Indexer

Program Usage:

./indexer <inverted-index file name> <directory or file name>

Program Description:

- Given a set of files, this program will parse the files and create an inverted index, which maps each token found in the files to the subset of files that contain that token.
- Indexer also maintains the frequency with which each token appears in each file.

Program Analysis:

- Indexer checks if the argument given is a directory or file. If it's a directory, indexer recursively indexes all files in the directory and it's sub-directories. If it's a file, that single file is indexed.
- A file is indexed by readsFile() which converts the file into a string the size of the file.[O(n) time and O(n) memory]
- The string is separated into tokens by TKGetNextToken(). [O(n) time]
- Each token is inserted into a hash table by addIndex() and SLInsert() inserts the token into a sorted list. [Worst case O(n^2)]
- Once all the tokens are processed, mergeSort() sorts the tokens in ascending order and SortWriteToFile() writes the results to the file previously determined. Sorting the tokens using mergeSort() is O(nlog(n)) time.
- Lastly, freeing memory allocated for the hash table and all its nodes. [O(n*m), n = number of tokens, m = average number of indexes per token]

Data Structures:

- Hash map used to store a list of unique tokens. Stores a list of unique tokens and maps to a list of indexes which store file path and token frequency. [O(1) time and O(n) memory for the hashmap]
- Linked list used to store tokens in alphabetical order.

Outside Contributions:

- UTHash – Hash table for C structures
- Author: Troy Hanson
  Source: https://github.com/troydhanson/uthash

Authors:

- Mauricio Trajano - Email: mauriciot1993@gmail.com
- Paul McNeil - paul.mcneil@rutgers.edu