→ ~~Understand requirements~~

Storage & File Structure.

→ Free list

→ Variable length Records.

→ Sequential file organisation

Important

Indexing & Hashing

→ Ordered indices

→ B^+-Tree Index files ← Most effective

→ B tree index files

→ Static Hashing

→ Dynamic Hashing

→ Comparison of Ordered Indexing & Hashing

→ Index Definition of SQL

→ Multiple key Access.

Search Key → attribute or set of attribute used to look up records in a file.

{Can be primary key or ~~can~~ candidate key or Super key}

→ Need Not to be.

A file can have more than one index i.e. multiple Search Keys.

Clustering index also known as primary index.

Clustering index

dense — All search key values

non dense — sparse — only few.

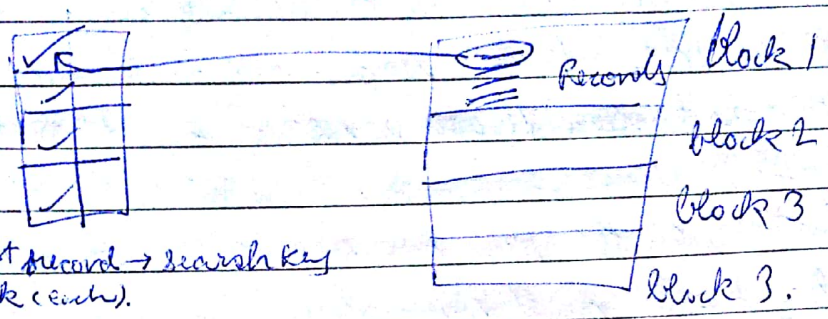
Kaun?

How to decide

Teacher's Signature

→ If index stores an entry for each block of the file, no change needs to be made to the index unless a new block is created.

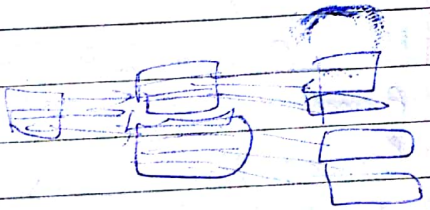
→ If a new block is created, the first search key value appearing in the new block is inserted into the index.



*After adding new entry sort then choose 1st record → search key of Block (each).

On the other hand, if the record has the least search key value in its block, the system updates the index entry (from shade)

Multilevel indexing:



Multiway Search tree.

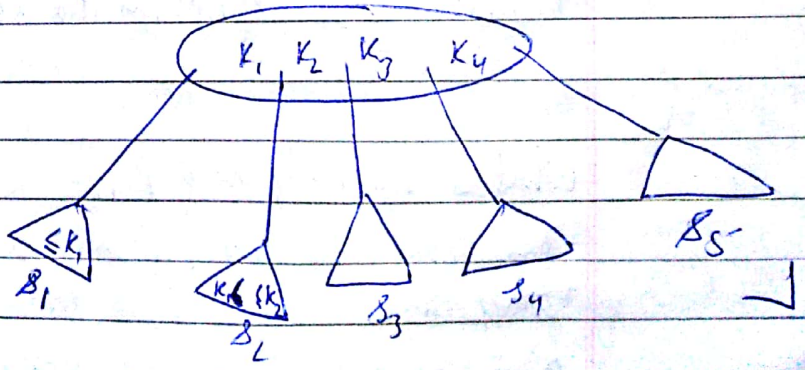
→ A multiway search tree of order n is a general tree in which each node has n fewer shades & contains one fewer (n-1) key then it has subtree keys / data value

→ If a node has 4 subtrees then No. of keys will be 3.

→ If $S_1, S_2, \ldots S_n$ are the n subtrees of a node containing $K_1, K_2, K_3, \ldots K_{n-1}$ keys in ascending order.

→ All trees in a subtree S_1 will be less than or equal to K_1 similarly all the keys in a subtree S_j are greater than K_{j-1} & less than or equals to K_j.

$K_1 < K_2 < K_3 < K_4$



BST is of order n.

Teacher's Signature

all the keys in a sub tree Sm are greater than

B+ tree.

It is a balanced multiway search tree.
Here balanced means every path from the root node of the tree to the leaf node is of same length. and that ~~same~~ length is height of B+ tree.

In this, all the keys ~~are~~ are maintained in the leaf node and some of the leaf nodes are replicated into non leaf node.
The leaf nodes are linked together to provide a sequential path to traverse the keys in the B+ tree.

How B+ tree look like?
A node contain n-1 keys & maximum n pointers.
Search key values are actually ~~kept~~ kept in a sorted order. this means.

if $i < j$ then. $K_i \leq K_j$

For a leaf node, P_i points to either a file record with search key values with K_i or a bucket of pointers each of which points to a file ~~with~~ which points to file records K_i

P_1	K_1	P_2	K_2	----	K_{n-1}	P_n

If a B+ tree is a dense index then all the search key value will be in the leaf node

in sparse → few of the

What is the purpose of P_n.
In these nodes P_n is used to chain together the leaf nodes in ~~search~~ search key order. This ordering allows for efficient sequence processing of the file.

Teacher's Signature

* In a leaf node, Any node in B+ tree of order can have max $n-1$ keys.

Minimum $\lceil \frac{n-1}{2} \rceil$ keys.

$\lceil x \rceil$ = least integer greaterthan x

$\lceil 1.5 \rceil = 2$

* A non leaf node or may hold upto n no. of pointers. internal or root node.

Internal → Maximum : n pointers ⇒ Maximum Keys ⇒ $n-1$.

Minimum no. of pointers $\lceil n/2 \rceil$

Minimum Keys = $\lceil n/2 \rceil - 1$.

* In a root Node.

Maximum pointers: n.

maximum keys : $n-1$

minimum pointers = 2.

min Key := 1.

→ Let us consider a node containing n no. of pointers.

for $i = 1 \ldots n-1$

Pointer P_i points to the subtree that contains subtree values less than K_i or greaterthan equal to K_{i-1}.

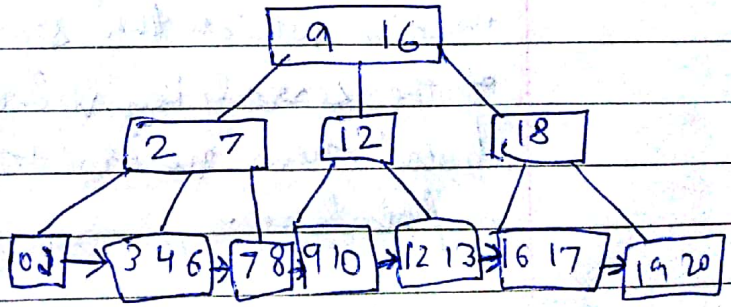
All the values are greaterthan equal to K_{n-1} in P_n.

eg, Order 4

Max Keys = 3

Leaf node = $\lceil 3/2 \rceil = 2$

Internal node = $\lceil 4/2 \rceil - 1 = 1$



Query

How to search in B+ tree.

Find all the file records with the search key value = V.

Teacher's Signature

Step 1. First we examine root node looking for the smallest root node greater than V. Suppose it is K_i

Step 2. We can follow the pointer P_i to another node. If we donot find no such value the you will follow the pointer P_{n}.

Step 3. On the node, we reached above again we look for smallest root node greater than V and once again we follow the pointer as above.
Iterate it until you reach leaf node.
At leaf node if we find value V then pointer P_i direct us file or bucket of pointers.
record

Valid in Dense index } Suppose Values not found, in the leaf node it means that no such record exist in the file.

Thus in processing a query, we traverse in the tree from root the reach leaf node.

No. of comparision = length of the path from the root node to the leaf node

Right of a complete

(a) Insertion Query.

(1) Using a searching technique we find the leaf node in which the search key value would appear. If the search key value in a leaf node, we add new record to the file & add to the bucket of pointers.

(2) If the search key value doesnot appear in B+ tree then the following actions are taken

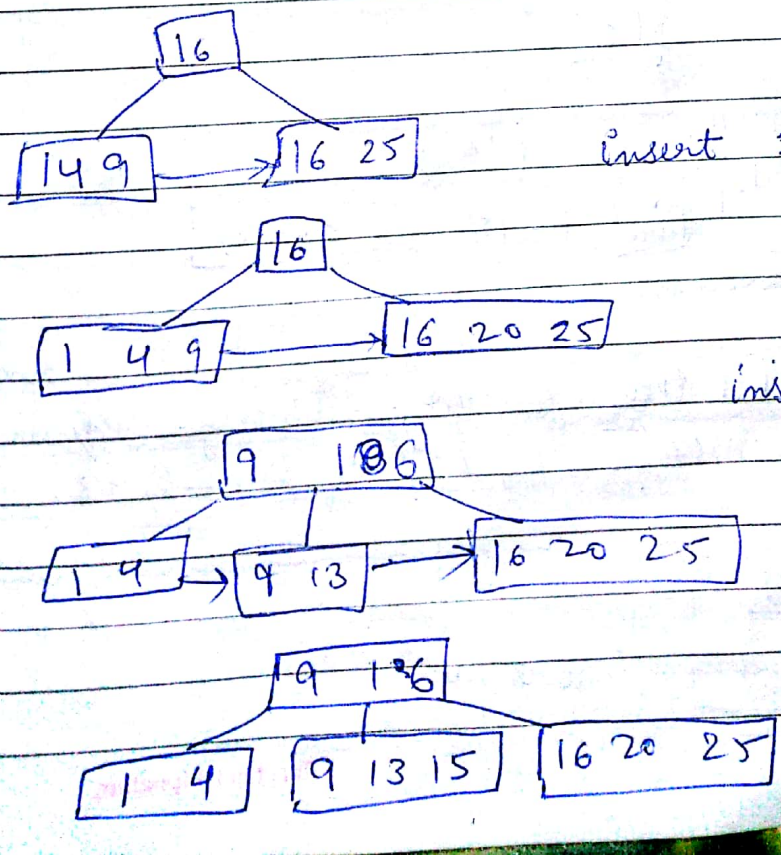
(a) If the leaf node is not full i.e. no. of keys are less than $(n-1)$ then we insert the value in the

Teacher's Signature

leaf node & position if such that the search key are still in sorted order. Then insert the new record in the file or create a bucket of pointers.

② If the leaf node is full (i.e. it is not $(n-1)$), we split the node. We take all n search key values including newer one in a sorted form & put the first $\lceil n/2 \rceil$ d value in the existing node & remaining values in a new node, we then insert the lowest search key value of the new node into the parent of the leaf node that was split. This lowest search key value will also be there in the new node. If the parent node is also full then the parent node is also splitted. In the worst case all the nodes must be splited. If the root is itself splited then a new root is created and an entire tree become deeper.
When a non leaf node is splited in a process, starting $\lceil n/2 \rceil$ values are placed in and $(\lceil n/2 \rceil + 1)$th key goes to the parent and then rest of the values go into the new node.

Eg →



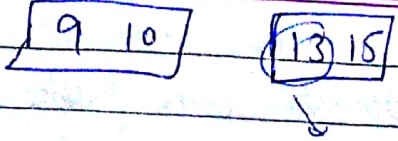
Teacher's Signature

Now inserting 10.

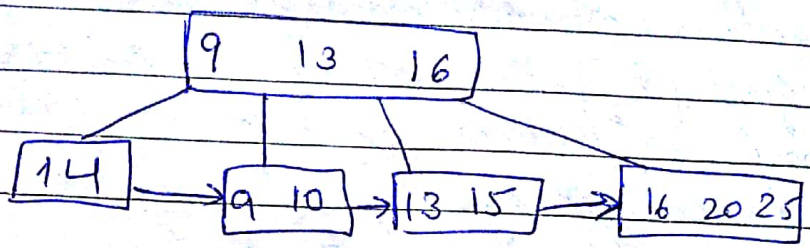
9 10 13 15

DATE: / /
PAGE NO.:

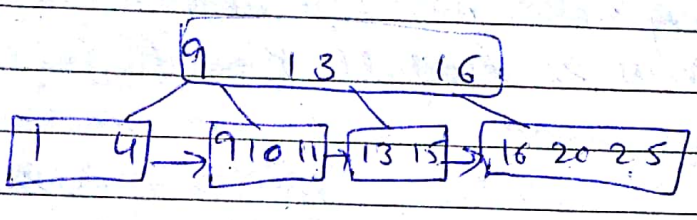
$\lceil m/2 \rceil$ in first rest in second



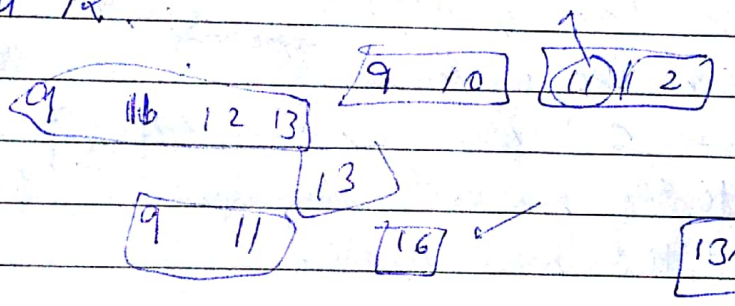
Second ka first element will go up.



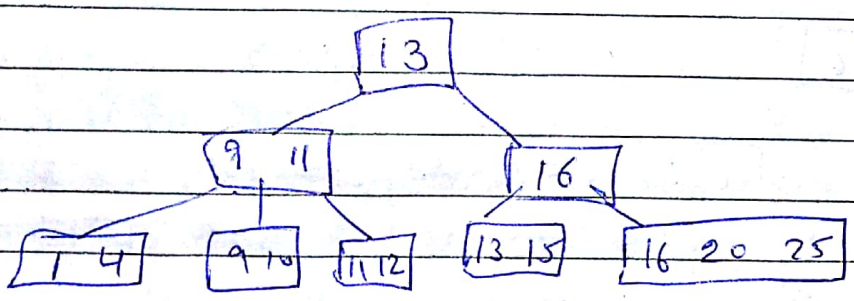
insert 11



insert 12.

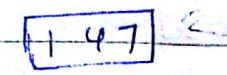


as non leaf node.



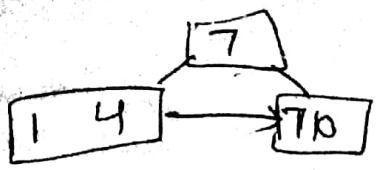
Q Construct a B+ tree for the following sequence order 4.

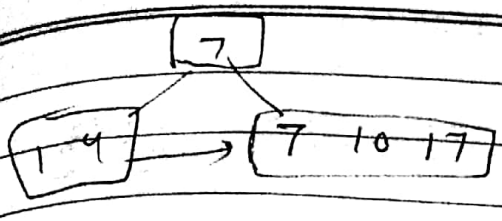
1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42.

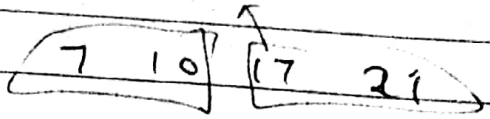


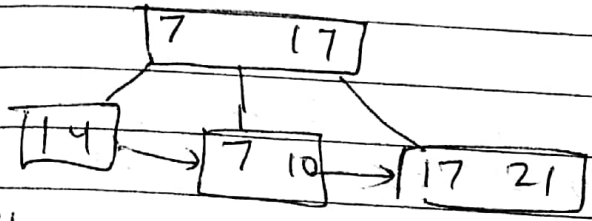
insert 10.

Teacher's Signature

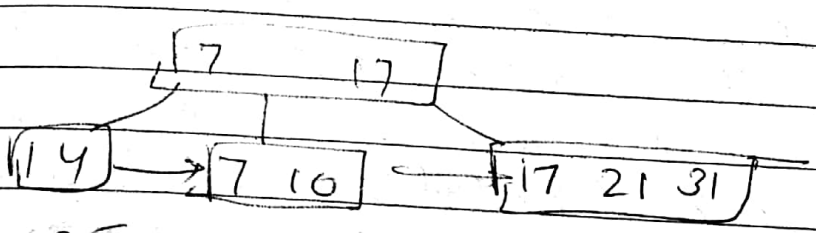




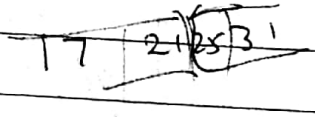


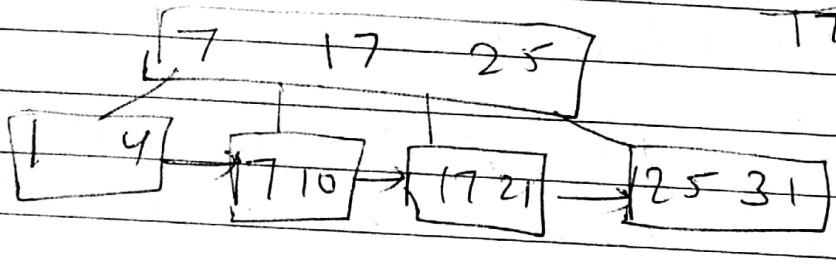


insert 31

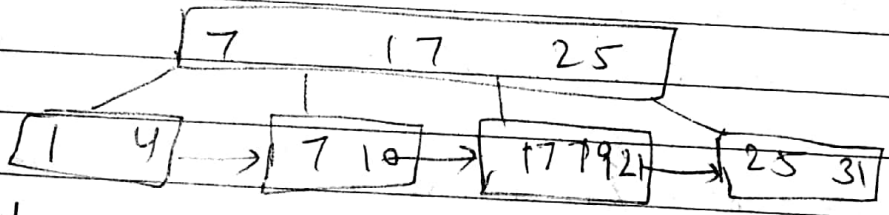


insert 25

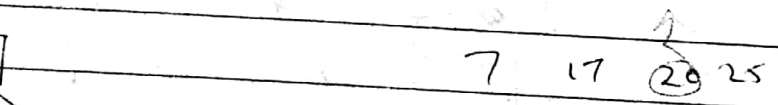


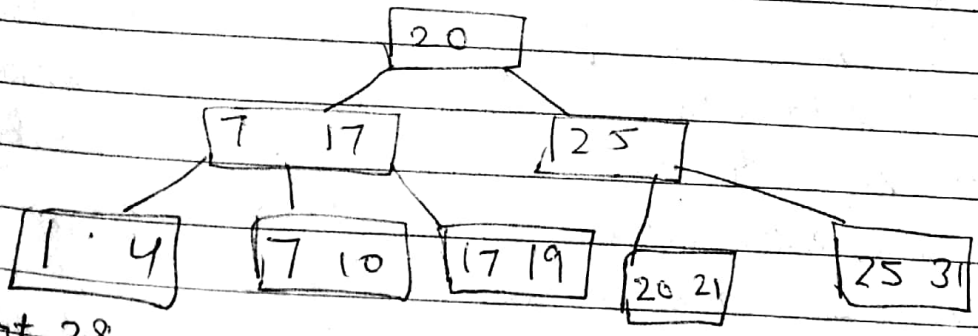


insert 19

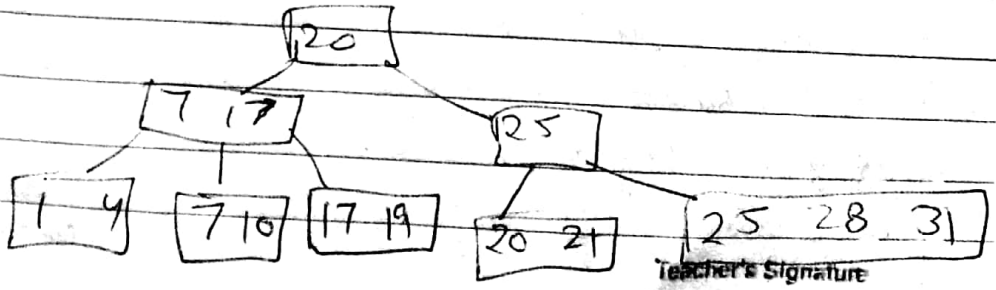


insert 20



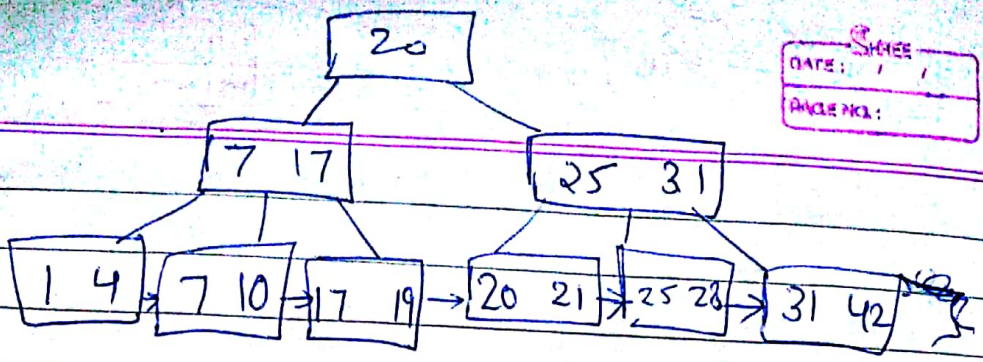


insert 28



Teacher's Signature

Insert 42.



Deletion

using a searching technique, we find the record we want to deleted and delete it.

① If a leaf node contains atleast $\lceil \frac{n-1}{2} \rceil$ keys after the deletion. We remove a search key value from the leaf node, if there is no bucket associated with that search key value or if bucket becomes empty after deletion of record.
else.

② When a key is deleted and a number of a keys in a leaf node drops below $\lceil \frac{n-1}{2} \rceil$ Situation called as Underflow.
Following actions are taken.

2a) Examin left or right sibling of underflow node in that if left sibling has greater than $\lceil \frac{n-1}{2} \rceil$ keys then transfer last key from the left sibling to the underflow node. After that update the search key value in the parent node as leftmost key of the right node.
Otherwise if right sibling has enough $\lceil \frac{n-1}{2} \rceil$ keys then transfer first key to the underflow node. and update search key in the parent as left key of right node.
Otherwise if left sibling has minimum no. of

Redistribution

Teacher's Signature

keys then merge left sibling ~~or~~ with underflow node. After that delete a ~~node~~ key from parent node.

otherwise if Right sibling has minimum no. of keys then merge right sibling with underflow node after that delete a key from parent node.

(iii) If a parent of a leaf node (non leaf node) also have the minimum has minimum no. of keys then the deletion is performed on non leaf node.

~~If a parent~~ As follows:

If left sibling is greater than $\lceil \frac{m}{2} \rceil - 1$ keys then transfer last key of parent from the left sibling to the key in parent node and the old key at the parent node becomes the first key of the underflow node.

otherwise if right sibling has enough keys (greater than $\lceil \frac{m}{2} \rceil - 1$) then transfer first key from right sibling to the key in parent node and the old key in the parent node becomes last key of underflow node.

otherwise if left sibling has minimum no. of keys then merge left sibling and underflow node along with the key with the parent node.

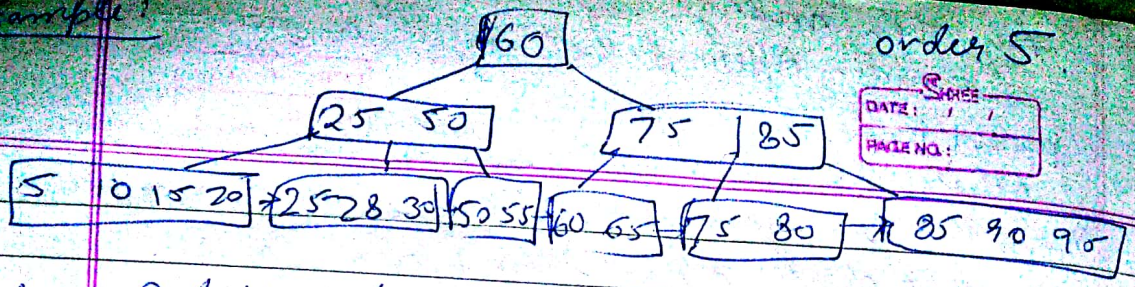
otherwise if right sibling has minimum no. of keys then merge right sibling and under flow node along with the key with the parent node.

~~If directly to~~

When a key is deleted from a non leaf node then we replace the value by the left most entry in the right sub tree.

Teacher's Signature

Example:



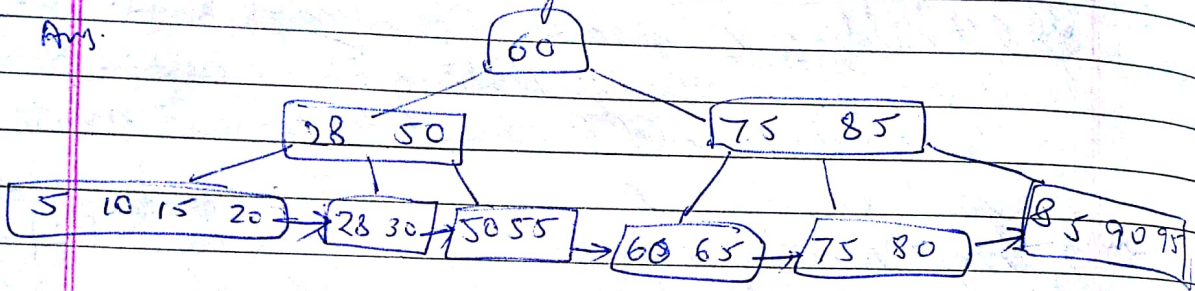
order 5

DATE: / /
PAGE NO.:

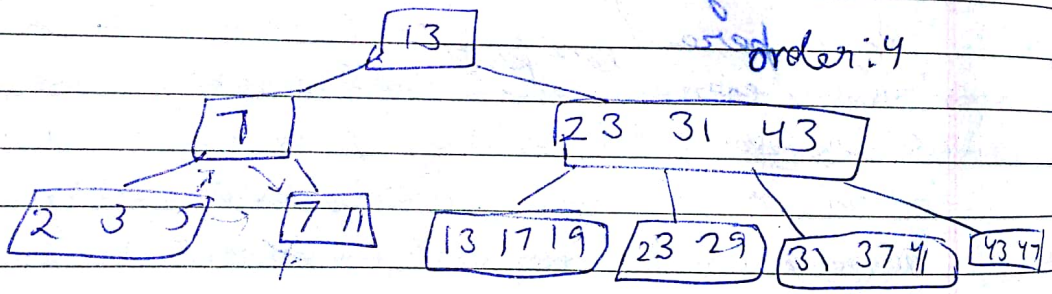
Q. Delete 25 { assuming 25 has the only one record }.

Order 5 ⇒ Max key = 4
leaf node min = 2
non leaf node = 2.

Ans.



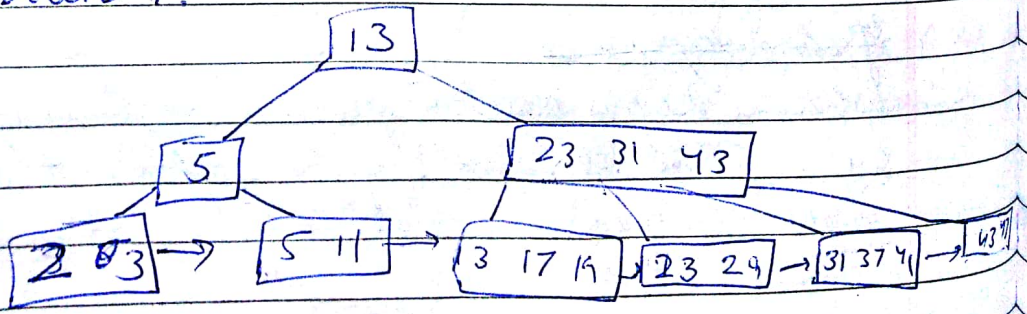
Ex ⇒



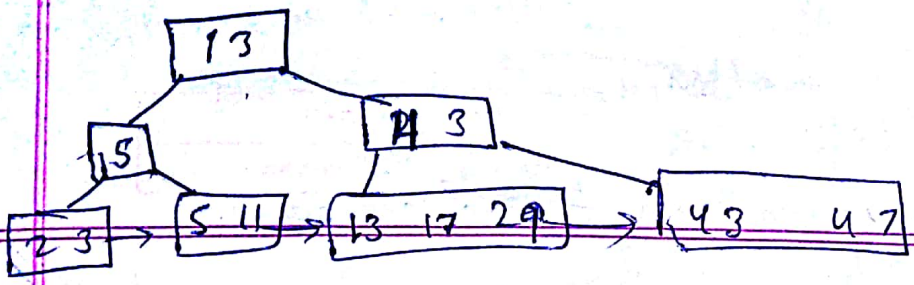
order: 4

if we delete 17 from this Btree then 13 17 19 will be 13 19
Now no other change in tree.

then if we delete 7.

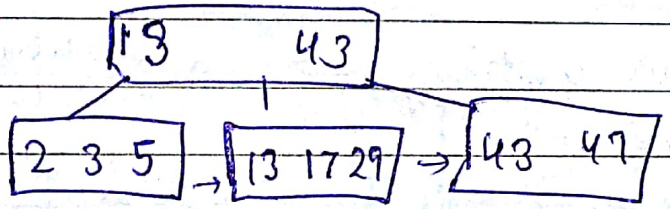


Teacher's Signature



deleting (11). 2, 3, 5 merging ⇒ Parent remove.

→



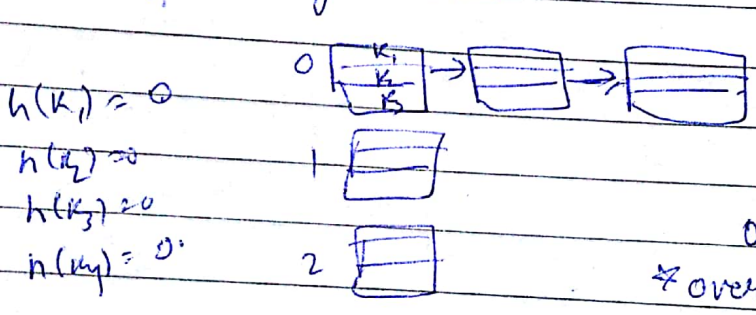
height of tree will get reduced.

Another type of searching ~~index~~ technique

- Hashing → Static
  → Dynamic

Ideal hash function → each bucket should assigned
→ Uniform → Same No. of search key values.
→ Random, so each bucket will have the same number of records assigned to it irrespective of the actual distribution of search key values in the file.

Probability of bucket overflow (in DSM (collision)) can be reduced using overflow buckets.



closed hashing

overflow chaining
* overflow buckets of a given bucket are chained together in a linked list.

Dynamic Hashing :-

Sample account file

A 211	Brighton	750
A 901	Downtown	500
A 110	Downtown	600
A 215	Mianus	700
A 102	Perryridge	400
A 201	Perryridge	900
A 218	Perryridge	700
A 222	Redwood	700
A 306	Round Hill	350

h (Branch name) :-

~~h(B) = 0~~ h(Round) = 0

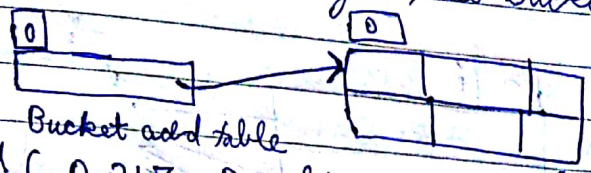
~~h(M) = 1~~ h(Perry) = 1

~~h(P) = 1~~

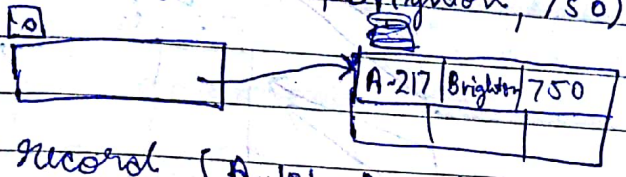
Date: / /

Page No.:

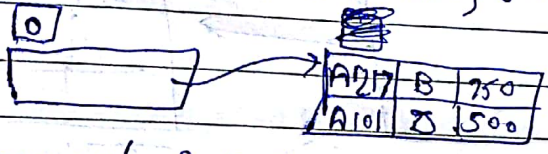
Initially there is no record in file, so bucket add table will be initial



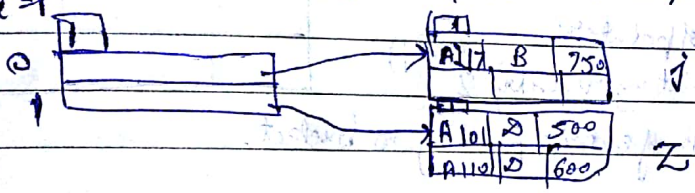
① Insert record (A-217, Brighton, 750)



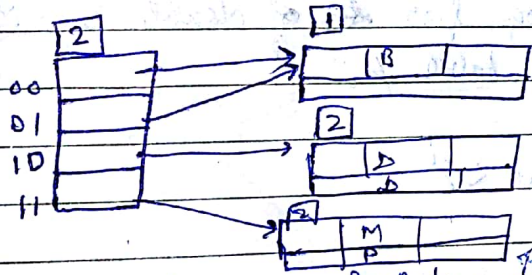
② Insert record (A-101, Downtown, 500)



③ insert record (A-110, Downtown, 600) ↓ splitting



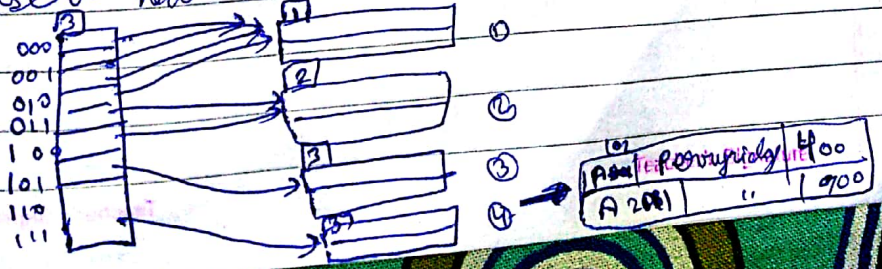
④ Insert record (A215, Miam, 700).



No. of entries which are pointing to bucket is given by $2^{(i-j)}$

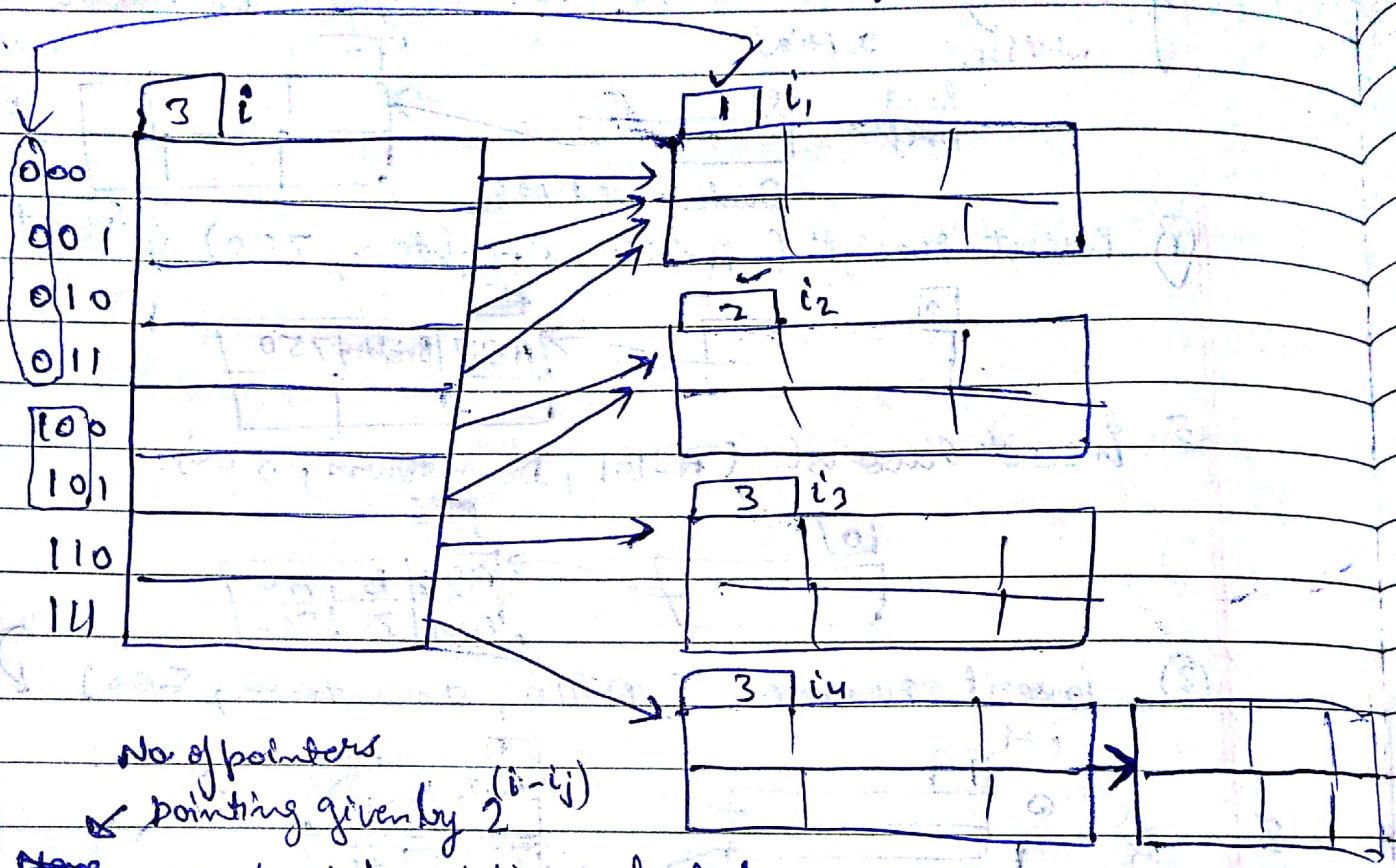
⑤ Insert new entry (A-102, Perryridge, 400)

⑥ → Insert new entry (A201, Perryridge, 900).



④ Inserting (A-218), Perryridge, 700).

Now, after spliting also problem is not solved which means we will add overflow bucket.



No. of pointers pointing given by $2^{(i-i_j)}$

~~None~~ ~~Above~~ No. of entries pointing to bucket.

agar $i > i_j$ ⇒ Wali ~~bucket~~ bucket full ho gayi aur phir add karege to split to hoga par ~~size~~ of double size Nahi karege. Bucket add. table ~~ki~~

∴ this No. of Bucket $< 2^i$.

Transaction Management :→

Transactions → collection of operations ~~that perform a single unit of operation.~~

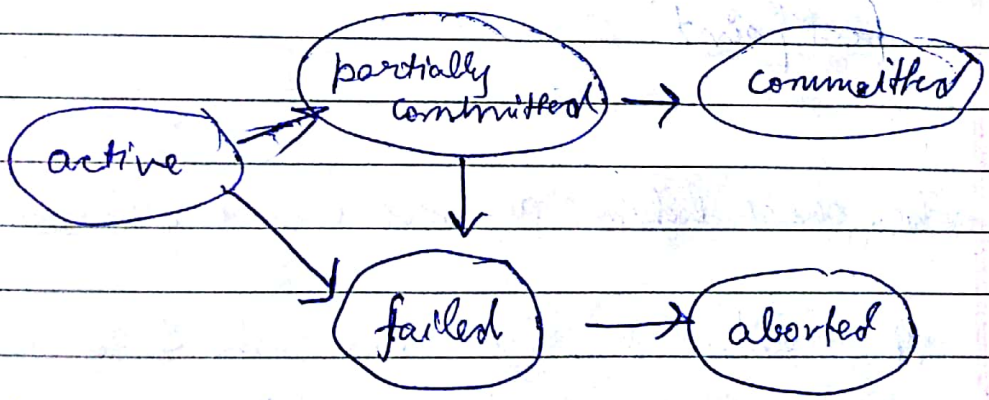
A transaction is a unit of power execution that accesses & possibly updates various data items.
TO preserve the integrity of the data the database must be ensure.

A tomicity
C onsistency
I solation
D urability

#tran action that completes its execution ~~successfully~~ successfully committed.

Once the changes caused by an aborted transaction have been undone the transaction is rolled back.

committed ko kabhi rolled back nahi kar sakte.



State Diagram of transaction

Pungnering :

In operating system, we will learn about time scheduling

Serial schedule

Conflict Serializability

If a schedule S can be transformed into a schedule S' by a series of swaps of non conflicting instructions, we say that S and S' are conflict equivalent.

* It is possible to have two schedules that produce the same outcomes, but that are not conflict equivalent.

Lock based protocols.

Lockpoint

Two phase locking does not ensure freedom from deadlocks.

upgradation → convert a lock S to lock X.
in first phase.

All locks are released after commit or abort.

Schedule:
A bundle of transactions executing together.

Consistency of Database

* Serial schedule: If a schedule is serial there doesnot exist any risk or any possibility of inconsistency, because if you are considering any one transaction at a time then there is no risk of having any kind of problem.

Disadvantage of serial ⇒ there is no concurrency.

If we have n transaction then → different serial schedules are possible ⇒ $n!$

transaction → $T_1, T_2, \_\_\_\_, T_n$, No. of Non Serial schedule

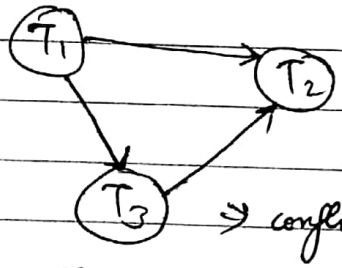
No. of instructions → n_1, n_2, n_n

$$= \frac{(n_1+n_2+\_\_n_n)!}{n_1!\ n_2!\ \_\_\ n_n!} - n!$$

* जहाँ Cycle Hai ⇒ conflict Serializable Nahi Har

Eg.

T_1	T_2	T_3
R(X)		
	R(Y)	
		R(Y)
	W(Y)	
W(X)		
		W(X)
	R(X)	
	W(X)	



⇒ conflict serializable

$T_1 \rightarrow T_3 \rightarrow T_2$

order of serializability

Teacher's Signature