BSCCS2001: Graded Solutions
Week 9

1. Consider a B^+-tree index to be built on the attribute *empID* of a table **Employee**, with the following properties:

[MCQ :2 points]

- The length of the attribute *empID* is 8 bytes.
- The size of each child pointer is 12 bytes.
- The size of each disk block size is 512 bytes.

With the given information, what is the best choice for the order of the non-leaf nodes of the B^+-tree?

○ 12

✓ 26

○ 32

○ 43

Solution: Since the index has to be built on the attribute *empID* having size 8 bytes, the size of each search-key is also 8 bytes.
If p is the required order of the non-leaf nodes of a B^+-tree, the maximum number of keys in a node is $p-1$ and the maximum number of child pointers in a node is p.
The following equation must hold:
$\textit{size-of-key-field} \times (p-1) + \textit{size-of-child-pointer} \times p \leq 512$
$\Rightarrow p = \left\lfloor \frac{520}{20} \right\rfloor \Rightarrow p = 26$

2. Consider a bitmap index, on the attribute *grade* of a table **Student**, with the following properties:

[MCQ :2 points]

- The distinct values of the attribute *grade* are: A, B, C, D and F.
- The size of the bitmap index file is 200 bytes.

How many rows are there in the table **Student**?

○ 25

○ 120

[✓ 320]

○ 512

Solution: A bitmap index on an attribute, having k distinct values and n rows, has size $s = n \times k$ bits.
Thus, $n = (s \times 8)/k = (200 \times 8)/5 = 320$. The number of rows is 320.

3. Consider a B-tree of order 17. What are the maximum and minimum number of keys that can be placed in a leaf node?

[MCQ: 1 point]

○ max = 17, min = 8

[✓ max = 16, min = 8]

○ max = 16, min = 1

○ max = 17, min = 9

Solution: Given the order of B-tree $p = 17$, the maximum number of keys in leaf node is $p - 1 = 17 - 1 = 16$, and the minimum number of keys in leaf node is $\left\lceil \frac{(p-1)}{2} \right\rceil = \lceil \frac{16}{2} \rceil = 8$.

4. Consider a B^+-tree index built on the key attribute of a data file having 62,50,000 records. Let the order of the B^+-tree be 50.

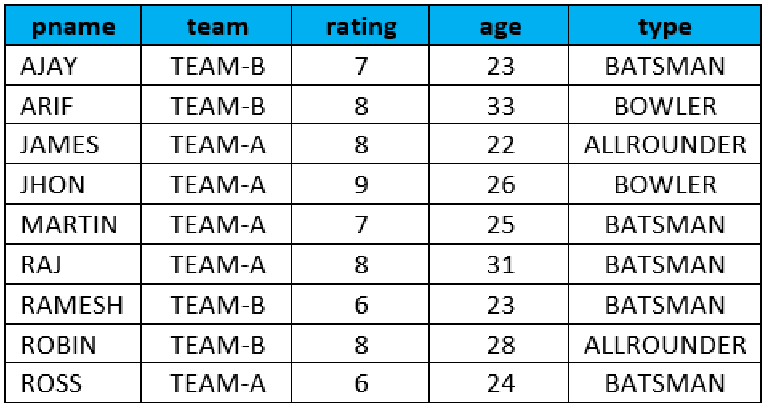
[MCQ: 2 points]

Find out the maximum number of nodes to be accessed to search a key from the given B^+-tree.

- ○ 7
- ○ 23
- [✓ 4]
- ○ 5

Solution: The maximum number of nodes to be accessed depends on the height of the given B^+-tree, which is $log_{50}(62, 50, 000) = 4$.

Consider the table **Player** given in Figure 1 and answer the questions 5 and 6.



pname	team	rating	age	type
AJAY	TEAM-B	7	23	BATSMAN
ARIF	TEAM-B	8	33	BOWLER
JAMES	TEAM-A	8	22	ALLROUNDER
JHON	TEAM-A	9	26	BOWLER
MARTIN	TEAM-A	7	25	BATSMAN
RAJ	TEAM-A	8	31	BATSMAN
RAMESH	TEAM-B	6	23	BATSMAN
ROBIN	TEAM-B	8	28	ALLROUNDER
ROSS	TEAM-A	6	24	BATSMAN

Figure 1: Table **Player**

5. Identify the correct SQL statement to build a bitmap index **team_name** on attribute *team* for the table **Player**.

[MCQ: 1 point]

- ○ `CREATE BITMAP INDEX team_name ON team`
- ○ `CREATE BITMAP INDEX team ON team_name`
- [✓ `CREATE BITMAP INDEX team_name ON Player(team)`]
- ○ `CREATE BITMAP INDEX Player(team) ON team_name`

Solution: The syntax to create a BITMAP index for the given attribute is:

```
CREATE BITMAP INDEX team_name ON Player(team)
```

6. If an ordered index is to be built on the attribute *pname*, then choose the suitable index(es) from the following.

[MSQ: 1 points]

- √ Primary index and sparse index
- ◯ Secondary index and sparse index
- √ Primary index and dense index
- ◯ Secondary index and dense index

Solution: Since the index is an ordered index, and the entries for the attribute *pname* are also in alphabetical order (which is also the only possible primary key in the given table), the index constructed on *pname* is a primary index. A primary index can be dense or sparse.

7. Consider the hash functions given below.

[MSQ: 3 points]

- $h_1(n) = n\ (mod\ 7)$,
- $h_2(n) = n^2 + 1\ (mod\ 7)$,
- $h_3(n) = 3n + 3\ (mod\ 7)$,
- $h_4(n) = round(\frac{n}{2}) + 2\ (mod\ 7)$.

Identify the hash function(s), that can generate unique hash values for the following search values: 43, 65, 89, 10, 12, 48, 31.

○ $h_1(n)$

○ $h_2(n)$

○ $h_3(n)$

✓ $h_4(n)$

Solution: Consider $h_1(n) = n\ (mod\ 7)$, the hash values are: $h_1(43) = 1, h_1(65) = 2, h_1(89) = 5, h_1(10) = 3, h_1(12) = 5, h_1(48) = 6, h_1(31) = 3$. Thus, it cannot generate unique values.
Consider $h_2(n) = n^2 + 1\ (mod\ 7)$, the hash values are: $h_2(43) = 2, h_2(65) = 5, h_2(89) = 5, h_2(10) = 3, h_2(12) = 5, h_2(48) = 2, h_2(31) = 3$. Thus, it cannot generate unique keys.
Consider $h_3(n) = 3n + 3\ (mod\ 7)$, the hash values are: $h_3(43) = 6, h_3(65) = 2, h_3(89) = 4, h_3(10) = 5, h_3(12) = 2, h_3(48) = 0, h_3(31) = 5$. Thus, it cannot generate unique values.
Consider $h_4(n) = round(\frac{n}{2}) + 2\ (mod\ 7)$, the hash values are: $h_3(43) = 2, h_3(65) = 6, h_3(89) = 4, h_3(10) = 0, h_3(12) = 1, h_3(48) = 5, h_3(31) = 3$. Thus, it generates unique values.

8. Consider an extendable hashing scheme with a hash function that generates 16-bit hash values. The hash values for the search keys to be inserted are generated as shown in Table 8.

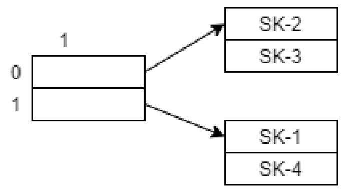
[NAT: 3 points]

Search key	Hash Value
SK-1	1001 0000 1001 0101
SK-2	0011 0110 1001 1111
SK-3	0001 1110 1101 1000
SK-4	1100 0110 1000 1010
SK-5	1011 1010 1101 1001
SK-6	0101 1111 1111 0000
SK-7	1001 0100 1101 0011
SK-8	1011 1100 1111 1100

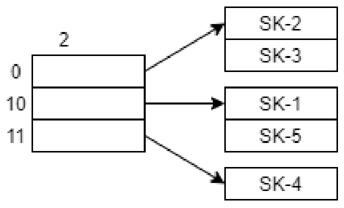
Let the maximum number of records that can be accommodated in a disk block be 2. Determine the length of hash prefix for the given scheme.

Answer: 3

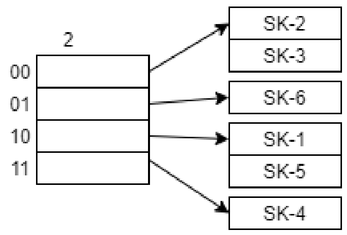
Solution: Initially the hash prefix is 0.
Hash structure after insertion of SK-1, SK-2, SK-3 and SK-4:



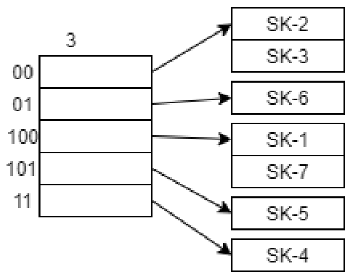
Hash structure after insertion of SK-5:



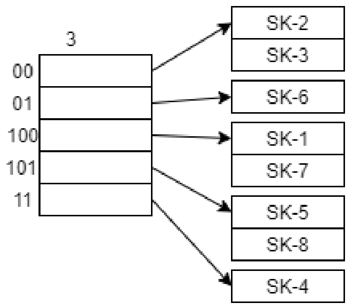
Hash structure after insertion of SK-6:



Hash structure after insertion of SK-7:



Hash structure after insertion of SK-8:



9. Consider the given instance of a 2-3-4 tree and answer the question that follows.

[MCQ: 2 points]

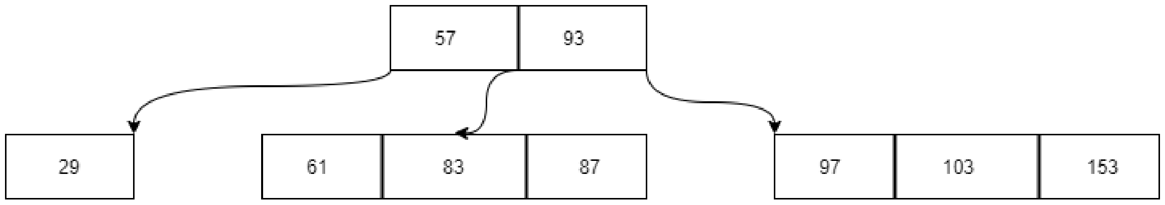


Figure 2: Instance of 2-3-4 tree

If we perform the following operations (in the given order) in the instance given in Figure 2, then how many leaf nodes are there in the resultant 2-3-4 tree?

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

◯ 6

✓ 8

◯ 7

◯ 9

Solution: Figure 3 shows the resultant 2-3-4 tree.
It has 8 leaf nodes: 29, 38, 61, 87, 89, 97, 100, 153.

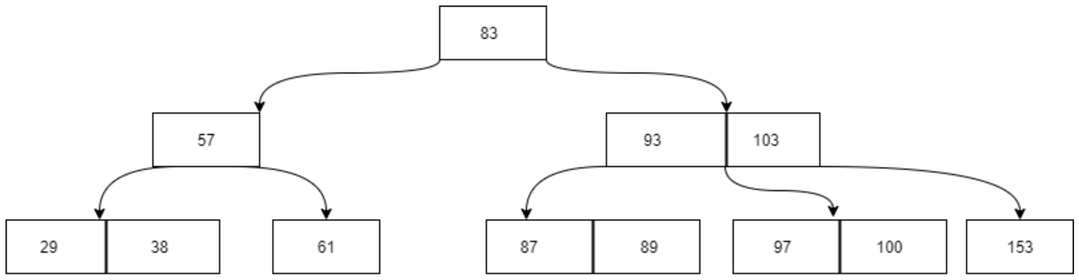


Figure 3: Resultant tree

10. Consider the B^+ tree given in Figure 4.

[NAT: 3 points]

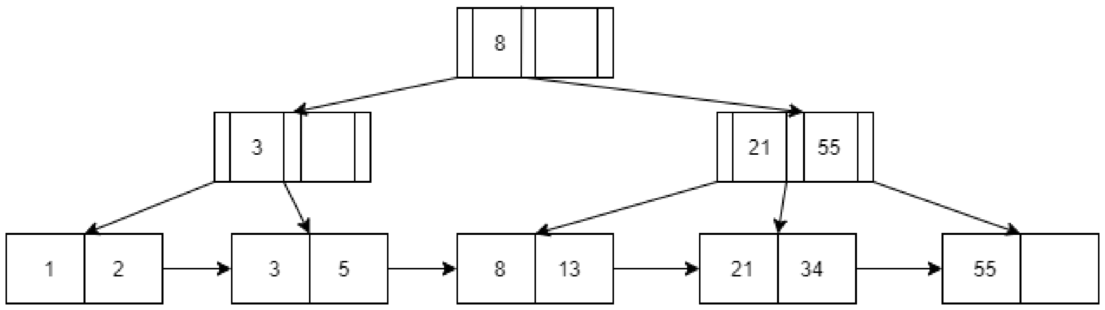


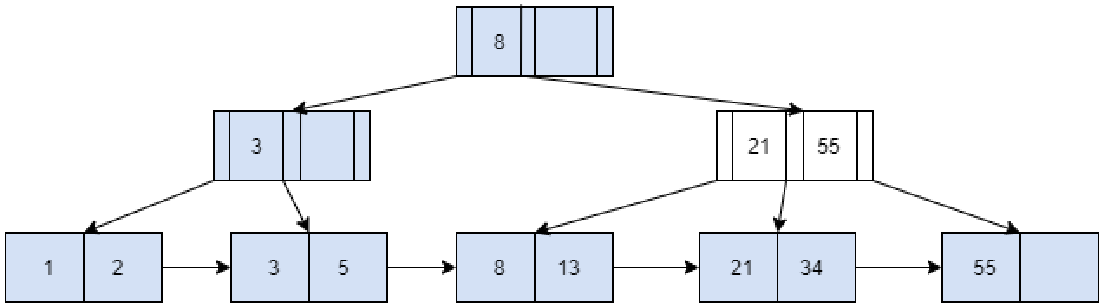
Figure 4: B^+-tree

Let n_1 be the minimum number of nodes (including the root node) that should be accessed in order to fetch all records with a search key greater than or equal to 1 and less than or equal to 55.
Let n_2 be the minimum number of nodes (including the root node) that should be accessed in order to fetch all records with a search key greater than or equal to 5 and less than 34.
What will be the value of $n_1 + n_2$?

- ○ 11
- ✓ 12
- ○ 9
- ○ 15

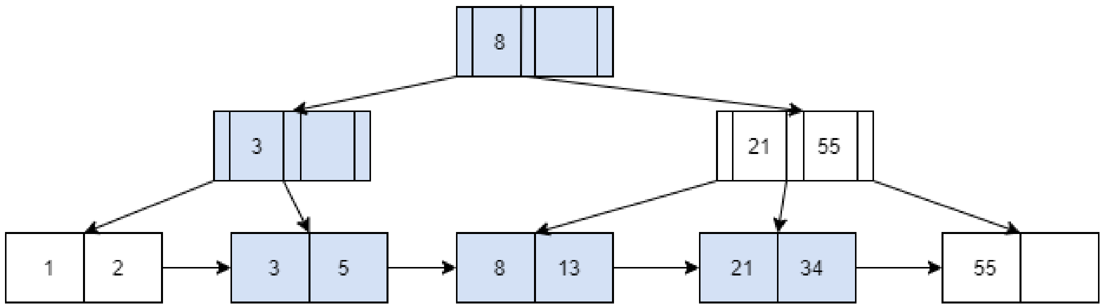
Solution:

- Computing n_1: We search for a node with value 1 in the leaf nodes. For this, we start at the root, which has a node with value 8 and then move to the left child, whose node has value 3. Then, we further traverse to the left child which has the node with value 1. From then, we traverse linearly, one leaf node after the other until we finally reach 55. The blue coloured nodes in the figure show the nodes to be traversed.



Hence the minimum number of nodes (including the root node) that should be accessed is 7.

- Computing n_2: We search for a node with value 5 in the leaf nodes. For this, we start at the root, which has a node with value 8 and then move to the left child, whose node has value 3. Then, we further traverse to the right child which has the node with value 5. From then, we traverse linearly one leaf node after the other until we finally reach 34. The blue coloured nodes in the figure show the nodes to be traversed.



Hence the minimum number of nodes (including the root node) that should be accessed is 5.

That is, $n_1 = 7$ and $n_2 = 5 \Rightarrow n_1 + n_2 = 7 + 5 = 12$.