Introduction to Database Systems

Individual Homework 2: B+ Tree Index

1. Introduction

In the lecture, you had learned what an index is in the database, the purpose of building an index, as well as a common form of index in the database, which is B+ tree. In this homework, you are required to implement a B+ tree class and use it as a simple index for given data. Please continue reading for details.

2. Tasks

You need to write two files, "**index.h**" and "**index.cpp**", which are the header file and the cpp code of your index class, the class is named "**Index**".

The provided files are explained below:

- makefile
    - TAs will use provided makefile to compile your code when grading
- main.cpp
    - contains the main function which will be used to test your homework
    - main function will read all files needed for you, as well as record time used by each task
    - **DO NOT** modify this file (except uncomment the test script)
- utils.cpp
    - contains some utility functions for file reading and used time output
    - **DO NOT** modify this file
- utils.h
    - header file for utils.cpp
    - **DO NOT** modify this file
- data.txt

    - contains the data used in the homework
    - each line consists of a key and a value, both are integer and are separated with a “,”
    - the data is not sorted on key or value
    - will be loaded as two vectors of integer in the main function
    - note that the data TAs use to test your program will not be identical to the one provided
- key_query.txt
    - contains the queries to test your index
    - each line consists of a key, which is an integer
    - will be loaded as a vector of int in the main function
    - some of the key may not exist in the data
    - note that the queries TAs use to test your program will not be identical to the one provided
- key_query_ans.txt
    - answer to key_query.txt
- range_query.txt
    - contains the queries to test your index on range search
    - each line consists of two keys, which are integers, the first one is always smaller than second one, they are separated by a “,”
    - will be loaded as a vector of pair of two integers in the main function
    - note that the queries TAs use to test your program will not be identical to the one provided
- range_query_ans.txt
    - answer to range_query.txt

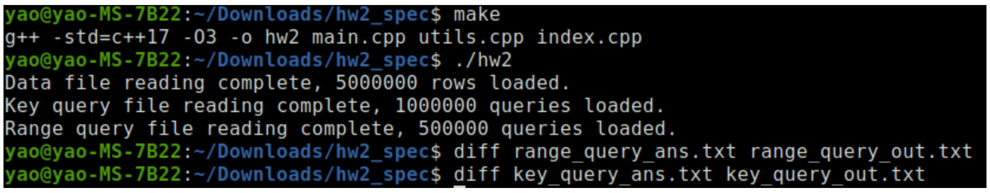
You need to write two files, the detail requirements are explained below:

- index.cpp
    - the code for your index, this file must contain a class named “**Index**”
    - Index class is an implementation of B+ tree index
    - At least four function needed to be implemented (add more if you need)
        - constructor Index()
        - key_query
        - range_query
        - clear_index
    - Index(num_rows, key, value) takes three inputs, which are an integer indicating the number of data rows, a vector of integer represent keys and a vector of integer represent values. You need to construct your B+ tree index in this function by inserting the key, value pairs into the B+ tree **one by one**.
    - key_query(query_keys) takes one input, which is a vector of integer indicating the key used for query. In this function you need to output a file named “key_query_out.txt”, each row consists of an integer which is the value corresponding to the keys in query_keys (output -1 if the key is not found).
    - range_query(query_pairs) takes one input, which is a vector of pairs of two integers indicating the range of query. In this function you need to output a file named “range_query_out.txt”, each row consists of an integer which is the **MAXIMUM** value in the given query key range (output -1 if no key found in the range).
    - clear_index() takes no input, you need to free all the memory used by your B+ tree index in this function.
- index.h
    - header file for index.cpp

To summarize, you need to implement a B+ tree index class which is capable to insert some key, value pairs into it, then process (two types of) queries with the index, a function to release used memory is also required (we do not require your class to be capable of deleting instance in the B+ tree).

Please note that the data files TAs use to test your program will be different to the ones provided, even in the number of rows and queries. Remember to make your program work with these differences in data.

If you successfully run the code and do the tasks, your program will output 3 files, “key_query_out.txt”, “range_query_out.txt” (generated by functions you implemented) and “time_used.txt” (automatically generated, you do not need to implement this). Below figure is the example way to examine your correctness.



```
yao@yao-MS-7B22:~/Downloads/hw2_spec$ make
g++ -std=c++17 -O3 -o hw2 main.cpp utils.cpp index.cpp
yao@yao-MS-7B22:~/Downloads/hw2_spec$ ./hw2
Data file reading complete, 5000000 rows loaded.
Key query file reading complete, 1000000 queries loaded.
Range query file reading complete, 500000 queries loaded.
yao@yao-MS-7B22:~/Downloads/hw2_spec$ diff range_query_ans.txt range_query_out.txt
yao@yao-MS-7B22:~/Downloads/hw2_spec$ diff key_query_ans.txt key_query_out.txt
```

3. Grading

In this homework, the correctness of your code only makes part of your score, to encourage you to optimize the index (e.g. search algorithm, tree order), part of your score will be evaluated by the time used for the subtasks. The details are explained in the table below:

Description	Score(%)
Correctness of “key_query_out.txt”	20%
Correctness of “range_query_out.txt”	20%
Correctness of B+ tree implementation	20%
Time used to build index	10%
Time used to process key query	15%
Time used to process range query	15%

There will be a -10% penalty if your clear_index() is wrong.

You will get at least 60 points if your output and implementation are correct, the other 40 points are based on performance of your code. For each subtask, we will rank all time used for the subtask of all students (wrong answer or no submission will not be included), then give score based on the PR(percentile rank) value:

- PR >= 90: you get 100% of the performance score
- 60 <= PR < 90: you get 75% of the performance score
- 20 <= PR < 60: you get 50% of the performance score

- PR <= 20: you get 25% of the performance score
- If your answer/implementation is wrong, you will not get any points in the performance score

With the exception of "correctness of B+ tree implementation", the homework will be revised automatically with a script. We will use your "index.h" and "index.cpp" with other provided codes to compile the program, we also provide the makefile we will use for your reference. The C++ version used will be C++17. Please make sure your code works with provided files and given settings.

4. Discussion

TAs had opened a channel **HW2 討論區** on New E3 forum of the course, you can post questions about the homework on the forum. TAs will answer questions as soon as possible.

Discussion rules:

1. Do not ask for the answer to the homework.
2. Check if someone has asked the question you have before asking.
3. We encourage you to answer other students' questions, but again, do not give the answer of the homework. Reply the messages to answer questions.
4. Since we have this discussion forum, do not send email to ask questions about the homework unless the questions are personal and you do not want to ask publicly.
5. If you are not familiar with asking questions precisely, you can refer to **how to ask questions the smart way**.

5. Submission

1. The deadline of this homework is **5/12 (Wed.) 23:55:00,** no submission allowed after **5/16 (Sun.) 23:55:00**.
2. The submission only requires two files: index.h and index.cpp
3. You should put your "index.h" and "index.cpp" into one folder, the folder should be named as "**HW2_XXXXXXX**" where XXXXXXX is your student ID.
   ex: **HW2_123456**

   Then compress your folder into one `zip` file. Submit it to New E3 System with the format **HW2_XXXXXXX.zip** where XXXXXXX is your student ID.
   ex: **HW2_123456.zip**

   We **only accept one zip file**, each wrong format or naming format cause -10 points to your score (after considering late submission penalty).

4. Late submission lead to score of (original score)*0.85^{days}, for example, if you submit your homework right after the deadline, your get (original score)*0.85 points.
5. **0 points will be given to Plagiarism**. If the codes are similar to other people and you can’t explain your code properly, you will be identified as plagiarism. TAs will strictly examine your code. It is important that you must write your code by yourself.
6. If there is anything you are not sure about submission, ask in the discussion forum.