WARNING: SPECIFICATIONS NOT COMPLETE. CHECK BACK LATER FOR FINAL VERSION.

1. Overview

Zipf’s law, named after the Harvard linguistics professor George Zipf, is the observation that frequency of occurrence of some event ($P$), as a function of the rank ($i$) when the rank is determined by the above frequency of occurrence, is a power-law function $P_i \approx 1/i^a$ with the exponent ($a$) close to unity. The most famous example of Zipf’s law is the frequency of English words\(^1\). For example in Project Gutenberg’s\(^2\) copy of «The Complete Works of William Shakespeare»\(^3\), we see the following statistics:

% wc shakespeare.txt  
124456 901325 5707111 shakespeare.txt  
% zipf.perl -l shakespeare.txt -o shakespeare.freq  
% head -10 shakespeare.freq  

27644 3.0714% the  
26739 2.9709% and  
20684 2.2981% i  
19200 2.1332% to  
18174 2.0192% of  
14620 1.6244% a  
13653 1.5169% you  
12481 1.3867% my  
11124 1.2359% that  
10968 1.2186% in

Poloñius: What do you read, my lord?  
Hamlet: Words, words, words.  


2. Program Specification

**NAME**

zipf.jar – calculate the Zipf distribution for some input text.

**SYNOPSIS**

zipf.jar [-l] [-o outputfile] [-c hyphenchars] [filename…]

**DESCRIPTION**

The zipf.jar utility reads in words from its input files and counts the number of occurrences of each word. It then prints out the words that it has found along with a count of the number of times each word appears and its percentage occurrence in the total list. Output is sorted in descending order by count. For words whose count is the same, output is sorted in ascending lexicographic (not alphabetical) order.

The output format is firstly an integer right justified with leading spaces in a field of width 10, and followed by a space. Then a floating point number in fixed point format right justified in a field of width 10 with four digits to the right of the decimal point, followed by a percent sign and four spaces. Finally, the word is printed followed by a new line.

**OPTIONS**

Any word from the command line which begins with a minus sign «-» is taken to be an option string. Options may be given as separate words or concatenated together into a single word. However, if a word consists of a single minus sign with nothing following it, it is an operand, not an option.

---

1. http://linkage.rockefeller.edu/wli/zipf/  
Debug output is printed to the standard error. The exact format of this debug output is not specified and is of use only to the programmer of this utility.

All words read in are converted to lower case before being added to the frequency table. Without this option, upper and lower case letters are considered different.

Instead of writing output to the standard output, output goes to the filename specified with this option. If the letter «o» is followed by any characters in the current word, then all of those characters constitute the output file name. Otherwise the output filename is the next word on the command line.

Special characters may occur inside a word if they are preceded and followed by a letter. By default the hyphen («-») and apostrophe («’») are special and no others are. This allows hyphenated words and contractions. If special characters are specified, they replace the hyphen and the apostrophe, although these characters may, of course, be specified in the option. The operand may immediately follow the letter «s» in the option word or be the entire next command line word.

All operands are names of files that are read in sequence. If any file name is specified as a minus sign («-»), the standard input is read at that position. If no filename operands are specified, the standard input is read.

0 Successful execution without errors.
1 An error occurred, such as an invalid option, or a file that could not be accessed.

3. Implementation Strategy

4. Data Structure

The most important data structure to be used in this assignment is the HashMap, discussed in class. It will make use of collision resolution by chaining. Initially, you should make use of java.util.HashMap while you work on your main program. What that is done, write your own version of HashMap and use it instead. Caution: If you submit a program that uses java.util.HashMap you will lose half the points for the assignment.

You will begin with the following:

class HashMap{
    public static class Entry{
        private Object key;
        private Object value;
        private Entry link;
        public Object getKey();
        public Object getValue();
    };
    public Object get{ Object key };
    public Object put{ Object key, Object value };
    public Iterator iterator();
};