CS143: Database Systems
Homework #4

1. We want to store the table created by the following SQL statement into a disk.

```
CREATE TABLE Class(
  dept  CHAR(2),
  cnum  INTEGER,
  sec   INTEGER,
  unit  INTEGER,
  year  INTEGER,
  quarter INTEGER,
  title CHAR(30),
  instructor CHAR(20)
)
```

6 surfaces/disk * 10,000 tracks/surface * 500 sectors/track * 1KB/sector = 30GB/disk

10ms (seek time) + 5ms (rotational delay) + 0.02ms (transfer time) = 15.02ms

(1024 bytes/block)/(1 tuple/72 bytes) = 14 tuples/block.
(1024 tuples/table)/(14 tuples/block) = 72 blocks/table.

We need to store tuples for 1,000 classes that have been offered so far. 10 classes are offered every year. The tuples are stored in random order (i.e., they are not sequenced by any attribute). A disk of the following parameters is used for storing the table.

- 3 platters (6 surfaces)
- 10,000 cylinders
- 500 sectors per track
- 1024 bytes per sector
- 6,000 RPM rotational speed
- 10ms average seek time

10ms (seek time) + 5ms (rotational delay) + 72*0.02ms (transfer time) = 16.44ms

24* (10ms (seek time) + 5ms (rotational delay) + 3*0.02ms (transfer time)) = 361.44ms

(a) What is the capacity of this disk? 30GB/disk

(b) What is the average time to read a random sector from the disk? 15.02ms

(c) Assume one disk block corresponds to one disk sector. How many disk blocks are needed to store the above table with 1,000 tuples? 72 blocks

(d) We want to run the following query by scanning the entire table.

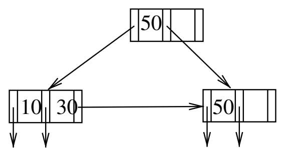
```
SELECT * FROM Class WHERE year = 2005
```

Assuming that all blocks for the table is allocated sequentially, how long will it take to run the query? Assume that the disk head is not on the same track where the first block of the table is stored. 16.44ms

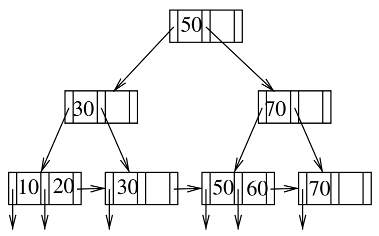
(e) Now assume that due to frequent updates to the table, disk blocks are allocated such that, on average, sequentiality is broken every three blocks. That is, the table is stored in 24 randomly located "clusters" of 3 consecutive blocks. Assuming that we scan the entire table to execute the above query, how long will it take? 361.44ms

(f) Now assume that we have a B+tree on the year attribute and the tree has already been loaded into main memory. None of the disk blocks containing the Class table has been cached in main memory. What is the expected time to run the above query? Is it helpful to create a B+tree to run this query? 150.2ms. Since the tuples are not clustered by the search key, we will need to do 10 random IOs to retrieve all 10 tuples. Therefore, 10 * (10ms+5ms+0.02ms) = 150.02ms. If all blocks are allocated sequentially, using the index may actually slow down the query execution.

2. Consider the following two B+trees for this problem.



(a)



(b)

(a) Show the final B+tree structure after we insert 60, 20, and 80 into Figure (a) in the given order.

(b) Show the final B+tree structure after we delete 20, 10, and 70 from Figure (b) in the given order.

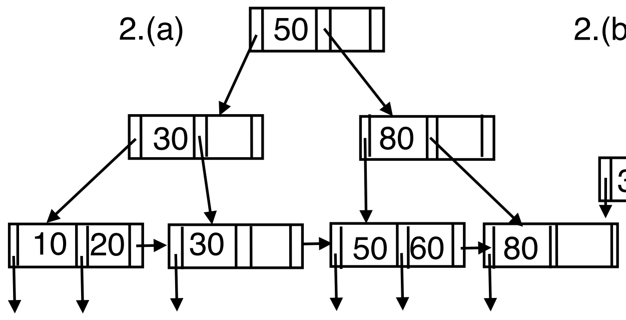
3. Consider a B+tree that indexes 300 records. Assume that $n = 5$ for this B+tree (i.e., each node has at most 5 pointers), what is the minimum and maximum height (depth) of the tree? (A tree with only the root node has a height of 1.)

4. Consider the following key values:

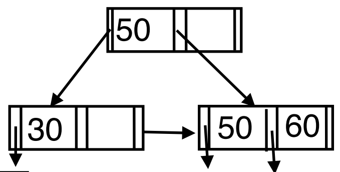
```
106, 115, 916, 0, 96, 126, 16, 15, 31
```

These keys are to be inserted in the above order into an (initially empty) extendible hash table. The hash function $h(n)$ for key n is $h(n) = n \bmod 256$; that is, the hash value is the remainder when the key value is divided by 256 (2^8). Thus, the hash value is an 8-bit value. Each block can hold 3 data items. Draw the extendible hash table after all data items are inserted. Show the keys themselves in the buckets, not the hash value. The bucket numbers are drawn from the bits at the high order end of the hash value. Be sure to indicate i for the directory, the number of hash value bits used. Also indicate i for each bucket, the number of hash function bits that are used for that bucket.

2.(a)

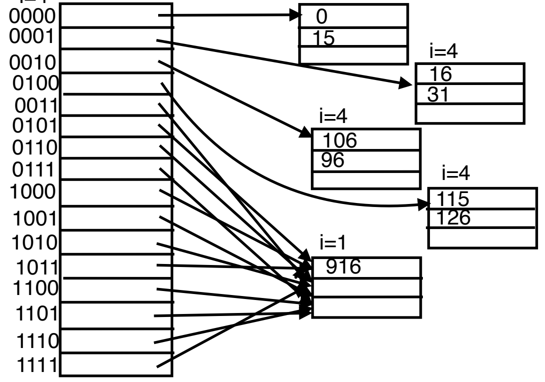


2.(b)



3. Minimum 4. (maximum 4 record pointers per node at leaf. 300/4 = 75 leaf nodes are needed when ful. maimum branching factor 5 at non-leaf nodes. 75/5 = 15 nodes are needed at level 2. 15/5 = 3 node are needed at level 3. One more level of root node that points to these three nodes.)
Maximum 5. (minimum 2 record pointers per node at leaf. 300/2 =150 leaf nodes. minimum branching factor 3 at non-leaf nodes. 150/3 = 50 nodes at level 2. 50/3 = 16 nodes at level 3. 16/3 = 5 nodes at level 4. 5/3 = 1 nodes at level 5. Since there is only one node at level 5, this is the root node.)

4.
H(106) = 106 = 01101010
H(115) = 115 = 01110011
H(916) = 148 = 10010100
H(0) = 0 = 00000000
H(96) = 96 = 01100000
H(126) = 126 = 01111110
H(16) = 16 = 00010000
H(15) = 15 = 00001111
H(31) = 31 = 00011111


i=4
0000
0001
0010
0100
0011
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
i=4
0
15
i=4
16
31
i=4
106
96
i=4
115
126
i=1
916