Carnegie Mellon University
Department of Computer Science
15-415/615 - Database Applications
C. Faloutsos, A. Pavlo , Fall 2015

Homework 4 (by Yujing Zhang)
Due: **hard copy**, at **3:00pm, Oct. 14, 2015**

VERY IMPORTANT: Deposit **hard copy** of your answers, in class. Please

1. **Separate** your answers, on different page(s) for each question (staple additional pages, if needed).
2. **Type** the full info on **each** page: your **name**, **Andrew ID**, **course#**, **Homework#**, **Question#** on each question.

Reminders

- *Plagiarism*: Homework is to be done **individually**.
- *Typeset* all of your answers, please.
- *Late homeworks*: Standard policy: email (a) to all TAs, (b) with the subject line exactly `15-415 Homework Submission (HW 4)`, and (c) the count of slip-days you are using.

For your information:

- Graded out of **100** points. **6** questions total. Expected effort: ≈ 3-6h.
- **Solutions** for odd numbered exercises are available on the web: `http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/solutions/ans3ed-oddonly.pdf` You are strongly encouraged to use them.

Revision : 2015/10/06 14:37

Question	Points	Score
Buffers	15	
B Tree	20	
B+ Tree	30	
Extendible Hashing	10	
Linear Hashing	10	
Sorting	15	
Total:	100	

Question 1: Buffers . . . . [15 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Consider a buffer pool with 4 frames (F1-F4); and a file with pages numbered from P1 to P7 inclusive. Consider the 10 read-requests for the pages, as shown in the table below. Consider three different buffer replacement policies: least recently used (LRU), most recently used (MRU), and 'clock'. Assume the following:

- The buffer pool is initially empty.
- The frames are unpinned, immediately after use.
- The pointer for the 'clock' policy, starts at frame F1 at the very beginning; then, it moves whenever we need to replace the contents of a frame, as the textbook says.

(a) **[5 points]** For each policy, which page is evicted at which time as requests come in? Fill in the table below. (Put a "-" (dash) if nothing is evicted at certain timestamp)

timestamp	*request*	*evicted page* (LRU)	*evicted page* (MRU)	*evicted page* (Clock)
t1	P1			
t2	P2			
t3	P3			
t4	P2			
t5	P3			
t6	P4			
t7	P5			
t8	P1			
t9	P6			
t10	P7			

(b) **[5 points]** Draw the final state of the buffer for each policy.

LRU: | | | | |

MRU: | | | | |

Clock: | | | | |

(c) **[5 points]** For each of the three policies, draw the final states of the buffer for a sequential scan of pages numbered from P1 to P15 inclusive. (That is, scanning 15 pages from page P1 to page P15, in order)

LRU: | | | | |

MRU: | | | | |

Clock: | | | | |

Question 2: B Tree . . . . [20 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

(a) Produce the most dense possible B-tree of order $d = 2$ (at most 4 keys per page), containing as keys the integers 1 through 24 inclusive. For example, the most dense B-tree with keys from 1 to 9 has height h=2, and is shown in Figure 1:

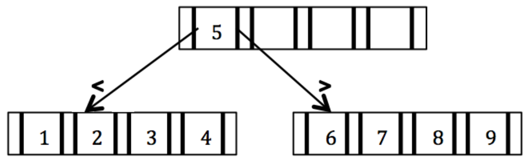

Figure 1: dense B-tree of order d=2, with height h=2

For your drawing convenience, here is the MSWord template for tree nodes:: `http://www.cs.cmu.edu/~christos/courses/dbms.F15/hws/HW4/tree-template.docx`

i. [**2 points**] How many nodes does the structure have?

ii. [**3 points**] Draw the final structure.

(b) Produce the most *sparse* B-tree of order $d = 1$, containing the keys 1 through 15 inclusive.

i. [**2 points**] How many nodes does the structure have?

ii. [**2 points**] What is the height of the structure?

iii. [**1 point**] How many keys are in the root?

iv. [**2 points**] Which key(s) is/are in the root?

(c) Consider an empty B tree of order $d = 3$. Using the standard B-tree algorithm given in the foils (2-to-1 split, no deferred splits), insert keys from 1 to 32 (inclusive), in order.

i. [**2 points**] What is the height of the final structure?

ii. [**1 point**] How many keys are in the root?

iii. [**2 points**] Which key(s) is/are in the root?

iv. [**1 point**] How many keys are in the last (= right-most) leaf node?

v. [**2 points**] Which key(s) is/are in the last leaf node?

Question 3: B+ Tree . . . . [30 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

Consider the following B+ tree of order d=2 and height h=3 levels, shown in Figure 2. Please make the following assumptions:

- (A) For each part of the problem, disregard previous parts and apply the instruction on the tree structure in Figure 2.
- (B) With respect to "≥", follow the convention used in the textbook, and in Figure 2, that is, the left pointer is for <, the right one for ≥.
- (C) In case of underflow, if you can borrow from both siblings, choose the one on the *left*.
- (D) In case of overflow, promote the key in the middle.
- (E) If a large part of the tree remained unchanged, you may type '(unchanged)', instead of drawing it, to make your solution cleaner.

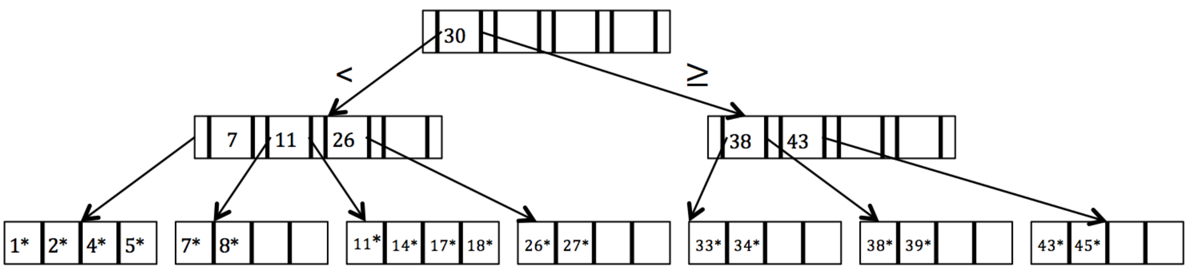

Figure 2: B+ tree of order d=2.

For each question except for the last one ('e') below, draw the tree after the specified operation, always starting from the tree of Figure 2.

(a) **[5 points]** Insert 10*.

(b) **[5 points]** Starting from the B+tree of Figure 2, insert 20*.

(c) **[5 points]** Starting from the B+tree of Figure 2, delete 7*. (according to assumption C)

(d) **[5 points]** Starting from the B+tree of Figure 2, delete 17*, 18*, and 26*.

(e) Consider the 12 records with keys 1*, 2*, . . . , 12*. Suppose they are bulk-loaded in an *initially empty* B+ tree of order d=1, using the algorithm from the *textbook*.

   i. **[3 points]** How many levels are in the resulting tree? (The B+ tree of Figure 2 has height h=3 levels; The fully-packed leaves are at level '1').

   ii. **[3 points]** How many keys are in the root?

   iii. **[4 points]** Which key(s) is/are in the root?

Question 4: Extendible Hashing .... [10 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

This is a modification of Exercise 11.1, p. 386-387 of the textbook. Assume we have the following records where we indicate the hashed key in parenthesis (in binary). Consider an Extendible Hashing structure.

- Each bucket can hold up to 3 records.
- Initially the structure is empty (only one empty bucket).

Consider the result after the records above have been inserted in the order shown, using the lowest-bits for the hash function, as the textbook does. That is, records in a bucket of local depth d, agree on their **rightmost** d bits.

'a' [00001]

'b' [01000]

'c' [00000]

'd' [10100]

'e' [10101]

'f' [00010]

'g' [11000]

'h' [10001]

(a) **[2 points]** What is the global depth of the resulting directory?

(b) **[3 points]** Which record causes the first split? Which record causes the second split?

(c) After inserting records 'a'-'h', suppose that we insert record 'i' [10000].
   i. **[2 points]** How many buckets will we have now?
   ii. **[2 points]** What is the local depth of the bucket containing record 'i'?
   iii. **[1 point]** Which other keys are in the same bucket with record 'i'? (Type '*alone*', if 'i' is by itself.)

Question 5: Linear Hashing .... [10 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

For the same records as in Question 4 ('a' - 'h'), consider a linear hashing structure where pages can hold up to 3 records. A bucket consists of 1 main page, and zero or more overflow pages. Initially the structure is empty (only one empty bucket/page). Consider the result after the records above have been inserted in the order shown.

- Use the algorithm in the textbook (p. 380 and on, section 11.3)
- Assume that a split is 'triggered' whenever the insertion of the new record caused the creation of an overflow page.

(a) [**3 points**] How many buckets will we have? (In the textbook, there are 5 buckets in Figure 11.9, that is, overflow pages don't count as 'buckets'.)

(b) [**4 points**] List all the records in the bucket that contains record 'b' (including records in the overflow pages for that bucket).

(c) In the hash table that contains records 'a'-'h', insert records 'j' [11100], 'k' [00011] in this order.

i. [**1 point**] How many buckets will we have?

ii. [**2 points**] List all the elements in the bucket which contains the record 'k' [00011] (including records in any overflow page(s) for that bucket). Type 'alone', if 'k' is by itself.

Question 6: Sorting .... [15 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

- *Hint*: This is a modified version of Exercise 13.1 in the textbook.
- Feel free to study the answers to the odd-numbered exercises, that are on the web.

Suppose you have a file with $N = 8 * 10^7$ pages, and B buffers (each able to hold one page).

(a) [**4 points**] What is the total I/O cost of sorting the file using the two way merge sort(with B=3)?

(b) Suppose we have B=100 buffers, and we are using the external sorting algorithm discussed in chapter 13.3 (p.427, Figure 13.6) of the textbook.

  i. [**3 points**] How many runs will you produce in the first pass?

  ii. [**4 points**] Now assume that we have a disk with *16msec* I/O time. What is the total running time to sort the file?

  iii. [**4 points**] With B=100 buffer pages, what is the maximum size N_{max} of the file (in number of pages) that we can sort with 4 passes?