Assignment 5

CIS451/551, Fall 2006
due 5:00pm Thursday, November 16

1. A primary index can be either sparse or dense, but a secondary index must necessarily be dense.
   a. Please explain why.

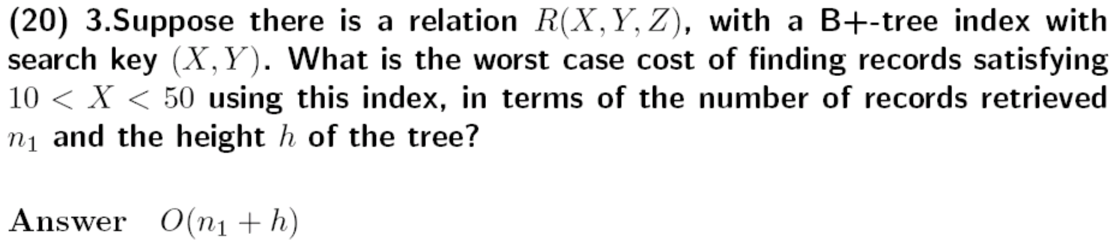
   b. Now explain why a B+ Tree is like a sparse, secondary, multi-level index (drawing a picture might help). Does this contradict your previous explanation? Why not? (HINT: Identify what part of the tree specifically is sparse versus what part is dense.)

2. Construct a B+ Tree for the following set of key values: (30,0,29,1,27,2,21,3,15,5,6). Assume the index is initially empty and you are constructing the tree based on insertions in the order given. Use *n = 4* as your node size (the number of pointers in the node).

   a. Delete the key values (in this order): 21, 29. Show the resulting tree. (HINT: Check the algorithm carefully to see what to do with the internal node containing "21" and "29" .)

   b. Delete 30. Show the resulting tree.

3. a. Last year, we gave the problem and answer below. Please explain why the answer provided is correct. We suggest that you sketch a figure of a relevant, hypothetical B+ Tree and use it in your explanation.



(20) 3.Suppose there is a relation $R(X, Y, Z)$, with a B+-tree index with search key (X, Y). What is the worst case cost of finding records satisfying $10 < X < 50$ using this index, in terms of the number of records retrieved n_1 and the height h of the tree?

Answer $O(n_1 + h)$

   b. Please explain why a B+ Tree is a better choice than a Hash index to handle frequent range queries on the search key such as: *select * from account where branch < "Perryridge"* or the one given above in (3a).

4. Suppose that we use extendable hashing on a file that contains records with the following search-key values:

   ```
   5, 6, 9, 10, 18, 31, 39, 42, 55, 57
   ```

   Show the extendable hash structure for this file if the hash function is

   h(x) = x mod 7

   and each bucket can hold 3 records.

5. ***Extra credit***. Convince us whether the following statement is true or false. When constructing a hash mapping to *n* buckets using the mod function (i.e., a function form *h(x) = x mod n)* you will generally have a better (random and/or uniform) distribution if *n* is prime.