CS245 Database System Principles - Winter 2001

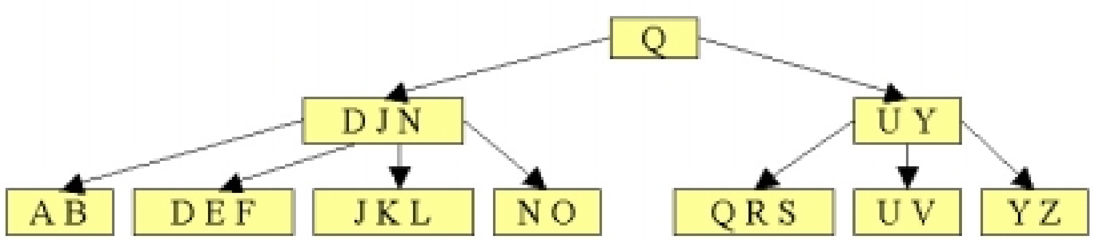
- PLEASE write your serial number on the top of your first page. If you have not received your serial number by e-mail, send a message to **orkut@stanford.edu**.
- This assignment is due in class on Thursday, Feb 1st.
- State all assumptions.
- Subscribe to **cs245-win01** to receive clarifications and changes.
- Email questions to **cs245ta-win01@lists.stanford.edu**.

Assignment 3

Problem 1 (30 points)

1. Consider a B+ tree of order 4. Show the results of inserting the keys:
   6, 14, 13, 8, 3, 9, 7, 15, 16, 17, 10, 11, 12, 1, 2, 18, 19, 4, 20, 5
   in order into an empty B+ tree. Only draw the configurations of the tree just before some node must split, and also draw the final configuration.

2. Consider the following B+ tree of order 4. Show the results of deleting *Q*, *N* and *V*, in order. Draw the configurations after each deletion.



Problem 2 (25 points)

Consider an indexed sequential file consisting of 4,000,000 blocks. Each block contains 10 fixed size records. For each key value found in the file, there are at most 5 records that share that value.

For this problem, assume that:

- Pointers to blocks are 8 bytes long
- Pointers to records are 9 bytes long (block pointer plus 1 byte offset)
- Index blocks are 4096 bytes
- There is no spanning of records in the file.
- There is no spanning of "records" in the index, e.g., if a [key, pointer] does not fit in an index block, it must move to another block.
- Search keys for file records are 12 bytes long.

a. Assume that the file and index blocks are stored contiguously on disk. How large (in blocks) would a sparse one-level, primary index be? Design the index so it would use the minimum amount of space.

b. Assume that the file and index blocks are *not* contiguous on disk. How large (in blocks) would a sparse one-level, primary index be? Design the index so it would use the minimum amount of space.

c. Suppose you now construct a second level index on the index of part (b). How large would it be, in blocks? Also assume you want to minimize space (but no spanning).

d. Next you construct a one level, dense secondary index. Compute its minimum size (in blocks). The secondary keys are also 12 bytes, and index blocks are contiguous. Also assume that no buckets are used; if multiple values of a secondary key occur in the file, the keys are replicated in the index. (We do not expect many duplicates, so it is not worth optimizing for them.)

e. Suppose you construct a second level index for the index of part (d). How large would it be in blocks? Also assume you want to minimize space (but no spanning).

Problem 3 (25 points)

Consider an index organized as a B+ tree. The leaf nodes contain pointers to a total of *N* records, and each block that makes up the index has *m* pointers. We wish to choose the value of *m* that will minimize search times on a particular disk device with the following characteristics:

- For the disk that will hold the index, the time to read a given block into memory can be approximated by *(70 + .05*m)* milliseconds. The *70* milliseconds represent the seek and latency components of the read, the *.05*m* milliseconds is the transfer time. That is, as *m* becomes larger, the larger the block will be and the more time it will take to read it into memory.)
- Once the block is in memory a binary search is used to find the correct pointer. So the time to process a block in main memory is $a + b\log_2 m$ milliseconds, for some constants *a*, *b*.
- The main memory time constant *a* is much smaller than the disk seek and latency time of *70* milliseconds.
- The index is full, so that the number of blocks that must be examined per search is $\log_m N$.

Answer the following:

1. What value of m minimizes the time to search for a given record? An approximate answer is OK. The value you obtain should be independent of *b*. (HINT: If you come up with an equation which is hard to solve algebraically, try plugging in values to locate the root of the equation.)

2. What happens as the seek and latency constant (*70*ms) decreases? For instance, if this constant is cut in half, how does the optimum *m* value change?

Problem 4 (20 points)

a. What is the general expression for the minimum number of record pointers an order *n* B+ tree can contain, when it has *j* levels?

b. If we are told that an order n B+ tree points to r records, then what is the maximum number of levels, j, it may have? That is, derive an expression of the form $j \le f(n,r)$. Note that this is a bound on the number of B+ tree nodes we need to examine for looking up a record (given its key value), when the file we are indexing contains r records.