Programming Assignment 1 Report:

Google Search PageRank Implementation

Neeval Kumar

CS 146A, Section 8

4:30 PM – 5:30

1. **Design and Implementation**

In this assignment, I heavily relied on objects and arrays to do the actual sorting. I created an array of objects (the UrlPageRank object) that would hold the Url and the PageRank factors and it’s sum. I then applied the heapsort methods to this array of objects and using the UrlPageRank methods to print the each Url and PageRank sum. For the heap priority queues, I reversed the array and then applied the 3 methods to the array. The benefit of priority queues is that we can skip the buildMaxHeap step, so after using the methods, the heap should satisfy the maxheap priority already. I just applied the heapsort function afterwards and printed.

1. **List of Classes/Methods/Subroutines**

WebCrawler class:

This class’s purpose was to search the keyword inputted and then create a HashSet of its multiple String urls. This was standard and I just used the one given by Professor Wu.

UrlPageRankClass:

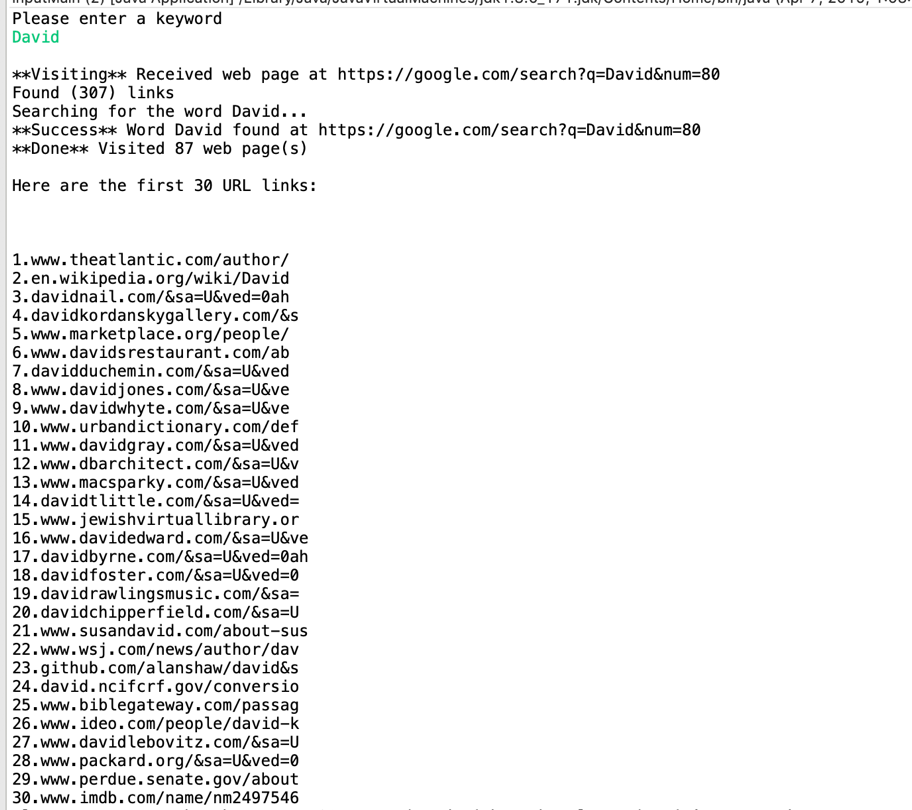
This class was mainly to create an object that would be able to hold the Url, taken by each index of the Hashset from Web Crawler. It would then store 4 PageRank factors (that were randomized) into the object and have a getSum method to return the sum of all those 4 PageRank scores.

HeapSort class: Most important class – (indexes were changed for java implementation)

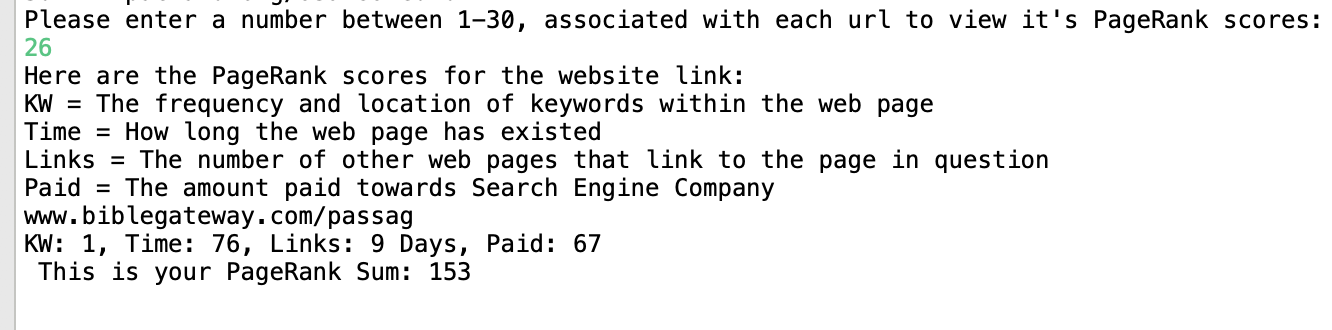
* HeapSort method – to receive array, and then divide it in half so we can access parent nodes. Decrementing down we apply maxHeapify to swap the values so we can have an array that satisfies the heap property. And then we build the BuildMaxHeap.
* MaxHeapify – Basic method to check if the left and right child nodes are less than the parent node. If not, it recursively swaps the child and parent node. Pretty much step-by-step similar to book’s pseudo code.
* BuildMaxHeap – The actual building of the heap that occurs. A for loop to decrement builds each node with a value attached to it. It also runs MaxHeapify to make sure the heap property is satisfied.
* PrintMapKey –to print the Map of the sorted PageRankFactors and its associated Url
* HeapMaximum – return the root of the Array
* HeapExtractMax – to remove the and return the root of the array, or in this case, removing the Url and its associated Page Rank factor and returning both
* HeapIncreaseKey – To increase the PageRankFactor of a Url, therefore increasing its position in the priority queue.
* MaxHeapInsert – To insert a Url and its associated with Page Rank Factor into the priority queue.

InputMain class: (Note. I used a series of Scanners so the user interface is more structured)

1. I first asked the user to input a keyword to search using web crawler.
2. I created an UrlPageRank object and and array that would store its PageRankFactors and Sum.
3. I filled the array with the objects
4. I then asked the user to input a number between 1-30 to receive its PageRank score and url
5. I then sort the list and print it to the user.
6. I create a new heap of the first 20 UrlPageRank objects I extracted from the previous sort
7. I ask the user to insert a new Url and print the new Array with the Url and its PageRank score inserted
8. I ask the user to whether return and extract the top of the heap and print the array afterward.
9. I ask the user to type a url number and a key to increase the objects sum by.
10. I then print the new array with the position change of the increased object.
11. **Self-Testing Screenshots**
12. **Enter a keyword and search for keyword**

****

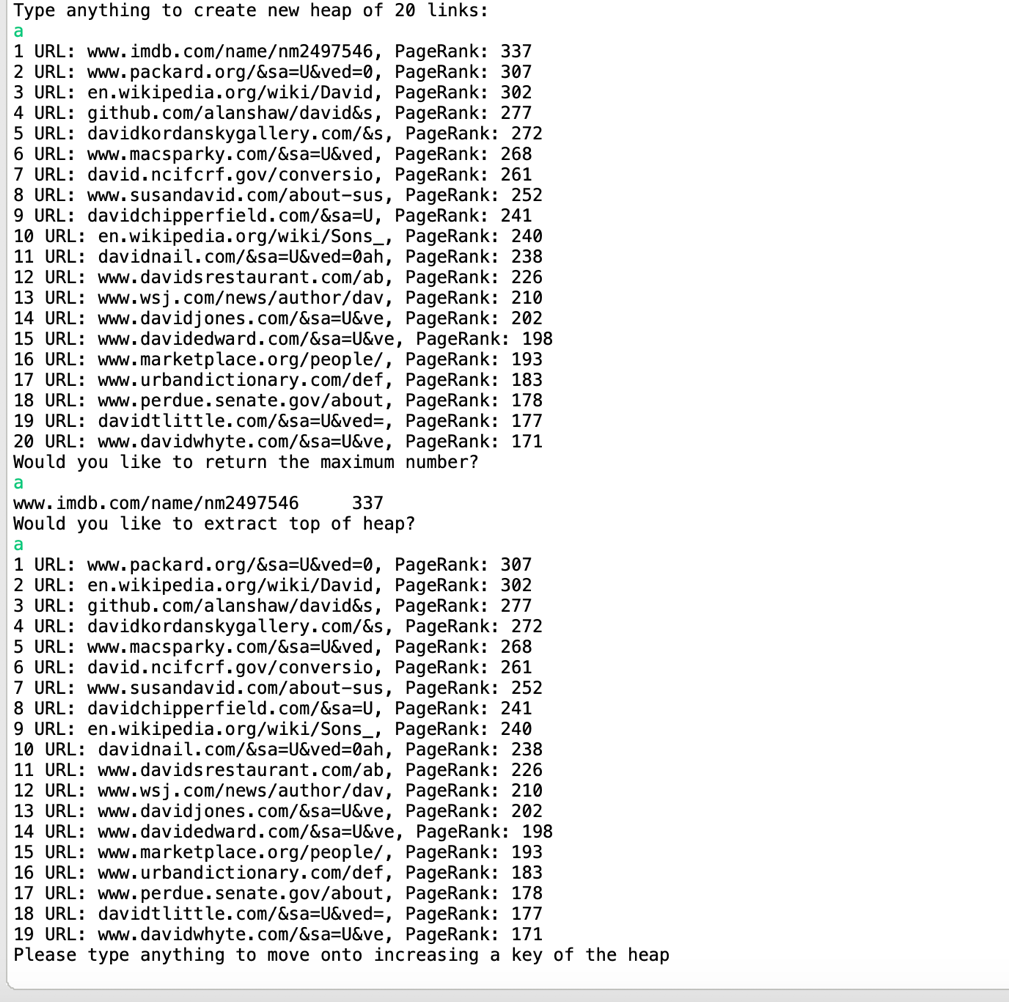
1. **Enter a number from numbers above to view its PageRank score**

****

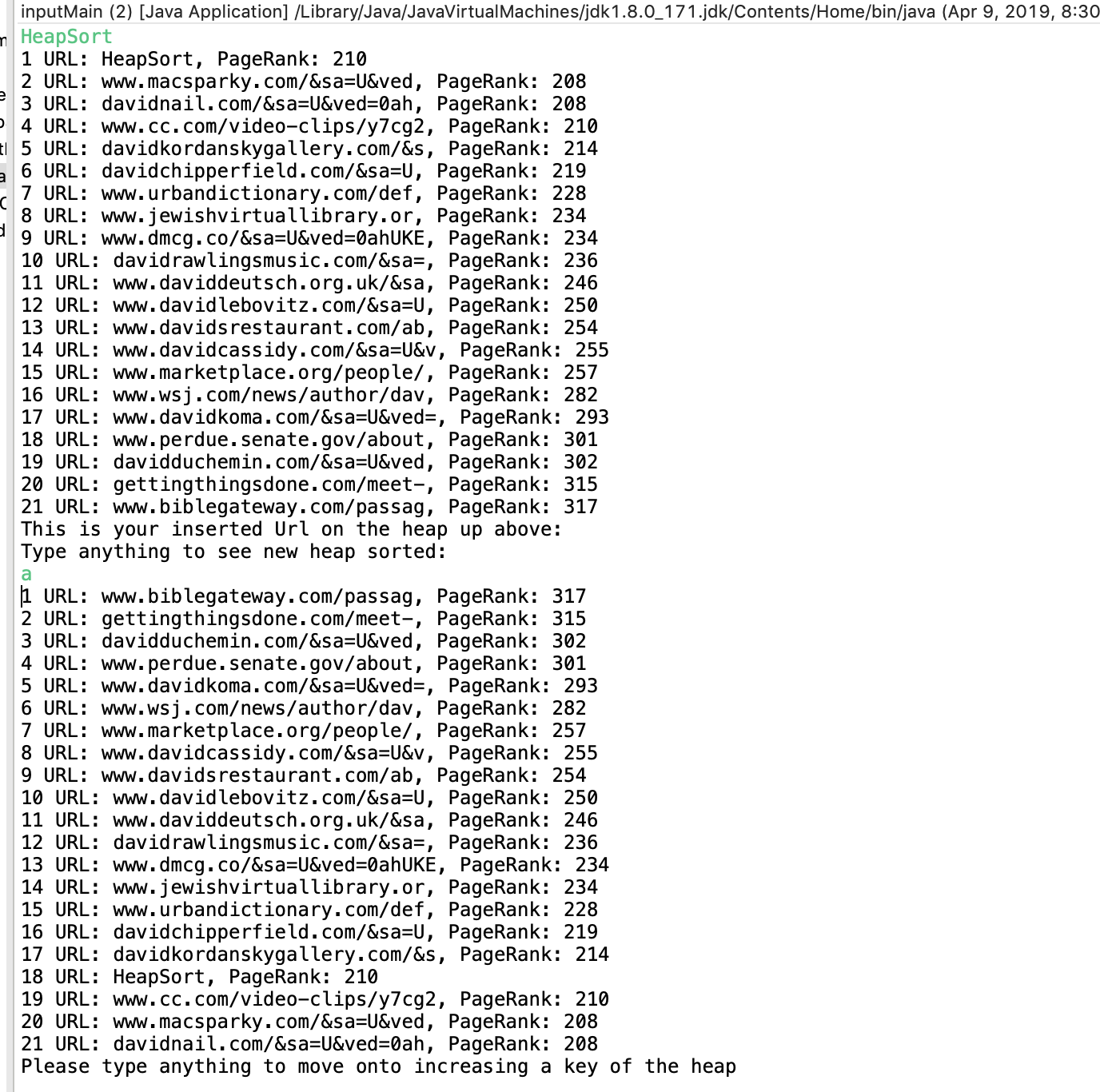
1. **Print Sorted Urls by PageRank score, as can see, from largest to smallest**

****

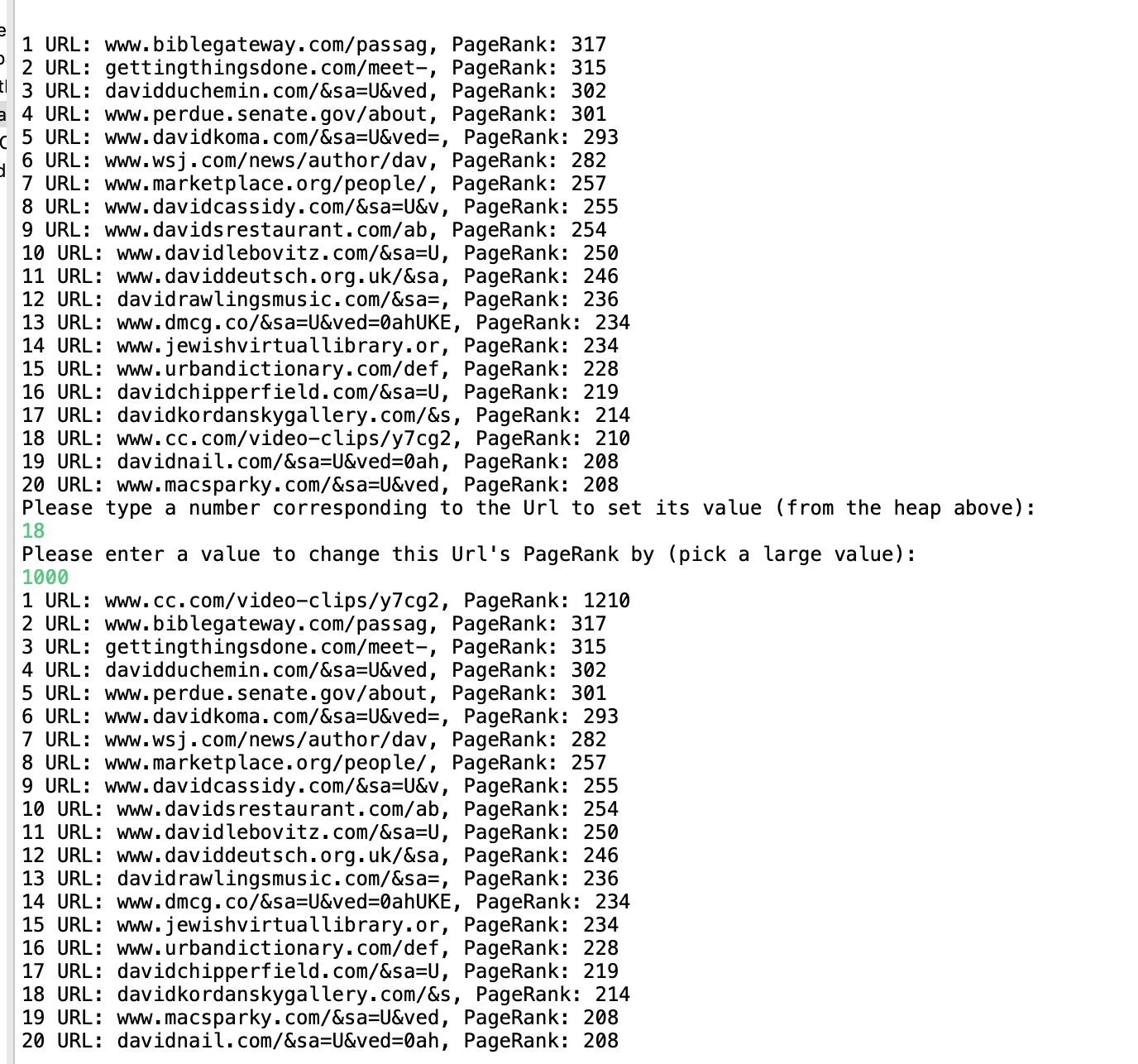
1. **Create new heap of first 20 links and ask for maximum and ask to extract it.**

****

1. **I ask the user to input an url and then have the new url inputted into the back of the queue. I then ask the user to continue to view the sorted heap. As can seen, I input the url HeapSort which has a pagerank of 210. It is then sorted in 18th place due to its Pagerank.**

****

1. **I ask the user to input a number corresponding to the url number and and a key to increase this url by. As can seen, I input 18 which was cc.com with a PageRank of 210. After increasing this PageRank by 1000, the new heap shows cc.com as the first url at a score of 1210.**

****

1. **Step-by-Step Procedure to unzip Files, Install Application and Test Code**
2. **First extract the file to unzip it (if on mac just double click the zipped file)**
3. **A new folder will pop up. Go to its contents.**
4. **You will see the java classes and the jar file.**
5. **After looking over the java classes, or testing them out yourself in eclipse by dragging them into a new project, you can test the console in terminal or command prompt for PC’s.**
6. **Using ls and cd, make your way to the jar file.**
7. **Type “java -jar PA1Redo.jar “.**
8. **You can now run the console commands seen from the screenshots above.**
9. **Make sure to follow the directions in the console or reference the screenshots above.**
10. **Problems Encountered During Implementation:** I spent a large amount of time trying to figure how to implement a way to transfer the PageRank score sums into an array, and keep all its references to it’s URL link. This required a large amount of research towards Object Oriented Programming, and how object arrays work, which I only briefly learned before. I also struggled a bit with the indexes of the Heapsort functions, granted the pseudo code was not adjusted to java code. To solve this problem, I did a lot of trial and error to sort my array. There was a period of time where it would only execute the MaxHeapify function but would not do the Heapsort or my insert function would not swap the parent node with the inserted node. Therefore, I used print statements to see that the problem lied in an index problem. Afterwards , most of my problems lied in syntax which was pretty solvable as long I surfed through Java’s API pages, even though it was quite time consuming.
11. **Lessons learned**: I struggled a lot in the beginning, but I also learned a lot. My Data Structures teacher was not that strict, so this assignment allowed me to polish some of my skills in Data Structures, Object Oriented, and of course the most important part of this assignment, my skills in Algorithms. Some main key concepts I took away from this assignment was using Objects to store data, creating user interactive input commands (use of Scanner), and commenting and indenting my code. I also learned a little about how PageRank actually functions and what type of work I would do in these big companies as a Computer Scientist. This assignment was the closest assignment I’ve ever done to a company’s coding “project”, because it required quite a bit of time. I can now see the amount of effort I have to put it in and manage my time in the upcoming Programming Assignments and hopefully future coding projects.