Database Systems Technology (CSC443-W2019)
Assignment 2

Submission Deadline: Mar 03, 2019 11:59 PM

1 External Sorting - 35%

Attached is very simple heap file (**names.db**). It is just a series of fixed-length records. The length of each record is **64 bytes** and they have the following 4 fields:

1. First Name (12 bytes)
2. Last Name (14 bytes)
3. Email (38 bytes)

Although each field has a fixed length but the content can have different length. The remaining of the field is filled by ASCII(0). For example the "First Name" field of the first record has the following format:

48 65 72 6D 69 6C 61 00 00 00 00 00 = Hermila

This file has no header or any extra information. It is just a sequence of records.

Write a program that accepts the following parameters and uses the "External Sorting" algorithm we discussed in the class to sort the file based on the given field and creates another sorted DB file:

1. inDB (the name of the DB file to be sorted)
2. outDB (the name of the sorted DB file with the same format as inDB)
3. B (number of buffer pages in memory)

4. pSize (data page size that is a multiply of 64)
5. field (the index of the field to be sorted by; 0=First Name, 1=Last Name, 2=Email)

You must only use **B** number of pages in memory (as buffer for the data pages) and should read and write from/to the DB files page by page. The program needs to print the number of passes and the total number of pages read or written (separately).

Run your program for the combination of the following parameters:

1. pSize = 512, 1024, 2048
2. B = 3, 10, 20, 50, 100, 200, 500, 1000, 5000, 10000
3. field = 1

Plot a line graph with x-axis as the number of buffer pages (B) and y-axis as the total pages read or written. This plot should show two line graphs for different page sizes in one plot.

2 Hash Indexing - 40%

Write a program to create hash index for the same DB file explained in the previous section (**names.db**). The program should accept the following input parameters:

1. inDB (the name of the DB file to be sorted)
2. indexFile (the name of the index file to be created)
3. indexType (the type of hash index: 0=Static, 1=Extendible, 2=Linear hashing)
4. buckets (the initial number of buckets that is a power of 2)
5. pSize (data page size that is a multiply of 64)
6. field (the index of the field to be sorted by; 0=First Name, 1=Last Name, 2=Email)

Your program should do the following:

1. Store the index structure in a separate file named by the given input parameter (**indexFile**).
2. Use alternative 2 for data entries (key, rowid).
3. Use the same page size as **pSize** for the index pages.
4. Design your own format for the index file but make sure you have all the required info in the index file (e.g. page size, index type, any required directory, ...). You need to explain in details your selected format and the reasons behind your choices. Use diagrams as much as possible to make it more readable.
5. You don't need to create the index file record by record. You can construct the index for all records in memory and then write them to the index file at the end.
6. For the main hash function use **MD5** hash function. You don't need to implement your own MD5 function. You can use the built-in/existing functions in Python or C++.
7. It may not be possible to have only one data entry page per bucket in Extendible Hashing. Increase the size of the directory to get the minimum possible number of pages per bucket.
8. For linear hashing, the splitting process is triggered only when a new overflow page is created.
9. Your program needs to print the following info:
   (a) the final number of buckets
   (b) the number of regular index pages
   (c) the number of overflow pages
   (d) histogram of number of index pages per bucket. Use a proper bin size to get 10 bins.

Run your program to create three index files for the following parameters:

1. pSize = 1024
2. field = 0
3. indexType = 0, 1, 2

For each run, plot the histogram of number of index pages per bucket.

3 Query by Index - 25%

Write a program to make a query and prints the records found. The program will accept the following input parameters:

1. dbFile (the database file)
2. indexFile (the index file you created in the previous section)
3. field (the index of the field to be sorted by; 0=First Name, 1=Last Name, 2=Email)
4. value (the value to be searched, field = value)

Your program need to print the following info:

1. All the fields of all found records
2. The number of pages read or written separated by their type (i.e. index page, data page)

Run you program against the three index files you created in the previous section to find "First Name = Nona"

4 Submission

You need to submit the following to MarkUs before deadline:

1. A report including all the results from all questions. You need to include the screenshots of when you run your programs and the output that is printed. The report should be in PDF format.
2. You need to attach the source code of all scripts and programs. They should be written modular and should be self-documented.
3. Do not include the generated data or index files.