CIS/CSC384

HomeWork 7: Due Apr 20, 2010 before class
You will turn in the assignment via bb. Turn in a single word/pdf document.

1. Use the following data about a table to answer the questions below. [2 pts]

   Row size = 100 bytes
   Number of rows = 100,000
   Physical Record Size = 4,096 bytes
   Primary Key size = 6 bytes
   Pointer size = 4 bytes

   (a) How many rows (logical records) can be fit into a physical record? Assume that only complete rows are stored in a physical record.

   Answer: 4096/100 = 40 rows in a physical record.

   (b) What is the number of physical records needed for a sequential file?

   Answer: 100,000 rows/40 rows in a PR = 2500 physical records

   (c) If an unordered sequential file is used, calculate the number of physical records that need to be accessed to find a key (find rows with specified key values) that does not exist in the table.

   Answer: If it does not exist, we must scan all the physical records. Therefore 2500 physical records will be accessed.

   (d) In a node in the B+ tree we can hold x key values and (x + 1) pointers. Suppose we build a B+ tree index on the primary key (key size = 6 bytes), what is the maximum possible value for x?

   Answer: We can have max of 409 keys and (409 + 1) pointers – they require 4094 bytes (which is <= the size of a physical record).

2. Consider the following query:
   SELECT * FROM Customer
   WHERE CustBalance > 5000 AND CustState='CO'

   Suppose that the total number of rows in the Customer table is 7240. The CustBalance ranges from 0 – 10000. [1 pt]

   (a) Using the uniform distribution assumption, state how many rows are expected to satisfy the condition CustBalance > 5000 (see that the condition CustState='CO' is not used).

   Answer: 50% [given by (10000 – 5000)/(10000 – 0)] of the Customers are expected to satisfy the CustBalance > 5000 condition. In other words, 3620 Customers satisfy.
   Grading – it is ok if the students use (10000 – 5001)/10000

(b) Use the following histogram to estimate the number of rows expected to satisfy the condition CustBalance > 5000.

Histogram for CustBalance

Range	Number of Rows
0 – 100	1000
101 – 250	950
251 – 500	1050
501 – 1000	1030
1001 – 2000	975
2001 – 4500	1035
4501 - 10000	1200

Answer: Use uniform distribution in the range, and we get the expected number of customers = 1200 * (10000 – 5000)/(10000 – 4501) = 1091
Grading: Again it is ok if the students use 1200 * (10000 – 5001)/(10000 – 4501) – Look to see that they understand the main idea.

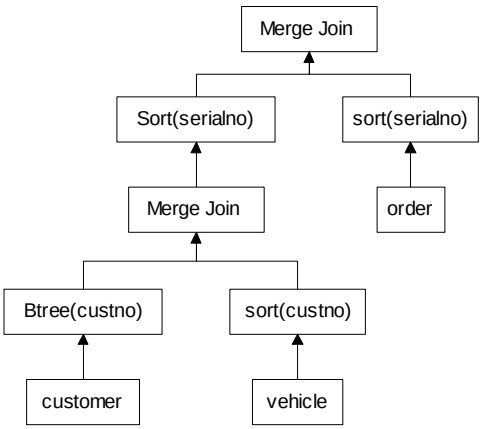
3. Consider the following query:

```
SELECT OrdNo, OrdDate, Vechicle.ModelNo
FROM Customer, Order, Vehicle
WHERE CustBalance > 5000 AND Customer.CustNo = Vehicle.CustNo
        AND Vehicle.SerialNo = Order.SerialNo
```

Draw the access plan showing the following:

You access Customer using a Btree on CustNo, you sort vehicle on CustNo
Now you join Customer and Vehicle using Sort Merge Join (the Btree on CustNo gives you the customers sorted by CustNo, and we sorted vehicles on CustNo)
You sort order by serialNo, and you sort the result of the above Sort Merge Join on serialNo.
Now you join the order table with the result of the first sort merge join using another sort merge join. [1 pt]

Answer:



4. For the following indexes, and the query conditions, indicate if the index matches the condition. [3 pts]

   (a) B tree Index on TrdDate. Query Condition: TrdDate BETWEEN '1-Oct-2006' AND '31-Oct-2006'

   Answer: Yes.

   (b) B-tree Index on CustPhone. Query Condition: CustPhone LIKE '(810)%'

   Answer: Yes

   (c) B-tree Index on TrdType. Query Condition: TrdType <> 'BUY'

   Answer: No

   (d) BITMAP Index on BondRating. Query Condition: BondRating IN ('AAA', 'AA', 'A')

   Answer: Yes

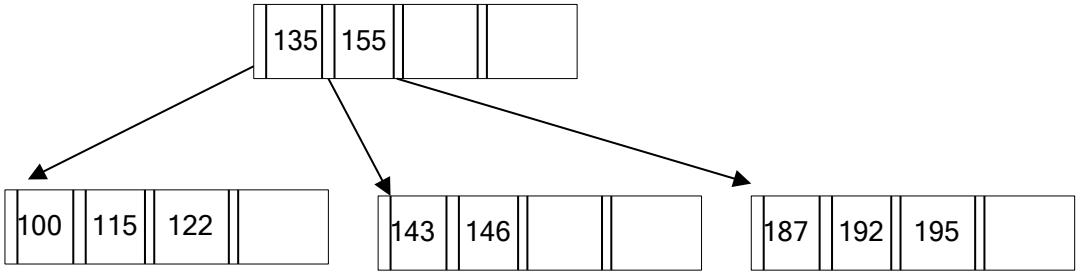
   (e) B-tree Index on <CustState, CustCity, CustZip>. Query Condition: CustState ='CO' AND CustCity = 'Denver'

   Answer: Yes, both the conditions together match the index.

   (f) B-tree Index on <CustState, CustCity, CustZip>. Query Condition: CustState IN ('CO', 'CA') AND CustZip LIKE '8%'

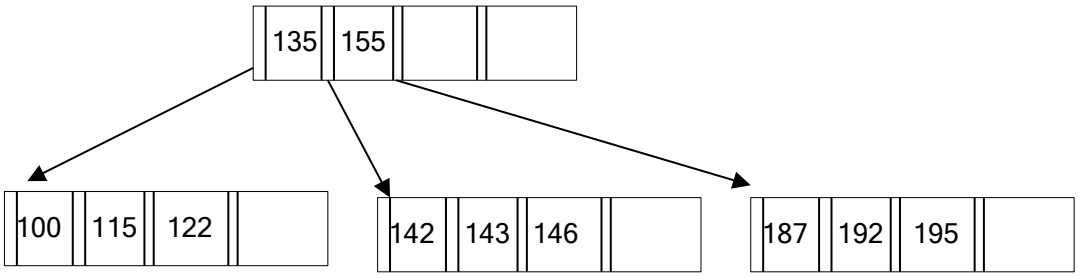
   Answer: Yes, only the first condition matches the index.

5. Consider the B tree shown below.

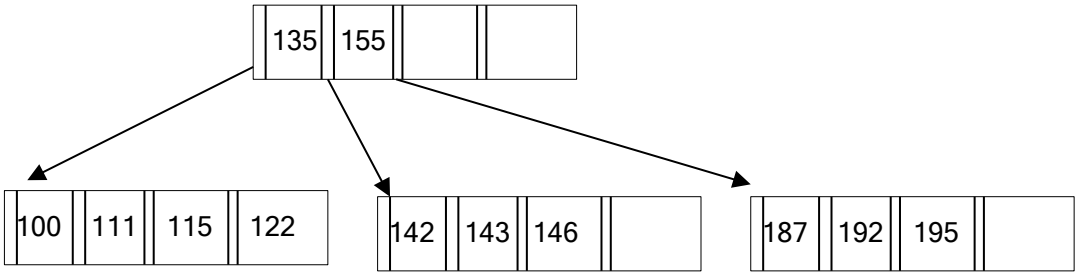


You are going to insert the following keys. Show the B tree after each insertion.
142, 111, 134, 170, 175, 127, 137, 108, 140 (Note that first you insert 142, then to the B tree resulting after inserting 142, you insert 111, to the resulting B tree you insert 134 and so on..) [2 pts]

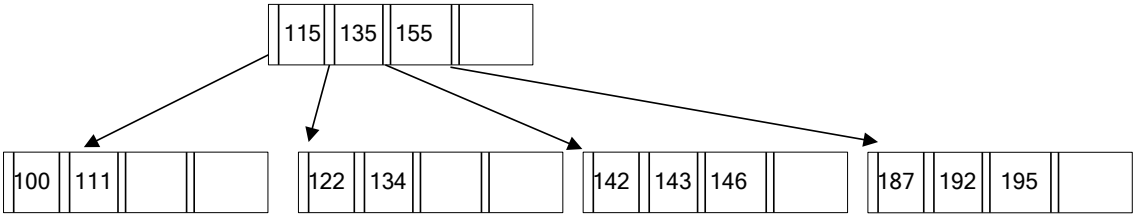
- Key 142 is added to the middle node before 143 as shown below.



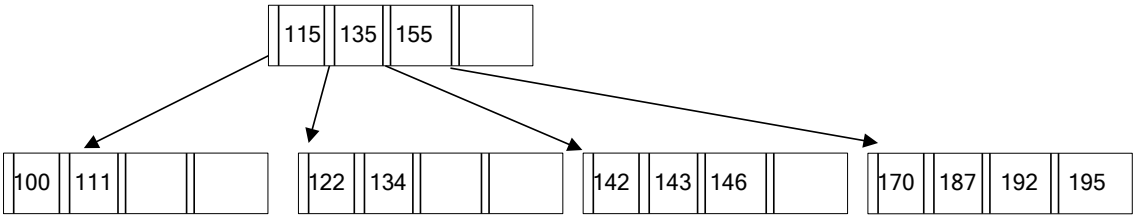
- Key 111 is added to the left most node between 100 and 115.



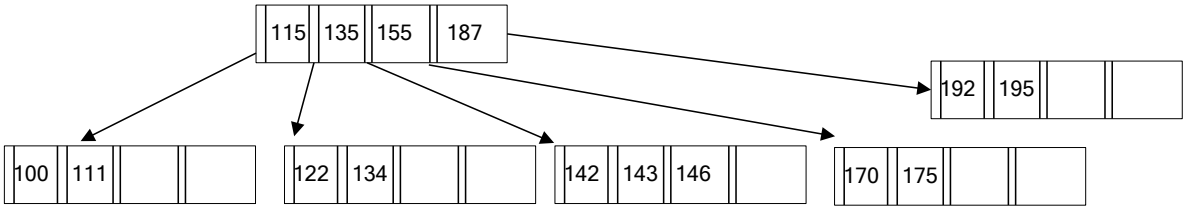
- Key 134 causes a split of the left-most node as shown below.



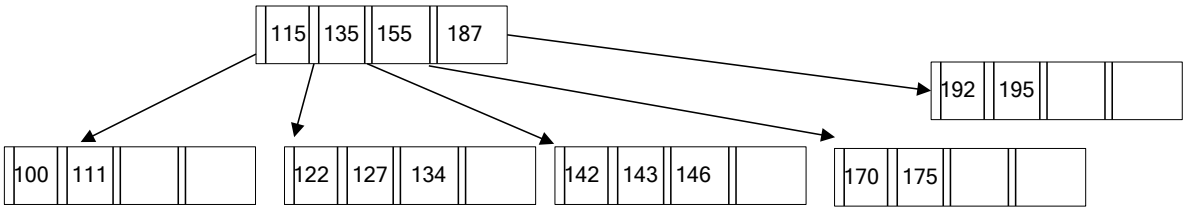
- Key 170 is added to the right-most node as shown below.



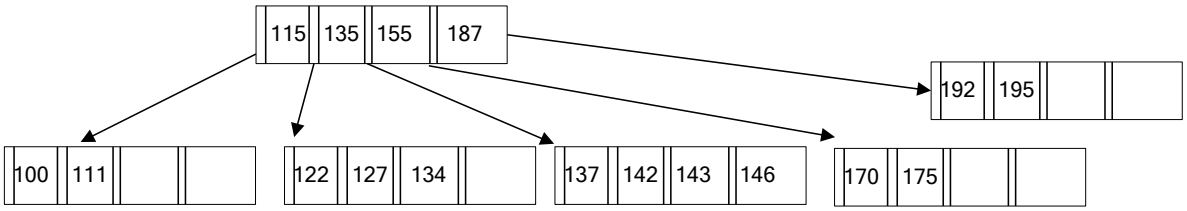
- Key 175 causes a split of the right-most node as shown below.



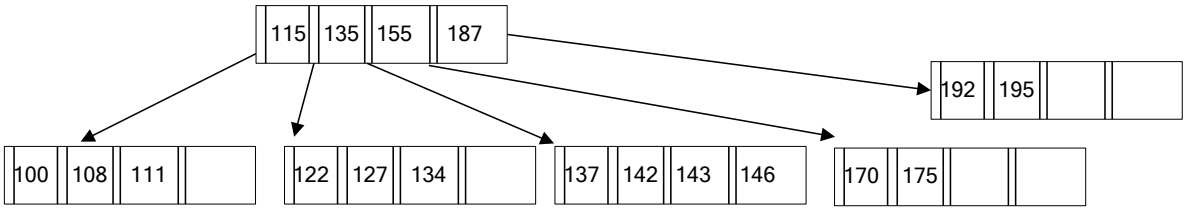
- Key 127 is added to the second node as shown below.



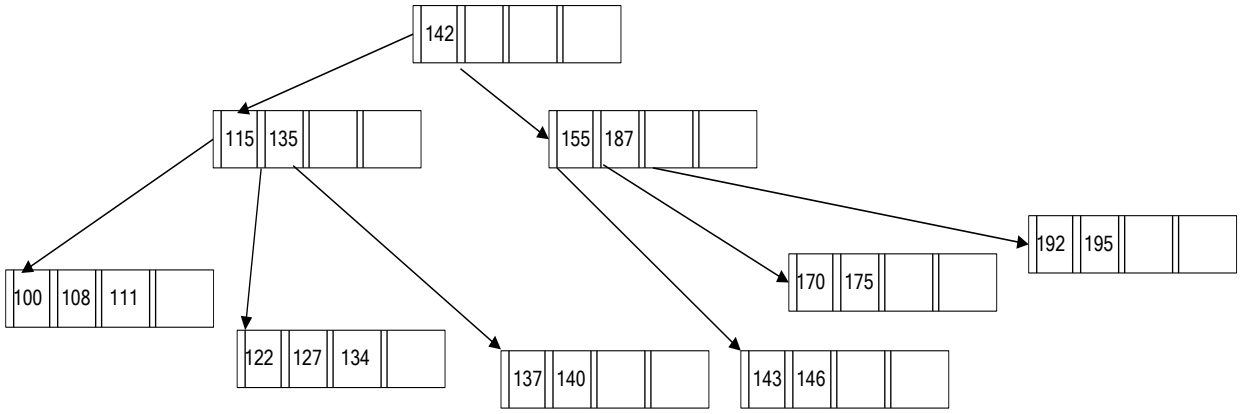
- Key 137 is added to the middle node as shown below.



- Key 108 is added to the left-most nods as shown below.

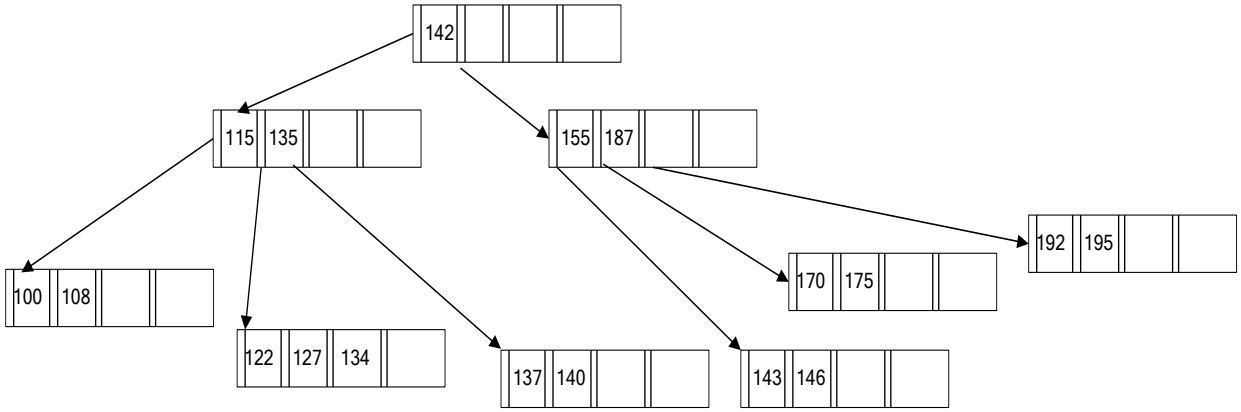


- Key 140 causes two node splits resulting in a new level in the Btree.

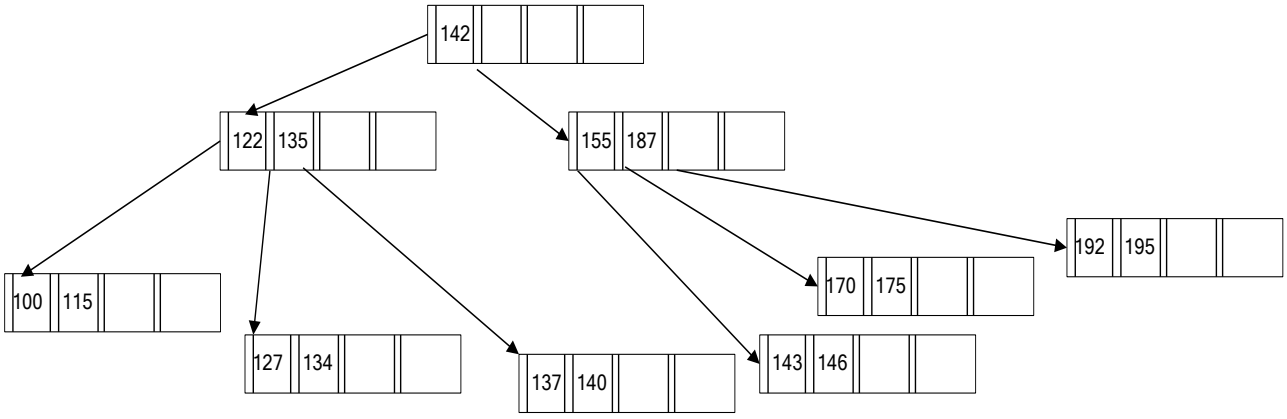


6. Consider the B tree resulting from the problem above. Delete the following keys one after the other and show the B tree after each deletion. Keys to be deleted: 111, 108, 137. [1 pt]

- Key 111 is removed from the first node as shown below.



- Key 108 causes a redistribution of keys among the left-most leaf nodes and the parent node as shown below.



- Key 137 causes a node concatenation at two levels resulting in a tree that loses a level.