CS245 Database System Principles

Midterm Exam   Winter 2001

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		10
2		10
3		10
4		15
5		10
6		15

Problem 1 (10 points)

State if the following statements are TRUE or FALSE. If the statement is false, briefly explain why.

a) Insertion order into a B+ Tree will effect the tree's end structure.

b) B+ trees are typically more efficient than linear or extensible hashing for all types of queries.

c) The time for a request to wait to be serviced may be longer using the elevator algorithm than using the first-come-first-serve algorithm.

d) Consider relations $R(A,B)$ and $S(B,C)$ where $T(R) = 5000$, $T(S) = 3000$, and B is a primary key on S. The expected number of tuples in $R \bowtie S$ is less than or equal to 3000.

e) Consider two relations $R(A, B, C)$ and $S(A, D, E)$, sharing a common attribute A. It is known that $R.A$ is a foreign key referencing $S.A$, and that $S.A$ is the primary key of relation S. Then the estimated size of $R \bowtie S$ is $T(R)$.

f) For any data file, it is possible to construct two separate sparse first level indexes on different keys.

g) For any data file, it is possible to construct two separate dense first level indexes on different keys.

h) For any data file, it is possible to construct a sparse first (lower) level index and a dense second (higher) level index. Both indices should be useful.

i) For any data file, it is possible to construct a dense first (lower) level index and a sparse second (higher) level index. Both indices should be useful.

j) $\pi_S(E_1-E_2) = \pi_S(E_1) - \pi_S(E_2)$

Answers:

a) **True**. Different orders will cause different splits to occur.

b) **False**. Hashing is more efficient for single point queries.

c) **True**.

d) **False**. V(S, B) = 3000 since S is the key. T(R) * T(S) / V(S, B) = 5000* 3000 / 3000 = 5000.

e) **True**

f) **False**. Needs to be sorted on two different field

g) **True**.

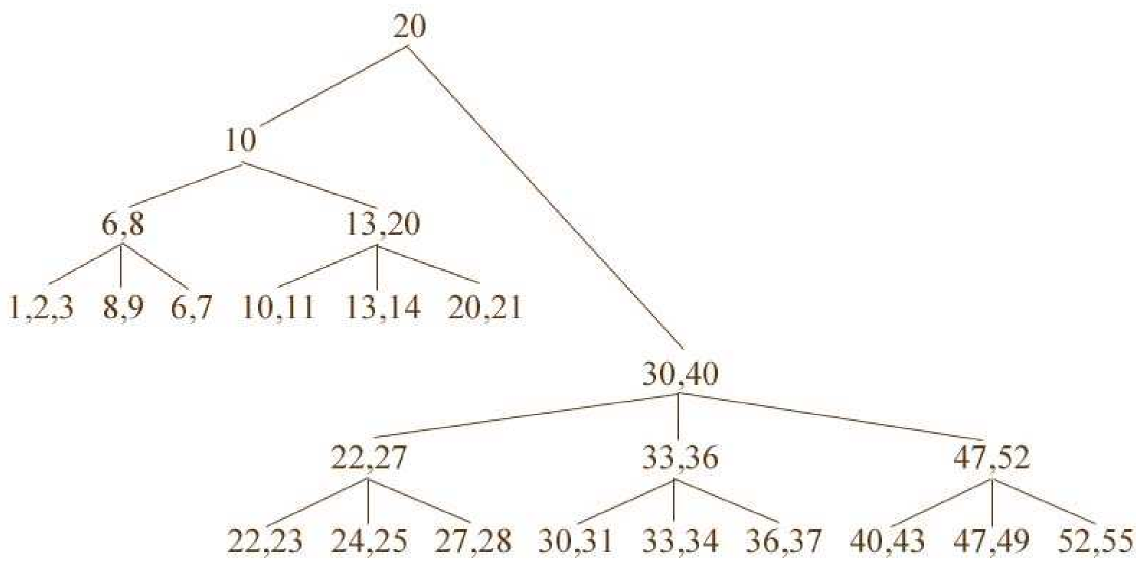
h) **False**. Dense second level index does not add any value

i) **True**

j) **False**. Pushing the select may remove additional tuples

Problem 2 (10 points)

Find any/all violations of a B+ tree structure in the following diagram. Circle each bad node and give a *brief* explanation of each error. Assume the order of the tree is 4 (*n=4; 4 keys, 5 pointers*)



Answer:

1. Interior Node 10: below min key req (3 points).
2. Interior Node 13,20: key 20 duplicated from root (3 points).
3. Leaf nodes 8,9 and 6,7: swapped positions – leaf key 6 less than parent key 8 (2 points)
4. Leaf node 20,21: both keys are not less than root.= (3 points).
5. Leaf node 22,23: key 22 not less than parent key 22 (2 points).

Note that (2) and (4) can be corrected by making root = 22

Problem 3 (10 points)

A database system uses a variant of B-trees to maintain sorted tables. In this variant, the leaves contain the records themselves, not pointers to the records. A leaf can contain as many records as will fit in a block (allowing for a sequence pointer). Non-leaf nodes are also one block in size, and contain as many keys (and appropriate pointers) as will fit. Assume the following:

1. Blocks are 4096 bytes.
2. Each record is 300 bytes long.
3. A block pointer is 10 bytes.
4. A record pointer is 12 bytes.
5. A key for the index is 8 bytes long.
6. The nodes are 85% occupied. For example, if a leaf can hold 100 records, it will only hold 85. If a non-leaf can hold 100 keys, it will only hold 85. (For 85% calculations, round to the nearest integer.)
7. The indexed file has 1,000,000 records.

How many blocks will this index use?

Answer:

Leaves can hold $\text{floor}((4096-10)/300)=13$ records. If leaves are 85% occupied, we can hold 11 records per node. This means the leaves will require 90910 blocks. For interior nodes we need to determine the value of n from $(n+1)*10+n*8 <= 4096$ which gives us $n=227$. If interior nodes are 85% occupied they actually use 194 pointers. Thus our first level above the leaves will require 469 nodes. The level above that will require 3 nodes which are all children of the root. The total is 90910+469+3+1=91383 blocks

Common Mistakes:

- Using record pointers (instead of block pointers) for the interior nodes.
- The number of interior nodes = $1 + K^2 + K^3$ where K is the number of pointers per interior node. It is possible that this answer was based on the reasoning that *every* node including the *root* must have K=194 pointers. But then this is clearly not possible. Solutions that highlighted this descrepamcy would have received full credit as well.
- (minor mistake) assuming that the number of keys = number of pointers in an interior node.
- Full credit was given to solutions that "rounded" the fractional in ways different than ours. Full credit was also given to solutions that assumed that "average" occupancy is 85%.

Problem 4 (15 points)

For this problem, consider hash indexes with the following characteristics:

1. A block can hold 50 value/pointer pairs. (The hash structure does not contain the records themselves, only pointers to them.)
2. The file being indexed has 1000 records.
3. The indexed field can contain duplicates.

a) For a linear hash index with an average occupancy (utilization) of 50%, how many blocks will the buckets require in the worst-case?

b) In case (a), what is the worst-case number of I/Os to look up a value (including the record itself)?

c) For an extensible hash index, what is the worst-case (largest possible) size of the directory? If there is no limit, state so.

d) For an extensible hash index, what is the best-case (smallest possible) size of the directory?

Answers:

a) Worst case is when all records have same key value. They will placed in one block, plus overflow blocks, a total of 1000/50 = 20 blocks. To get 50% utilization, the hash table must have 40 blocks, 39 empty and one full (with 19 overflow chained). (4 points)

Common Mistakes: None

b) 21 IOs: 1 to get to bucket, plus 19 to get to last block in the overflow chair, plus 1 to get to record. (4 points)

Common Mistakes:
- assuming overflow blocks were only at 50% utilization – not possible in linear hashing for this to happen
- assuming that all records were evenly distributed over 40 main blocks and none in overflow
- assuming to read one block required reading each record individually (i.e. 50 I/Os for one block instead of just 1 I/O for the whole block)
- forgetting that you have to read the first block and all overflows and just accounting for the 19 overflow blocks.
- forgetting to add one I/O to get the record itself

c) There is no limit; worst case is again when all records have same key (3 points)

Common Mistakes:
- assuming that there could only be one directory split per record inserted
- trying to calculate a limit based on buckets as in the homeworks and not realizing that there is no bucket limit --> thus no directory limit

d) In best case, directory has 32 entries. (20 blocks are needed to hold data, so 16 entires are not enough to point to these 20 blocks.) (4 points)

Common Mistakes:
- not realizing that the directory must be a power of 2.
- giving the size as 5 - instead of 2^5 = 32.
- using a different number of needed buckets (instead of 20) and then calculating the closest power of 2. Partial credit was given to all those who showed work so I could see where the assumed needed buckets came from.
- trying to calculate size with blocks instead of the number of entries.

Problem 5 (10 points)

Parts *a* and *b* below refer to disk with an actual (formatted) capacity of *8 gigabytes* (2^{33} bytes). The disk has *16 surfaces* and *1024 tracks*. The disk rotates at *7200 rpm*. The average seek time is *9 ms*. The block size is *8KB*.

For each of the following questions, state if there is enough information to answer. If there is, give the answer. if not, explain what is missing.

- **a)** What is the capacity of a single track?
- **b)** Suppose we are reading a file F that occupies exactly one entire track. Assume that a track can only be read starting at a particular position on the track. How long does it take to read the entire file sequentially?
- **c)** How long does it take to read a block?

Answer:

a) 8GB/(16*1024) = (2^33)/(2^4)*(2^10) = 2^19 bytes/track

b) 21.5 ms
60 s / 7200 = 8.3 ms / rotation

9 + 8.3 / 2 + 8.3 = 21.5 ms

c) Not enough information. We need to know the percentage of overhead per track.

Common Mistakes:

- When computing the time for read the file in part(b), seek time is neglected or rotational latency is mis-calculated as the time for one rotation (instead of half of that). Any of these costs 3 points.

- Didn't realize that part (c) can't be solved without being given the percentage of overhead in one track. 2 points were deducted for this. For those people mentioned the necessity of this parameter, whether or not they give an answer assuming there is no gaps in between blocks, no points were deducted.

Problem 6 (15 points)

Consider a relation *R(A,B)* stored using a partitioned hash organization. The hash table contains 1024 blocks (buckets), and the tuples (records) are stored in these blocks.

To store tuple *(a,b)*, we apply hash function *h1* for *a*, obtaining *X* bits. Then we apply hash function *h2* on *b*, obtaining *10-X* bits. The bits are concatenated, obtaining a 10 bit hash that identifies the block where *(a,b)* is placed..

Suppose that 20% of the queries on R are of the form Q_1: `SELECT * from R` where *A=a*, and 80% of the queries are of the form Q_2: `SELECT * from R where B=b` where *a* and *b* are constants.

a) How many blocks are accessed by the Q_1 queries? How many by the Q_2 quereis? (your answer will be a function of *X*.)
b) Write an expression that gives the expected number of blocks that must be accessed (on average).
c) What *X* value minimizes the expected number of IOs? (Hint: Recall that the derivative (with respect to *z*) of 2^{az+b} is $a2^{az+b}ln2$.)

Answer:

a) For Q1, 2^(10-X) blocks; for Q2, 2^X

b) 0.2*2^(10-X) + 0.8* 2^X

c) X = 4

Common Mistakes:

- Some students transposed the numbers of buckets needed for the two queries, giving an answer of X = 6, instead of 4. This error could be caused either by overlooking the given conditions in the problem, or having an incorrect understanding of partitioned hashing scheme. 8 points are deducted for this.