McMaster University
CS/SE 4AD3 Winter 2024
Assignment 2
Due: March 7, 2024 at 10:00pm

February 14, 2024

I. Cost Analysis (25 marks)

Question 1 (10 marks)

Suppose we have an Alternative-2 unclustered index on (assignment_id, student_id) with a depth of 3 (the root is at depth 1). Here's the schema:

```
CREATE TABLE Submissions (
 record_id integer UNIQUE,
 assignment_id integer,
 student_id integer,
 time_submitted integer,
 grade_received byte,
 comment text,
 regrade_request text,
 PRIMARY KEY(assignment_id, student_id)
);
CREATE INDEX SubmissionLookupIndex ON
             Submissions (assignment_id, student_id);
```

Assume the table takes up 12 MB on disk (1 MB = 1024 KB, the 12MB does not include the index, only the heap files in which the records are stored) and that page size is 64 KB. (This includes extra space allocated for future insertions.)

(a) (4 marks) We want to scan all the records in Submissions. How many I/Os will this operation take? Show the derivation of your answer.

(b) (4 marks) How many I/Os will the below UPDATE statement take? Show the derivation of your answer.

```
UPDATE Students SET grade_received=85
WHERE assignment_id=20 AND student_id=12345;
```

(b) (2 marks) In the worst case, how many I/Os does it take to perform an equality search on grade_received? Briefly justify your answer.

Question 2 (15 marks)

For the following questions, consider this SQL table, and the following three queries:

```
CREATE TABLE Students (
sid INT PRIMARY KEY,
name VARCHAR(20),
email VARCHAR(20),
GPA float,
years_enrolled INT);
```

- Query A:

  ```
  SELECT * FROM Students WHERE years_enrolled >= 2
  ```

- Query B:

  ```
  DELETE FROM Students WHERE sid = 123456
  ```

- Query C:

  ```
  INSERT INTO STUDENTS VALUES (9999, 'Joe', joe@b.com, 3.0, 2)
  ```

For the following questions, assume there are no indexes, and we leave each page about 2/3 full.

(a) (5 marks) Order these queries in ascending order by the total number of I/Os required to execute them; ties can be listed in any order. Justify your answer.

(b) (10 marks) If Students table is arranged in a sorted file, sorted on sid, what is the order of queries? If there is a tie, write both answers Justify your answer.

II. Buffer Management (12 marks)

Consider the MRU buffer replacement policy, where you have 4 buffer pages, with the access pattern {A F A D G D G E D F}. The number of rows indicate the number of buffer frames, and each column represents a single access. If there is a hit, indicate this with an asterisk (*) and if there is a miss, indicate the new value. Assume buffer page 1 contains A, buffer page 2 contains B, buffer page 3 contains C, and buffer page 4 contains D.

(a) Fill in the following table showing the series of hits and misses, and pages that are brought into the buffer pool.

(b) What is the hit rate at the end of the access pattern?

A										
B										
C										
D										

III. External Sorting (73 marks)

In lectures, we discussed external merge sort that sorts files too large to fit into memory. Most external sorting methods use the following general strategy. Make a first pass through the file to be sorted, breaking it into blocks about the size of the memory pages, and sort these blocks. Then, merge the sorted blocks together, if necessary, by making several passes through the file, creating successively larger sorted blocks until the whole file is sorted.

In this part of the assignment, you will implement external merge sort to sort a file of fixed size records, stored in a file on disk, using a given amount of main memory. The input data to your code will run into Gigabytes, so you should make sure your program can handle data sizes of more than 1GB. You are free to choose any programming language from C/C++, Java or Python. In the Assignment folder, you will find an example C template (template.C) file containing sample class templates that you will need to populate.

You are free to add our own classes, e.g., buffers of records that will handle input/output. Reading and writing is done using the read and write system calls. The IO streams need to maintain a size B buffer inside them so that they can read or write B size blocks on disk at a time. Note that you are given M = size_of_memory = `amt_of_mem` and the choice of B should be such that all the buffers in your code do not make your code allocate more than size `amt_of_mem`.

Requirements:

1. You are not permitted to use any static, constant variables in your code.

2. Use underscore separators or camel case to separate words in your variable names.

3. Your code should not use more than 2x the input file size of disk space at any given time. For example, if your input file is 1 GB, your code cannot create temporary files of size 3GB on disk. The maximum amount of disk space your code can use at any given time is 2GB. You may destroy the input file while creating the output file if that helps.

4. You must create a file that documents your functions, their inputs/outputs, and its functionality. This file should be no longer than 2 pages. Name this file `docs.pdf`.

5. Compress all your files into a tar file, `sorting.tar`, and include a makefile to create an executable that the TA can run. It should include a README that provides instructions on how to run the code on an input file (details below). For example, if the input file is `input.dat` where each record is 100 bytes, key size is 8 bytes, sorted in ascending order (indicated with 1), with 32MB memory, the command to run is:

   ```
   tar -xvf sorting.tar
   ```

```
make
extsort input.dat output.dat 100 8 32 1
```

6. You may assume you have at least 32MB of memory.

Input files. In the assignment folder, you will find SortGen.tar that produces example input files. The example run outputs 100 records.

```
tar -xvf SortGen.tar
make
./SortGen 100 input.dat
```

There is also a sample input file, `sample.dat` generated by SortGen with 2000000 records that you can use to test your code with a memory size of 16MB. You can check if your code can sort this data by keeping memory usage below 16MB. Key sizes are 8 bytes.

Evaluation:

1. We will evaluate your code on the db4ad3.cas.mcmaster.ca server.
2. You can monitor memory usage using the `top` command. The first evaluation criteria is that the file is sorted correctly. Even if you have a fast algorithm, but the file is not sorted, this will be penalized. Correctness is the first priority. You can use `df` to check disk space on Linux.
3. Ensure your code is well documented.
4. As a baseline, your implementation should be faster than randomized quicksort.
5. You may use any programming language of your choice as long as the TA can run your executable (as in Step 5 of Requirements) on the db4ad3.cas server.

Grading

This assignment is worth 14% towards your final grade.
You may work in groups of up to 2 members for this assignment. Clearly indicate the names and student numbers of all your group members in a cover sheet along with your typed answers in file `costBuf.pdf`.

Submission

All the below files are to be submitted on Avenue in the Assignment 2 folder.

- For Part I, II: Submit your **typed** answers in a file called `costBuf.pdf`. **Handwritten answers will not be accepted.**
- For Part III: Compress all your files into a file named `sorting.tar`. Ensure your code is executable on the class db4ad3.cas.mcmaster.ca server. Non-executable code (that do not run on the server) will not be marked, and will receive a grade of zero. Submit your documentation in a file `docs.pdf`.