Written Assignment #2

Due: Oct 18 (Tuesday) in Class.

Problem 1. [20pts]

Consider a disk with a sector size of 500 bytes, 1000 sectors per track, 2000 tracks per surface, five double-sided platters (i.e., 10 surfaces). The disk platters rotate at 6000 rpm (revolutions/rotations per minute). The average seek time is 4ms (millisecond).

Suppose that the block size is 4000 bytes (4 Kbytes). Consider a **Sorted File** with 10^6 records of 100 bytes each that is stored on the disk. No record is allowed to span over two pages. All pages in this file are stored next to each other on the disk. Answer the followings (*you do need to show your equation/calculation*).

1. What's the total capacity of this disk (in bytes)?[2 points]
2. How many pages are there from each track, each cylinder, and the entire disk? [3 points]
3. How many pages are required to store the file? [3 points]
4. Assuming the disk arm head starts at a random location, how long does it take to read the content of the first 100 pages in the file? [4 points]
5. Suppose each record in the file is a Course record with the schema **Course(<u>cid</u> int, name char[92], capacity int)**. int is 4 bytes. Assuming the disk arm head starts at a random location and the file is **sorted by cid**, How long does it take to answer the following query? **Briefly justify your answer**.
   - select * from Course where name like '%Database%'. [4 points]
   - select * from Course where cid = 5530. [4 points]

Problem 2. [10pts]

In this exercise, you will compare the LRU (least recent used) and MRU (most recent used) buffer management algorithms for a given workload. Here are the assumptions you must make:

- Assume there are four page slots your buffer manager must manage: P1, P2, P3, and P4.
- All four slots are empty to start.
- When the buffer pool has unused slots (such as at the beginning, when all four slots are empty), it will put newly read data in the leftmost empty slot (e.g., if slots 2 and 3 are free, it will use slot 2).
- The pages to be read from disk are labeled A through G.
- For each access the page is pinned, and then immediately unpinned.

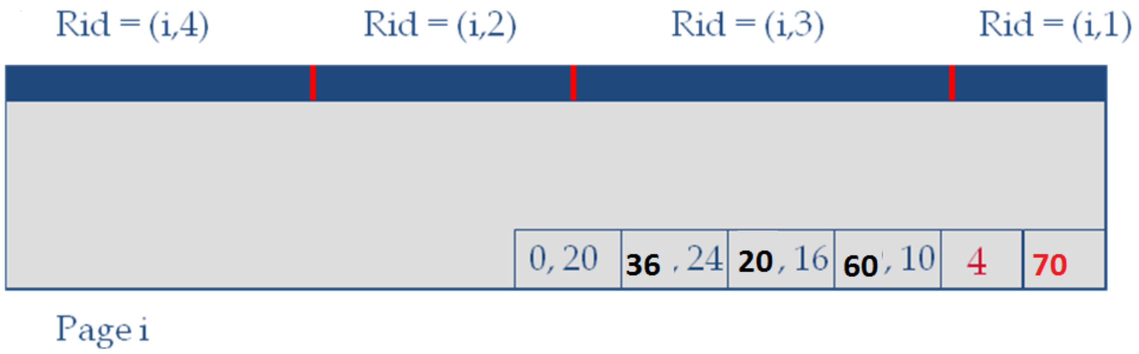
LRU					
Time	Page Read	Buffer Frame			
		P1	P2	P3	P4
1	G				
2	F				
3	E				
4	D				
5	G				
6	C				
7	F				
8	E				
9	D				
10	C				
11	D				
12	B				
13	A				
14	D				
15	F				
No. of Pages Evicted:					

MRU					
Time	Page Read	Buffer Frame			
		P1	P2	P3	P4
1	G				
2	F				
3	E				
4	D				
5	G				
6	C				
7	F				
8	E				
9	D				
10	C				
11	D				
12	B				
13	A				
14	D				
15	F				
No. of Pages Evicted:					

Below are two tables for describing the contents of the buffer pool at each time step. A page is read at the beginning of each time step. You should record, in the table, the contents of each Buffer Page after the new page has been read in.

Problem 3. [10pts]

Suppose we store the pointer (or starting address of a record) in a page by using the offset to the beginning of the page (i.e., the first byte in a page has an offset of 0, the 2nd byte in a page has an offset of 1, and so on and so forth). Each slot entry is of the format: (record offset, record length). Consider the variable-length record page organization instance below:



1) Show the content of the page after the deletion of Rid=(i,3)

2) Now, after step 1) has been completed, show the content of the page after the insertion of a new record to this page with 30 bytes. And what's the Rid of this new record after the insertion is done?

Problem 4. [30pts]

Suppose that the page size is 1K bytes, and the each record size is 100 bytes. Given 1 million records in a table, answering the following questions.

1) Suppose the table is stored in a heapfile, how many pages are in this file?

2) What's the IO cost of inserting and deleting a record?

3) We will build an unclustered index B+ tree on this table over an attribute A, how many data entries are contained in the leaf level?

4) Suppose A is integer type of 4 bytes, whats the size of a data entry (assuming a rid/pid is 4 bytes)?

5) Suppose each leaf level page is 70% filled, how many leaf level pages are needed?

6) Whats the size of one index entry?

7) What is the height of the B+ tree (suppose each index level page is also 70% filled)?

8) Given a query: find all data entries with 100<A<500, suppose there are 300 matching data entries for this query, whats the IO cost of this query?

9) Given a query: find all records with 100<A<500, suppose there are 300 matching records for this query, whats the IO cost of this query?

10) With this unclustered index, what's the IO cost of inserting and deleting a record?

11) Answer 1, 3-10 again suppose now we are using a clustered index over A.

12) Answer 1, 2, 9 again suppose now we are using a sorted file where **each page is 70% filled**.

13) Answer 1, 2, 6, 7, 8, 9 again suppose now we are using a clustered file, and each page is 70% utilized.

Problem 5. [10pts] Suppose that we are using extensible hashing on a file that contains records with the following search-key values: (2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, 9208).

Load these records into a file in the given order using extensible hashing. Assume that every block (bucket) can store up to four (4) values. Show the structure of the directory every 3 insertions, and the global and local depths. Use the hash function: h(k) = k mod 128 and then apply the extensible hashing index

Problem 6. [10pts] Suppose we are using linear hashing as discussed in class. Assume two records per bucket and the hash function $h(x) = x$. The intermediate hash functions $h_i(x)$, are given by the last i bits of $h(x)$. Assume a split is initiated whenever an overflow bucket is created. Starting with an empty hash file with the hash function $h_1(x)$ and two buckets show the significant intermediate steps (overflows and splits) when keys are inserted in the order given below:

(1,3,2,8,6,10,9,7,5,11).

Problem 7. [10pts] Suppose H is a k-universal family. Then

a) H is also $(k-1)$-universal.

b) For any $x \in U$ and $a \in [M]$, $\Pr[h(x) = a] = 1/M$

c) H is a universal hash family.

Prove these claims.