BSCCS2001: Practice Solutions
Week 9

1. Consider a B-tree of order 17.

[MCQ: 2 points]

What are the minimum number of child pointers and keys that can be placed in a non-root node?

- √ min. number of child-node pointers = 9,
  min. number of keys = 8.
- ◯ min. number of child-node pointers = 2,
  min. number of keys = 1.
- ◯ min. number of child-node pointers = 9,
  min. number of keys = 9.
- ◯ min. number of child-node pointers = 17,
  min. number of keys = 16.

Solution: A non-root node of a B-tree of order $p = 17$ has minimum $\lceil \frac{p}{2} \rceil = \lceil \frac{17}{2} \rceil = 9$ child-node pointers and $\lceil \frac{p-1}{2} \rceil = \lceil \frac{17-1}{2} \rceil = 8$ keys.

2. Consider a non-empty B^+-tree of order 17.

[MCQ: 2 points]

What are the maximum and minimum number of keys that can be placed in the root node?

- ◯ max. number of keys = 16,
  min. number of keys = 8.
- ◯ max. number of keys = 17,
  min. number of keys = 8.
- √ max. number of keys = 16,
  min. number of keys = 1.
- ◯ max. number of keys = 16,
  min. number of keys = 2.

Solution: A non-empty B^+-tree must have a minimum of one key and a maximum of $p - 1$ keys, where p is the order of the B^+-tree. Thus, in this case the maximum number of keys = 16, and the minimum number of keys = 1.

3. Suppose a data file has 1,00,000 records. Each disk block contains 10 such records (records are unspanned and of fixed-length) and the total space required to store the file is 100 MB (megabytes). We consider a primary (sparse index with an index entry for every block in file) index is constructed on the data file. The key field is 30 bytes and pointer field is 10 bytes for each entry of the index. How many blocks are required to store the index?

[NAT: 2 points]

Answer: 382

Solution: Since the data-file has 1,00,000 records and each disk block contains 10 such records, the total number of blocks required to store the entire data-file is $\frac{100000}{10} = 10000$.
The 10,000 blocks are stored in 100 MB (megabytes). Thus, each disk block size is

$$\frac{100 \times 2^{20}}{10000} = 10485.76 \text{ bytes.}$$

But we must consider the nearest smaller whole number as the usable block size because the records are unspanned.
Therefore, block size = 10485 bytes.
A dense index on the primary key requires one entry for each record in the given relation (thus, 1,00,000 entries) and the size of each entry is:
size-of-key-field + size-of-pointer-field $= 30 + 10 = 40$ bytes.
So, the space required to store the index is $40 \times 100000 = 4000000$ bytes.
Thus, the number of blocks required to store the index is

$$\left\lceil \frac{4000000}{10485} \right\rceil = 382$$

.

Consider a multilevel index with four levels as L_1, L_2, L_3 and L_4. Let L_1 be the innermost and L_4 be the outermost levels. Let the index blocking factor or the maximum number of entries held by a block be 50. With the given information, answer the questions 4 and 5.

4. If the number of blocks in level L_1 is 1,00,000, then how many blocks are required at L_2, L_3 and L_4?

[MCQ: 2 points]

- √ Number of blocks at L_2 is 2000,
  Number of blocks at L_3 is 40,
  Number of blocks at L_4 is 1
- ◯ Number of blocks at L_2 is 2000,
  Number of blocks at L_3 is 50,
  Number of blocks at L_4 is 1
- ◯ Number of blocks at L_2 is 2000,
  Number of blocks at L_3 is 200,
  Number of blocks at L_4 is 4
- ◯ Number of blocks at L_2 is 20000,
  Number of blocks at L_3 is 2000,
  Number of blocks at L_4 is 10

Solution: Number of blocks in L_2 is $\lceil \frac{100000}{50} \rceil = 2000$.
Number of blocks in L_3 is $\lceil \frac{2000}{50} \rceil = 40$.
Number of blocks in L_4 is $\lceil \frac{40}{50} \rceil = 1$.

5. Given the multilevel access structure, the number of block accesses required to read a record is .....

[MCQ: 2 points]

- √ 5
- ◯ 4
- ◯ 3
- ◯ 17

Solution: For each level, there is a block access and access to the block having the target record. Thus, the number of block accesses required is 5.

6. Consider a B-tree based index with order $p = 19$. Assume each node of the B-tree is 73% full. If the height of the tree is 3, then what is the maximum number of keys that can be accommodated in the given B-tree?

[MCQ: 2 points]

- √ 38,415
- ◯ 8,663
- ◯ 12,765
- ◯ 68,605

Solution: On an average, each node will have 0.73×19 or approximately 14 tree pointers. Thus, on an average, each node will have 13 keys.

Level	**Nos. of nodes**	**Nos. of Keys**	**Nos. of pointers**
Level-0 (root)	1	$1 \times 13 = 13$	$1 \times 14 = 14$
Level-1	14	$14 \times 13 = 182$	$14 \times 14 = 196$
Level-2	196	$196 \times 13 = 2,548$	$196 \times 14 = 2,744$
Level-3	2,744	$2,744 \times 13 = 35,672$	

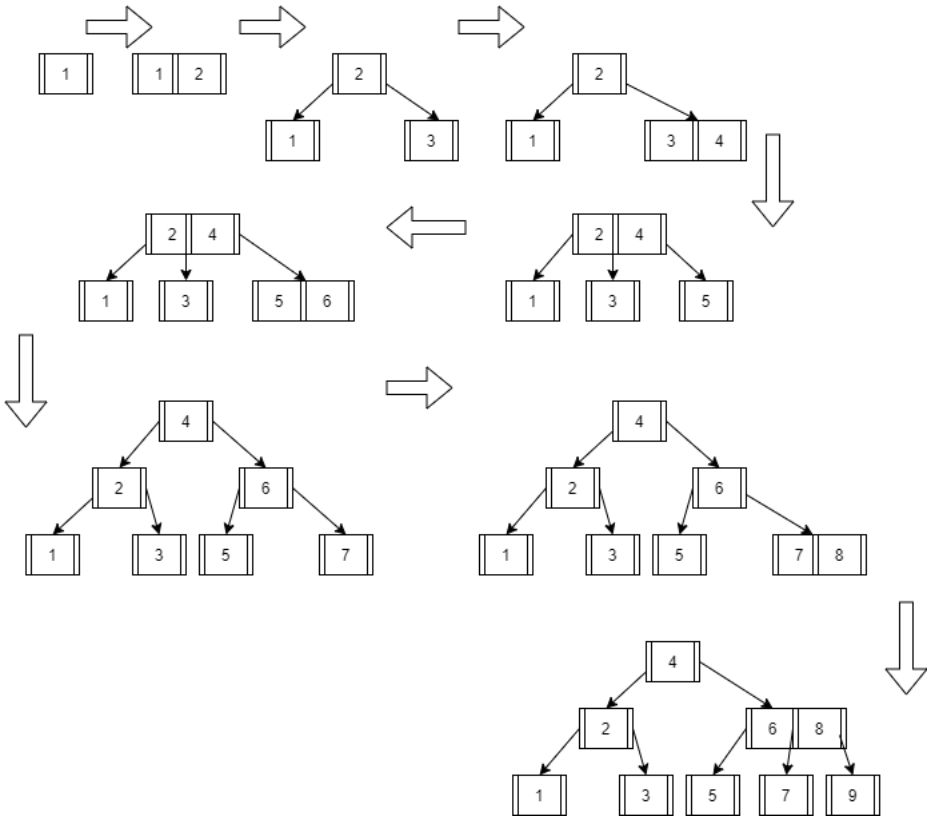
Thus, the number of keys that can be accommodated is $13 + 182 + 2,548 + 35,672 = 38,415$.

7. Consider a B-tree of order 3. If we have 9 elements to be inserted into the tree, then what will be the maximum number of splits that take place in the nodes?

[NAT: 2 points]

Answer: 5

Solution: Assume that the elements are $1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9$. The elements are in ascending order such that the B-tree becomes skewed, and the maximum number of splits take place. The steps for insertion are as shown in the figure, and it may be observed that the number of splits is 5.



8. Consider the instance of a 2-3-4 tree given in Figure 1 and answer the question that follows.

[MCQ: 2 points]

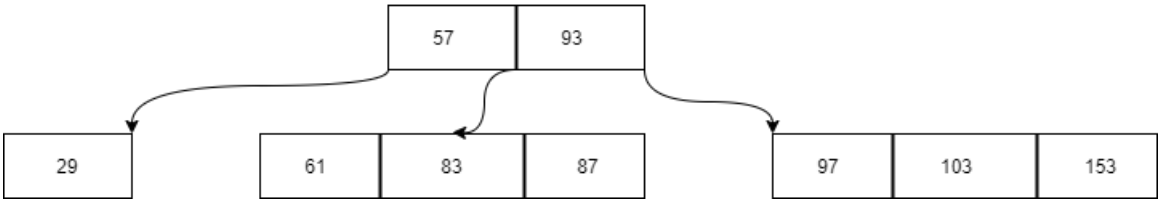


Figure 1: Instance of 2-3-4 tree

If we perform the following operations (in the given order) in the instance given in Figure 1, then choose the correct statement(s) about the resultant 2-3-4 tree.

- Insert value 38
- Insert value 89
- Insert value 100

◯ Insertion of value 38 will lead to a split.

√ Insertion of value 100 will lead to a split.

√ Insertion of value 89 will lead to a split.

◯ Insertion of value 100 will not lead to a split.

Solution: When value 38 is inserted, we can add it to the leaf node beside 29 and no split is required. When value 89 is inserted, the block containing values 61, 83, 87 will need to be split. When value 100 is inserted, the block containing values 97, 103, 153 will need to be split. Figure 2 shows the resultant 2-3-4 tree.

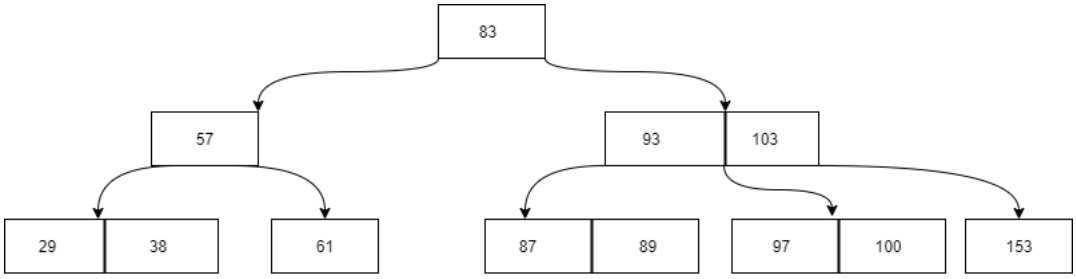


Figure 2: Resultant tree

9. Consider the table **Food**(*Fid, Fname, Foodtype, Rating*). There can only be two types of food - Dessert or Soup. The rating value is an integer in the range 1 to 5.

[MCQ: 2points]

Let the bitmap indexes for columns *Foodtype* and *Rating* be as given below:
Foodtype: 1010 for Dessert, 0101 for Soup
Rating: 0101 for 1, 0010 for 2, 0110 for 3, 0000 for 4, 1000 for 5
In order to find the *Fname* of an item that is a dessert with a rating of 2, which of the following operations will be performed?

- ◯ Logical AND operation on 0101 and 0010
- √ Logical AND operation on 1010 and 0010
- ◯ Logical OR operation on 1010 and 0010
- ◯ Logical XOR operation on 1010 and 0010

Solution: Dessert has a bitmap index of 1010 and Rating 2 has a bitmap index of 0010. Hence, in order to find *Fname* of an item that is a dessert and has a rating of 2, we must perform the Logical AND operation on 1010 and 0010.

10. Consider a B^+ tree in which the maximum number of keys allowed in a node is 7. Which among the following options are correct about the given tree?

[MSQ: 2 points]

- ◯ The minimum number of keys in any non-root node is 2.
- √ The B^+ tree has a maximum of 8 child pointers.
- ◯ The B^+ tree has a maximum of 7 child pointers.
- √ The minimum number of keys in any non-root node is 3.

Solution: Let the order of the given B^+ tree be p. Then, the maximum number of keys will be (p–1). Hence the number of keys in the B^+ tree is 8. The minimum number of keys in any non-root node will be $\lceil \frac{p-1}{2} \rceil = 3$.

11. From among the given options, choose the incorrect statement(s) about static and dynamic hashing.

[MSQ: 2 points]

- ◯ Static hashing uses a fixed hash function to partition the set of all possible search-key values into subsets, and then maps each subset to a bucket.
- √ Static hashing is a method of hashing in which a variable number of buckets are allocated to a file to store the records.
- ◯ Dynamic hashing may use a second stage of mapping to determine the bucket associated with some search-key value.
- ◯ In dynamic hashing, memory is well utilized as it grows and shrinks with the data. Hence, there will not be any unused memory.

Solution: Static hashing is a method of hashing in which a fixed number of buckets are allocated to a file to store the records.
The other statements follow from definitions of static and dynamic hashing.

Consider the given table **USER** and answer the questions 12 and 13.
USER(*User_ID, User_name, User_city, User_country, User_skill*)

[MCQ: 2 points]

12. Choose the correct SQL statement from the options to create an index with the name 'idx_userid', for the attribute *User_ID*.

- ◯ `CREATE INDEX AS idx_userid ON USER (User_ID);`
- ◯ `CREATE INDEX idx_userid AS USER (User_ID);`
- ◯ `CREATE INDEX AS idx_userid ON User_ID;`
- √ `CREATE INDEX idx_userid ON USER (User_ID);`

Solution: The syntax to create index on a table is:

```
CREATE INDEX index_name ON table_name (indexed_attribute_name);
```

13. Choose the correct SQL statement from the options to remove the index with the name 'idx_userid' on *User_ID*.

- ◯ `REMOVE INDEX ON idx_userid;`
- ◯ `DELETE INDEX ON User_ID;`
- √ `DROP INDEX idx_userid;`
- ◯ `DROP INDEX OF USER ON idx_userid;`

Solution: The syntax to remove an existing index on a table is:

```
DROP INDEX index_name;
```

14. Which of the following statements is/are correct about the B^+ tree data structure that is used for creating an index of a relational database table?

[MSQ: 2 points]

- √ B^+ Trees are considered *balanced* because the lengths of the paths from the root to all leaf nodes are equal.
- ◯ Non-leaf nodes have pointers to data records.
- ◯ B^+ Trees are considered *balanced* because the number of records in any two leaf nodes differ by at most 1.
- √ Key values in each node are kept in sorted order.

Solution: The statements follow from the definition and structure of a B^+ tree.