CS 245
Midterm Exam – Winter 2000

This exam is open book and notes. You have 70 minutes to complete it.

Print your name:____

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signed:____

Problem	Points	Maximum
1		20
2		20
3		20
4		10
5		20
Total		90

Problem 1 (20 points)

Suppose that we have 4096-byte blocks in which we store records of 816 bytes each. The block header consists of a 2-byte counter (indicating the number of entries in the offset table), plus an offset table (or directory), as in Figure 3.8 in the textbook and in slide 52 of Notes 3. (The slide number may be off depending no which version of the slides you have. It is the slide with header "Ex #1 Indirection in block".) In the offset table, each record pointer is 2 bytes.

On an average day, two records per block are inserted, and one record per block is deleted. When a record is deleted, a tombstone is left in the offset table. For specificity, assume that the deletion on any day occurs before the insertions.

If a block is initially empty, on what day will there be no more room for the day's insertions? (On the first day, day #1, there is no deletion, and two records are inserted.)

ANSWER : 4

On the first day, the block has a counter, 2 pointers, and 2 records. Therefore, a total of $2 + 2 \times (816 + 2) = 1638$ bytes is used up. Every day,

- We add two records : $+2 \times (816 + 2) = 1636$ bytes
- Delete one record : -816 bytes

Increase in size is $1636 - 816 = 820$ bytes per day. Notice that we **don't** reclaim the space allocated to the record pointer since that space is now occupied by the tombstone. After n days the size of the block is $S(n) = 1638 + 820 \times n$. The condition $S(n) \leq 4096$ gives $n \leq 2.998$. Hence, we can perform the operations for 2 additional days (i.e, until the 3rd day) without any problem. On the fourth day, we will be unable to make the second insertion.

Problem 2 (20 points)

Suppose that you have a document repository, and we build an inverted index. We have the following information:

- There are 10,000 documents in the repository.
- There are 10,000 unique words in the documents.
- On the average, each word occurs 1000 times in the collection.
- Words are represented in the index in a 10 byte fixed-length string. (If the word is longer, it is truncated. If the word is shorter, the word is padded with blanks.)
- Block pointers are 4 bytes long. Assume pointers to disk records are also 4 bytes.
- Document identifiers are 4 bytes long
- All disk blocks are 4096 bytes

The unique words are indexed with a B+ tree. A record pointer in a leaf of this tree points to the inverted list for the word.

Each entry in the inverted list (a posting) consists of (i) a document identifier; (ii) one byte for the type of occurrence (e.g., in the abstract); and (iii) two bytes for the position of the occurrence. If a word occurs more than once in a given document, there are different entries in the inverted list No compression is done on the lists.

(a) How many levels does the B+ tree have? **ANSWER:** 2

Let n be the order of the B+ tree. Then, to fit $n+1$ pointers (each of size 4 bytes) and n keys (each of size 10 bytes) in a block of size 4096 bytes, we require that $4(n+1)+10n \leq 4096$ giving an integral value of $n = 292$. Therefore, the B+ tree has $\lceil \log_{292} 10000 \rceil = 2$ levels.

(b) On the average, how large is each inverted list (for one word), in blocks? **ANSWER:** 2

Each entry in the inverted list is $4+2+1 = 7$ bytes long. Therefore, each block can hold $\lfloor \frac{4096}{7} \rfloor$ $= 585$ bytes. Since the inverted list for a word is 1000 entries long (on the average), the required size in blocks is $\lceil \frac{1000}{2} \rceil = 2$.

(c) On average, how many IOs must be done to answer the query "get all documents with the word 'cat' "? Assume that only the root of the B+ tree is in memory. **ANSWER:** 3

We need 1 IO for retrieving the leaf of the B+ tree (the root already being in memory) and 2 IOs for retrieving the inverted list associated with a word (from the answer to part (b)). Hence, a total of 3 IOs.

(d) On the average, how many documents are returned by the query "get all documents with the word 'cat' "? Do not try to compute the answer, just show the expression for computing the value.

The key is to realize that a word might occur multiple times in a document. Let us assume that each word is equally likely to occur at any position in any one of the documents in the repository. Now, since there are 10000 unique words, the probability that the first word in a certain document is *cat*, is $\frac{1}{10000}$. Therefore, the probability that the first word is not *cat* is $1 - \frac{1}{10000}$. Hence, the probability that a document does not contain *cat* is $P_{\wedge cat} = \left(1 - \frac{1}{10000}\right)^{1000}$ (we raise to the power 1000 since each document has, on the average, 1000 words). Thus, the probability that a document contains *cat* is $1 - P_{\wedge cat}$, giving us the following expression for the expected number of documents :

$$10000 \times \left(1 - \left(1 - \frac{1}{10000}\right)^{1000}\right)$$

Problem 3 (20 points)

This question deals with a B+-tree of order 3. Figure 1 depicts various stages in the life of such a B+-tree. Each figure has one or more keys missing. Complete the figures based on your knowledge of how the B+-tree would be restructured after each insertion and deletion.

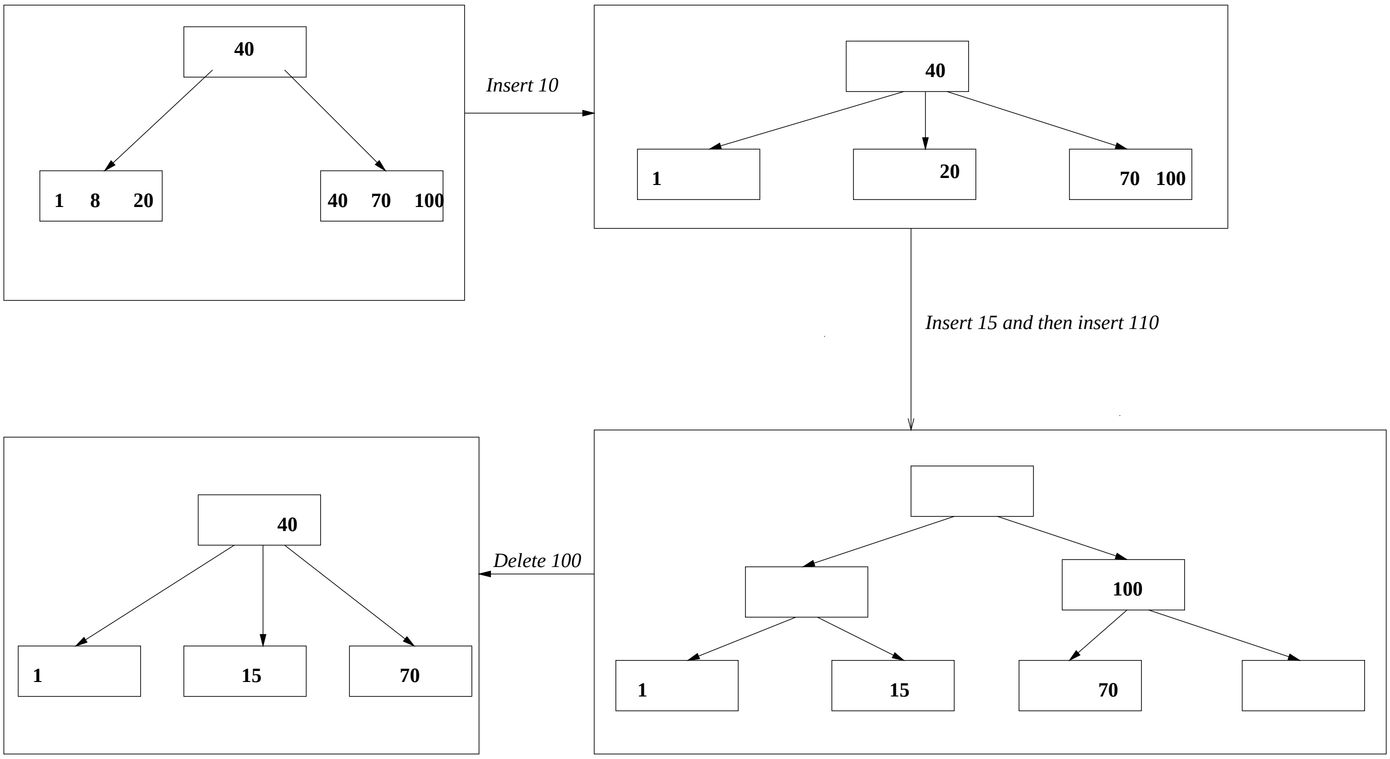
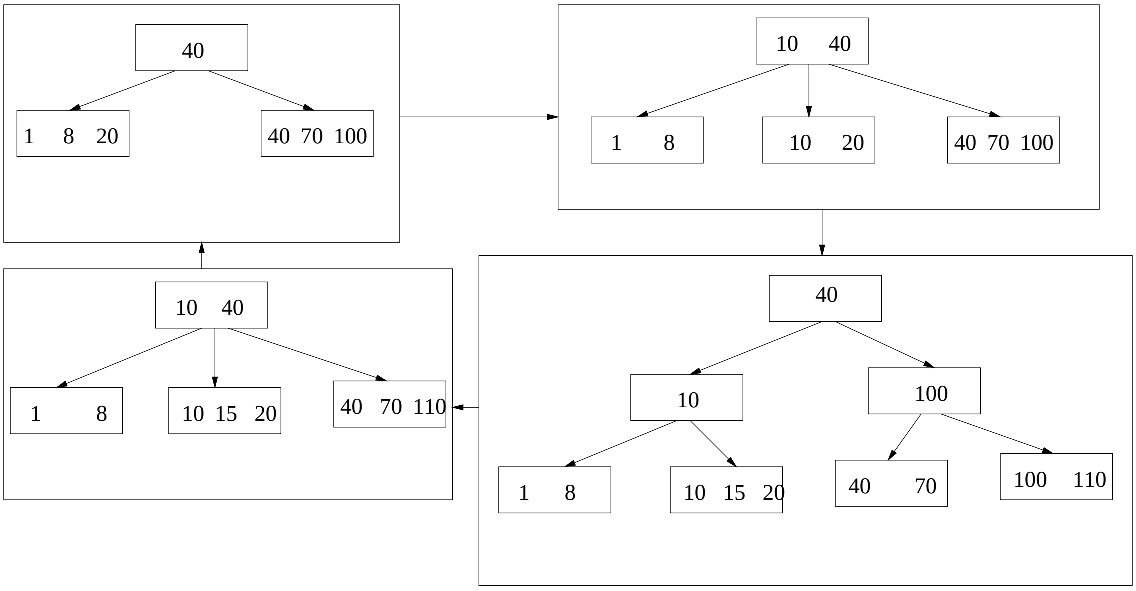


Figure 1: B+-tree of order 3



Problem 4 (10 points)

Problem 3

Indicate (YES or NO) which of the following expressions hold if relations can be bags and the union operator has the "sum" semantics.

(a) $\sigma_C(R \cup S) = (\sigma_C R) \cup (\sigma_C S)$

ANSWER: **Yes**

(b) $\sigma_{\neg(C \wedge D)} R = \sigma_{\neg C} R \cup \sigma_{\neg D} R$

ANSWER: **No** (tuples which satisfy both $\neg C$ and $\neg D$ will be counted twice by right hand side but only once by left hand side)

(c) $R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$

ANSWER: **No** (because of tuples in $R \cap S \cup T$)

(d) $R \cup S = S \cup R$

ANSWER: **Yes**

Problem 5 (20 points)

Consider the join of two relations R and S. Each relation has 10,000 tuples that are stored contiguously on disk, occupying 1000 blocks (10 tuples per relation). The join will be performed with 201 blocks of available memory.

(a) What is the minimum amount of memory needed to perform a hash-join, as described in class (no optimizations)?

ANSWER: k, the number of buffers for partitioning the relation, must be greater than the square root of 1000 blocks (31.66). Thus, we need 32 memory buffers, plus one to read the relation, for a total of 33 buffers.

(b) Using this hash-join algorithm, what is the *minimum* number of buckets that can be produced? How large is each bucket?

ANSWER: We can have 5 buckets, each with 200 blocks. We cannot have fewer buckets, because then each would be greater than 201 blocks, and we could not fit a bucket in memory for the join phase.

(c) Using this hash-join algorithm, what is the *maximum* number of buckets that can be produced? How large is each bucket?

ANSWER: We can have 200 buckets, each with 5 blocks. We cannot have more because we would not have enough memory to partition the relations in the first phase.

(d) If we use the optimized hash-join (where some buckets are kept in memory), how many buckets are kept in memory? Please explain how memory is utilized in this case.

ANSWER: Any number of buckets in memory was counted as correct, as long as they fit in memory and as long as memory utilization was described. Two common answers: (i) k =6, keep one bucket in memory; memory accounting: 1 bucket with 167 blocks, 5 blocks for partitioning to the remaining buckets on disk, and 1 block for reading R (or S), for a total of 173 blocks in use. (ii) $k = 32$, keep five buckets in memory; memory accounting: 5 buckets with 32 blocks each, 27 blocks for partitioning to the remaining buckets on disk, and 1 block for reading R (or S), for a total of 188 blocks in use.

(e) How many IOs are performed in the optimized hash-join? Do not count the IOs to write the final result. Please explain how you obtain your IO count.

ANSWER: This answer can be computed in a variety of ways, depending on what approximations are made. Here are two examples, using the previos two scenarios: (i) $1000+167\times 5$ IOs to read and bucketize R; the same amount to bucketize S; 167×5 to read back the R buckets; plus the same amount for the S buckets; for a total of 5340 IOs. (i) $1000 + 32 \times 27$ IOs to read and bucketize R; the same amount to bucketize S; 32×27 to read back the R buckets; plus the same amount for the S buckets; for a total of 5456 IOs.