CS2022 Programming Project
Project Report

Part A - The Design

The abbreviated idea of this e-book searching tool can divide into three major parts.

Step 1 - Read the input file and store them in a manner which can retrieve them efficiently.

Step 2 - Search the relevant Keywords and Page numbers efficiently. And sort the relevant page numbers.

Step 3 - Execute the queries and output the results.

The program execution starts with reading the first line of the input file. As soon as it reads the command “Keyword List Start” it start to read the keywords and the corresponding page numbers until it reach the command “Keyword List End”. As the keyword and the corresponding page number is in the same line of the input text file, the StringTokenizer (java.util.StringTokenizer) has been used to read them separately. The keywords and the page numbers will be inserted to two separate Hash Tables to store and retrieve data efficiently. For each keyword program will calculate a hash value, and that hash value will be the index (key) of the storing point of the hash table. First hash table, the key will be generated from the **Keyword** and the data item will be the **Page Number.** For the second hash table the key will be generated from the **Page Number** and the data item will be the **Keyword.**

Step 2

After that, the program starts to read the queries and understand them. If user wants to find the keywords in an appropriate page (Queries starts with **Keywords**), the program will access the second hash table and provide all the keywords in that page.

If the user wants to find any page number for an appropriate keyword, first of all, the program will insert all the pages which contain that keyword to a linked list. While inserting, the program uses the insertion sort algorithm to insert the page numbers to the linked list in ascending order (The program do not sort the page numbers unnecessarily). So if the user wants the first page (Queries starts with **First**) which includes the appropriate keyword, then the program will provide the first element of the linked list.

If the user wants all the page numbers (Queries Starts with **List**) which includes the appropriate keyword, then the program will provide the complete linked list in sorted order.

Part B - The Data Structures and Algorithms Used

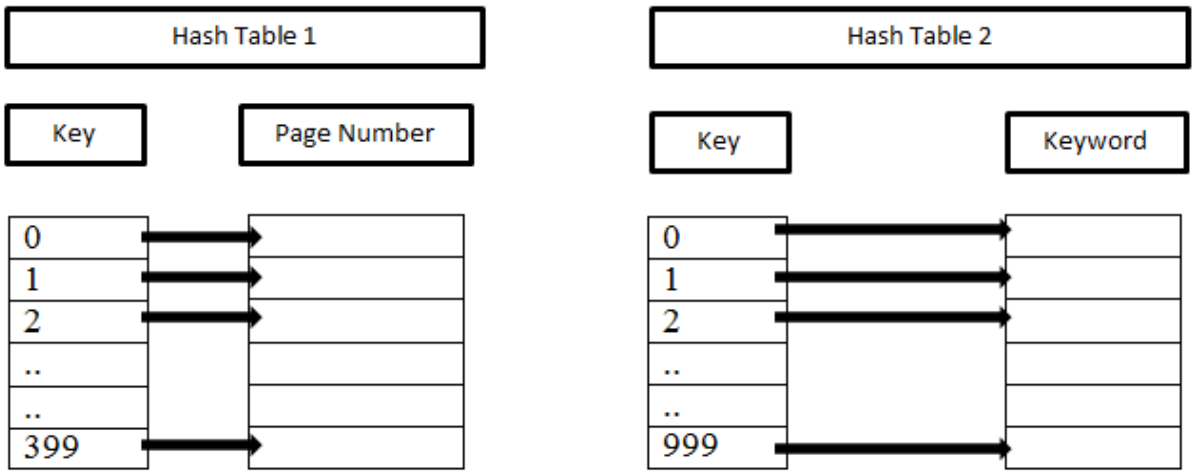
Data Structures

- Hash Table
- Linked List

Sorting Algorithm

- Insertion Sort*(modified version of insertion sort for Linked List insertion)*

Hash Table is a data structure which offers very speed insertion and searching. So no matter how many data items there are insertion and sorting consumes constant time for each operation (O(1) in big O notation). In practice this is just a few machine instructions. Hash tables are significantly faster than trees which operates in relatively speed time (O(logN) time). Hash tables are not only fast but also they are easy to program.
So in the program I used two separate hash tables for store the data items. In the first hast table the key will be generated according to the keyword and it will save the appropriate page number as data. In the second hash table the key will be generated according to the page number and it will save the appropriate keyword as data. If same keyword or page number occurs more than one time the program will use **linear probing** to save them in an efficient manner. Also I choose the hash array size double than the required size to **avoid clustering**. The final structure is shown below.



As user need the list of page numbers in a sorted order (Queries 'First' & 'List'), I have implement a **Sorted Linked List.** There are many reasons to choose a sorted linked list as follows. In unordered array, searching is slow, where as in ordered array, Insertion is slow, also the size of an array can't be changed after its created. So if we use a linked list instead of array then all the above problems are solved easily. Also the linked lists are surprisingly versatile and conceptually simpler than some other popular structures such as trees. However the major reason to use a linked list is the space efficiency of it.
While inserting items to the linked list, I used a modified version of insertion sort to create the sorted linked list. It is true that general unordered array insertion sort algorithm requires $O(N^2)$ time in the worst case scenario. But in this algorithm it uses only O(N) time to insert a item to the linked list in the worst case scenario.
My algorithm do not sort all the keywords. It's a waste of time. Even though there exists 500 or more keywords as inputs, if there are only 3 operations in the queries my algorithm will only sort those 3 keywords which were given in the operation instead of sorting all the 500 keywords.
The time complexity of inserting an item to the hash table is O(N), searching an item in the hash table O(N), adding item to the sorted linked list O(N). So finally the time complexity of my complete program is O(N) in worst case scenario.

Part C – Assumptions

- Assume that the input file will be in the same format which is given in the sample inputs.
- Assume that the maximum number of page numbers is 200.
- Assume that the maximum number of keywords is 500.

Part D – The Problems Faced

In the first version of this program, I used only one hash table and the key is based on the input keyword. So I understood that when we searching keywords in an appropriate page number, it searches the complete hash table and it is a worst case scenario (actually it behaves like a simple array). So I understood that instead of using one hash table we can use two hash tables, one to search keywords and the other to search page numbers. After including the second hash table the time complexity of the program improves significantly.

Part E – Discussion

In this program I have used insertion sorting algorithm to sort the page numbers. But I have to admit that it is not an efficient way for sorting. Instead of insertion sort if I use heap sort or merge sort then the time complexity will be a better value. But in this scenario I used insertion sort to add elements to a linked list, so using heap sort or merge sort while inserting items to a linked list simultaneously will be very much complex. The simplicity of a program is also an important issue. However if we use heap sort or merge sort we can easily improve the time complexity of this algorithm.