Written Assignment #3
CAS CS 460/660: Introduction to Database Systems
Fall 2017

Due: Wednesday, Nov 15, 2017, in class.

Problem 1. [20pts]

Consider a disk with sector size of 512 bytes, 1000 tracks per surface, 50 sectors per track, five (5) double-sided platters, and average seek time of 8 msec. Suppose that the disk platters rotate at 7400 rpm (revolutions per minute).

Suppose also that a block (page) of 2048 bytes is chosen. Now, consider a file with 150,000 records of 100 bytes each that is to be stored on such a disk and that no record is allowed to span two blocks. Also, no block can span two tracks.

1. How many records fit onto a block?
2. How many blocks are required to store the entire file? If the file is arranged "sequentially on the disk, how many cylinders are needed?
3. How many records of 100 bytes each can be stored using this disk?
4. What time is required to read a file containing 100,000 records of 100 bytes each sequentially? You can make an assumption about how long it takes moving the heads from one cylinder to the next here.
5. What is the time required to read a file containing 100,000 records of 100 bytes each in a random order? To read a record, the block containing the record has to be fetched from the disk. Assume that each block request incurs the average seek time and rotational delay. You need also to include the transfer time for each block.

Problem 2. [20 pts]

Consider a B+-tree of order two (d=2). Thus, the maximum number of pointers per node is 5 and the maximum number of entries is 4.

1. Show the results of entering one by one the keys that are three letter strings:
   (era, ban, bat, kin, day, log, rye, max, won, ace, ado, bug, cop, gas, let, fax)
   (in that order) to an initially empty B+-tree. Assume that you use lexicographic ordering to compare the strings. Show the state of the tree after every 4 insertions.
2. What is the utilization of the tree? The utilization of the tree is defined as the total number of entries in all the nodes of the tree (both leaf and non-leaf nodes) over the maximum number of entries that the same nodes can store.
3. Can you give a different insertion order of the same values that can create a tree with different height? If yes, provide the order. If not, explain why.

Problem 3. [20 pts]

Suppose that we are using extensible hashing on a file that contains records with the following search-key values:

(449, 124, 654, 831, 1016, 176, 285, 468, 615, 340, 331, 135, 667, 818, 117, 429)

Load these values into a file in the given order using extensible hashing. Assume that every block (bucket) of the hash index can store up to four (4) values.

Show the structure of the hash index after every 4 insertions, and the global and local depths. Use the hash function: $h(K) = K \mod 128$ and then apply the extensible hashing technique. Using this function, every number is mapped first to a number between 0 and 127 and then we take its binary representation. Then, the extensible hashing technique is applied on the binary representation. Furthermore, initially, you start with a single bucket and a single pointer and the global and local depths are zero (0).

Problem 4. [20 pts]

Assume that you have just built a B+-tree index using Alternative (2) on a Heap file containing 40,000 records. That is, the actual records are stored in another file outside the index that has no particular order since it is a Heap file. Also, Alternative (2) means that we have one data entry at the leaf nodes of the B+-tree for every record.

The search key field for this B+-tree index is a 40-byte string, and it is a candidate key (therefore unique). Pointers (i.e., record ids and page ids) are 10-byte values. The size of one disk page is 1000 bytes. The index T was built in a bottom-up fashion using the bulk-loading algorithm, and the nodes at each level were filled up as much as possible.

1. How many levels does the resulting tree T have?
2. For each level of the tree, how many nodes are at that level?
3. How many levels would the resulting tree T' have if key compression is used and it reduces the average size of each key in an entry to 10 bytes?
4. How many levels would the resulting tree T" have without key compression but with all pages 70 percent full?
5. Assume that each record is 100 bytes and is stored in the Heap file with disk pages of 1000 bytes each. Assume that you use the first B+-tree T that you computed, where the nodes were filled up as much as possible. Consider now a range query that uses this tree and has selectivity (reduction factor) 1% (i.e., 0.01). How many I/Os you need to perform in order to retrieve all the records that satisfy the range query using this tree? Explain.

Problem 5. [20 pts]

Consider the join $R \bowtie S$ where the join predicate is $R.a = S.b$, given the following metadata about R and S:

- Relation R contains 20,000 tuples and has 10 tuples per page (block)
- Relation S contains 5,000 tuples and has 10 tuples per page (block)
- Attribute b of relation S is the primary key for S, and every tuple in S matches 3 tuples in R
- There exists a unclustered B+-tree index on $R.a$ with height 3
- There exists a clustered B+-tree index on $S.b$ with height 2
- The main memory buffer can hold 5 blocks

Answer the following questions:

1. If $R \bowtie S$ is evaluated with a block nested loop join, which relation should be the outer relation? Justify your answer. What is the cost of the join in number of I/O's?
2. If $R \bowtie S$ is evaluated with an index nested loop join, what will be the cost of the join in number of I/O's? Which index you will use? Show your cost analysis.
3. What is the cost of a plan that evaluates this query using sort-merge join. Show the details of your cost analysis.
4. Evaluate the cost of computing the $R \bowtie S$ using hash join assuming: i) The main memory buffer can hold 202 blocks, ii) The main memory buffer can hold 4 blocks.