Arnesh Sahay & Krishna Yellayi Systems Programming
Assignment 4: Search November 14, 2014

Design

Data structures: This assignment required us to create a search function that worked off the output given by indexer.c and allowed the user to search with the 'AND' and 'OR' logical operators. The data structures implemented in our indexer included a hashtable of words and linked lists of filenames, both of which were reused in creating this application. The hashtable and file linked list functions were restructured from indexer.c into util.c. The hashtable was an array of size 36 elements, each corresponding to integers or letters (1-9 & a-z). The words were hashed to each of the 36 buckets using the first letter of the char string as the key, and stored in alphanumerical order in the linked list of words corresponding to each bucket in the hashtable. Further, each word was attached to a linked list of filenames in which the word appeared. The search function searches through the hashtable to find the word and prints the linked list of documents in which the word appears.

Functionality: The inverted index output by indexer.c is parsed and stored into the hashtable data structure described above. The loadFile function performs this task by parsing the inverted index file, hashing words to the hashtable, and appending filenames to the linked list of documents in which the word appears. Featured in this program are 'sa' and 'so," two types of search queries containing multiple terms. The search function 'sa' searches for documents in which both search terms appear, and the search function 'so' searches for all documents in which either or both search terms appear.

Memory: When the program is valgrinded, the memory usage comes out to 880 Bytes of data.

Complexity Analysis

The worst case time complexity to search a term in the hashtable is O(n), thanks to the hash function. Due to the hash function, the program starts searching for the in the correct bucket and only searches the terms which begin with the same alphanumeric character, effectively reducing the list of terms to search through by a magnitude of 36 (assuming equivalent terms beginning with each alphanumerical character). The search function also has to traverse the filename list to return the list of documents in which the term appears, effectively adding a time complexity of O(m). Therefore, assuming there are n total terms to search through and m documents in which the n terms appear, the worst case time complexity is O(n x m).