New York University Ab New York University Abu Dhabi

CS-UH 3010 - Fall 2021

Programming Assignment 1

Mengjie Shen-ms11141

**The structure of the project included:**

* Makefile: for easier compilation
* Header file: tree.h

I define three struct inside, which is llNode (node for linked list), node (b+ tree node) and h\_node (node for calculating course total times of enrollment). I also listed all the functions that I implemented.

* C code:

main.c: The main function with interface to run my program

dictionary.c: compute the top-k (number) more popular courses taken by all students in the semester.

insertion.c: The function used for inserting single value to the b+ tree.

function.c: All the basic implementation function for b+ tree.

**Design Idea:**

I used a b+ tree with linked list linking to my leaf node to retrieve and store data. For the node in the b+ tree, there is two pointers: \*\*pointers and \*\*record. In the non-leaf node, I use \*\*pointers to point to the corresponding child nodes and let the \*\*record remains null. In the leaf node, I use \*\*record to point to the corresponding linked list. I set the \*\*pointers to be null, except the last pointer of each node pointing to the next node. There is also a pointer called next in each node, this is used for enqueue and dequeue operation in printing the tree.

Therefore, I put record with the same student ID in one linked list and link the list to the leaf. When inserting a new record, I will find if the key is already existed in the tree. If so, I will directly append the record in the corresponding linked list. If not, I will add a new key to the correct node. If the node is already full, I will split the node.

For calculating the top k courses, I defined another linked list using h\_node to construct a “dictionary” (Since it’s a linked list, the run time of finding key is O(n) instead of O (1)). Every time, I insert a new data, I insert the record into the b+ tree and update the dictionary as well. When I need to get the top k courses, I will use the merge sort to sort the dictionary key and get the first k key-value pairs.

For printing the tree, I used Breadth First Search and constructed a queue. By using enqueue() and dequeue() to print the node.

For calculating single student’s GPA or finding the information of a single student, I start from the root node and comparing the key value. If the key value of the target is larger, I goes to the right pointer, otherwise, I go to the left pointer.

For calculation several student’s GPA or finding the information of several students, I first find the leaf node of the starting student ID and goes right until the key value is larger than the end student ID.

For memory free, I use recursion to goes to the leaves. I first freed the linked list and then free the corresponding node.

**To start the program:**

Step 1: Go to the directory

Step 2(optional): You can change the size and input file on line 18 in the form of

./main -k <size> -f <fileName>

Step 3: Run command line to start: make run

After the program starts to run, besure to type in “in” to initiate the program

Step 4: To finish the program: type in "e" to exit

Besure to run command line: make clean to remove the output file

**Difficulties and what I learned:**

This is actually my first time writing so many lines of c code. Because back to my computer architecture courses in NYU Shanghai, the work is usually twenty or thirty lines of code at most. I found it very challenging as well as rewarding.

The biggest problems I met is the segmentation fault/ pointer allocation problem. Since I need to read the file using a while loop. Every time the value needs to be renewed. However, I was renewing the value from the original pointer, instead of allocating a new memory for a new record. This leads to the problem that the value in the tree was overwritten by the value I was about to insert. I was stuck at this bug for hours. I found that the most difficult part of programming in C is to understand the pointer problem and memory allocation.

**Citation:**

1. For the b+ tree, I used part of the code from <https://www.geeksforgeeks.org/insertion-in-a-b-tree/>.
2. For the merge sort used to calculate the top k courses, I used part of the code from <https://www.geeksforgeeks.org/merge-sort-for-linked-list/>
3. For reading in the .csv file, I got knowledge from <https://stackoverflow.com/questions/13465264/reading-csv-file-into-c-linkedlist>