Keyspace Construction

Hetal Patel - hp373
Rushabh Jhaveri - rrj28
Systems Programming, Spring 2018

Synopsis

This project's target is to create an indexer that tokenizes the files and produces an inverted index of how many times the word occurred in each file, sorted by word.
This project is composed of the following core files:

- index.c - this file contains the core logic for creating an indexer
- index.h - header file for index.c
- strArr2.c - this file is the stringsorter file from Asst0 with a couple changes made
- strArr2.h - header file for stringsorter

Design and Working

`void processAndReadFile(const char *, char *, char *, off_t);`

processAndReadFile gets executed when argv[2] is a file. It takes the filename, the output filename, the buffer and input file size as parameters. The function opens the file, reads it and stores the tokens in the buffer. The buffer is then passed as a parameter in the tokenizer function in strArr2.c file.

The function ***`extern char ** tokenizer(char *)`*** returns an array with sorted words. This sorted array is then passed as a parameter in ***`char ** getDistinctWords(char **, int, int *)`*** and ***`void getWordCount(int[], char **, int, char **, int)`*** functions to get distinct words and their frequencies.

`void writeToFile(int[], char **, int, const char *, char *);`

processAndReadFile calls writeToFile function and passes word count array, distinct words array and it's length, input file name and output file name. This function open the file to write to. If the file already exists, if asks the users permission to overwrite the file.

If the user enters ‘y’, the function loops through word count array and distinct words array and prints it to the file.

void processAndReadDir(const char *, char *, char *);

This function gets executed when argv[2] is a directory. It first opens the directory and then reads through it. If the encounters a file, it send it to processAndReadFileFromDir function. If it encounters a directory, it recursively calls itself and performs the entire process again.

void processAndReadFileFromDir(char *, char *, char *, off_t);

This function takes in the input filename, the output filename, the buffer, and file size. The buffer stores the tokens read from the input file. The buffer is then passed as a parameter in the tokenizer function in strArr2.c file.

The function ***extern char ** tokenizer(char *)*** returns an array with sorted words. For every word in the file, a hash index is then created by ***extern int getHashMapIndex(char *).*** After that, ***extern void buildHashMap(char*, char *, int)*** function is called and the word, filename and the respective hash index is passed and a hash map is created. The build hash map function calls ***extern void doInsertionSort(char *, char *, int)*** function and sorts the words along with the filename as it builds up the map.

After the hashmap has been created for every file in the directory, a master word linked list is created by ***extern void buildsortedMasterWordList()*** function. At the end, ***extern void printSortedMasterList(char *)*** function is called by main. This function writes to the output file passed as a parameter.

Analysis

The runtime analysis of this project is O(n^2), as in the worst case, we are iterating through a double loop.

The spatial analysis of this project is also O(n^2).