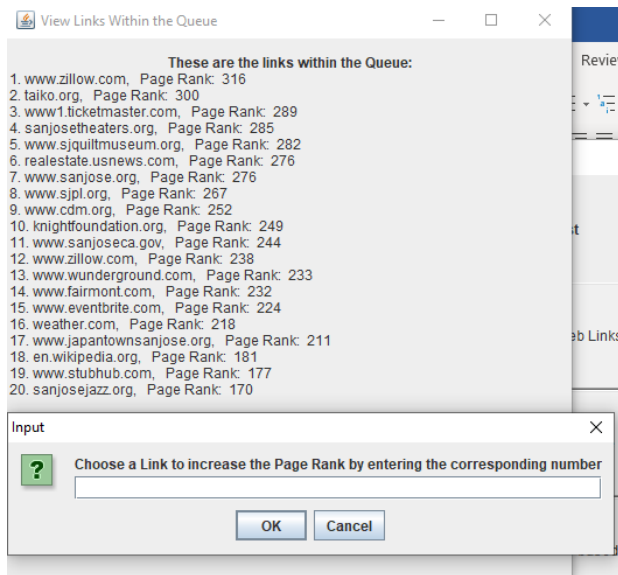
Since google.com is at the top now. When the user presses “View the Top Link of the Queue” button, it should return google.com.



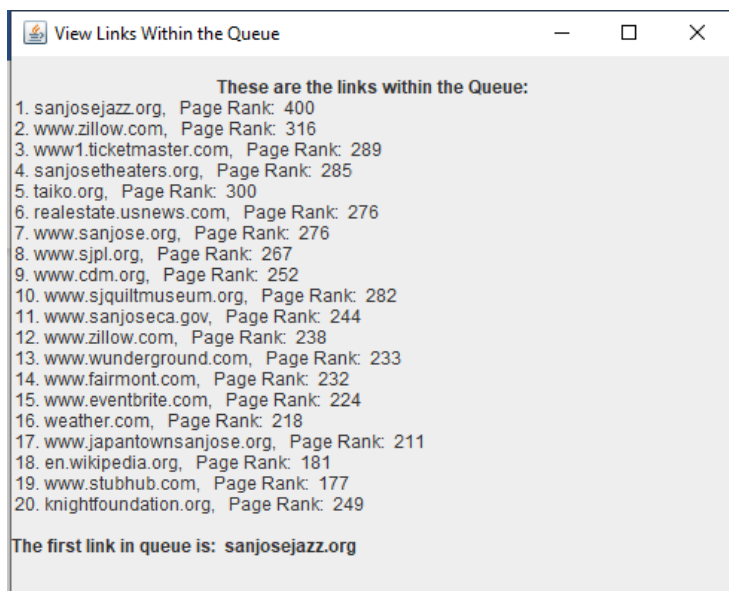
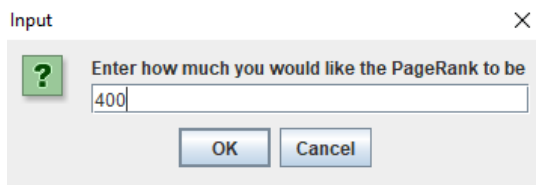
The link is then extracted and the queue is updated as it calls upon heapExtractMax as it can be seen in the image below



F. Allowing Users to choose any of the links and increase its PageRank



This is prompted when the user presses “Increase the Page Rank of a Link within the Queue” button. In this case we will be raising the 20th link to the maximum PageRank of 400 to see if the queue will move it to the top of the queue.



The Link’s page rank is increased to 400 and is put into the top of the Queue as `heapIncreaseKey` was called.

V. How to Install the Application and Test the Codes

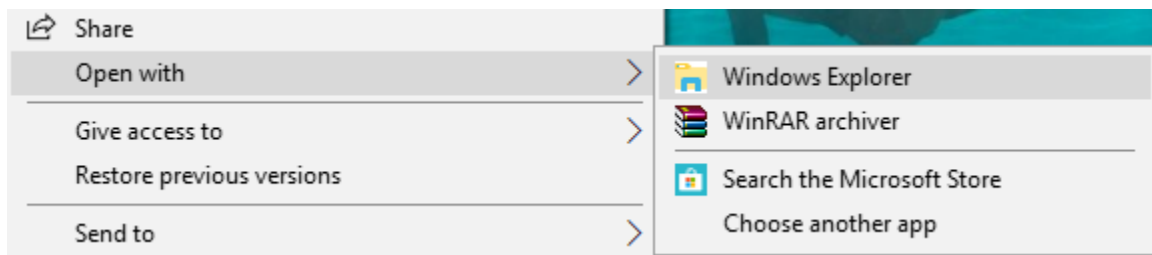
This section will demonstrate how to install the application and test the code of the programming assignment.

A. Required Programs

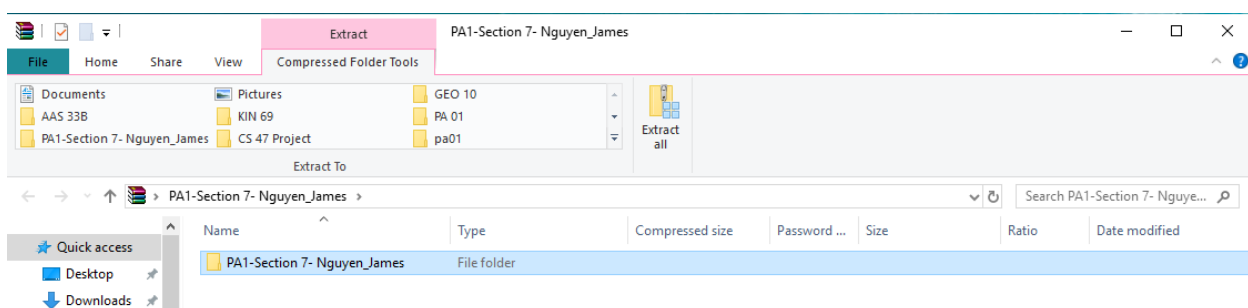
In order to test the application and the code, you will need to download Java which can be found at <https://www.java.com/en/download/>. Furthermore, you will also need to have Eclipse downloaded which can be found at <https://www.eclipse.org/downloads/>. You will also need to download jar file of the j.soup library in order to test the code which can be found at <https://jsoup.org/download>.

B. Extracting the Folder

In order to run the application you will need to extract the zipped folder first. In order to do this, you first have to left click on the folder, and then under the drop-down menu, you will select open with, and then Windows Explorer.



Once you have opened it with Windows Explorer, you can then click the extract all button on the top of the window.



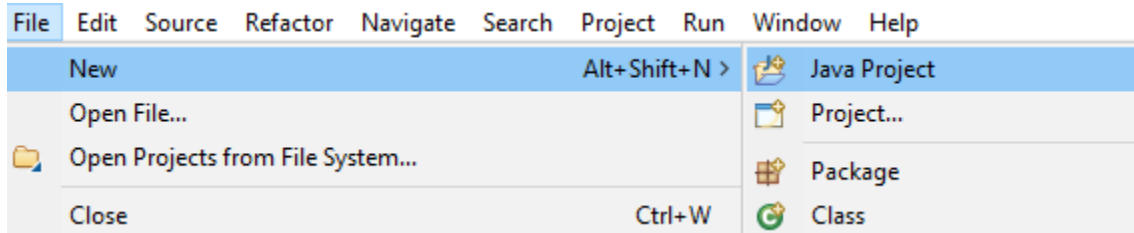
Then a pop up window will ask you to select a destination window to extract the files to. Once it has been extracted, a folder containing the extracted files will pop up.

C. Running the Application

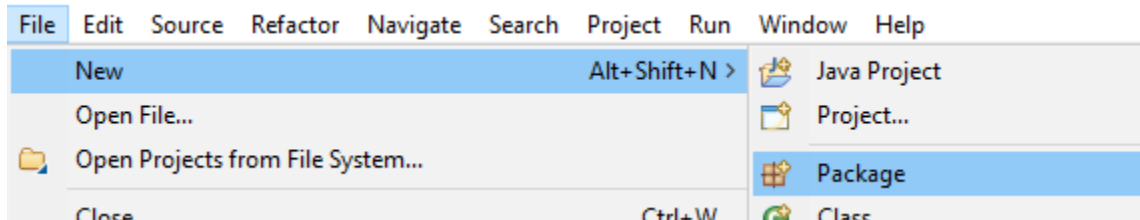
To run on the application, double click on the “Google Simulation.jar”. There will be a pop-up menu which will ask you to enter a keyword to search the web. Once you inputted your search and pressed ‘OK’, a pop-up menu will show up in which you can pick a choice and either view the list of the links and queue or edit the queue.

D. Testing the Code in Eclipse

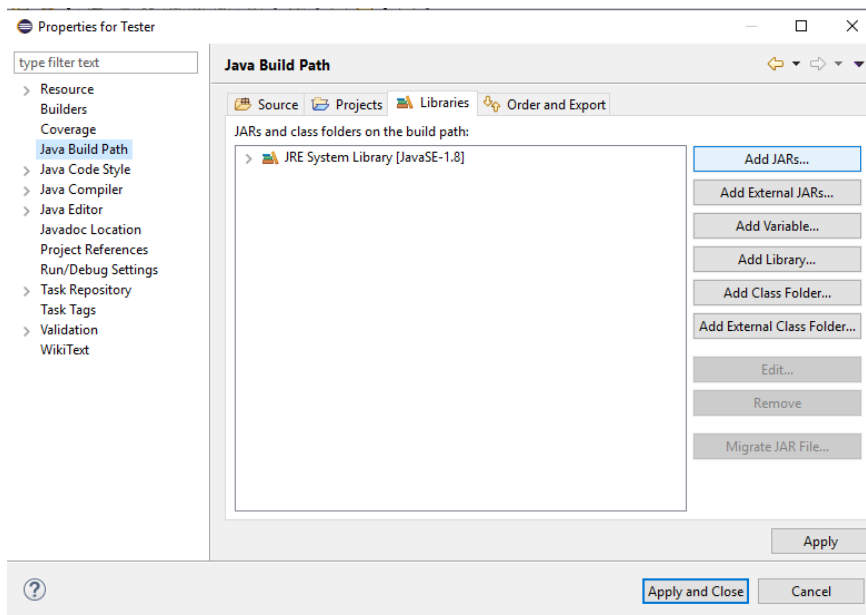
In order to test and view the code, you will have to open Eclipse first. After selecting a workspace in Eclipse, you will first press ‘File’ on the top left corner, then select new, and then Java Project.



You can name this new Java Project ‘Tester’. Next you will then go back to ‘File’, select new, and this time, you will select ‘Package’



You will then name this Package ‘pa01’. Once this has finished, you can then drag and drop the *.java files into the package. Next you will then left click on the project that you have made, and then select ‘Properties’ on the drop-down menu. You will then select ‘Java Build Path’, and then add the jsoup library by selecting ‘add JARs’ and selecting the jar file of the jsoup you have downloaded.



You can then press the green play button to run the main method of the ‘Google Simulation’ class, or you can view the code of the program.

VI. Problems Encountered during the Implementation

During the implementation of the Google Simulation, there were many problems that occurred and needed to be debugged or fixed.

A. Implementing HeapEditor

When I was implementing Heap Editor, I had many errors when I tried transferring the pseudo code of the textbook into Java Code. Since the pseudo code has its index starting at 1 and the Java language started their index at 0, the equations, for instance, to find the parent or children nodes were very different even though the indexes were just 1 apart. As a result, as I implemented the heap editor class and the different methods, I sometimes would calculate the index incorrectly causing errors such as the incorrect result, or Null Pointer Exceptions.

In order to solve this problem, I took a visual approach of drawing out the heaps and labeling the corresponding index to each node and going through each line of code and tracing through to make sure the index logic is correct.

B. Getting false length for Array

In some of the methods for example like buildMaxHeap, one would need to set the heap size to the length of the array you are using to build the max heap. In my case, my array.length returned an inaccurate representation of the number of elements it has, since it counted the null spaces within the array. As a result, the heap size would thus be inaccurate.

In order to solve this problem, I created a method to get the true length of the array by going through the array until I hit a null space within the array.

C. First Time Use of the Java Libraries of JFrame and JOptionPane

In this programming assignment, it was my first time using the java libraries of JFrame and JOptionPane to create a user interface for users. As a result, I faced several problems. The first of these problems is that at many times, the windows are not able to be opened as since I forgot to set the visibility of the window.

At other times, the window may be opened but the size of the window is too small to show anything. Furthermore, the JOptionPane input window that I created only return strings, in which I needed to use ParseInt. I also had problems with the JOptionPane input window as since it took in strings also, I needed to many different boundaries to not allow the input window from accepting string as it would return an Exception.

D. Use of Old Java Libraries

The Web Crawler Class that was provided with programming assignment required me to down the j.soup library online. At first I downloaded a j.soup library that was outdated, and even

though the web crawler was able to run at first, I found that it had a flaw in which I wasn't able to search any phrases with spaces with the j.soup library.

In order to fix this solution, I had to research the j.soup library online and found that I needed a new j.soup library which would allow me to use the Web Crawler to search for phrases with spaces.

VII. Lessons Learned

Throughout this programming assignment, I learned many different concepts and gained many new skills, as well as gained motivation regarding my coding skills.

A. How Google Search Engine Works

With this programming assignment, I was able to speculate and have more insight about what was happening behind the scenes when enter a word into the Google Search Engine. Before the assignment, I didn't know that Google had to do so many steps in the background. I also didn't know that Google has to take so much factors to determine the PageRank of a Page. This assignment has really opened my eyes and made me appreciate the Google Search Engine a little more.

B. Making a User Interface Within Java

Before this programming assignment, I didn't really know how to make a user interface in Java. I didn't know how to open another window from Java for a user to use. However, this programming assignment made me step out of my comfort zone and learn how to do something that I've never done before. Through research online, I was able to learn how to efficiently create a user interface and make it such that users are able to choose methods to do. If it wasn't for this programming assignment, I would probably be using System.in or System.out.println to show my methods.

C. Heap Sort and Heap Priority Queue

This programming assignment really led me to be able to learn the concepts of heap sort and the heap priority queue. By turning the pseudo code into Java code, and applying it to a real-life example, I was able to fully understand the different methods within each algorithm. Furthermore, by running it, I was able to see how fast and efficient using heap sort and heap priority queue was as, I didn't have to wait long for an action of the program to occur.

D. Motivation for my coding skills

This programming assignment really opened my eyes and showed me that I was meant to be a coder. Before the programming assignment, I usually had specific instructions to make applications like in CS 46B. However, with this programming assignment, I had the freedom to make as many classes and methods that I needed for my code to run. Since I was able to accomplish this programming assignment on my own, I am able to see that coding is really for me, and that I am even more motivated to carry on my coding journey!