CS245A Midterm Examination

Tuesday, February 10, 1998, 11AM–12:15PM

Directions

The exam is *open book*; any written materials may be used.

For each question, circle the letter (a), (b), (c), (d) or (e) of your chosen answer. Score = 5 times number right minus number wrong, so random guessing nets you nothing on the average.

If you wish to explain or demonstrate your solution to a problem for partial credit, you may use the backs of the pages (but warn us on the front).

Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

The first two questions refer to a disk with the following characteristics. There are 10 surfaces, each with 100 tracks. Each track is divided into 18 sectors, and a sector holds 512 bytes. Blocks consist of 2 sectors. 20% of the circumference of each track is occupied by gaps between the sectors. The disk rotates at 6000rpm; i.e., one rotation every 10 milliseconds.

Question 1: The capacity of the disk most closely approximates:

(a) 0.9 megabytes (b) 1.2 megabytes (c) 2.4 megabytes (d) 4.8 megabytes (e) 9.2 megabytes.

Question 2: The transfer time for one block most closely approximates:

(a) 0.4 milliseconds (b) 0.5 milliseconds (c) 0.8 milliseconds (d) 1.0 milliseconds (e) 1.2 milliseconds.

Question 3: Suppose blocks consist of 4096 bytes. We wish to store in one block records of 100 bytes (including record header). The block is organized with a block header consisting of 100 bytes *plus* whatever space is needed for an offset table; the offset table contains an offset for each record in the block. We assume an offset requires 4 bytes. The maximum number of records we can store in one block is:

(a) 36 (b) 37 (c) 38 (d) 39 (e) 40.

Questions 4 and 5 refer to a RAID disk system with four disks. Disks 1 and 2 are data disks; 3 and 4 are redundant. Disk 3 is a mirror of disk 1. Disk 4 holds the parity check bits for the corresponding bits of disks 2 and 3; that is, each bit of disk 4 is the modulo-2 sum of the corresponding bits on disks 2 and 3.

Question 4: If in some particular bit position, disk 1 holds `0` and disk 2 holds `1`, then the corresponding bit positions for disks 3 and 4, respectively, will be:
(a) `0` and `0` (b) `0` and `1` (c) `1` and `0` (d) `1` and `1` (e) It cannot be determined from the information given.

Question 5: For some pairs of disks we can recover from two simultaneous disk crashes, and for some pairs we cannot. For which of the following pairs can we *not* recover from the crash of both at the same time?
(a) 1 and 3 (b) 1 and 4 (c) 2 and 3 (d) 2 and 4 (e) 3 and 4.

Questions 6 and 7 are based on the following situation. Blocks are 1000 bytes long; there is no need for a block header. Records are 100 bytes long, of which 12 bytes are the key field. Pointers take 8 bytes. A sequential file (sorted by the key field) consists of 10,000 records; i.e., it occupies 1000 blocks.

Question 6: The number of blocks required for a *dense* index on this file is:
(a) 20 (b) 50 (c) 100 (d) 200 (e) 500.

Question 7: The number of blocks required for a *sparse* index on this file is:
(a) 10 (b) 20 (c) 50 (d) 100 (e) 200.

Questions 8 and 9 refer to an extensible hash table, in which blocks have a capacity of three records. Keys will be hashed to 4-bit sequences. Initially, there is a bucket array with two entries, for 0 and 1; each bucket is empty. We insert, in order, 16 records whose keys hash to 0000, 0001, 0010,..., 1111.

Question 8: On the insertion of which (hashed) key does the bucket array first grow to have size 8 (i.e., the first three bits of the hash function are used)?
(a) 0011 (b) 0101 (c) 0111 (d) 1000 (e) It doesn't happen for these 16 records.

Question 9: After inserting all 16 records, the number of nonempty blocks used to hold the records is:
(a) 6 (b) 8 (c) 10 (d) 12 (e) 14.

Question 10: Suppose that blocks have 1000 bytes of available space, keys are 12 bytes, and pointers are 8 bytes. A B+ tree on a certain data file has three levels of nodes. We assume it is maintained as described in class or the text; in particular, subminimum blocks due to deletion are merged and not allowed to exist. The *minimum* number of records in the data file is:

(a) 1250 (b) 14,400 (c) 15,000 (d) 15,625 (e) 120,050.

Question 11: We wish to build a hash table on a file of 1,000,000 records. 10 records fit in one block, and 50 key-pointer pairs fit in one block. Our hash table has 1000 buckets. "Keys" are both the primary key for the records and the search key for the hash table. You may assume a pointer to the first block of each bucket is always in main memory. Two possible data structures to store this file are:

A) Keep the records themselves in a chain of blocks, one chain for each bucket.

B) Store the records in a separate file, and have in the buckets a chain of blocks holding key-pointer pairs; associated with each key is a pointer to the record with that key.

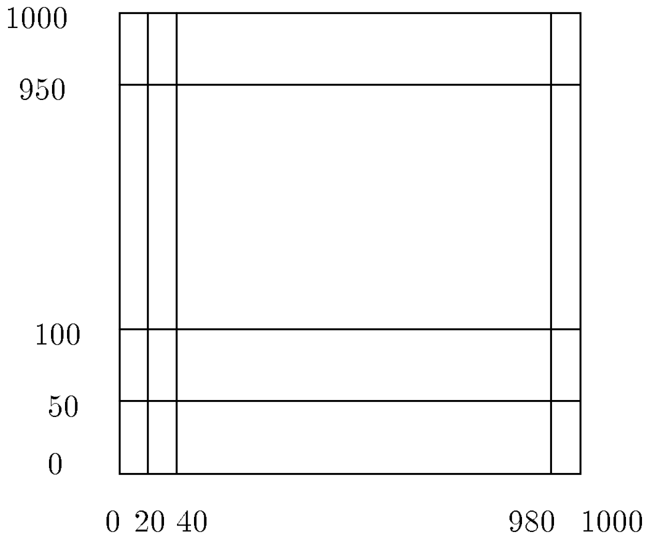
To look up a record given a key, we find the appropriate bucket and search down the chain of blocks for that bucket until we find the desired key. In scheme (B), we then follow the associated pointer to retrieve the record. Assume that on average we have to search half the records or key-pointer pairs in a bucket to find the key we are looking for. What is the ratio of the average number of disk I/O's required for structure (A) to that required for structure (B)?

(a) 0.5 (b) 1.1 (c) 2.0 (d) 2.5 (e) 4.5.

Question 12: We are merging two large, sorted files that are stored on a typical disk (you may take the Megatron 747 as "typical" if it is an issue). The system being used is "I/O bound"; it takes much more time to move data to or from the disk than it does to compute with the data in main memory. The files are each stored sorted on consecutive cylinders, and within cylinders sorted by track and then by block. A cylinder holds 1 megabyte. Unfortunately, there is only 3 megabytes of available main memory, so we use six half-megabyte buffers. Two are used to hold portions of the file currently being merged; one holds the output being constructed, two are used to receive the next half-megabyte portions of the files being merged, and the last buffer is used to hold previously formed output that is being written to disk. Which of the following would have the greatest effect on the time taken to complete this task?

(a) Double the size of main memory (b) Double the rotation speed of the disk (c) Halve the time it takes the head to move between cylinders. (d) Double the processor speed (e) Use the elevator algorithm to schedule I/O requests.

Questions 13 and 14 are based on a grid file that stores a relation $R(x, y)$. Attributes x and y are both real numbers in the range 0 to 1000. The grid partitions tuples of R into 1000 regions, using 20 stripes in the y dimension and 50 stripes in the x dimension, as suggested by the following picture.



Question 13: To answer the range query

```
SELECT *
FROM R
WHERE 310<x AND x<400 AND 520<y AND y<730;
```

the number of buckets (regions of the grid) that we have to search is:
(a) 20 (b) 22 (c) 24 (d) 25 (e) 30.

Question 14: We wish to do a nearest-neighbor query to find the (x, y) point in R closest to the point $(110, 205)$. We begin by searching the region with lower left corner at $(100, 200)$ and upper right corner at $(120, 250)$. We find that the closest point in this region is $(115, 220)$. The number of *other* regions we must search to confirm that this point is the nearest of all points is:
(a) 0 (b) 2 (c) 4 (d) 5 (e) 6.

Question 15: Suppose we store relation $R(a, b, c)$ in a partitioned hash table with 1024 buckets; that is, the hash function produces a string of 10 bits. Queries about R all specify exactly one of the attributes a, b, or c, and each of the attributes is equally likely to be specified. If the hash function produces 5 bits based only on the value of a, 3 bits based only on b, and 2 bits based only on c, then the average number of buckets that must be searched to find matching tuples is closest to:

(a) 140 (b) 180 (c) 220 (d) 260, (e) 510.

Question 16: Relation $R(a, b)$ is stored in 200 blocks, with 100 records (tuples) of R fitting in one block. Relation $S(c, d)$ is stored in 100 blocks, with 50 records (tuples) per block. When we compute $R \underset{b=c}{\bowtie} S$, we find that the average tuple of R joins with five tuples of S. How many blocks does the result of the join require?

(a) 500 (b) 750 (c) 1000 (d) 2000 (e) 3000.

Question 17: Suppose relation R occupies N and relation S occupies M blocks. We assume that $N \leq M$. Initially both relations are on disk, and we want to perform a nested-loop join in which we read each block of R and S only once (and do no other disk I/O other than writing the result). The minimum number of main-memory buffers (each the size of a block) that we need is:

(a) $N + 1$ (b) M (c) $N + M - 1$ (d) $N + M$ (e) NM.

Question 18: Let relation R have a primary key, so we can be sure it has no duplicates. Let relation S *not* have a declared primary key, so it may have duplicates. Indicate which of the following expressions is *not* equivalent to the other four:

(a) $\delta(R \times S)$ (b) $\delta(R) \times S$ (c) $R \times \delta(S)$ (d) $\delta\big(\delta(R) \times S\big)$ (e) $\delta\big(R \times \delta(S)\big)$.

Question 19: Let $R(a, b)$ be a relation, possibly with duplicates. Consider the following three expressions:

I. $\sigma_{a=1}(\gamma_{a,SUM(b)}(R))$

II. $\gamma_{a,SUM(b)}(\sigma_{a=1}(R))$

III. $\gamma_{a,SUM(b)}\Big(\delta(\sigma_{a=1}(R))\Big)$

Which of the following holds?

(a) All three are equivalent (b) No two of the three are equivalent (c) I and II are equivalent to each other and not equivalent to III. (d) I and III are equivalent to each other and not equivalent to II. (e) II and III are equivalent to each other and not equivalent to I.

Question 20: Let $R(a, b)$ and $S(b, c)$ be relations, with the following parameters: $N(R) = 50$; $N(S) = 100$; $V(R, a) = 5$, $V(R, b) = 10$; $V(S, b) = 10$; $V(S, c) = 20$. That is, R has 50 tuples, there are five different values for $R.a$, and so on. Using the approach in the notes (handout 16), the estimate of the size (number of tuples) of $\sigma_{a=1}(R \bowtie S)$ is:

(a) 20 (b) 50 (c) 100 (d) 200 (e) 500.