

CS411 Database Systems
Spring 2007

HW#3

Due: 1:55pm CST, 04/10/07

Note: Print your name and NetID in the upper right corner of every page of your submission. Handin your stapled homework to Trisha Benson in 4322 SC. In case Trisha is not in office, slide your homework under the door.

To grade homeworks faster, 'The homework is partitioned into three parts. **Please, submit each part separately.** For each part, make sure to write down your name and NetID.

This homework is partitioned into 3 parts as follows:

- Part 1: Problem 1 - Problem 3
- Part 2: Problem 4 - Problem 6
- Part 3: Problem 7 - Problem 8

There is a bonus credit of 2% if your homework is formatted correctly: Each part must be separable so that it can be independently graded. If you submit a paper copy then each part should i) be separately stapled and ii) include your netid. For I2Cs students: If you submit an electronic copy (or fax) then each part should start on a new page and include your netid at the top of each page.

Handwritten submissions will be graded but they will take longer to grade. For clarity, machine formatted text is preferable: Expect to lose points if your handwritten answer is unclear or misread by the grader.

Part 1

Problem 1 Merge-Sort Suggested reading: Chapter 11.4

Suppose we have a relation with 2.2 billion tuples and each tuple requires 300 bytes. We have a machine whose main memory and disk-block size (16,384 bytes) are sufficient to sort the 2,200,000,000 tuples using TPMMS (Two-Phase, Multiway Merge-Sort). Assume that no tuples are spanned over disk-blocks and there is no header information in a disk-block.

1. What is the minimum size of the main memory?
2. Assuming all results are written to disk, what is the number of disk I/O's needed to sort the 2,200,000,000 tuples using your answer above?

Problem 2 Pointer Swizzling Suggested reading: Chapter 12.3

Suppose that the important actions related to data storage take the following times, in some arbitrary time units:

- On-demand swizzling of a pointer: 30.
- Automatic swizzling of pointers: 10 per pointer.
- Following a swizzled pointer: 1.
- Following a unswizzled pointer: 10.

Suppose we design a pointer-swizzling control scheme like the following. At the beginning, we automatically swizzle 30% of the pointers and leave the rest unswizzled. Once a pointer is followed, we swizzle it by probability 0.5. If an unswizzled pointer has been followed twice, we swizzle it. Suppose there are 150 pointers in our data. The number of times that they are followed by a program is distributed according to the following histogram.

times of being followed	0	1	2	3
number of pointers	10	60	40	40

What's the **expected cost** of this program in terms of pointer following?

Problem 3 Data Representation Suggested reading: Chapter 12.4

An employee record consists of the following fixed-length fields: the employee's date of birth and social-security number, each 12 bytes long. It also has following variable-length fields: name and title. The records are augmented by an additional repeating field that represents employee evaluations. Each evaluation requires a date (12 bytes) and an integer result (4 bytes) of the evaluation. Pointers within a record requires 4 bytes, and the record length is a 4-byte integer. You may assume that no alignment of fields is required.

1. Show the layout of employee records if:

- (a) The variable-length fields and repeating evaluations are kept within the record itself
 - (b) The variable-length fields and repeating evaluations are stored outside of the record, with pointers to them in the record.
2. The evaluation results of an employee has a probability p of being queried. There are n evaluation results per student in average. The average lengths of name and title are 8 bytes and 12 bytes, respectively. Suppose the cost function of the above scheme (a) is len where len is the record length. The cost of scheme (b) is $len + 150 \times p$, for penalizing accessing evaluation results on separate blocks. What is the minimal value of n as a function of p , so that the above scheme (b) is better than scheme (a).

Part 2

Problem 4 Indexes on Sequential Files Suggested reading: Chapter 13.1

Given a relation of 100,000 tuples. Suppose each block could hold 5 tuples or 10 key-pointer pairs. The index is built on the key field of the relation (thus no duplicate search keys) and the file is sorted according to the key. Answer the following questions:

- a) How many blocks do we need if neither data nor index blocks are allowed to be more than 80% full? (consider dense index)
- b) How many blocks do we need if neither data nor index blocks are allowed to be more than 80% full? (consider sparse index)
- c) How many blocks do we need (with sparse index) if we use as many levels of index as is appropriate, until the final level of index has only one block?
- d) How many blocks do we need for a 2-level index of this relation (given that the first level is dense)?

Problem 5 B+tree Suggested reading: Chapter 13.3

Execute the following operations on Fig. 13.23 (page 635 of the textbook). Show the detailed steps, using Example 13.23 and Example 13.24 as examples.

- (a) Lookup the record with key 17.
- (b) Find out if record with key 50 exists.
- (c) Lookup all records in the range 20 to 42.
- (d) Lookup all records with keys less than 25.

Problem 6 B+tree Suggested reading: Chapter 13.3

Consider the B+ tree index that indexes 300 records.

- (a) If a B+ tree is order 9 (i.e., each node has at most 9 keys), what is the minimum and maximum height (depth) of the tree? (a B+ tree with only the root node has a height (depth) of 1.)
- (b) If this B+ tree has a height of 2, what is the minimum and maximum order?

Part 3

Problem 7 B+tree Suggested reading: Chapter 13.3

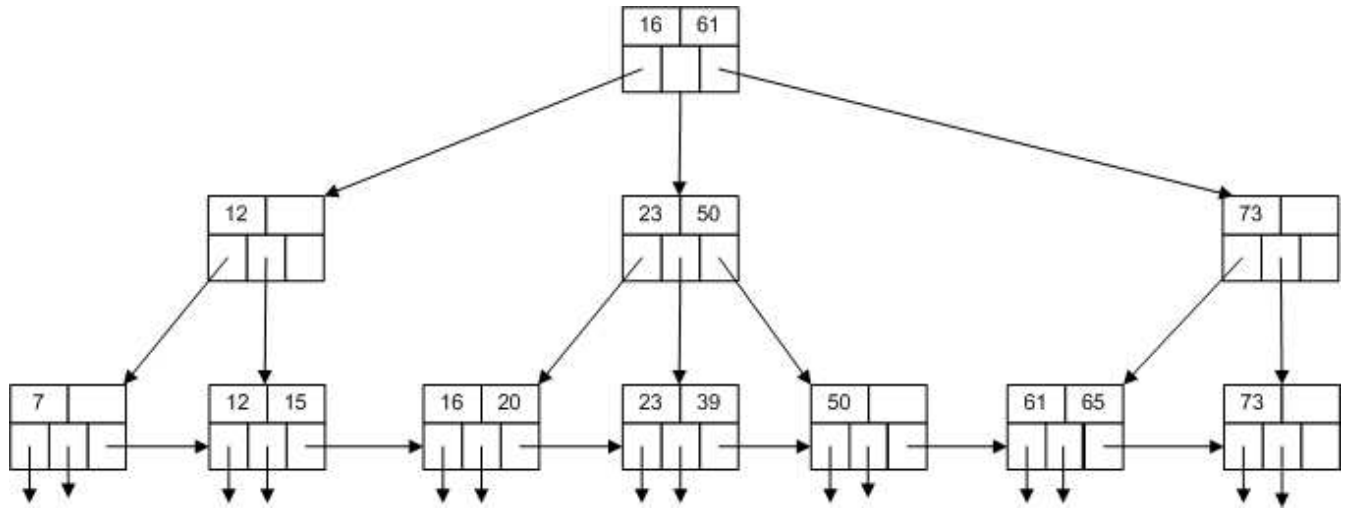


Figure 1: Original B+ Tree

- Consider the following B+ tree of order $n=2$ (2 keys, 3 pointers). Show the tree that would result from inserting a data entry with key 19 into this tree.
- Show the B+ tree that result from inserting a data entry with key 11 into original tree.
- Show the B+ tree after deleting the data entry with key 50 from original tree.
- Show the B+ tree after deleting the data entry 12 from original tree.

Problem 8 Extensible Hash Table Suggested reading: Chapter 13.4

Consider indexing the following key values using an *extensible hash table*. Suppose that we insert the keys in the order of

45, 36, 31, 56, 34, 62, 50, 23.

The hash function $h(n)$ for key n is $h(n) = n \bmod 16$; *i.e.*, the hash function is the remainder after the key value is divided by 16. Thus, the hash value is a 4-bit value. Assume that each bucket can hold 2 data items.

- (a) Draw the hash index (both the directory and the buckets), after the first four keys are inserted. Show the keys themselves in the buckets, as well as the hash values. Be sure to indicate the number of bits in the hash value that are used (in the directory as well as each bucket).
- (b) Draw the hash index after all the keys are inserted.