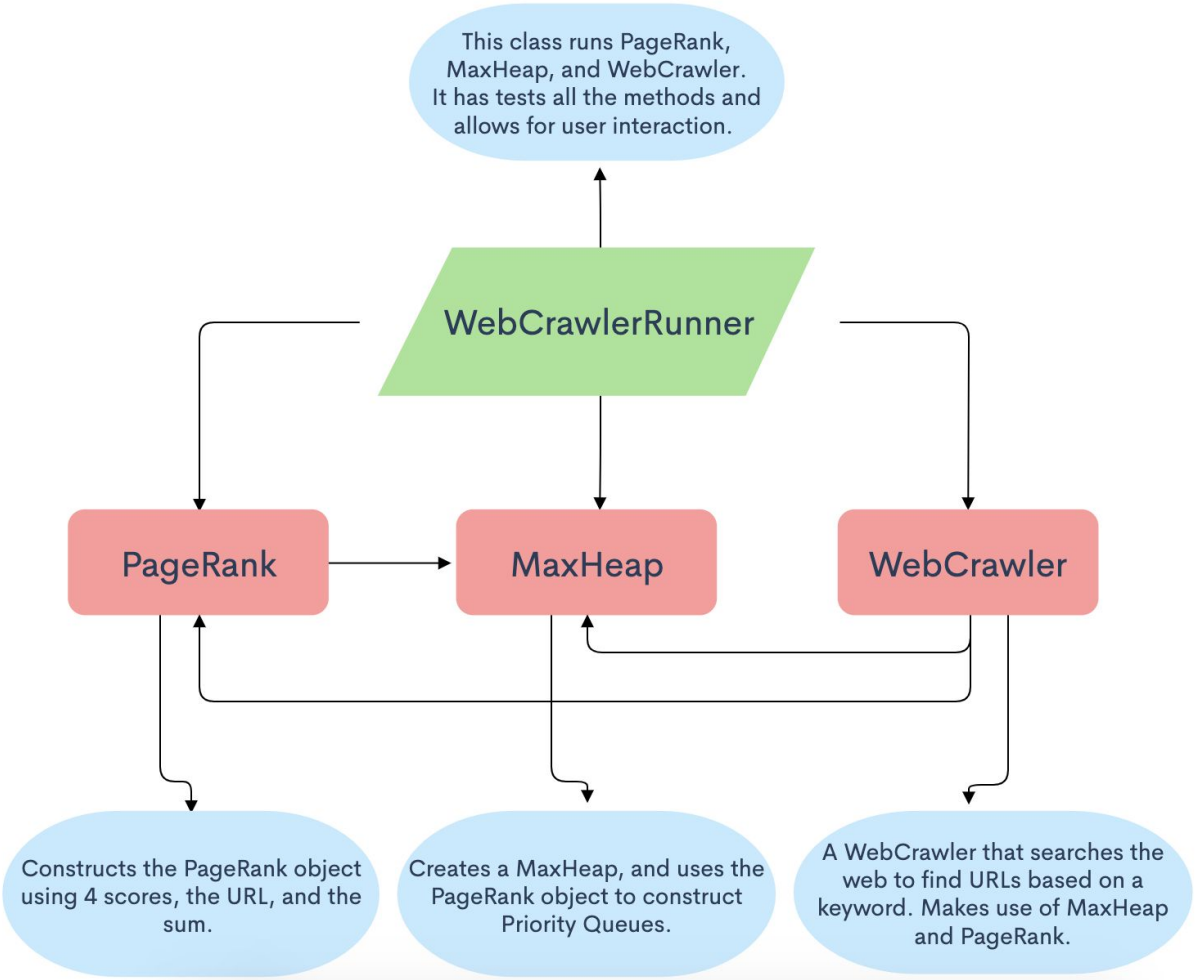
Programming Assignment - 1

Shivani Asokumar

Design and Implementation



Classes and Function Calls

PageRank: Constructs an object with 4 of the Page Rank scores from each category, a URL, and the sum of the 4 scores. This class is useful because in MaxHeap and WebCrawler, we can just build a PageRank object. Then, we can use an array of PageRank objects in order to build a heap and perform other operations.

1. getScore1(): gets the first score.
2. getScore2(): gets the second score.
3. getScore3(): gets the third score.
4. getScore4(): gets the fourth score.
5. getURL(): gets the url.
6. getSum(): gets the sum of the 4 scores.
7. setScore(): sets the URL to a new URL.
8. setSum(): sets the sum to a new sum.
9. compareTo(): compares the PageRank object first by sum then by url.
10. equals(): determines if two PageRank objects are equal.
11. hashCode(): returns a unique hashCode for each PageRank.
    PURPOSE: The purpose of all these methods is to have the correct getters and setters. That way, when I construct PageRank in another class, I am able to access the instance variables. In the MaxHeap class, when I need to swap two elements in an array, I can use the setters to do so.

MaxHeap: Constructs a MaxHeap and a Priority Queue. This method makes use of the PageRank class in order to construct a MaxHeap of PageRank objects. The purpose of this method is to ensure easy access to the Heap and Priority Queue from the WebCrawler class.

1. Parent(): gets the index of the parent in the heap. It is later useful in MaxHeapify method.
2. Left(): gets the index of the left node in the heap. It is later useful in MaxhHeapify method.
3. Right(): gets the index of the right node in the heap. It is later useful in MaxHeapify method.
4. BuildMaxHeap(): builds a max heap with a given array of PageRank objects. It implements the MaxHeapify(), which makes sure that the heap maintains the max heap property. This method is also useful for HeapSort().
5. MaxHeapify(): makes sure that the heap follows the Max Heap properties. This method is useful for building a Max Heap.
6. HeapSort(): sorts the given heap according to the PageRank sum. It orders the PageRanks from least to greatest sum. This is useful for allowing the users to see each of the PageRank object in the array.
7. HeapMaximum(): returns the PageRank with the greatest sum.

8. HeapExtractMax(): returns the PageRank with the greatest sum and it removes it from the heap. The heapSize is decreased by 1. This method is useful for users to see which PageRank is the greatest
9. HeapIncreaseKey(): increases the key value of a certain URL in the PageRank array in order to increase its priority in the queue. This methood is useful for allowing users to increase the priority of a certain URL.
10. MaxHeapInsert(): Inserts a PageRank object into the PageRank array. This method allows users to insert their own URLs into the array.

WebCrawler: Constructs a WebCrawler based on a keyword inputted by the user. It search for webpages containing that keyword, and it returns their URLs in a set. This class makes use of both the PageRank and the MaxHeap.

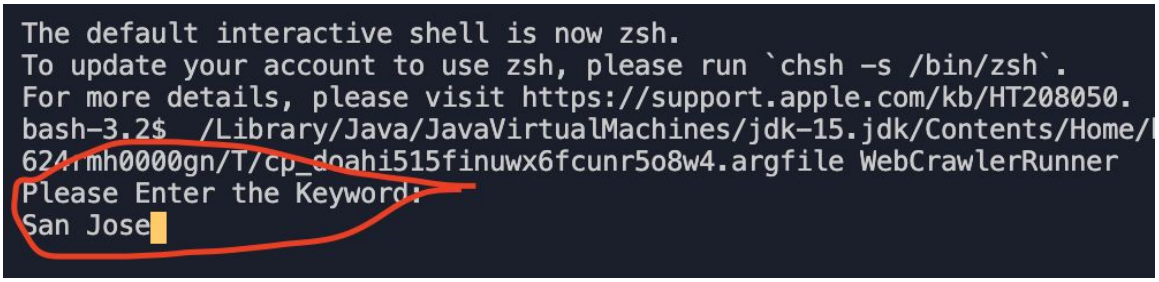
1. search(): starts the web crawler. It also searches the web using the keyword.
2. getDomainName(): uses pattern and matcher to extract the domain name of a URL.
3. getUrls(): gets a Set of 30 URL's based on the keyword that the user inputted. This is useful for the other methods which use the Set to construct an array of PageRank objects.
4. CalculatePageRank(): takes the Set with the 30 URLs, and constructs a PageRank object for each of them. It also computes the 4 scores, and the sum randomly for each of them. It returns an array of PageRank objects. This method is extremely useful for building heaps of PageRank objects.
5. crawl(): crawls the links and puts them in a set to keep. Makes an HTTP request for a web page based on a given URL.
6. searchForWord(): checks if the website contains the keyword.
7. StoreURLsinQueue(): stores the first 20 out of the 30 PageRank objects. This is useful for creating a Priority Queue.
8. PrintPageRankReverseOrder(): prints the given PageRank array in reverse order. This is useful when we run Heapsort, but we want to order it from greatest to least. This method allows the users to view the PageRank objects. This method is useful for debugging. Makes it easier to see the URL and the Sum.
9. PrintPageRank(): prints the given PageRank array in regular order. This method allows users to view the PageRank objects. This method is useful for debugging. Makes it easier to see the URL and the Sum.
10. PrintSet(): prints the given set so that it is easy to see the individual URLs.

WebCrawlerRunner: This method only contains a main method. It is useful for testing all of the methods from the other classes. It is also where the users get to interact with the code and make modifications to the Heap and Priority Queue.

Self-Testing Screenshots

Exercise Requirements.

1. **Using HeapSort to sort and display a list of web url links based on PageRank calculation:**
    a. **Allow users to enter keyword:**




```
// Enter Keyword
System.out.println("Please Enter the Keyword: ");
Scanner scan = new Scanner(System.in);
String keyword = scan.nextLine();
```

- There is no input for this exercise because we are simply asking the user to enter a keyword.
- I am using the Scanner to prompt the user, and I am later able to collect that information using scan.nextLine().

    b. **WebCrawler searches for the keyword on the inter to generate a list of web url links.**

```
1. www.sjusd.org/&sa=U&ved=2ah
2. sjmusart.org/&sa=U&ved=2ahU
3. www.sjsu.edu/&sa=U&ved=2ahU
4. cszsanjose.com/&sa=U&ved=2a
5. www.cnn.com/2020/10/22/us/s
6. sanjosetheaters.org/&sa=U&v
7. realestate.usnews.com/place
8. www.cdm.org/&sa=U&ved=2ahUK
9. www.visitcalifornia.com/pla
10. www.sanjose.org/&sa=U&ved=2
11. www.operasj.org/&sa=U&ved=2
12. en.wikipedia.org/wiki/Santa
13. www.sjica.org/&sa=U&ved=2ah
14. www.ndsj.org/&sa=U&ved=2ahU
15. en.wikipedia.org/wiki/Timel
16. www.city-data.com/city/San-
17. www.japantownsanjose.org/&s
18. www.redfin.com/city/17420/C
19. www.sanjosehotel.com/&sa=U&
20. www.fairmont.com/san-jose/&
21. worldpopulationreview.com/u
22. en.wikipedia.org/wiki/Histo
23. www.zillow.com/san-jose-ca/
24. collegefootballnews.com/202
25. www.sjquiltmuseum.org/&sa=U
26. www.nbcbayarea.com/tag/san-
27. en.wikipedia.org/wiki/Saint
28. www.facebook.com/CityofSanJ
29. broadwaysanjose.com/&sa=U&v
30. en.wikivoyage.org/wiki/San_
```

```
WebCrawler crawler = new WebCrawler(keyword);
crawler.search();
```

```
public void search() {
    String currentUrl = url;
    crawl(currentUrl);
    boolean success = searchForWord(keyword);
    if (success) {
        System.out.println(String.format("**Success** Word %s found at %s", keyword, currentUrl));
    }
    System.out.println(String.format("**Done** Visited %s web page(s)", urls.size()));

    System.out.println(" \nHere are the first 30 URL links: \n");

}
```

- The crawler uses the search method to search the web for URLs and returns them.
- In the above code, it takes in a keyword entered by the user and generates a list of 30 URLs.

c. **Allow users to display the search result containing a list of 30 URLs.**

Input: San Jose

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
bash-3.2$  /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/
624rmh0000gn/T/cp_doahi515finuwx6fcunr5o8w4.argfile WebCrawlerRunner
Please Enter the Keyword:
San Jose
```

Output

```
Would you like to view the 30 URLs? (Y):
Y
**Visiting** Received web page at https://google.com/search?q=San Jose&num=80
Found (139) links
Searching for the word San Jose...
**Success** Word San Jose found at https://google.com/search?q=San Jose&num=80
**Done** Visited 92 web page(s)

Here are the first 30 URL links:


1. www.sjusd.org/&sa=U&ved=2ah
2. sjmusart.org/&sa=U&ved=2ahU
3. www.sjsu.edu/&sa=U&ved=2ahU
4. cszsanjose.com/&sa=U&ved=2a
5. www.cnn.com/2020/10/22/us/s
6. sanjosetheaters.org/&sa=U&v
7. realestate.usnews.com/place
8. www.cdm.org/&sa=U&ved=2ahUK
9. www.visitcalifornia.com/pla
10. www.sanjose.org/&sa=U&ved=2
11. www.operasj.org/&sa=U&ved=2
12. en.wikipedia.org/wiki/Santa
13. www.sjica.org/&sa=U&ved=2ah
14. www.ndsj.org/&sa=U&ved=2ahU
15. en.wikipedia.org/wiki/Timel
16. www.city-data.com/city/San-
17. www.japantownsanjose.org/&s
18. www.redfin.com/city/17420/C
19. www.sanjosehotel.com/&sa=U&
20. www.fairmont.com/san-jose/&
21. worldpopulationreview.com/u
22. en.wikipedia.org/wiki/Histo
23. www.zillow.com/san-jose-ca/
24. collegefootballnews.com/202
25. www.sjquiltmuseum.org/&sa=U
26. www.nbcbayarea.com/tag/san-
27. en.wikipedia.org/wiki/Saint
28. www.facebook.com/CityofSanJ
29. broadwaysanjose.com/&sa=U&v
30. en.wikivoyage.org/wiki/San_
```

Input: Ice Cream

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208
bash-3.2$  /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/
624rmh0000gn/T/cp_doahi515finuwx6fcunr5o8w4.argfile WebCrawlerRun
Please Enter the Keyword:
Ice Cream
```

Output:

```
Would you like to view the 30 URLs? (Y):

**Visiting** Received web page at https://google.com/search?q=Ice Cream&num=80
Found (134) links
Searching for the word Ice Cream...
**Success** Word Ice Cream found at https://google.com/search?q=Ice Cream&num=80
**Done** Visited 85 web page(s)

Here are the first 30 URL links:

Y
1. www.goldenstateicecream.com
2. www.yelp.com/search%3Fcflt%
3. www.sanjose.org/listings/tr
4. www.hersheyicecream.com/&sa
5. www.rollbamaroll.com/2020/1
6. www.idfa.org/the-history-of
7. en.wikipedia.org/wiki/Froze
8. mitchellshomemade.com/&sa=U
9. en.wikipedia.org/wiki/Milk&
10. www.historysanjose.org/wp/p
11. www.museumoficecream.com/&s
12. www.dreyers.com/&sa=U&ved=2
13. creamnation.com/san-jose/&s
14. www.benjerry.com/&sa=U&ved=
15. postmates.com/merchant/smit
16. www.mercurynews.com/2017/06
17. www.graysicecream.com/&sa=U&
18. nypost.com/2020/10/23/this-
19. www.purityicecream.com/&sa=
20. www.instagram.com/sweetmelo
21. www.baskinrobbins.com/&sa=U
22. iciclescreamroll.com/&sa=U&v
23. www.cnet.com/news/this-app-
24. www.theverge.com/2020/10/22
25. www.foxnews.com/food-drink/
26. kellyshomemadeicecream.com/
27. www.facebook.com/treaticecr
28. www.velveticecream.com/&sa=
29. www.thrillist.com/news/nati
30. www.hudsonvilleicecream.com
```

```java
// Enter Keyword
System.out.println("Please Enter the Keyword: ");
Scanner scan = new Scanner(System.in);
String keyword = scan.nextLine();


// Set up WebCrawler
System.out.println("Would you like to view the 30 URLs? (Y): ");
WebCrawler crawler = new WebCrawler(keyword);
crawler.search();
String viewURLs = scan.nextLine();
Set<String> urls = crawler.getUrls();

// Allow users to see 30 URLs.
if(viewURLs.equals("Y") || viewURLs.equals("y") || viewURLs.equals("Yes") || viewURLs.equals("yes"))
{
    crawler.PrintSet(urls);
}
```

- The user is allowed to enter a keyword, and they can view the 30 resulting URLs.
- The crawler is first initialized, and then the search() method is called.
- We then collect the user data, and inside the if statement, we are printing the set of urls.

d. Assign 4 scores for the 4 PageRank factors to each URL.

Input: Ice Cream, Input for CalculatePageRank(): set of URLs().

```java
WebCrawler crawler = new WebCrawler(keyword);
crawler.search();
String viewURLs = scan.nextLine();
Set<String> urls = crawler.getUrls();
```

Output:

```
Would you like to view the 4 PageRank scores? (Y):
Y
=============
www.goldenstateicecream.com
Score 1: 96
Score 2: 27
Score 3: 79
Score 4: 85
=============
www.yelp.com/search%3Fcflt%
Score 1: 53
Score 2: 83
Score 3: 15
Score 4: 21
=============
www.sanjose.org/listings/tr
Score 1: 49
Score 2: 70
Score 3: 41
Score 4: 88
=============
www.hersheyicecream.com/&sa
Score 1: 23
Score 2: 12
Score 3: 21
Score 4: 29
=============
www.rollbamaroll.com/2020/1
Score 1: 8
Score 2: 42
Score 3: 21
Score 4: 3
=============
www.idfa.org/the-history-of
Score 1: 86
Score 2: 57
Score 3: 95
Score 4: 70
=============
en.wikipedia.org/wiki/Froze
Score 1: 5
Score 2: 68
Score 3: 69
Score 4: 54
=============
mitchellshomemade.com/&sa=U
Score 1: 45
Score 2: 58
Score 3: 51
Score 4: 4
=============
en.wikipedia.org/wiki/Milk&
Score 1: 62
Score 2: 63
Score 3: 34
Score 4: 13
=============
```

```
==========
www.historysanjose.org/wp/p
Score 1: 95
Score 2: 71
Score 3: 89
Score 4: 23
==========
www.museumoficecream.com/&s
Score 1: 41
Score 2: 39
Score 3: 18
Score 4: 5
==========
www.dreyers.com/&sa=U&ved=2
Score 1: 72
Score 2: 51
Score 3: 25
Score 4: 23
==========
creamnation.com/san-jose/&s
Score 1: 46
Score 2: 63
Score 3: 97
Score 4: 95
==========
www.benjerry.com/&sa=U&ved=
Score 1: 53
Score 2: 93
Score 3: 8
Score 4: 6
==========
postmates.com/merchant/smit
Score 1: 59
Score 2: 55
Score 3: 30
Score 4: 22
==========
www.mercurynews.com/2017/06
Score 1: 31
Score 2: 66
Score 3: 57
Score 4: 31
==========
www.graysicecream.com/&sa=U&
Score 1: 17
Score 2: 3
Score 3: 69
Score 4: 74
==========
nypost.com/2020/10/23/this-
Score 1: 7
Score 2: 96
Score 3: 24
Score 4: 97
==========
```

```
==========
www.purityicecream.com/&sa=
Score 1: 35
Score 2: 22
Score 3: 58
Score 4: 90
==========
www.instagram.com/sweetmelo
Score 1: 39
Score 2: 74
Score 3: 43
Score 4: 98
==========
www.baskinrobbins.com/&sa=U
Score 1: 17
Score 2: 41
Score 3: 93
Score 4: 86
==========
iciclescreamroll.com/&sa=U&v
Score 1: 66
Score 2: 23
Score 3: 72
Score 4: 26
==========
www.cnet.com/news/this-app-
Score 1: 55
Score 2: 9
Score 3: 52
Score 4: 71
==========
www.theverge.com/2020/10/22
Score 1: 14
Score 2: 89
Score 3: 49
Score 4: 60
==========
www.foxnews.com/food-drink/
Score 1: 16
Score 2: 44
Score 3: 56
Score 4: 49
==========
kellyshomemadeicecream.com/
Score 1: 27
Score 2: 8
Score 3: 96
Score 4: 81
==========
www.facebook.com/treaticecr
Score 1: 62
Score 2: 8
Score 3: 54
Score 4: 52
==========
```

```
=============
www.velveticecream.com/&sa=
Score 1: 16
Score 2: 2
Score 3: 88
Score 4: 70
=============
www.thrillist.com/news/nati
Score 1: 78
Score 2: 90
Score 3: 16
Score 4: 2
=============
www.hudsonvilleicecream.com
Score 1: 61
Score 2: 60
Score 3: 46
Score 4: 37
```

```
// Allow users to view the 4 PageRank scores
System.out.println("Would you like to view the 4 PageRank scores? (Y): ");
String viewPageRankScores = scan.nextLine();
PageRank[] p = crawler.CalculatePageRank(urls);

if(viewPageRankScores.equals("Y") || viewPageRankScores.equals("y") || viewPageRankScores.equals
("Yes") || viewPageRankScores.equals("yes")){
    for(PageRank pageRank : p){
        System.out.println("=============");
        System.out.println(pageRank.getURL());
        System.out.println("Score 1: " + pageRank.getScore1());
        System.out.println("Score 2: " + pageRank.getScore2());
        System.out.println("Score 3: " + pageRank.getScore3());
        System.out.println("Score 4: " + pageRank.getScore4());
    }
}
```

```java
public PageRank[] CalculatePageRank(Set<String> set){

    PageRank[] pageRankMapper = new PageRank[30];
    int count = 0;
    for (String url : set){

        // Generate random integer for PageRank score.
        Random rand = new Random();

        // These are the 4 scores for EACH url.
        int score1 = rand.nextInt(100) + 1;
        int score2 = rand.nextInt(100) + 1;
        int score3 = rand.nextInt(100) + 1;
        int score4 = rand.nextInt(100) + 1;

        int sum = score1 + score2 + score3 + score4;

        // Create PageRank object with url and sum
        PageRank pageRank = new PageRank(score1, score2, score3, score4, url, sum);
        pageRankMapper[count] = pageRank;

        count++;
    }

    return pageRankMapper;

}
```

```
 * Gets the first score.
 * @return the first score.
 */
public int getScore1(){
    return this.score1;
}

/**
 * Gets the second score.
 * @return the second score.
 */
public int getScore2(){
    return this.score2;
}

/**
 * Gets the third score.
 * @return the third score.
 */
public int getScore3(){
    return this.score3;
}

/**
 * Gets the fourth score.
 * @return the fourth score.
 */
public int getScore4(){
    return this.score4;
}
```

- The user is allowed to view the 4 PageRank scores for each URL.
- Then CalculatePageRank() is called.
- CalculatePageRank() takes in a Set of 30 urls, which we saw earlier, and it returns an array of PageRank objects each containing the 4 scores, the URL, and the sum.
- In our main method, we are getting each of the scores by using the getters from the PageRank class. Then, we are printing out the scores.
- The scores are completely randomized, which is why we are using a Random integer generator in our CalculatePageRank() method.

e. **Calculate the total PageRank score for each URL. Use HeapSort to sort the 30 URLs based on PageRank sum.**

Input:

```
PageRank[] p = crawler.CalculatePageRank(urls);
```

```
heap.Heapsort(p);
```

```
crawler.PrintPageRankReverseOrder(p);
```

- First, in order to calculate the total PageRank score, we are calling CalculatePageRank(), which takes in the Set of urls.
- Then, we use the output of that method p as the input for heapSort(), which sorts the p (least to greatest) based on the total PageRank score.
- Finally, we reverse the heap when we print so that it is sorted from greatest to least total PageRank score. This makes it easier to perform the next step where we have to display the PageRanks from greatest to least for the user.

Output:

```
=======Sorted Order=======
1. www.baskinrobbins.com/&sa=U, Page Rank Score: 312
2. postmates.com/merchant/smit, Page Rank Score: 287
3. www.historysanjose.org/wp/p, Page Rank Score: 280
4. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 271
5. www.goldenstateicecream.com, Page Rank Score: 270
6. www.cnet.com/news/this-app-, Page Rank Score: 267
7. www.idfa.org/the-history-of, Page Rank Score: 266
8. www.velveticecream.com/&sa=, Page Rank Score: 254
9. iciclescreamroll.com/&sa=U&v, Page Rank Score: 238
10. www.theverge.com/2020/10/22, Page Rank Score: 227
11. www.facebook.com/treaticecr, Page Rank Score: 217
12. www.hersheyicecream.com/&sa, Page Rank Score: 205
13. en.wikipedia.org/wiki/Gelat, Page Rank Score: 204
14. www.hudsonvilleicecream.com, Page Rank Score: 195
15. kellyshomemadeicecream.com/, Page Rank Score: 180
16. en.wikipedia.org/wiki/Milk&, Page Rank Score: 175
17. en.wikipedia.org/wiki/Froze, Page Rank Score: 165
18. wgntv.com/news/mcdonalds-fa, Page Rank Score: 165
19. www.foxnews.com/food-drink/, Page Rank Score: 164
20. www.mercurynews.com/2017/06, Page Rank Score: 162
21. www.purityicecream.com/&sa=, Page Rank Score: 162
22. www.benjerry.com/&sa=U&ved=, Page Rank Score: 159
23. www.graysicecream.com/&sa=U&, Page Rank Score: 154
24. creamnation.com/san-jose/&s, Page Rank Score: 131
25. www.yelp.com/search%3Fcflt%, Page Rank Score: 120
26. www.museumoficecream.com/&s, Page Rank Score: 103
27. mitchellshomemade.com/&sa=U, Page Rank Score: 94
28. www.sanjose.org/listings/tr, Page Rank Score: 75
29. www.rollbamaroll.com/2020/1, Page Rank Score: 91
30. nypost.com/2020/10/23/this-, Page Rank Score: 72
```

```java
System.out.println("=======Sorted Order=======");
heap.Heapsort(p);
crawler.PrintPageRankReverseOrder(p);
```

```
public void Heapsort(PageRank[] A){

    this.BuildMaxHeap(A);
    for (int i = A.length - 1; i > 0; i--){
        PageRank temp = A[0]; // Swaps A[0] and A[i]
        A[0] = A[i];
        A[i] = temp;
        heapSize = heapSize - 1;
        this.MaxHeapify(A, 0); //Changed from 1 to 0 to account for index = 0
    }
}
```

```
public void PrintPageRankReverseOrder(PageRank[] pageRank){
    int count = 1;
    Collections.reverse(Arrays.asList(pageRank)); // Riverses a given array.

    //Traverses the array.
    for(PageRank p : pageRank){
        System.out.println(count + ". " + p.getURL() + ", Page Rank Score: " + p.getSum());
        count++;
    }
}
```

- The HeapSort orders from least to greatest, and the PringPageReverseOrder reverses the heap.

f. Allow users to view the sorting result in sequence order based on PageRank score

Input:

```
PageRank[] p = crawler.CalculatePageRank(urls);
```

```
heap.Heapsort(p);
```

```
crawler.PrintPageRankReverseOrder(p);
```

Output:

```
Would you like to view the PageRank in sorted order? (Y):
Y
=======Sorted Order=======
1. www.baskinrobbins.com/&sa=U, Page Rank Score: 312
2. postmates.com/merchant/smit, Page Rank Score: 287
3. www.historysanjose.org/wp/p, Page Rank Score: 280
4. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 271
5. www.goldenstateicecream.com, Page Rank Score: 270
6. www.cnet.com/news/this-app-, Page Rank Score: 267
7. www.idfa.org/the-history-of, Page Rank Score: 266
8. www.velveticecream.com/&sa=, Page Rank Score: 254
9. iciclescreamroll.com/&sa=U&v, Page Rank Score: 238
10. www.theverge.com/2020/10/22, Page Rank Score: 227
11. www.facebook.com/treaticecr, Page Rank Score: 217
12. www.hersheyicecream.com/&sa, Page Rank Score: 205
13. en.wikipedia.org/wiki/Gelat, Page Rank Score: 204
14. www.hudsonvilleicecream.com, Page Rank Score: 195
15. kellyshomemadeicecream.com/, Page Rank Score: 180
16. en.wikipedia.org/wiki/Milk&, Page Rank Score: 175
17. en.wikipedia.org/wiki/Froze, Page Rank Score: 165
18. wgntv.com/news/mcdonalds-fa, Page Rank Score: 165
19. www.foxnews.com/food-drink/, Page Rank Score: 164
20. www.mercurynews.com/2017/06, Page Rank Score: 162
21. www.purityicecream.com/&sa=, Page Rank Score: 162
22. www.benjerry.com/&sa=U&ved=, Page Rank Score: 159
23. www.graysicecream.com/&sa=U&, Page Rank Score: 154
24. creamnation.com/san-jose/&s, Page Rank Score: 131
25. www.yelp.com/search%3Fcflt%, Page Rank Score: 120
26. www.museumoficecream.com/&s, Page Rank Score: 103
27. mitchellshomemade.com/&sa=U, Page Rank Score: 94
28. www.sanjose.org/listings/tr, Page Rank Score: 75
29. www.rollbamaroll.com/2020/1, Page Rank Score: 91
30. nypost.com/2020/10/23/this-, Page Rank Score: 72
```

```java
// Allow users to view the sorted PageRank
System.out.println("Would you like to view the PageRank in sorted order? (Y): ");
String sortedPageRank = scan.nextLine();
if(sortedPageRank.equals("Y") || sortedPageRank.equals("y") || sortedPageRank.equals("Yes") ||
sortedPageRank.equals("yes")){
    System.out.println("=======Sorted Order=======");
    heap.Heapsort(p);
    crawler.PrintPageRankReverseOrder(p);
}
```

```
public void Heapsort(PageRank[] A){

    this.BuildMaxHeap(A);
    for (int i = A.length - 1; i > 0; i--){
        PageRank temp = A[0]; // Swaps A[0] and A[i]
        A[0] = A[i];
        A[i] = temp;
        heapSize = heapSize - 1;
        this.MaxHeapify(A, 0); //Changed from 1 to 0 to account for index = 0
    }
}
```

```
public void PrintPageRankReverseOrder(PageRank[] pageRank){
    int count = 1;
    Collections.reverse(Arrays.asList(pageRank)); // Riverses a given array.

    //Traverses the array.
    for(PageRank p : pageRank){
        System.out.println(count + ". " + p.getURL() + ", Page Rank Score: " + p.getSum());
        count++;
    }
}
```

- This is similar to the previous step. The user is allowed to view the PageRank[] in sorted order. The greatest is at the top and the least is at the bottom of the list.
- HeapSort is used to sort from greatest to least PageRank score

2. **Using Heap Priority Queue to store web url links based on PageRank.**
   a. **Use Heap Priority queue to store the first 20 URLs out of the 30 into the Heap.**

Input: Ice Cream

```
PageRank[] r = crawler.StoreURLsInQueue(p);
```

- P is the sorted Heap containing 30 URLs.

Output:

```
======Priority Queue With 20 URLs
1. www.baskinrobbins.com/&sa=U, Page Rank Score: 312
2. postmates.com/merchant/smit, Page Rank Score: 287
3. www.historysanjose.org/wp/p, Page Rank Score: 280
4. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 271
5. www.goldenstateicecream.com, Page Rank Score: 270
6. www.cnet.com/news/this-app-, Page Rank Score: 267
7. www.idfa.org/the-history-of, Page Rank Score: 266
8. www.velveticecream.com/&sa=, Page Rank Score: 254
9. iciclescreamroll.com/&sa=U&v, Page Rank Score: 238
10. www.theverge.com/2020/10/22, Page Rank Score: 227
11. www.facebook.com/treaticecr, Page Rank Score: 217
12. www.hersheyicecream.com/&sa, Page Rank Score: 205
13. en.wikipedia.org/wiki/Gelat, Page Rank Score: 204
14. www.hudsonvilleicecream.com, Page Rank Score: 195
15. kellyshomemadeicecream.com/, Page Rank Score: 180
16. en.wikipedia.org/wiki/Milk&, Page Rank Score: 175
17. en.wikipedia.org/wiki/Froze, Page Rank Score: 165
18. wgntv.com/news/mcdonalds-fa, Page Rank Score: 165
19. www.foxnews.com/food-drink/, Page Rank Score: 164
20. www.mercurynews.com/2017/06, Page Rank Score: 162
```

```
PageRank[] r = crawler.StoreURLsInQueue(p);
System.out.println();
System.out.println("======Priority Queue With 20 URLs");
crawler.PrintPageRank(r);
```

```
public PageRank[] StoreURLsInQueue(PageRank[] pageRank){
    MaxHeap heap = new MaxHeap();
    PageRank[] result = new PageRank[20]; // Initialize array to store the 20 PageRanks.
    for(int i = 0; i < result.length; i++){
        PageRank p = pageRank[i]; // Transfers the objects over.
        result[i] = p;
    }

    return result;
}
```

- We are taking the original 30 sorted PageRanks from the array p. Then we are copying over the first 20 URLs from greatest to least into the queue.
- Then we are printing out the Priority Queue using PringPageRank.

b. Allow user to insert into max-heap a new PageRank containing a url and a key.

Input:

```
System.out.println("(Inserting WebURL to queue) Please enter URL: ");
String url = scan.nextLine();
System.out.println("(Inserting WebURL to queue) Please enter Sum: ");
int sum = scan.nextInt();
scan.nextLine();
PageRank rank = new PageRank(20,20,20,20,url, sum);
```

- The input is a PageRank that the user created

Output:

```
======Heap Insert=====

(Inserting WebURL to queue) Please enter URL:
DUMMY
(Inserting WebURL to queue) Please enter Sum:
500
1. DUMMY, Page Rank Score: 500
2. www.purityicecream.com/&sa=, Page Rank Score: 338
3. www.hersheyicecream.com/&sa, Page Rank Score: 304
4. www.theverge.com/2020/10/22, Page Rank Score: 251
5. www.historysanjose.org/wp/p, Page Rank Score: 258
6. postmates.com/merchant/mitc, Page Rank Score: 250
7. www.happyjoes.com/&sa=U&ved, Page Rank Score: 248
8. en.wikipedia.org/wiki/Milk&, Page Rank Score: 222
9. www.museumoficecream.com/&s, Page Rank Score: 219
10. zmenu.com/daybreak-donuts-a, Page Rank Score: 250
11. iciclescreamroll.com/&sa=U&v, Page Rank Score: 205
12. nypost.com/2020/10/23/this-, Page Rank Score: 191
13. wgntv.com/news/mcdonalds-fa, Page Rank Score: 191
14. www.goldenstateicecream.com, Page Rank Score: 182
15. www.cnet.com/news/this-app-, Page Rank Score: 177
16. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 175
17. www.yelp.com/search%3Fcflt%, Page Rank Score: 171
18. www.baskinrobbins.com/&sa=U, Page Rank Score: 157
19. www.foxnews.com/food-drink/, Page Rank Score: 152
20. www.thrillist.com/news/nati, Page Rank Score: 215
```

```
// Max Heap Insert
System.out.println("======Heap Insert=====");
System.out.println();

System.out.println("(Inserting WebURL to queue) Please enter URL: ");
String url = scan.nextLine();
System.out.println("(Inserting WebURL to queue) Please enter Sum: ");
int sum = scan.nextInt();
scan.nextLine();
PageRank rank = new PageRank(20,20,20,20,url, sum);
heap.MaxHeapInsert(r, rank);
crawler.PrintPageRank(r);
```

```
public void MaxHeapInsert(PageRank[] A, PageRank key){
    heapSize = A.length - 1;
    A[heapSize].setSum(Integer.MIN_VALUE);
    try {
        this.HeapIncreaseKey(A, heapSize, key); // Increases the key value for the last element.
    } catch (Exception e){
        System.out.println(e.getMessage());
    }

}
```

- We are asking the user for a URL.
- We are asking the user for a sum.
- Then we create a Page rank based on the URL and the Sum.
- We then call MaxHeapInsert to insert the PageRank into the Queue.
- Finally we are printing out the Queue.
- Our URL is placed at the top because it has the highest sum, and it is in sorted order.

c. Allow user to extract the maximum PageRank.

Input: Ice Cream

```
PageRank extractMax = heap.HeapExtractMax(r);
```

- The input is the Priority Queue that we already have.

Output:

```
======Queue Extract Max=====

DUMMY: 500
```

```
// Heap Extract Max
System.out.println("Would you like to Extract Max Number from queue? (Y): ");

String extractMaxAnswer = scan.nextLine();
if(extractMaxAnswer.equals("Y") || extractMaxAnswer.equals("y") || extractMaxAnswer.equals("Yes") |
| extractMaxAnswer.equals("yes")){
    System.out.println("======Queue Extract Max=====");
    System.out.println();

    try{
        PageRank extractMax = heap.HeapExtractMax(r);
        System.out.println(extractMax.getURL() + ": " + extractMax.getSum());
    } catch(Exception e){
        System.out.println(e.getMessage());
    }
    System.out.println();
}
```

```
public PageRank HeapExtractMax(PageRank[] A) throws Exception{
    if (heapSize < 1){ // If the heapSize is less than 1, throws Exception
        throw new Exception("Heap Underflow");
    }

    // Swaps A[0] and A[heapSize]
    PageRank max = A[0];
    A[0] = A[heapSize];
    heapSize--;

    // Maxifies the root element
    this.MaxHeapify(A, 0);
    return max;

}
```

- We allow the user to extract the maximum number.
- We return that PageRank that we extracted as we can see in the above images.
- The goal is to extract the max by removing the root element, then we MaxHeapify the Queue.

d. Allow the user to increase the key of any URL in the Priority Queue

Input:

```
System.out.println("(Increasing Key) Please enter index (1-19): ");
int index = scan.nextInt();
scan.nextLine();
System.out.println("(Increasing Key) Please enter URL: ");
String increaseKeyURL = scan.nextLine();
System.out.println("(Increasing Key) Please enter Sum: ");
int increaseKeySum = scan.nextInt();


PageRank increaseKey = new PageRank(0,0,0,0,increaseKeyURL,increaseKeySum);
```

```
heap.HeapIncreaseKey(r, index, increaseKey);
```

- The user inputs a URL that is already in the queue.
- The user inputs a key of their choice.
- Then we create a PageRank object using the URL and Sum.
- Then we call HeapIncreaseKey

Output:

```
======Queue Increase Key=====

(Increasing Key) Please enter index (1-19):
1
(Increasing Key) Please enter URL:
DUMMY
(Increasing Key) Please enter Sum:
900
1. DUMMY, Page Rank Score: 900
2. www.purityicecream.com/&sa=, Page Rank Score: 338
3. www.hersheyicecream.com/&sa, Page Rank Score: 304
4. www.theverge.com/2020/10/22, Page Rank Score: 251
5. zmenu.com/daybreak-donuts-a, Page Rank Score: 250
6. postmates.com/merchant/mitc, Page Rank Score: 250
7. www.happyjoes.com/&sa=U&ved, Page Rank Score: 248
8. en.wikipedia.org/wiki/Milk&, Page Rank Score: 222
9. www.museumoficecream.com/&s, Page Rank Score: 219
10. www.thrillist.com/news/nati, Page Rank Score: 215
11. iciclescreamroll.com/&sa=U&v, Page Rank Score: 205
12. nypost.com/2020/10/23/this-, Page Rank Score: 191
13. wgntv.com/news/mcdonalds-fa, Page Rank Score: 191
14. www.goldenstateicecream.com, Page Rank Score: 182
15. www.cnet.com/news/this-app-, Page Rank Score: 177
16. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 175
17. www.yelp.com/search%3Fcflt%, Page Rank Score: 171
18. www.baskinrobbins.com/&sa=U, Page Rank Score: 157
19. www.foxnews.com/food-drink/, Page Rank Score: 152
20. www.thrillist.com/news/nati, Page Rank Score: 215
```

```
// Heap Increase Key
System.out.println("======Queue Increase Key=====");
System.out.println();

System.out.println("(Increasing Key) Please enter index (1-19): ");
int index = scan.nextInt();
scan.nextLine();
System.out.println("(Increasing Key) Please enter URL: ");
String increaseKeyURL = scan.nextLine();
System.out.println("(Increasing Key) Please enter Sum: ");
int increaseKeySum = scan.nextInt();


PageRank increaseKey = new PageRank(0,0,0,0,increaseKeyURL,increaseKeySum);

try{
    heap.HeapIncreaseKey(r, index, increaseKey);
} catch (Exception e){
    System.out.println(e.getMessage());
}
crawler.PrintPageRank(r);
System.out.println();
```

```
public void HeapIncreaseKey(PageRank[] A, int i, PageRank key) throws Exception{
    if (key.getSum() < A[i].getSum()){ // If the give key is smaller than original key, throws
    Exception.
        throw new Exception("New Key is Smaller than Current Key");
    }

    A[i].setSum(key.getSum()); // Sets the new sum.
    A[i].setURL(key.getURL()); //Sets the new URL.

    while(i > 0 && A[this.Parent(i)].getSum() < A[i].getSum()){
        PageRank temp = A[i]; // Swaps A[i] and A[this.Parent(i)]
        A[i] = A[this.Parent(i)];
        A[this.Parent(i)] = temp;

        i = this.Parent(i);
    }
}
```

- We call HeapIncreasekey() after building our own PageRank object based on the user input.
- Then HeapIncreaseKey() increases the sum value of PageRank for the chosen PageRank object.

3. **Other Functions**
    a. **Build Max Heap**

    Input:

```
heap.BuildMaxHeap(r);
```

- R is the priority queue that we already have.

Output:

```
======Builds Max Heap======

1. www.purityicecream.com/&sa=, Page Rank Score: 338
2. www.hersheyicecream.com/&sa, Page Rank Score: 304
3. www.historysanjose.org/wp/p, Page Rank Score: 258
4. www.theverge.com/2020/10/22, Page Rank Score: 251
5. zmenu.com/daybreak-donuts-a, Page Rank Score: 250
6. postmates.com/merchant/mitc, Page Rank Score: 250
7. www.happyjoes.com/&sa=U&ved, Page Rank Score: 248
8. en.wikipedia.org/wiki/Milk&, Page Rank Score: 222
9. www.museumoficecream.com/&s, Page Rank Score: 219
10. www.thrillist.com/news/nati, Page Rank Score: 215
11. iciclescreamroll.com/&sa=U&v, Page Rank Score: 205
12. nypost.com/2020/10/23/this-, Page Rank Score: 191
13. wgntv.com/news/mcdonalds-fa, Page Rank Score: 191
14. www.goldenstateicecream.com, Page Rank Score: 182
15. www.cnet.com/news/this-app-, Page Rank Score: 177
16. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 175
17. www.yelp.com/search%3Fcflt%, Page Rank Score: 171
18. www.baskinrobbins.com/&sa=U, Page Rank Score: 157
19. www.foxnews.com/food-drink/, Page Rank Score: 152
20. bubsburgers.com/&sa=U&ved=2, Page Rank Score: 151
```

```
// Builds Max Heap
System.out.println("======Builds Max Heap======");
System.out.println();
heap.BuildMaxHeap(r);
crawler.PrintPageRank(r);
System.out.println();
```

```
public void BuildMaxHeap(PageRank[] A){
    heapSize = A.length - 1;

    for (int i = A.length - 1 / 2; i >= 0; i--){ // i >= 0 because our index starts at 0
        this.MaxHeapify(A, i);
    }
}
```

- Builds MaxHeap given a Priority Queue.
- We are then printing out that Max Heap so that it is easier to view.
- MaxHeap uses MaxHeapify to maintain the Max Heap properties.

b. Max Heapify

Input:

```
heap.MaxHeapify(r, 10);
```

- This is taking in the Priority Queue r that we previously build.
- Then we are passing an index value.

Output:

```
======MaxHeapify======

1. www.purityicecream.com/&sa=, Page Rank Score: 338
2. www.hersheyicecream.com/&sa, Page Rank Score: 304
3. www.historysanjose.org/wp/p, Page Rank Score: 258
4. www.theverge.com/2020/10/22, Page Rank Score: 251
5. zmenu.com/daybreak-donuts-a, Page Rank Score: 250
6. postmates.com/merchant/mitc, Page Rank Score: 250
7. www.happyjoes.com/&sa=U&ved, Page Rank Score: 248
8. en.wikipedia.org/wiki/Milk&, Page Rank Score: 222
9. www.museumoficecream.com/&s, Page Rank Score: 219
10. www.thrillist.com/news/nati, Page Rank Score: 215
11. iciclescreamroll.com/&sa=U&v, Page Rank Score: 205
12. nypost.com/2020/10/23/this-, Page Rank Score: 191
13. wgntv.com/news/mcdonalds-fa, Page Rank Score: 191
14. www.goldenstateicecream.com, Page Rank Score: 182
15. www.cnet.com/news/this-app-, Page Rank Score: 177
16. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 175
17. www.yelp.com/search%3Fcflt%, Page Rank Score: 171
18. www.baskinrobbins.com/&sa=U, Page Rank Score: 157
19. www.foxnews.com/food-drink/, Page Rank Score: 152
20. bubsburgers.com/&sa=U&ved=2, Page Rank Score: 151
```

```
// MaxHeapify
System.out.println("======MaxHeapify======");
System.out.println();
heap.MaxHeapify(r, 10);
crawler.PrintPageRank(r);
System.out.println();
```

```
public void MaxHeapify(PageRank[] A, int i){
    int l = this.Left(i);
    int r = this.Right(i);
    int largest;

    if (l < heapSize && A[l].getSum() > A[i].getSum()){ // Changed to < because getting IndexOutOfBounds
        largest = l;
    }
    else {
        largest = i;
    }

    if(r < heapSize && A[r].getSum() > A[largest].getSum()){ // Changed to < because getting
    IndexOutOfBounds
        largest = r;
    }
    if(largest != i){
        PageRank temp = A[i]; //Swaps A[i] and A[largest]
        A[i] = A[largest];
        A[largest] = temp;

        this.MaxHeapify(A, largest); // Maxifies largest.
    }
}
```

- We use MaxHeapify() to maintain the max heap properties.
- We start with a specific element in the Priority queue. In our case, we are starting at index 10. We check to see if that element follows the properties.
- Then we make recursive calls to make sure the entire queue is a Max Heap.

c. Heap Maximum

Input:

```
System.out.println(heap.HeapMaximum(r).getURL() + ": " + heap.HeapMaximum(r).getSum());
```

- We are getting the Maximum element in the Priority Queue r, which we have already built.

Output:

```
======Heap Maximum=====

www.purityicecream.com/&sa=: 338
```

```
// Heap Maximum
System.out.println("======Heap Maximum=====");
System.out.println();
System.out.println(heap.HeapMaximum(r).getURL() + ": " + heap.HeapMaximum(r).getSum());
System.out.println();
```

```
public PageRank HeapMaximum(PageRank[] A){
    return A[0]; // Returns the root element since its the max.
}
```

- HeapMaximum returns the root element because that has the maximum page rank score in the queue.
- Then we are printing out the URL and Sum so it is easy for us to view.

d. Heap Sort

Input:

```
heap.Heapsort(r);
```

- The input is the Priority Queue r, which we have already built.

Output:

```
======HeapSort=======

1. www.foxnews.com/food-drink/, Page Rank Score: 152
2. www.yelp.com/search%3Fcflt%, Page Rank Score: 171
3. www.baskinrobbins.com/&sa=U, Page Rank Score: 157
4. www.dreyers.com/&sa=U&ved=2, Page Rank Score: 175
5. www.cnet.com/news/this-app-, Page Rank Score: 177
6. www.goldenstateicecream.com, Page Rank Score: 182
7. wgntv.com/news/mcdonalds-fa, Page Rank Score: 191
8. nypost.com/2020/10/23/this-, Page Rank Score: 191
9. iciclescreamroll.com/&sa=U&v, Page Rank Score: 205
10. www.thrillist.com/news/nati, Page Rank Score: 215
11. www.thrillist.com/news/nati, Page Rank Score: 215
12. www.museumoficecream.com/&s, Page Rank Score: 219
13. en.wikipedia.org/wiki/Milk&, Page Rank Score: 222
14. www.happyjoes.com/&sa=U&ved, Page Rank Score: 248
15. postmates.com/merchant/mitc, Page Rank Score: 250
16. zmenu.com/daybreak-donuts-a, Page Rank Score: 250
17. www.theverge.com/2020/10/22, Page Rank Score: 251
18. www.hersheyicecream.com/&sa, Page Rank Score: 304
19. www.purityicecream.com/&sa=, Page Rank Score: 338
20. DUMMY, Page Rank Score: 900
```

```java
// HeapSort
System.out.println("======HeapSort=======");
System.out.println();
heap.Heapsort(r);
crawler.PrintPageRank(r);
System.out.println();
```

```java
public void Heapsort(PageRank[] A){

    this.BuildMaxHeap(A);
    for (int i = A.length - 1; i > 0; i--){
        PageRank temp = A[0]; // Swaps A[0] and A[i]
        A[0] = A[i];
        A[i] = temp;
        heapSize = heapSize - 1;
        this.MaxHeapify(A, 0); //Changed from 1 to 0 to account for index = 0
    }
}
```

- Heap Sort sorts the Page Rank from least to greatest based on the PageRank score.
- It Builds a max Heap, then switch the root element with the last element the heap.
- The reduces the size of the heap.
- Finally, it uses MaxHeapify to ensure that it is a MaxHeap.

Installation

1. Unzip the file. You should see .pdf, .java, and jar files.
2. Make sure you have an editor that supports Java. Now, right-click on the jar file => open with (Your editor).
3. You should now see 4 different classes: PageRank.java, MaxHeap.java, WebCrawler.java, and WebCrawlerRunner.java.
4. Navigate to WebCrawlerRunner.java.
5. Run the code.
    a. In the console, you will see the various prompts.
    b. Please respond to those prompts accordingly.

Problems Encountered

1. For MaxHeap.java, I initially followed the pseudocode from the textbook very strictly, but I ended up with a lot of IndexOutOfBounds exceptions. I especially had a lot of issues with heapSize because in BuildMaxHeap(), I set heapSize = A.length. So, I was able to fix this by setting heapSize = A.length - 1.
2. In WebCrawler.java, I used a HashMap to store the URL and the PageRank sums. Then, when I had to sort the map, I used a TreeMap. This caused a lot of issues later on because the TreeMap would ignore duplicate elements. When I had two sums that were the same, only one of the URL actually got counted. To bypass this problem, I decided to sort the HashMap using Collections.sort(). This too proved difficult because the goal of this assignment was to use HeapSort. However, I was unable to use HeapSort on a HashMap. Ultimately, I decided to delete all of my Maps,

and I just created a PageRank object. This made things a lot easier because I could just use Arrays.sort() to sort a PageRank[].

3. In MaxHeap.java, for HeapInsertKey(A, key), I was passing an integer value for the key parameter. I later realized that this is incorrect because I am no longer dealing with a Heap of integers. I have a Heap of PageRank objects. So, I changed it to PageRank key. I did the same thing for HeapIncreaseKey().

Lessons Learned

1. This assignment has been extremely helpful in solidifying my understanding of Heaps and Priority Queues. Before this assignment, I had a very abstract idea of how to implement these data structures in a real coding environment. I particularly appreciated the Web Crawler because it provides a real sense of how Priority Queues are implemented. It also helped me further understand how search engines work.
2. Another important concept that I learned relates to Java. I learned that you can use Objects from other classes, for example, I can create a PageRank object in the MaxHeap class without having to inherit PageRank. I was unaware of this before, and I inherited PageRank instead of just creating an instance of it.
3. A takeaway for me is that the best way to learn the concepts from the textbook is to implement them through programming. This assignment has motivated me to go back to the different data structures and algorithms we learned about such as sorting, and implement them through code. I think it would be really interested if I could incorporate them into solving actual problems.