Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2016, Prakash**

Homework 3: B+ Trees, Hashing, Bulk Loading
(due Feb 29th, 2016, 4:00pm, in class—hard-copy please)

Reminders:

a. Out of 100 points. Contains 4 pages.
b. Rough time-estimates: 4~6 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Lead TA for this HW: Sorour Amiri.

Q1. B+ Tree [24 points]

Assume the following B+ tree exists with d = 2, where left leaf page contains records below the key and right leaf page contains records equal or greater than the key.

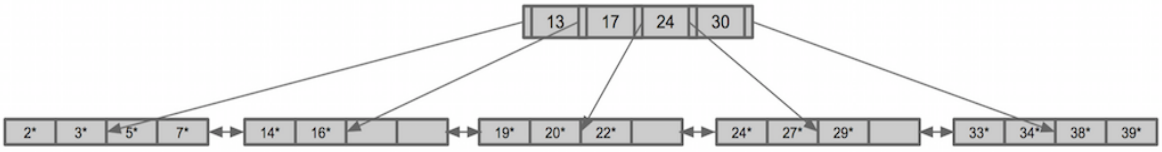

Figure 1. Initial B+Tree.

Sketch the state of the B+ tree for each operation. For each part of the problem, disregard previous parts and apply the instruction on the original tree structure above. Maintain at least 50% occupancy at each step and overflow triggered split.

Note: Use the insertion and deletion algorithms given in the textbook section 10.5 (page 349) and 10.6 (page 353) respectively. Root node can have 1 to 2d keys. During deletion redistribute the leaf pages wherever possible. In case of merge, if you can borrow from both siblings, choose the one on the right.

Q1.1. (4 points) Insert 17*.

Q1.2. (6 points) Starting from the B+Tree of Figure 1, Insert 31*.

Q1.3. (4 points) Starting from the B+tree of Figure 1, Delete 7*.

Q1.4. (4 points) Starting from the B+tree of Figure 1, Delete 24*.

Q1.5. (6 points) Starting from the B+tree of Figure 1, Delete 14*.

Q2. Bulk Loading a B+Tree [20 points]

Suppose we are bulk-loading an *initially empty* B+-Tree. Pages have 28 bytes to store information. A key value takes 8 bytes, and a pointer to a tree node or row takes 4 bytes. Bulk load the B+ tree with data entries 1*, 2*, …., 12* so that each leaf is **at least** half full using the algorithm outlined in Section 10.8.2 (Page 360) of the textbook.

Q2.1. (4 points) What is the order of the B+ tree? Also, how many keys and pointers per node can it hold? (Note: Recall that each node in a B+ tree is a page).

Q2.2. (3 points) How many levels are in the resulting tree? (e.g. The B+Tree in Figure 1 has 2 levels).

Q2.3. (5 points) Sketch the final B+ tree after bulk loading. No need to show each node, just enough for us to be convinced you have the right tree.

Q2.4. (3 points) Is this the *densest* possible (i.e. the most filled) B+-Tree tree with these keys? If not, sketch the densest possible tree.

Q2.5. (5 points) What is the minimum number of keys that must be added so that the height of the tree increases by 1? List these keys. Note1: There may be more than one correct answer. Note2: Please use the algorithm outlined in section 10.6 (page 353) of the textbook.

Q3. Linear Hashing [15 points]

Consider following linear hashing index to answer the questions. Assume that we split whenever a new key triggers the creation of a new overflow page.

Notes: Here h0 is (x mod 4). Currently, h0 is active, and no bucket is split. h1 is the next hash to be used (note that h1 would be (x mod 8)). Use the linear hashing algorithm outlined in Section 11.3 (page 379) of the textbook and Lecture 8: Indexes and Hashing (Linear Hashing).

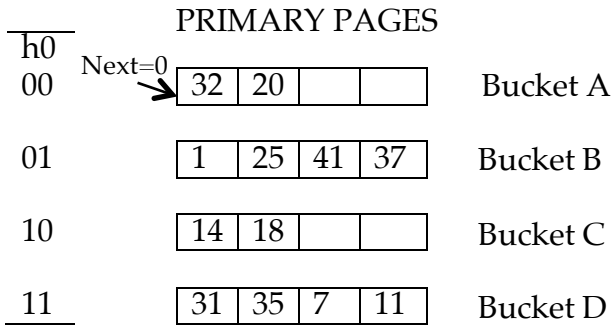

Figure 2. Linear Hashing

Q3.1. (2 points) What is the maximum number of data entries that can be inserted in the best-case scenario, before splitting a bucket?

Q3.2. (2 points) Which bucket would 64 be inserted into (use above mentioned Figure 2 as a starting point)?

Q3.3. (4 points) Show the file after inserting smallest possible single data entry whose insertion causes a bucket split and which bucket will be split (use above mentioned Figure 2 as a starting point).

Q3.4. (7 points) Show the file after inserting following entries 43, 29 (use above mentioned Figure 2 as a starting point). Assume that insertions are performed consecutively.

Q4. Extendible Hashing [16 points]

Q4.1. (10 points) Consider we are using extendable hashing on a file that contains records with the following search-key values:

(3, 9, 12, 15, 16, 17, 18, 20, 21, 34, 44, 49)

Sketch the extendable hash structure (in the same order shown) for this file if the hash function is $h(x) = x \bmod 8$ and buckets can hold up to three records. Initially, the structure is empty, and global depth is 1.

Note: Use a directory of pointers to buckets. The directory structure doubles when a bucket overflows. Use the extendible hashing algorithm outlined in section 11.2 (page 373) of the textbook and Lecture 8: Indexes and Hashing (Extendible Hashing).

Q4 .2 (3 points) What are the final global and local depths of all the buckets in the hash structure?

Q4.3 (3 points) Suppose following range query is given "select * from table where key> x and key<=y". Do you think this hash structure will efficiently answer the query? Why or why not?

Q5. External Sorting [25 points]

Suppose you have a file with 2×10^6 pages and assume that a two-way external merge sort algorithm is used:

Q5.1. (2 points) How many runs will you produce in the first pass?

Q5.2. (3 points) How many passes will it take to sort the file completely?

Q5.3. (5 points) What is the total I/O cost of sorting the file (with 3 buffer pages)?

Answer the following questions using the general external sorting algorithm outlined in section 13.3 of the textbook (page 424). Please write the formula you used in calculating the answers.

Q5.4. (5 points) How many buffer pages do you need to sort a file with 1010000 pages completely in just three passes? (Hint: ignore the celling function in the formula)

Q5.5. (3 points) Assume that the number of available buffer pages is the one you calculated in previous question (Q5.4.). What is the maximum size of the file (in number of pages) that we can sort with 2 passes?

Q5.6. (5 points) What is the total I/O cost of sorting a file with 2×10^6 pages with 17 available buffer page?

Q5.7. (2 points) Now assume that we have a disk with an average seek time of 12ms, average rotation delay of 6ms and a transfer time of 2ms for each page. Assuming the cost of reading/writing a page is the sum of those values (i.e. 20ms) and do not distinguish between sequential and random disk-access – any access is 20ms, what is the total running time to sort the file in Q5.6?