Tony Misic (500759917)
Thomas Liu (500717670)

# Assignment #1 Report

## Algorithms

The first algorithm written in invert.cpp is the parsing algorithm. In this algorithm, the provided "cacm.all" file is parsed line by line, each being checked for the fields: .I, .T, .W, .B, and .A. Using a series of loops and conditionals, when a certain field is found, the content beneath it is then parsed line by line into a vector where they will be processed before being added to a map.

The second algorithm in the mentioned file is *split*, a basic tokenization algorithm that takes vectors containing parsed sentences and splits them up word by word, returning a vector that contains an entry for every word in the processed sentences.

The third algorithm written is *delimit*, another tokenization algorithm that uses the library function "strtok." In this algorithm, each string entry inside the aforementioned vector is checked for basic delimiters. If a delimiter is found in the entry, it is removed and the string is broken up and returned as its separate components (i.e. Tom-Tony is returned as Tom and Tony separately).

The fourth algorithm in invert.cpp is *lower*, a basic normalization algorithm that converts every letter of a string to its lower case form.

The fifth algorithm implemented into invert.cpp is a C++ variation of the Porter Stemming Algorithm which takes terms and removes suffixes, leaving a stem. This is one of two algorithms that can be enabled or disabled using command line arguments.

The sixth algorithm implemented is *fill_stop_words*, an algorithm for parsing the data inside the provided "stopwords.txt" file into a map for comparisons.

Finally, the seventh algorithm implemented is *stop_word_removal*. Using a simple comparison, this algorithm compares query terms that have been parsed against the map of stop words created by *fill_stop_words*, removing any stop words found from the list of queries.

## Data Structures

The data structure of choice for this assignment were maps. The primary reasons for this choice was because maps do not require an extra hashing function, and they are self-sorting according to the key values in each of the <key, value> pairs that are stored, thus allowing for easy maintenance as well as quick random look-up. Another reason why maps were chosen to be the data structure used was because it allowed for one-to-one correspondence across multiple text files due to their nature. As previously mentioned, since they are self-sorting according to the key in <key, value> pairs, we used this to our advantage by organizing each generated text file the same way, ensuring a one-to-one correspondence across all files (i.e. Line #45 in postings.txt will contain all the necessary information about the query term found on line #45 in dictionary.txt).

## Running The Program

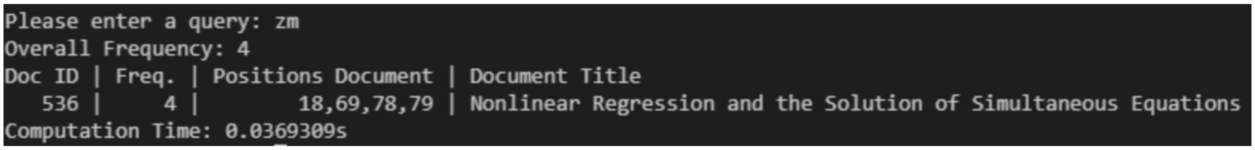
Requirement: Visual Studio Code with support for C++ installed.

1) Download and extract “cps842f19_a1_tmisic.zip” to the desktop of the machine.
2) Launch the application “Visual Studio Code”
3) Along the top taskbar, click File —> Open Folder —> Navigate to the Desktop —> Select “cps842f19_a1_tmisic” —> Select Folder
4) Along the top taskbar, click Terminal —> New Terminal
5) To compile the programs, enter into the terminal: “make all”
6) To generate the text files, enter into the terminal: “./invert [arg1] [arg2]”
    a) arg1 can be a 1 or a 0 to enable/disable Stop Word Removal
    b) arg2 can be a 1 or a 0 to enable/disable the Stemming Algorithm.
7) To run the query program, enter into the terminal: make test
8) To exit the query program, query “ZZEND” or press CTRL + C

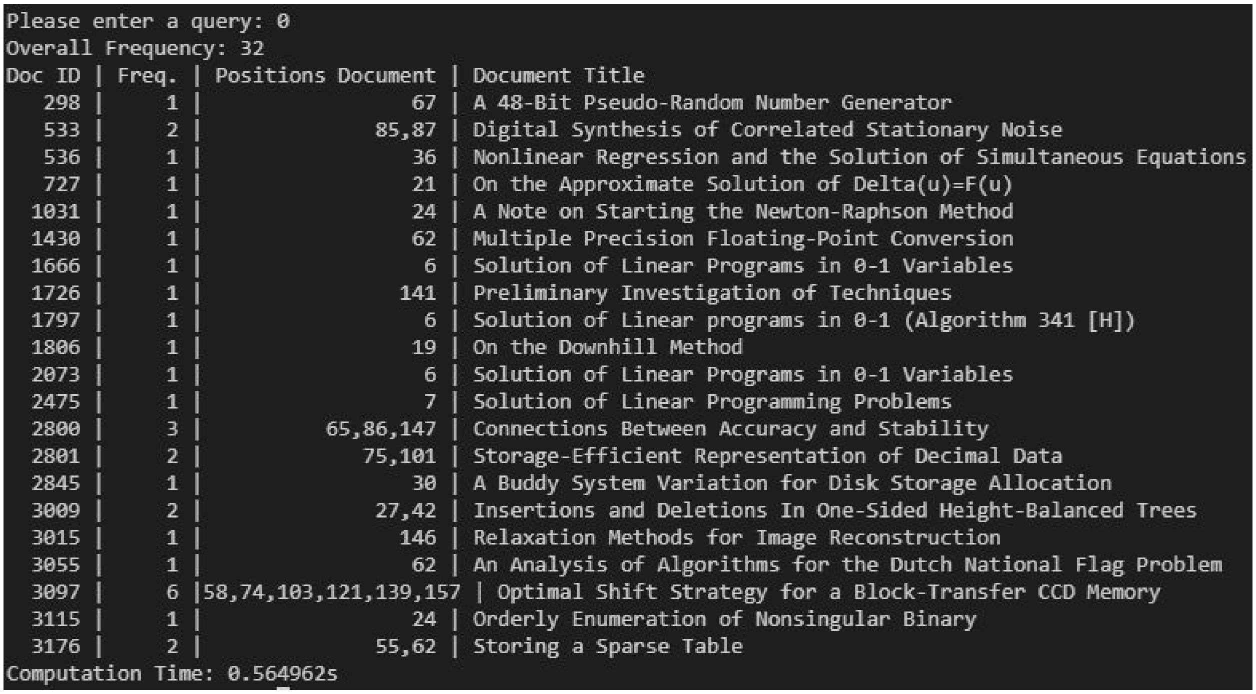
If the “make” command does not work:

1) Download and extract “cps842f19_a1_tmisic.zip” to the desktop of the machine.
2) Launch the application “Visual Studio Code”
3) Along the top taskbar, click File —> Open Folder —> Navigate to the Desktop —> Select “cps842f19_a1_tmisic” —> Select Folder
4) Along the top taskbar, click Terminal —> New Terminal
5) To compile “invert.cpp” as well as required header files, enter into the terminal: g++ -std=c++11 -pedantic -I. invert.cpp -o invert porter2_stemmer.o
6) To generate the text files, enter into the terminal: ./invert.exe [arg1] [arg2]
    a) arg1 can be a 1 or a 0 to enable/disable Stop Word Removal
    b) arg2 can be a 1 or a 0 to enable/disable the Stemming Algorithm.
7) To compile “test.cpp,” enter into the terminal: g++ -std=c++11 -pedantic -I. -o test test.cpp
8) To run the test executable, enter into the terminal: ./test
9) To exit the query program, query “ZZEND” or press CTRL + C

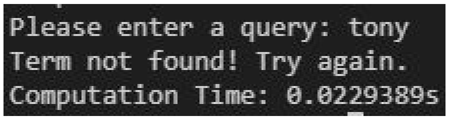
## Sample Outputs



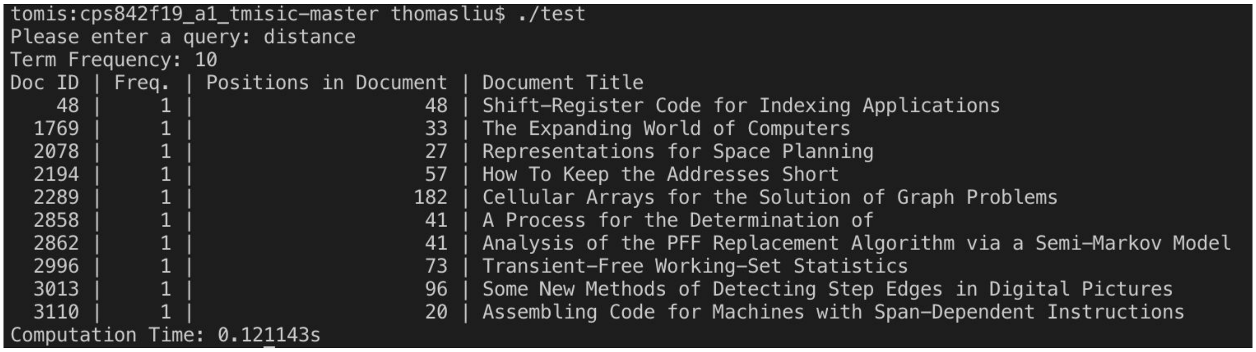
```
Please enter a query: zm
Overall Frequency: 4
Doc ID | Freq. | Positions Document | Document Title
   536 |     4 |        18,69,78,79 | Nonlinear Regression and the Solution of Simultaneous Equations
Computation Time: 0.0369309s
```



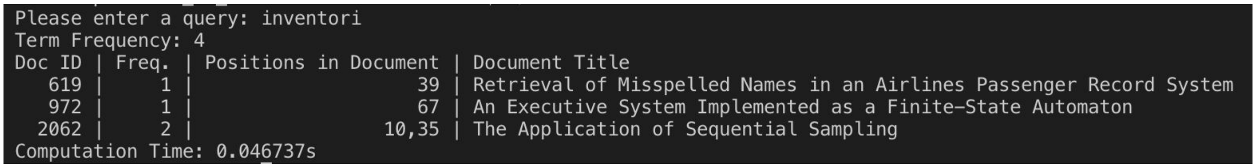
```
Please enter a query: 0
Overall Frequency: 32
Doc ID | Freq. | Positions Document | Document Title
   298 |     1 |                 67 | A 48-Bit Pseudo-Random Number Generator
   533 |     2 |              85,87 | Digital Synthesis of Correlated Stationary Noise
   536 |     1 |                 36 | Nonlinear Regression and the Solution of Simultaneous Equations
   727 |     1 |                 21 | On the Approximate Solution of Delta(u)=F(u)
  1031 |     1 |                 24 | A Note on Starting the Newton-Raphson Method
  1430 |     1 |                 62 | Multiple Precision Floating-Point Conversion
  1666 |     1 |                  6 | Solution of Linear Programs in 0-1 Variables
  1726 |     1 |                141 | Preliminary Investigation of Techniques
  1797 |     1 |                  6 | Solution of Linear programs in 0-1 (Algorithm 341 [H])
  1806 |     1 |                 19 | On the Downhill Method
  2073 |     1 |                  6 | Solution of Linear Programs in 0-1 Variables
  2475 |     1 |                  7 | Solution of Linear Programming Problems
  2800 |     3 |          65,86,147 | Connections Between Accuracy and Stability
  2801 |     2 |             75,101 | Storage-Efficient Representation of Decimal Data
  2845 |     1 |                 30 | A Buddy System Variation for Disk Storage Allocation
  3009 |     2 |              27,42 | Insertions and Deletions In One-Sided Height-Balanced Trees
  3015 |     1 |                146 | Relaxation Methods for Image Reconstruction
  3055 |     1 |                 62 | An Analysis of Algorithms for the Dutch National Flag Problem
  3097 |     6 |58,74,103,121,139,157 | Optimal Shift Strategy for a Block-Transfer CCD Memory
  3115 |     1 |                 24 | Orderly Enumeration of Nonsingular Binary
  3176 |     2 |              55,62 | Storing a Sparse Table
Computation Time: 0.564962s
```



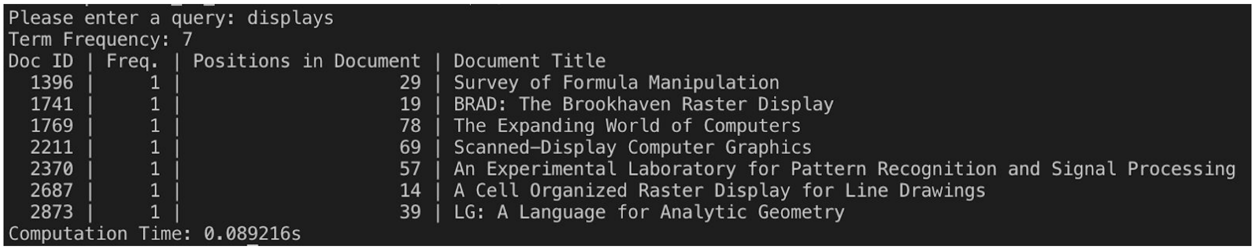
```
Please enter a query: tony
Term not found! Try again.
Computation Time: 0.0229389s
```



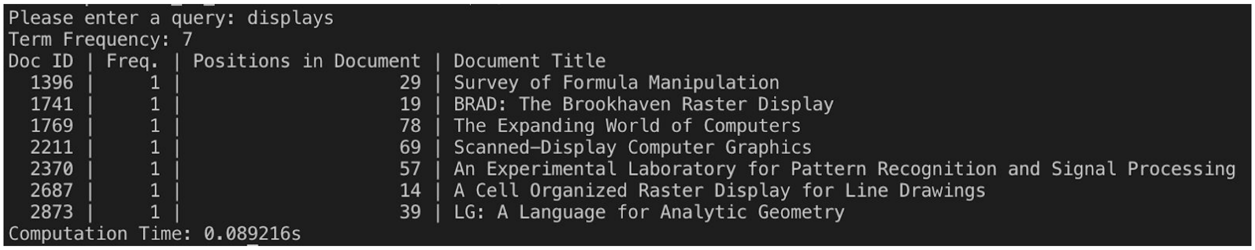
```
tomis:cps842f19_a1_tmisic-master thomasliu$ ./test
Please enter a query: distance
Term Frequency: 10
Doc ID | Freq. | Positions in Document | Document Title
    48 |     1 |                    48 | Shift-Register Code for Indexing Applications
  1769 |     1 |                    33 | The Expanding World of Computers
  2078 |     1 |                    27 | Representations for Space Planning
  2194 |     1 |                    57 | How To Keep the Addresses Short
  2289 |     1 |                   182 | Cellular Arrays for the Solution of Graph Problems
  2858 |     1 |                    41 | A Process for the Determination of
  2862 |     1 |                    41 | Analysis of the PFF Replacement Algorithm via a Semi-Markov Model
  2996 |     1 |                    73 | Transient-Free Working-Set Statistics
  3013 |     1 |                    96 | Some New Methods of Detecting Step Edges in Digital Pictures
  3110 |     1 |                    20 | Assembling Code for Machines with Span-Dependent Instructions
Computation Time: 0.121143s
```



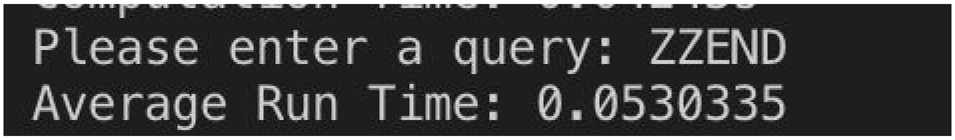
```
Please enter a query: inventori
Term Frequency: 4
Doc ID | Freq. | Positions in Document | Document Title
   619 |     1 |                    39 | Retrieval of Misspelled Names in an Airlines Passenger Record System
   972 |     1 |                    67 | An Executive System Implemented as a Finite-State Automaton
  2062 |     2 |                 10,35 | The Application of Sequential Sampling
Computation Time: 0.046737s
```



```
Please enter a query: displays
Term Frequency: 7
Doc ID | Freq. | Positions in Document | Document Title
  1396 |     1 |                    29 | Survey of Formula Manipulation
  1741 |     1 |                    19 | BRAD: The Brookhaven Raster Display
  1769 |     1 |                    78 | The Expanding World of Computers
  2211 |     1 |                    69 | Scanned-Display Computer Graphics
  2370 |     1 |                    57 | An Experimental Laboratory for Pattern Recognition and Signal Processing
  2687 |     1 |                    14 | A Cell Organized Raster Display for Line Drawings
  2873 |     1 |                    39 | LG: A Language for Analytic Geometry
Computation Time: 0.089216s
```



```
Please enter a query: displays
Term Frequency: 7
Doc ID | Freq. | Positions in Document | Document Title
  1396 |     1 |                    29 | Survey of Formula Manipulation
  1741 |     1 |                    19 | BRAD: The Brookhaven Raster Display
  1769 |     1 |                    78 | The Expanding World of Computers
  2211 |     1 |                    69 | Scanned-Display Computer Graphics
  2370 |     1 |                    57 | An Experimental Laboratory for Pattern Recognition and Signal Processing
  2687 |     1 |                    14 | A Cell Organized Raster Display for Line Drawings
  2873 |     1 |                    39 | LG: A Language for Analytic Geometry
Computation Time: 0.089216s
```



```
Please enter a query: ZZEND
Average Run Time: 0.0530335
```