CSCI 2110 – Data Structures in Java
Assignment A4

The goal of this assignment is to practice on Heap and HashMaps.

Q1) (50 points) Word Frequency

1) (25 points) Write a program to count the frequency of occurrence of each word in an input text file. Use the *java.util.HashMap* class to store the words and their frequencies.
   a. Your program gets a file name, "*document.txt*", and reads the content of the file.
   b. Your program removes all **non-alphabet ([^a-zA-Z])** characters and converts the rest to lower case.
   c. The third step is to tokenize the text into words, and insert or update the frequencies of words in the *HashMap*. Tokenization means to split a sentence into its words:

```
String[] result = "this is a test".split("\\s");
for (int x=0; x<result.length; x++)
    System.out.println(result[x]);


//output:
   this
   is
   a
   test
```

   Each entry in the *HashMap* is a pair of string and integer:
   <*word*, *frequency*>

2) (10 points) After building the *HashMap*, sort the frequencies in a descending order and print the top 20 most frequent words along with their frequencies. Append the output to your code or submit it as a text file.
3) (10 points) After building the *HashMap*, sort the frequencies in an ascending order and print the top 20 least frequent words along with their frequencies. Append the output to your code or submit it as a text file.
4) (5 points) Create a log-log plot of frequencies. The horizontal axis is the log(rank order) and the vertical axis is the log(frequency), where the most

frequent word has rank order 1. For more information about this plot, please read the following Wikipedia article. You need to upload your plot as an image along with your codes.

https://en.wikipedia.org/wiki/Zipf%27s_law

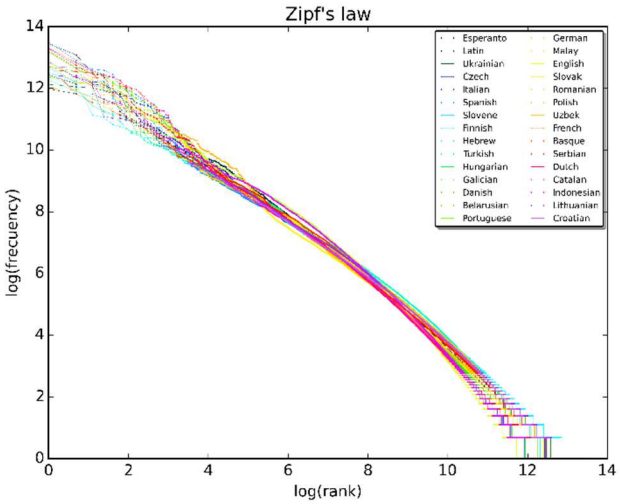

Figure 1: Zipf's law log-log diagram [1]

*Please download the input text file *document.txt*.

[1] https://en.wikipedia.org/wiki/Zipf%27s_law

Q2) (50 points) The goal of this question is to practice on Heaps. An implementation is provided in *Heap.java*. It is based on max heap. **Using that class**, you answer the following questions. If you need to add any auxiliary methods, make them protected or private.

1) (20 points) Similar to the max heap implementation in *Heap.java*, implement a min heap class. The new generic class, named *MinHeap.java*, should be implemented using an Arraylist<T>. Remember that you need to modify *siftUp*, *siftDown*, and replace *deleteMax* by *deleteMin*.

2) (30 points) Create a main method in test class *testHeaps*
   a. (15 points) To test the max heap implementation, use the method *add* to build a max heap by inserting 30 random integer numbers. Use the method *nextInt()* from the *java.util.Random* package for this purpose.

Perform successive *deleteMax* to obtain a descending order of numbers. Append the order as comment to your code. Also print and append as comment the level ordering of numbers using methods of the max heap. Is the level ordering like the descending order? Explain your observation.

b. (15 points) To test the min heap implementation, use the method *add* to build a min heap by inserting 30 random integer numbers. Use the method *nextInt()* from the *java.util.Random* package for this purpose. Perform successive *deleteMin* to obtain an ascending order of numbers. Append the order as comment to your code. Also print and append as comment the level ordering of numbers using methods of the min heap. Is the level ordering like the ascending order? Explain your observation.