**Core content homework assignment 5 (CC5)**

Qu 1. Some Dickinson students have developed a new SQL database appliance called DsonDB. DsonDB uses B-trees for column indexes, with 64-bit pointers to disk blocks. The DsonDB appliance uses an array of disks that have blocks of size 8 KiB. The Math/CS club has used DsonDB to create a database of mathematical theorems. The table storing names and descriptions of the theorems is defined as follows.

CREATE TABLE `theorem` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`theorem\_name` char(200) NOT NULL,

`description` text DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `theorem\_name` (`theorem\_name`)

)

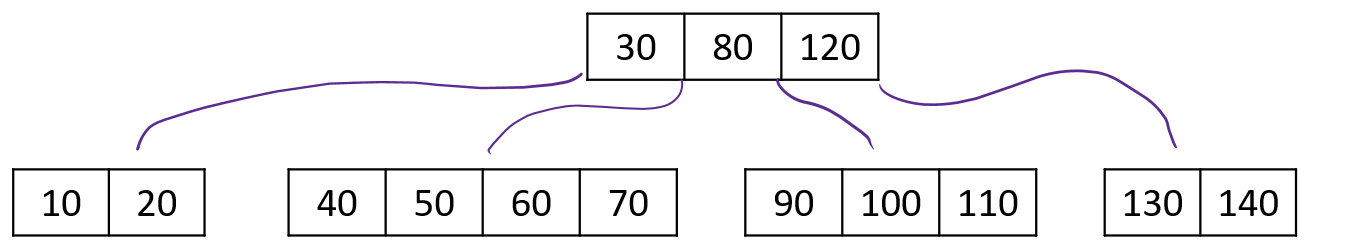
(a) (10 points) What order of B-tree would be used by DsonDB for the theorem\_name index?

(b) (10 points) Suppose there is a maximum of 10,000 theorems in the database. In the worst case, how many disk accesses would be needed to retrieve the description of a theorem based on its name? (You may assume the root node is at least half full.)

(c) (10 points) As part of an honors project, a senior math student uses some automated theorem-proving software to add a billion new theorems to the database. In the worst case, how many disk accesses are now required to retrieve the description of a theorem based on its name? (You may assume the root node is at least half full.)

(d) (10 points) If your answers to the previous two parts of this question are correct, you would have discovered that when we scale the database from 10,000 items to 1 billion items, the number of disk accesses required to retrieve an item increases by less than a factor of 2. So the database size increased by a factor of 100,000 while our access time increased by a factor of less than two. In a few sentences of your own words, explain how and why the database achieved this remarkable scalability of performance.

Qu 2. (10 points) Consider the following B-tree. It is an order 2/4 B-tree, meaning the non-root nodes contain a minimum of two and maximum of four keys.



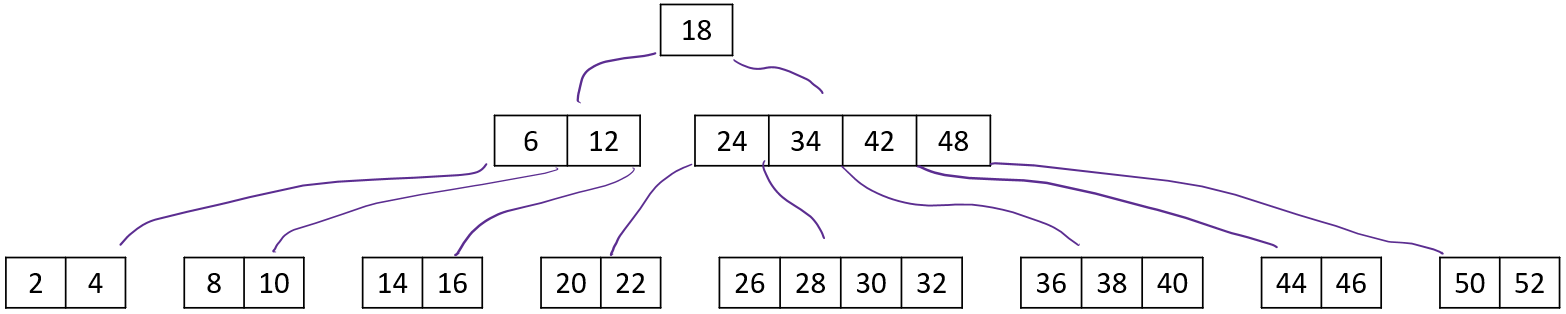
The above B-tree can also be represented in string format as follows:

30,80,120

10,20; 40,50,60,70; 90,100,110; 130,140

What is the state of the B-tree after we insert first the key 105 then the key 45? You may draw your answer graphically or give it in string format.

Qu 3. (10 points) Consider the following B-tree. It is an order 2/4 B-tree, meaning the non-root nodes contain a minimum of two and maximum of four keys.



The above B-tree can also be represented in string format as follows:

18

6,12; 24,34,42,48

2,4; 8,10; 14,16; 20,22; 26,28,30,32; 36,38,40; 44,46; 50,52

What is the state of the B-tree after we insert first the key 33? You may draw your answer graphically or give it in string format.

Qu 4. (40 points; 10 points for each part) For the purposes of this question, imagine a database similar to the textbook's wine database, except that the number of products and suppliers is much larger. There are still only four product types (red, white, rose, sparkling), but there are 100,000 products and thousands of suppliers. Statistical analysis of the database has shown that the number of products in each of the four product types is approximately equal and that 99% of products are usually available (i.e. their available quantity is non-zero). Suppose that the following query is issued to the database.

SELECT \* FROM `product` p WHERE p.AVAILABLE\_QUANTITY=0 AND p.PRODTYPE='red'

In each of the following four scenarios, what strategy is the query optimizer likely to use for executing this query? In each case, briefly justify your answer, and give the approximate total number of table rows that is likely to be retrieved during query processing.

(a) The AVAILABLE\_QUANTITY and PRODTYPE columns are both indexed.

(b) Only the AVAILABLE\_QUANTITY column is indexed.

(c) Only the PRODTYPE column is indexed.

(d) Neither column is indexed.

Qu 5. (30 points; 10 points for each part) As in the previous question, imagine a database similar to the textbook's wine database, except that the number of products and suppliers is much larger. Let denote the number of purchase orders and denote the number of suppliers. Consider the following join query.

SELECT \* FROM `purchase\_order` p, `supplier` s WHERE p.SUPNR=s.SUPNR

We assume that the purchase\_order table has non-unique index on SUPNR and the supplier table has a primary index on SUPNR.

Use big- notation to give a reasonable estimate of the asymptotic running time of this query, in terms of and , in the following three scenarios. Briefly justify your answer in each case.

(a) The query is accomplished via a nested-loop join with the purchase\_order table being the outer table.

(b) The query is accomplished via a sort-merge join.

(c) The query is accomplished via a hash join.

Total points on assignment: 130 points