CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | |

Third Internal Test

Sub:	**File Structures**						Code:	15IS62
Date:	21/ 05 / 2018	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	**ISE**

Answer Any **FIVE FULL** Questions

		Marks	OBE CO	RBT
1 (a)	What are the properties of B-Tree? Explain deletion, merging and redistribution of elements on B-Tree with suitable example.	[10]	CO3	L2

Ans: For a Btree of order m, Btree has the following properties.

1. Every page has a maximum of m children.
2. Every page, except for the root and the leaves, has a minimum of m/2 children
3. The root has a minimum of 2 children (unless it is a leaf).
4. All of the leaves are on the same level
5. The leaf level forms a complete, ordered index of the associated data file.

The rules for deleting a key *k* from a node *n* in a B-tree are as follows:

1. If *n* has more than the minimum number of keys and the *k* is not the largest in *n*, simply delete *k* from *n*.
2. If n has more than the minimum number of keys and the *k* is the largest in *n*, delete *k* and modify the higher level indexes to reflect the new largest key in *n*.
3. If *n* has exactly the minimum number of keys and one of the siblings of *n* has few enough keys, merge *n* with its sibling and delete a key from the parent node.
4. If *n* has exactly the minimum number of keys and one of the siblings of *n* has extra keys, redistribute by moving some keys from a sibling to *n*, and modify the higher level indexes to reflect the new largest keys in the affected nodes.

Example for deletion, merging and redistribution.

		Marks	CO	RBT
2 (a)	With a neat diagram, explain simple prefix B+ Tree and its maintenance.	[10]	CO3	L2

Ans:

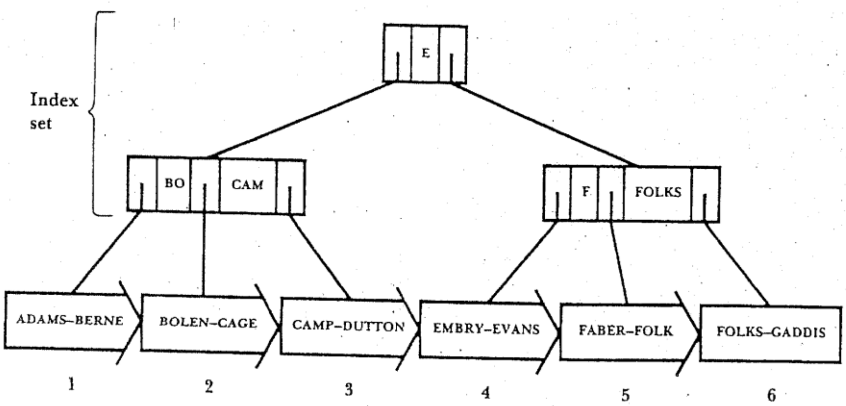


Figure 10.7 A B-tree index set for the sequence set, forming a simple prefix B+ tree.

Btree indexset taken together with sequence set forms a file structure called a simple prefix B+ tree. The modifier simple prefix indicates that the index set contains shortest separators or prefixes of the keys rather than copies of the actual key.

Maintenance:

1. **Changes Localized to Single Blocks in the Sequence Set**

Additions, deletions, and updates in the sequence set which affect only a single block do not affect the index set.

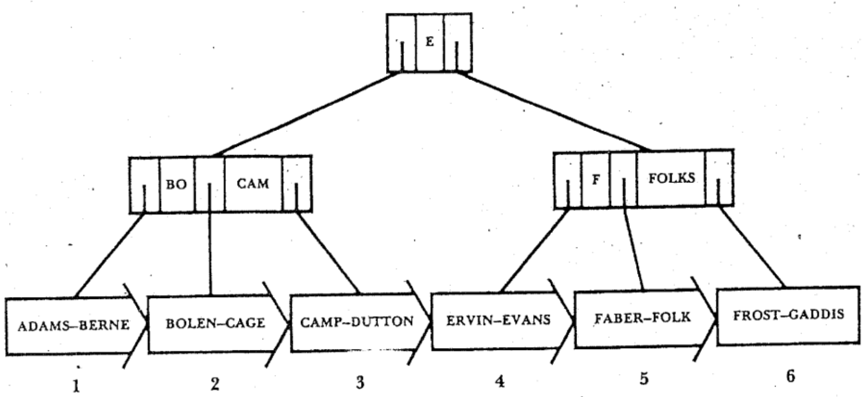


Figure 10.8 The deletion of the EMBRY and FOLKS records from the sequence set leaves the index set unchanged.

2. **Changes involving multiple blocks in the sequence set**

When addition to the sequence set results in split in the sequence set, deletion in sequence set which results in merger, or changes in sequence set resulting in redistribution requires involvement of more than one block set and corresponding changes in the index set as well.

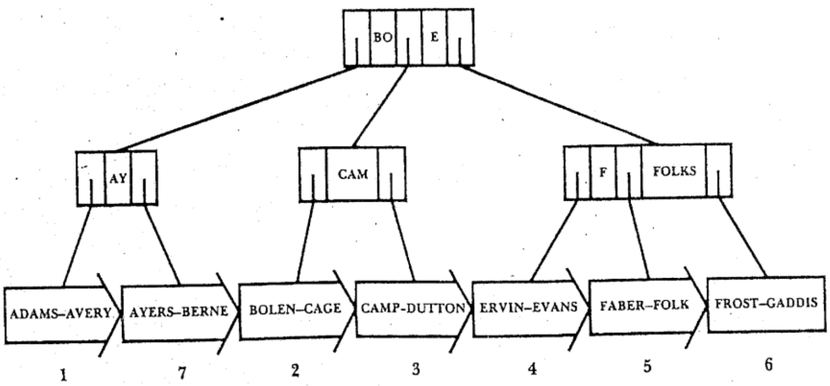


Figure 10.9 An insertion into block 1 causes a split and the consequent addition of block 7. The addition of a block in the sequence set requires a new separator in the index set. Insertion of the AY separator into the node containing BO and CAM causes a node split in the index set B-tree and consequent promotion of BO to the root.

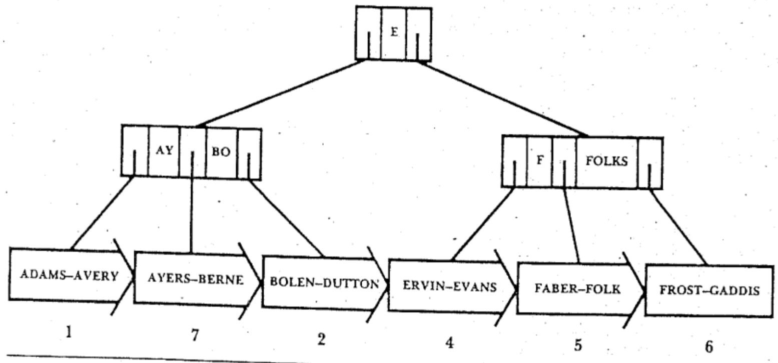


Figure 10.10 A deletion from block 2 causes underflow and the consequent merging of blocks 2 and 3. After the merging, block 3 is no longer needed and can be placed on an avail list. Consequently, the separator CAM is no longer needed. Removing CAM from its node in the index set forces a merging of index set nodes, bringing BO back down from the root.

3 (a)	Show the step by step construction of B-Tree of order four that results from loading the following set of keys in order: CSDAMPIBWNGURKE	[10]	CO3	L3
4 (a)	Suppose there are 1000 addresses allocated to hold 800 records in a randomly hashed file and that each address can hold one record. Compute the following values: i) The packing density ii) The expected number of addresses with no records assigned to them. iii) The expected number of addresses with exactly one record assigned. iv) The expected number of addresses with one or more synonyms v) The expected number of overflow records	[10]	CO4, CO5	L3

Ans:

i) Packing density:

800/1000=0.8

ii) $P(0)= (0.8)^0 \times e^{-(0.8)}/0!= .449*1000=449$

iii) $P(1)= (0.8)^1 \times e^{-(0.8)}/1!=.313*1000=313$

iv) $P(2)= (0.8)^2 \times e^{-(0.8)}/2!=.095*1000=95$

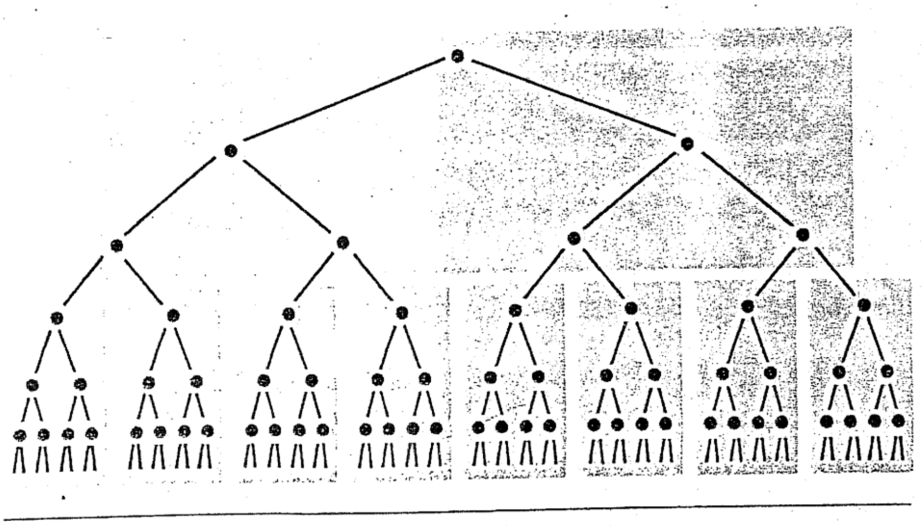
$P(3)= (0.8)^3 \times e^{-(0.8)}/3!=.038*1000=38$

$P(4)= (0.8)^4 \times e^{-(0.8)}/4!=0.007*1000=7$

$P(5)= (0.8)^5 \times e^{-(0.8)}/5!=0.001*1000=1$

address with one or more synonyms=95+38+7+1=141

v) (95*1)+(38*2)+(3*7)+(1*4)=196

5 (a)	With a neat diagram, explain paged binary trees. What are its advantages and disadvantages?	[10]	CO3	L2
Ans:	 Figure 9.12 Paged binary tree. AVL trees tackle the problem of keeping an index in sorted order cheaply. They do not address the problem regarding the fact that Binary Searching requires too many seeks. Paged Binary trees addresses this problem by locating multiple binary nodes on the same disk page. In a paged system, you do not incur the cost of a disk seek just to get a few bytes. Instead, once you have taken the time to seek to an area of the disk, you read in an entire page from the file. Adv: When searching a Binary Tree, the number of seeks necessary is log2(N+1). It is logk+1(N+1) in the paged version. Dis Adv: Balancing a paged binary tree can involve rotations across pages, involving physical movement of nodes.			
6 (a)	Explain B-Tree methods for search() and findleaf() with necessary C++ code.	[10]	CO3	L3

```
template <class keyType>
int BTree<keyType>::Search (const keyType key, const int recAddr)
{
   BTreeNode<keyType> * leafNode;
   leafNode = FindLeaf (key);
   return leafNode -> Search (key, recAddr);
}

template <class keyType>
BTreeNode<keyType> * BTree<keyType>::FindLeaf (const keyType key)
// load a branch into memory down to the leaf with key
{
   int recAddr, level;
   for (level = 1; level < Height; level++)
   {
      recAddr = Nodes[level-1]->Search(key,-1,0);//inexact search
      Nodes[level]=Fetch(recAddr);
   }
   return Nodes[level-1];
}
```

Figure 9.18 Method BTree::Search and BTree::FindLeaf.

The search operation on a b-tree is analogous to a search on a binary tree. Instead of choosing between a left and a right child as in a binary tree, a b-tree search must make an n-way choice. The correct child is chosen by performing a linear search of the values in the node using find leaf function as above. After finding the value greater than or equal to the desired value, the child pointer to the immediate left of that value is followed. If all values are less than the desired value, the rightmost child pointer is followed. Of course, the search can be terminated as soon as the desired node is found. Since the running time of the search operation depends upon the height of the tree, B-Tree-Search is O(log n).

7 (a)	What is collision? Explain the double hashing and chained progressive overflow collision resolution techniques.	[10]	CO4	L2

Collisions

synonyms

Keys which hash to the same value.

collision

An attempt to store a record at an address which does not have sufficient room ie already occupied by another record which is a synonym.

Double Hashing: A method of open addressing for a hash table in which a collision is resolved by searching the table for an empty place at intervals given by a different hash function, thus minimizing clustering.

Linear probing collision resolution leads to clusters in the table, because if two keys collide, the next position probed will be the same for both of them.

The idea of double hashing: Make the offset to the next position probed depend on the key value, so it can be different for different keys

Need to introduce a second hash function H 2 (K), which is used as the offset in the probe sequence.

For a hash table of size M, H 2 (K) should have values in the range 1 through M-1; if M is prime, one common choice is H2(K) = 1 + ((K/M) mod (M-1))

Example:

k (key)	ADAMS	JONES	MORRIS	SMITH
$h_1(k)$ (home address)	5	6	6	5
$h_2(k) = c$	2	3	4	3

Hashed file using double hashing:

0	
1	
2	
3	
4	
5	ADAMS
6	JONES
7	
8	SMITH
9	
10	MORRIS

Chained Progressive overflow:

It is similar to progressive overflow except that synonyms are linked together with pointers. The objective is to reduce the search length for records within clusters.

Progressive Overflow

	data
⋮	⋮
20	ADAMS
21	BATES
22	COLES
23	DEAN
24	EVANS
25	FLINT
⋮	⋮

Chained Progressive Overflow

	data	next
⋮	⋮	⋮
20	ADAMS	22
21	BATES	23
22	COLES	25
23	DEAN	-1
24	EVANS	-1
25	FLINT	-1
⋮	⋮	⋮

8 (a) What is indexed sequential access? Explain the block splitting and merging due to insertion and deletion in sequence set. CO2 L2

- In indexed and tree structure based access user had to choose between viewing a file from an indexed point of view or from a sequential point of view. In Indexed sequential access we are looking for a single organizational method that provides both of these views simultaneously.

Sequence set:

- A sequence set is a set of records in physical key order which is such that it stays ordered as records are added and deleted.
- Since sorting and resorting the entire sequence set as records are added and deleted is expensive, we look at other strategies. In particular, we look at a way to localize the changes.

- The idea is to use blocks that can be read into memory and rearranged there quickly. Like in B-Trees, blocks can be split, merged or their records re-distributed as necessary.

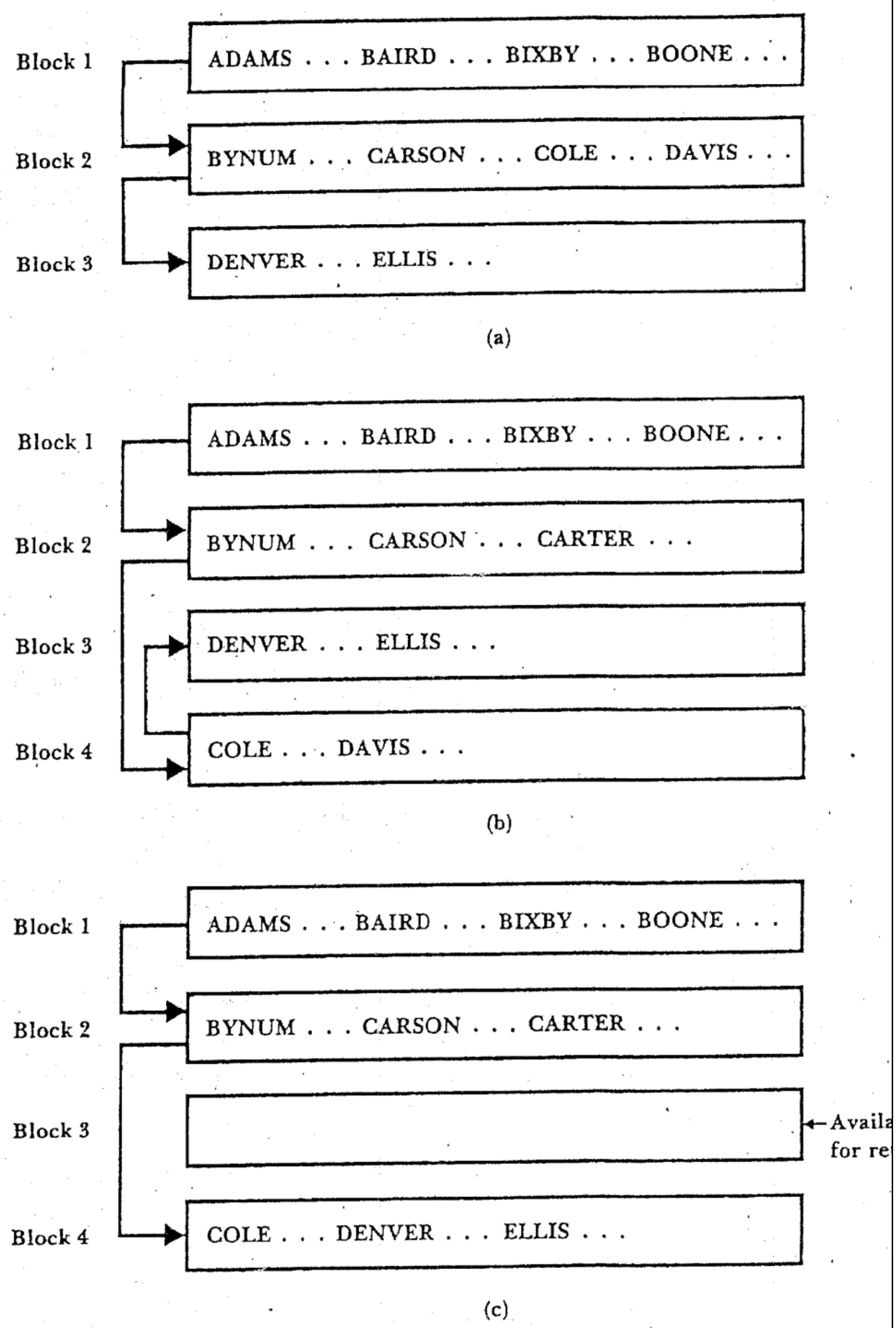


Figure 10.1 Block splitting and merging due to insertions and deletions in the sequence set. (a) Initial blocked sequence set. (b) Sequence set after insertion of CARTER record—block 2 splits, and the contents are divided between blocks 2 and 4. (c) Sequence set after deletion of DAVIS record—block 4 is less than half full, so it is concatenated with block 3.