COSC-4411: Assignment #2

Due: Friday 15 October 2004

---

1. **I/O.** *I/O, I/O, it's off to work we go.*

   Consider file **R** which has 10,000,000 records, and that we have a B+ tree index on this file with the search key as **R**'s primary key. Assume that the order (d) of the B+ tree is 75, and that the index pages are 70% full, on average.

   a. Assume that the B+ tree index above for **R** is of alternative *one*; that is, the leaf pages of the B+ tree contain the data records themselves (at 20 data records per page).

      You want to find all records with search key $\geq x$ and search key $\leq y$. Say that 1,000 data records qualify. How many I/O's does it cost you to retrieve these?

   b. Assume that the B+ tree index above for **R** is of alternative *two* and is unclustered; that is, the leaf pages of the B+ tree contain data entries that are $\langle \mathsf{key}, \mathsf{rid} \rangle$ pairs and not the data *records* themselves. Say that 50 data entries fit per page, and say that your buffer pool is absurdly small with just five frames.

      Again, you want to find all records with search key $\geq x$ and search key $\leq y$, and 1,000 data records qualify. How many I/O's does it cost you to retrieve these?

   c. Now assume that we have an index on **R**'s primary key instead that is an extendible hash index. Assume that the index is of alternative *two* and is unclustered. We want to find the record with search key value x. How many I/O's does it cost you to retrieve the record?

---

2. **B+ Trees.** *Root-bound.*

   a. Starting from scratch, show how a B+ tree of order one would appear after adding keys 1, 2, 3, 4, 5, 7, and 8 in that order.

   b. Now add key 6 to the tree from 2a. Do not do any redistribution.

   c. If you did a bulk build in 2a instead, would the tree have looked different?

---

3. **Page Layout.** *Open-ended.*

   The page layout scheme described in the textbook for variable length records places a limit in advance on how many records can be placed on the page. This is because the slot array is of fixed length.

   Briefly describe a new, reasonable scheme that would eliminate this restriction.

   What are the disadvantages of your scheme compared with the textbook's?

4. **Buffer Pool & Replacement Strategies.** *Pinned and pinned again.*

   Some pages in the buffer pool have at some time or another a pincount of more than one. Dr. Dogfurry has noted that one can consider pages pinned more times than other pages as more "popular". So we should favour them, leaving them in the buffer pool longer.

   Design a replacement strategy that does that. How long a page remanins in the buffer pool should be correlated with how many times it has been pinned while in the buffer pool *this* time. Hint: Think about the *clock* strategy.

5. **Compression.** *Paradox?*

   Dr. Dogfurry is consulting for Applications, Inc. They are exploring ways to save on disk-space usage for their data-intensive applications. So naturally, Dr. Dogfurry suggested that the applicaton server use a compression algorithm on programs' data whenever it is to be written to disk. Then, of course, the applicaton server would need to uncompress the data on the fly when it reads it back into main memory.

   Dr. Dogfurry said this was a trade-off: you save disk-space; but you lose time. Applications are slowed down because of the compression and decompression that must be done.

   However, once Applications, Inc., implemented this, they found it sped up the application server! Why did it speed up?