Carnegie Mellon University
Department of Computer Science
15-415/615 - Database Applications
C. Faloutsos, A. Pavlo , Fall 2016

Homework 4 (by Kai Kang)
Due: **hard copy**, at **3:00pm, Oct. 12, 2016**

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

1. **Separate** your answers, on different page(s) for each question.
2. **Type** the full info on **each** page: your **name**, **Andrew ID**, **course#**, **Homework#**, **Question#** on each of the 5 pages.

Reminders:

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

For your information:

- Graded out of **100** points; **5** questions total
- Rough time estimate: $\approx$ 5-10 hours - 1-2 hours per question
- **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 : 2016/10/05 01:36

Question	Points	Score
ISAM	20	
B Trees	20	
B+ Trees	25	
Extendible Hashing	15	
Linear Hashing	20	
Total:	100	

Question 1: ISAM . . . . . [20 points]

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

(a) Consider the following ISAM tree Figure 1 as defined in the text book, page 342, where the keys are characters and they are ordered alphabetically. For each operation, please plot the resulting tree. You can just draw the interesting subtree. Note that each operation starts with the original tree. There are 2 tiers of non-leaf pages. To save your time, you may use the `draw.io` template at: `https://drive.google.com/file/d/0B4lG-7EeCB0cSGpiSUt5TWcwbmc/view?usp=sharing`

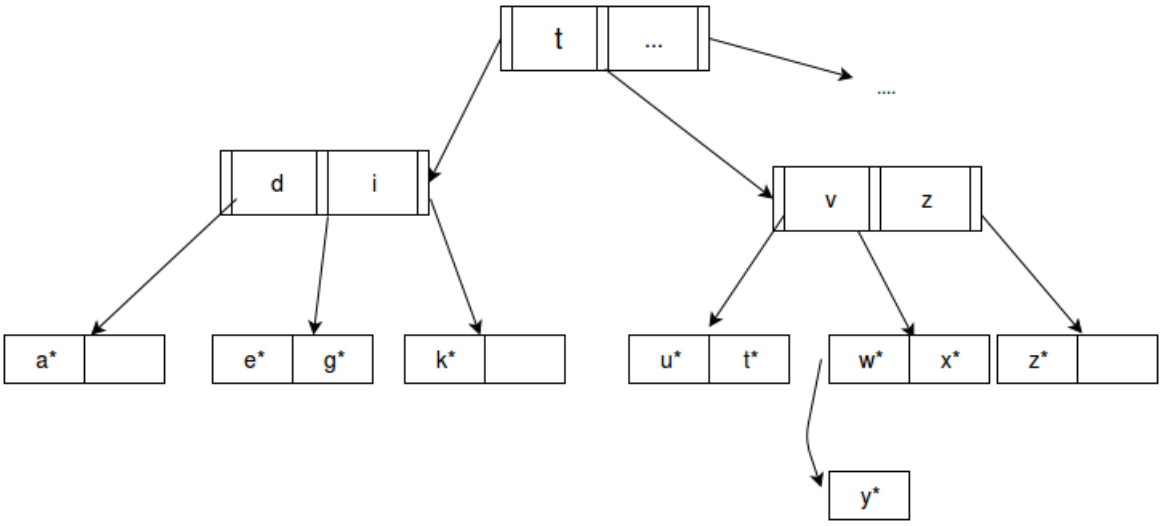

Figure 1: ISAM data structure

i. **[2 points]** Start with the original tree, insert key j*
ii. **[2 points]** Start with the original tree, delete key u*
iii. **[3 points]** Start with the original tree, insert key f*
iv. **[3 points]** Start with the original tree, delete key y*

(b) Answer the following True/False questions about ISAM:

i. **[2 points]** ☐ T ☐ F The number of primary leaf pages is fixed at creation.
ii. **[2 points]** ☐ T ☐ F Searching in a chain of overflow pages is cheap because those overflow pages are sequentially stored in adjacent storage locations.
iii. **[2 points]** ☐ T ☐ F ISAM leaf nodes have pointers to other leaf nodes.
iv. **[2 points]** ☐ T ☐ F The ISAM insertion algorithm is easier to implement than that of B+ tree.
v. **[2 points]** ☐ T ☐ F A range query on a file organized under ISAM, is always more costly than the same query on the same file, organized as a (clustering index) B+ tree.

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

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

We will see the properties of the densest possible B-trees, the sparsest possible ones, and the trees created by inserting keys in sorted order. For example, the most dense B-tree of order $d = 2$ that hosts 9 keys (say, the integers from 1 to 9), has height h=2, and is shown in Figure 2.

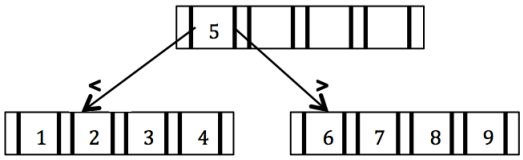

Figure 2: most dense B-tree of order d=2, with height h=2

(a) Consider B-trees of **order** $d = 5$. (Notice that we ask for a different order, than the tree in the Figure). Consider the most dense B tree of such order d=5, that contains all the integer keys 1, 2, . . ., 120. Answer the following questions:

i. **[3 points]** How many nodes does the structure have?

ii. **[3 points]** List the keys in the root node

(b) Answer the following questions about B-trees, of **order** $d = 4$. (Notice that we ask for a different order than before.)

i. **[1 point]** If B-tree T is a most sparse B-tree of height h=1, how many nodes does it have?

ii. **[3 points]** Repeat, for height h=2: If B-tree T is a most sparse B-tree of height 2, how many nodes does it have?

iii. **[4 points]** Repeat, for height h=3: If B-tree T is a most sparse B-tree of height 3, how many nodes does it have?

(c) **[6 points]** Now, consider B-trees of order $d = 2$. Starting from an *empty* such tree, insert the integers 1, 2, 3, . . ., in sorted order - which is the first key that makes the tree height = 3?

Question 3: B+ Trees ..... [25 points]

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

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

- With respect to "$\geq$", follow the convention used in the textbook, and in Figure 3, that is, the left pointer is for $<$, the right one for $\geq$.
- In case of underflow, if you can borrow from both siblings, choose the one on the *right*.

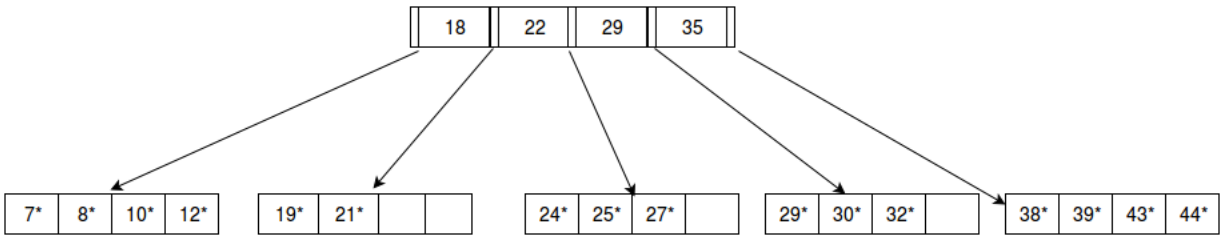

Figure 3: A B+ tree of order d=2.

For all questions below, use the standard B+-tree algorithm given in the foils and the textbook (on insertions: 2-to-1 split, no deferred splits; on deletions: no underflowing pages). For your drawing convenience, you may use the `draw.io` template, at: `https://drive.google.com/file/d/0B4lG-7EeCB0caEhrTlFCU05LQ0k/view?usp=sharing`

In all cases, start from the B+ tree of Figure 3.

(a) **[5 points]** Start from the original B+ tree; insert 22*.

(b) **[5 points]** Start from the original B+ tree; insert 36*.

(c) **[5 points]** Start from the original B+ tree; delete 12*.

(d) **[5 points]** Start from the original B+ tree; delete 29*.

(e) **[5 points]** Start from the original B+ tree; delete 19*.

Question 4: Extendible Hashing . . . . . [15 points]

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

Answer the following questions about extendible hashing:

(a) **[3 points]** Consider an extendible hash table, where the last insertion forced its directory to double in size, from n_{old}=8 to n_{new}=16 directory entries - how many buckets have exactly one directory entry pointing to them?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 8 ☐ 16 ☐ undefined

(b) **[3 points]** ☐ T ☐ F Every bucket with only one directory entry pointing to them, is guaranteed to be at least half full.

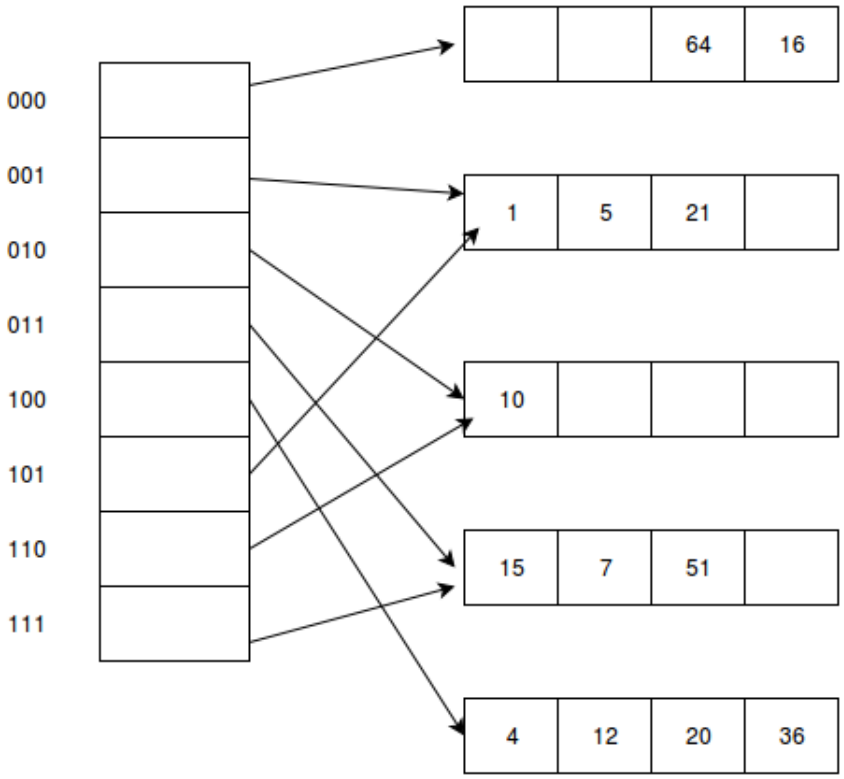

Figure 4: Extendible hashing

(c) Answer the following questions about Figure 4.

i. **[5 points]** Suppose we insert keys 8, 18, 23, 11, 42, in that order. Which key causes the first split?

ii. **[4 points]** Starting from the hash table of Figure 4, delete keys 36, 12, and plot the resulting table. You may use the `draw.io` template at: `https://drive.google.com/file/d/0B4lG-7EeCB0cUkFQd3hmdU9PdWM/view?usp=sharing`

Question 5: Linear Hashing.....[20 points]

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

Answer the following True/False questions about linear hashing:

(a) **[3 points]** ☐ T ☐ F Linear hashing always splits the bucket that overflows.

(b) **[3 points]** ☐ T ☐ F Like ISAM, linear hashing will suffer from long chains of overflow pages, if there are too many insertions and too few deletions.

(c) Answer the following questions for the hash table of Figure 5. This is a modified version of Exercise 11.9 from the textbook, p.389. As in the textbook, assume that a bucket split occurs whenever an overflow page is created.

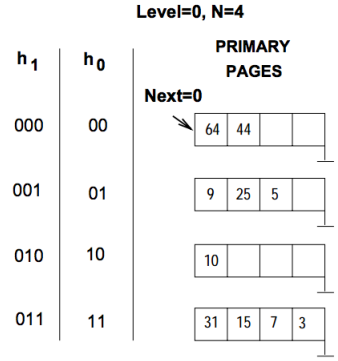

Figure 5: Linear Hashing

(d) **[2 points]** Which bucket will key 11 go to?

(e) **[6 points]** Starting from the hash table of Figure 5, what is the smallest key > 60, whose insertion will cause a split?

(f) **[6 points]** Starting from the hash table of Figure 5, plot the final hash table, after inserting $35, 17, 21$. Remember to indicate the new hash function (if any), and to update the "Next" pointer, if needed. You may use the `draw.io` template, at: `https://drive.google.com/file/d/0B41G-7EeCB0cN21Mak0xSldteGM/view?usp=sharing`