School of Computer Science
University of Guelph

CIS*6030 Information Systems

Fall 2022
Instructor: Fangju Wang

Assignment 1 (100%)

Text file A1_data.txt contains about 46,880 lines of character strings. Each line is a data record. A record has two fields: Field 1 – the first string, Field 2 – the second and third strings, Field 3 – all the rest strings. The values of Field 1 are unique. This field is the key of records.

Question 1 (60%)

Write a program that reads the text file, stores the records in a database file, and creates a B^+-tree to index the records.

In the database file, records are sorted by Field 1. The records are variable-length. The block size of the database file is 1024 bytes (1K). When the database file is initially created, each block should have some free space for possible later insertions. However, the total free space of the database file should not be greater than 20% of the file size.

Create a B^+-tree on Field 1, and store it in a disk file. The B^+-tree stores key-pointer pairs. The parameter m of the B^+-tree is 8. The block size of the index file is 512 bytes. Store a tree node in a block.

Write another program for searching, insertion and deletion. It can do the following:

1. Searching a record by entering a key value,
2. Range searching by using the index,
3. Inserting a new record,
4. Deleting a record with a key value, and
5. Display all the records sequentially ordered by the first field.

This program should have a menu for the above. A GUI is not required.

When doing a **search**, the program reads ONLY the tree nodes accessed and the data block that contains the target record. Each time a tree node or a data block is read in RAM, all of the keys at it are displayed. This must be done so that your program can be correctly graded. *Your program should report the number of disk reads for each search operation.* When a record is found, the program should display the results like

```
Field 1: nwlrbbmqb
Field 2: the information
Field 3: published in this undergraduate calendar outlines the
```

When doing **insertion** and **deletion**, your program can read the entire tree into the main memory, and write it back to disk when the modification is completed. It is not required to display any information. Your program should be able to handle the cases that a data block is full in insertion and a data block becomes empty in deletion. Remember that the records must always be sorted by the first field.

In short, for this question, you write two programs. The first program creates a database file and a B^+-tree, and writes them to disk. The second program accesses the tree and database file on disk to perform search, insertion, deletion, etc., and writes back to disk when needed.

Question 2 (40%)

Create a linear hash table to store the data in A1_data.txt on disk.

For the hash table, design a hash function, a bucket structure, and a search algorithm. Choose a bucket/block size. Please briefly describe and justify your designs and choices in your *readme* file.

Write a program to implement the hash table. The program stores the hash table on disk.

Write another program to access the hash table on disk, to perform search and insertion. The program is not required to do deletion. *Your program should report the number of disk reads for each search operation.*

In your *readme* file, please also briefly compare search performance of the hash table and the B^+-tree indexing, in term of the numbers of disk reads.

Note:

Write your programs in C or C++. Pack your work into a tar file containing your source code, makefile, and a *readme* file, which includes your name and ID, and your description and discussion for Question 2. Submit your tar file to Moodle.socs.uoguelph.ca by the due time.

Make sure your code can be compiled and executed on the SoCS Linux systems. There should be a *makefile* for compiling the programs. Any compilation warning will result in a mark deduction. Use the *-Wall* compilation option to check warnings. Please add adequate comments in your code. Please tell how to compile and execute your programs in your *readme* file.

- This assignment must be completed **individually**.
- Write your own code to implement your B^+-tree and hash table. Do **not** use any code from any sources.
- All submissions will be checked by anti-plagiarism software.

- Late submissions will not be graded.

Submission due time: 23:59, September 30, 2022 (Friday).