D.Hepsi Priya

B180418CS

B-Batch

DBMS LAB- EX 6 AND 7.

Introduction:

B- Tree:

- B-tree is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in logarithmic time.
- The B-tree generalizes the binary search tree, allowing for nodes with more than two children and more than one key.
- B -tree is a balanced m-way tree,where m is the order.
- All the leaf nodes are on the same level and maintains sorted order.

Properties:

If 'm' is the order,then

- Every node has maximum of 'm' children.
- Minimum children : For Leaf node : 0 Children

For Root node:2 Children

For interal nodes: ceil (m/2) Children

- For every node has maximum of 'm-1' keys.
- Minimum keys : For Root Node : 1

  For all other nodes : ceil (m/2) - 1

Insertion key into B/B+ Tree:

- Insertion should always take place in leaf node.
- The search keys in the B-tree are unique that they are **no duplicate keys** in the B-tree**.**

Insertion steps:

1.Traverse the B-tree in order to find the appropriate leaf node at which the node can be inserted.

2.If the leaf node contain less than m-1 keys then insert the element in the increasing order.

3.Else, if the leaf node contains m-1 keys, overflow condition occurs then do follow the following steps.

- Insert the node in the increasing order of the keys.
- Split the new node into two nodes at the median. In general if the order is even we can select the median in two ways **: left baised and right baised.**

  Example: 7,13,10,16 of order 4.

  So the median can be 13,10 based on the baised version.
- The median value is sent to the parent node.
- And if the parent node also has 'm-1' keys then repeat the same steps for

splitting.

Construction of a B-tree:

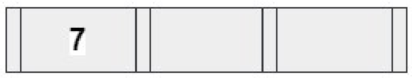
1. Initialise the construction with an empty tree. Create a new node with the key given and this will be the root node of the tree.
2. The keys to the parent node are based on the order,as we know the maximum keys are 'm-1' if the order is 'm'.
3. After the parent node is filled in sorted order. We insert the next key as the child using the Binary seacrh logic.
4. Now we insert keys using the rules we have discussed earlier in the insertion process.

Problem:

Construct a B-tree of order = 4 having keys 7, 13, 16, 24, 01, 03, 04, 05, 10, 11, 14, 15, 18, 19, 20, 21, 22, 25, 26.

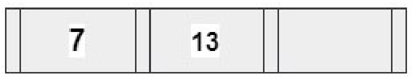
1. Insertion of 7:

7 is the first new key in the new node so its the root node and is also considered as leaf since it doesn't have child node. Its both root and leaf node.



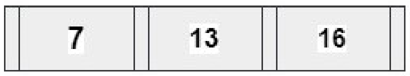
2.Insertion of 13 :

The element '13' can be added to the existing leaf node.This node has an empty position so it still acts as root as well as leaf node.



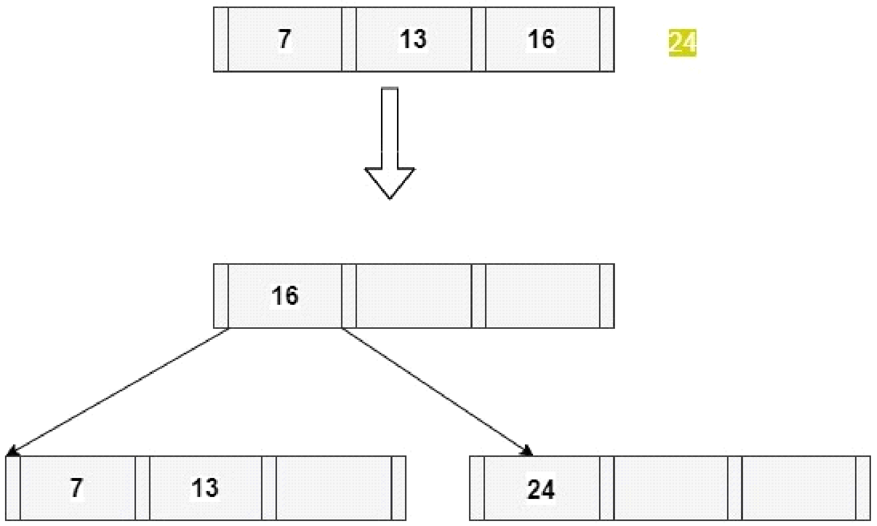
3. Insertion of 16 :

The element '16' can be added to the existing leaf node. So its added and it no more acts as leaf node.



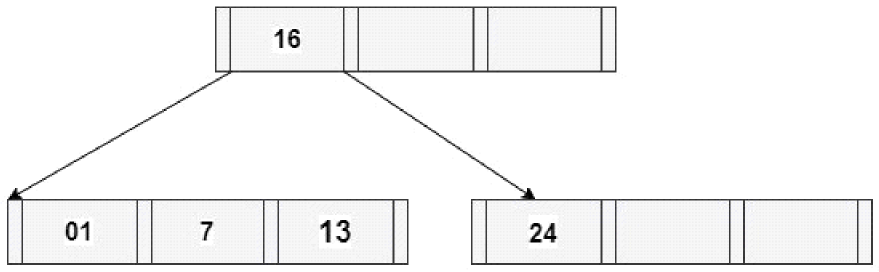
4.Insertion of 24:

Element `24` is added to the existing leaf node. Here, we have only one node and that node acts as the root and also as a leaf. This leaf node doesn't have an empty position. So, we split that node by sending middle value `16` to its parent node after sorting with the new element. But here, this node doesn't have a parent. So, this middle value becomes a new root node for the tree.



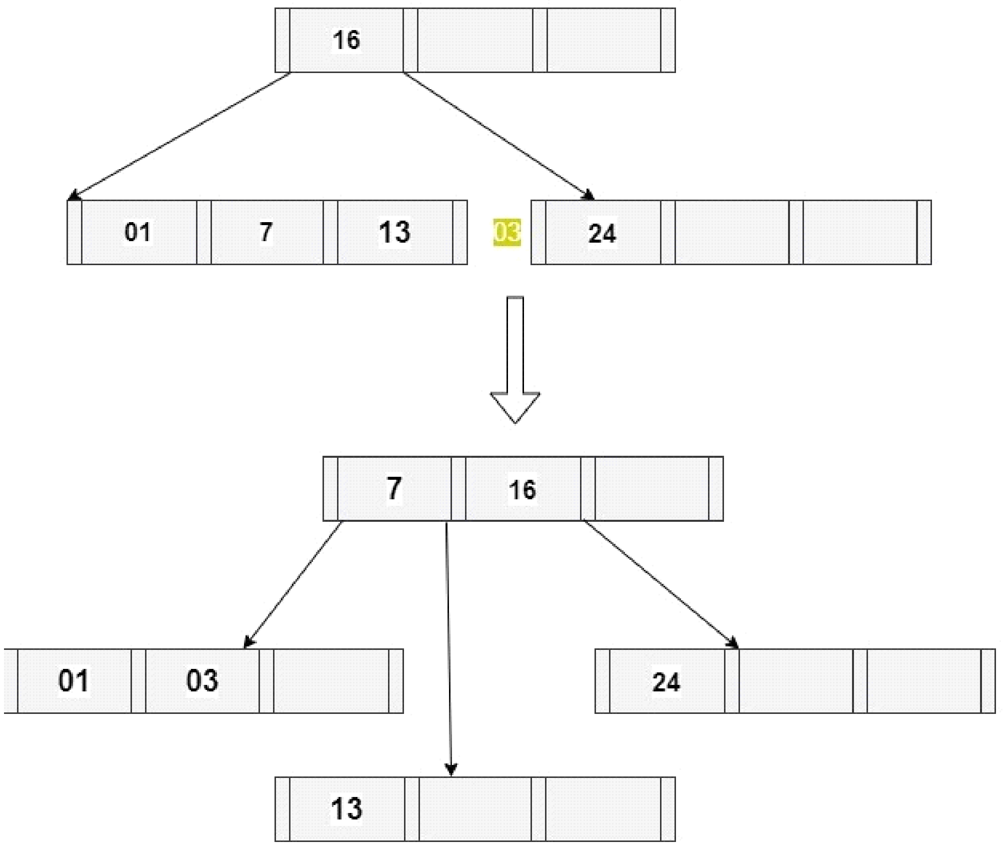
5.Insertion of 01:

Element '01' is smaller than the root '16'. '01' should be added to the leaaf node.So it is added to the leaf node with the values of '7' and '13' and its empty having a key space so '01' is inserted.



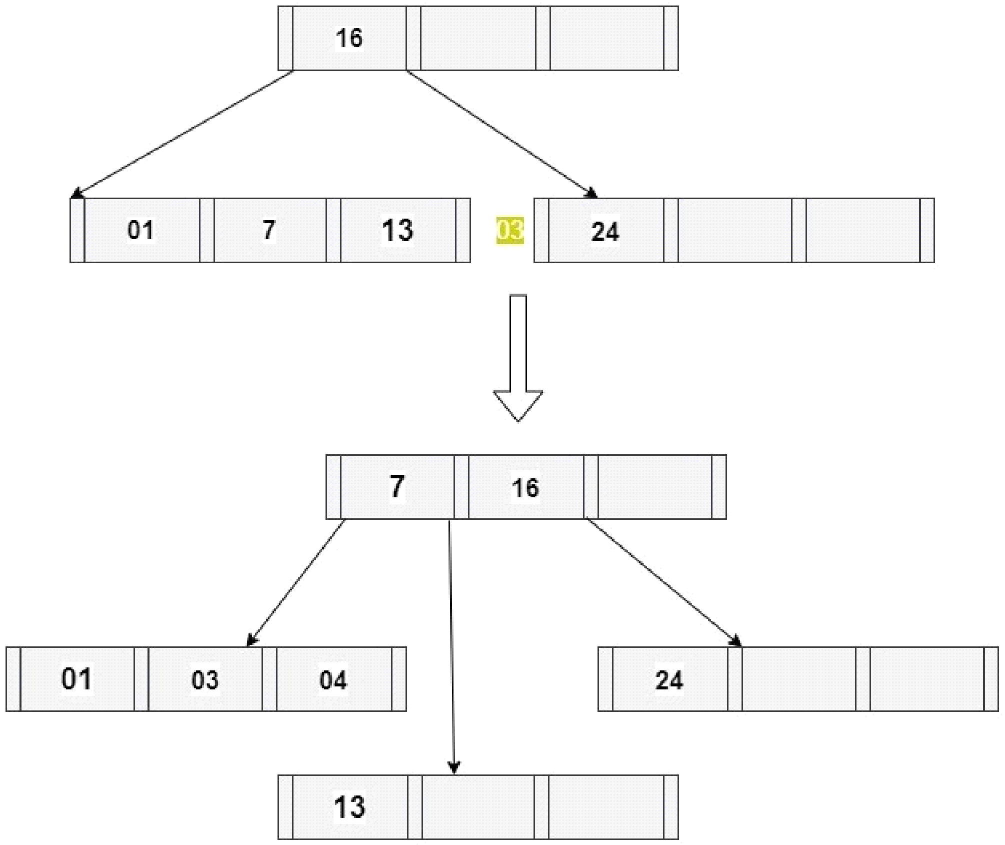
6.Insertion of '03':

Element of '03' is lessthan '16' it goes to the towards left child. And for the insertion to the leaf node with the values of '1','7' and '13' have no empty key space for insertion so **we split the node by the median to its parent node having '16'** and arrange the keys in sorted order.So here '7' is added to the parent node and '01' and '03' are left leaf nodes for '7' and '13' is the right leaf node.



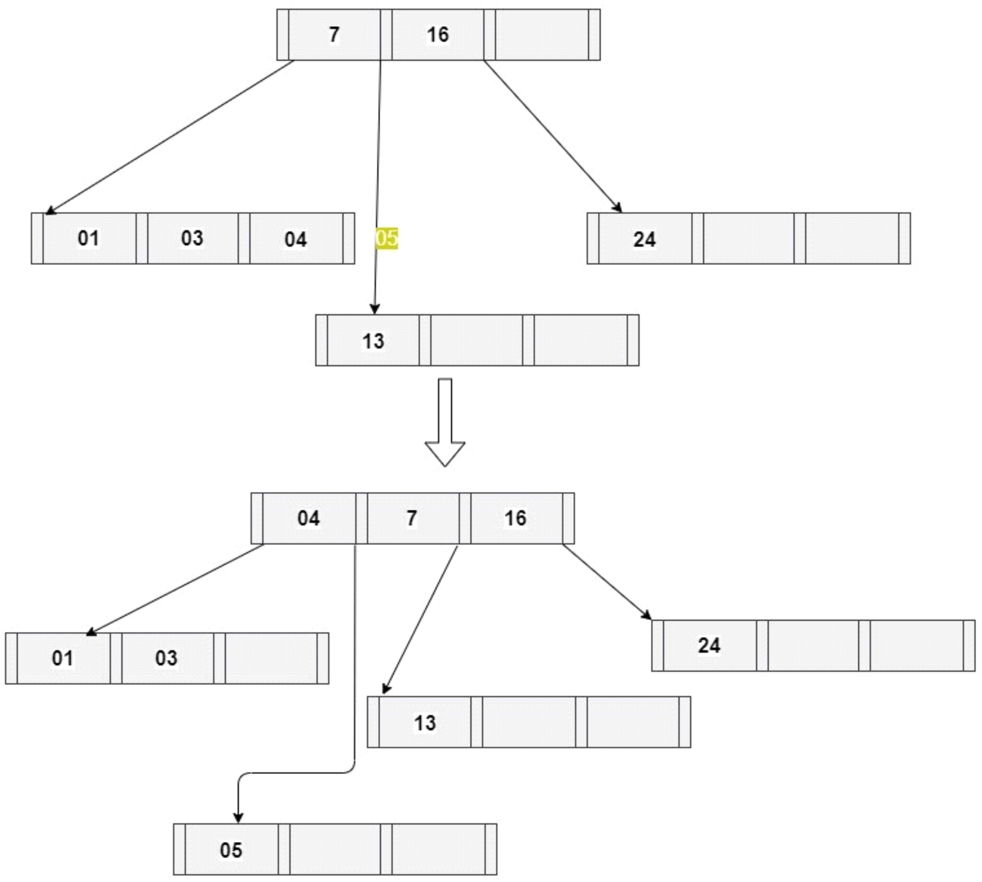
7.Insertion of '04':

Element '04' is lessthan '16' so it goes towards left and '07' is not leaf node and is greater than '04' so it goes to the left leaf node which has '01','03' keys. So here we insert '04' in sorted order.



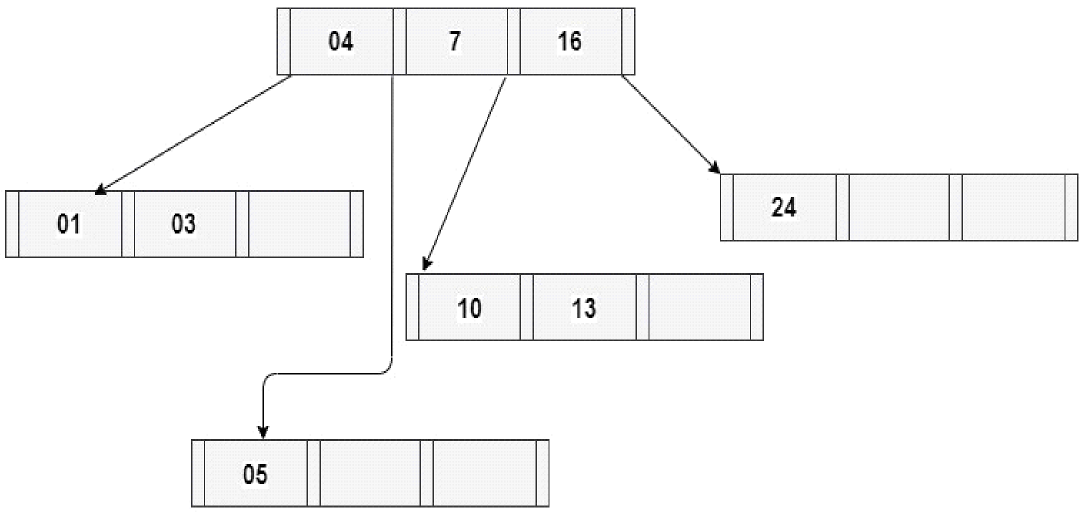
8.Insertion of '05':

Eleement '05' is lessthan '16' and '7' so it goes toward the left leaf node.But the leaf node having '01','03','04' are filled. So we split at the median and add it to the parent node havving '7','16'. And after splitting '04' has left leaf node with '01','03' and '05' on the right leaf node.



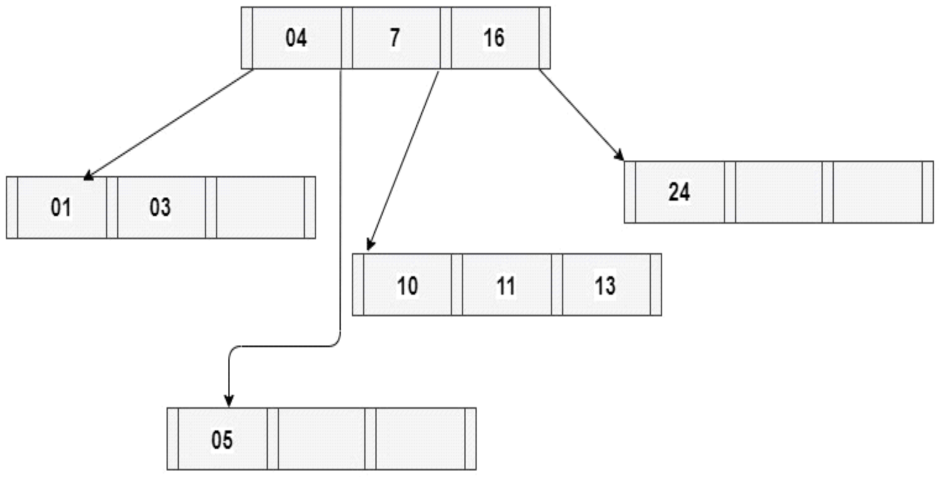
9.Insertion of '10':

Element `10` is greater than `07` and smaller than `16` and it is not a leaf node. So, we move to the right of root node of `07` or left node of `16`.So, we reach a leaf node with value `13` , it has an empty position. So, we can insert the value here at an empty position in sorted order.



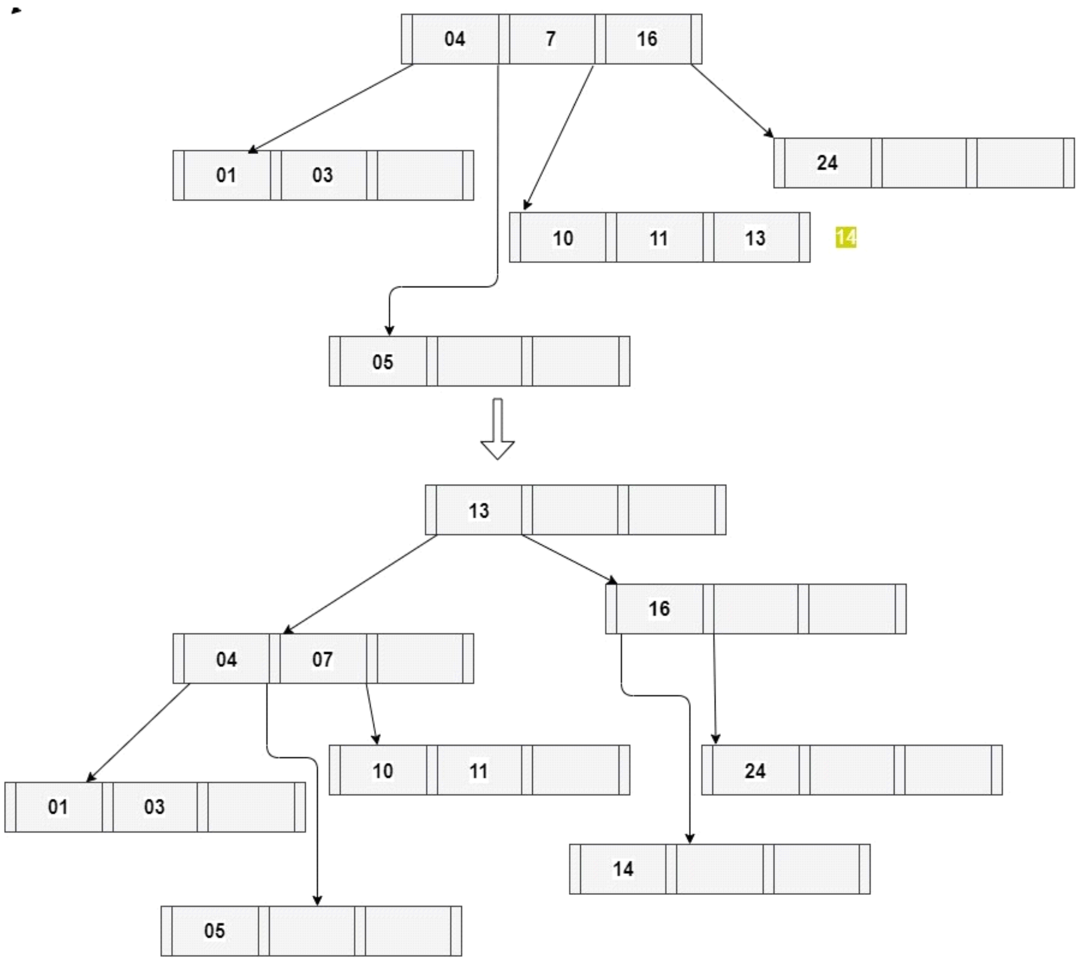
10.Insertion of '11':

Element `11` is greater than `07` and smaller than `16` and it is not a leaf node. So, we move to the right of root node of `07` or left node of `16`.So, we reach a leaf node with value `10 & 13` , it has an empty position. So, we can insert the value here at an empty position in sorted order.



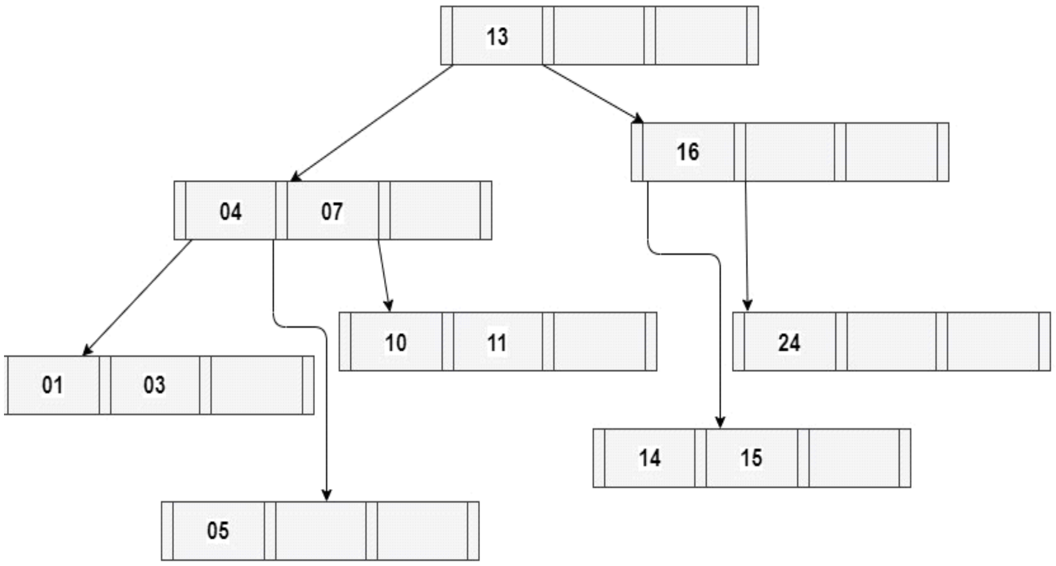
11.Insertion of '14':

First, the element `14` is compared, it's less than `16` and greater than `07,so the element goes to right node of `07` or left node of `16`.So, the leaf node is reached.As there is no space to fill the coming element we keep the middle element `13`after sorting into the parent node. Here for sending the middle node into parent node as the parent is already full,then the parent node middle element is send to top as there is parent node then it can be considered as parent node and splitting takes place to the left of `13` are `04 & 07` and to the right are `16`To the right of `07` are `10 &11` and to the left of `16` is `14`.



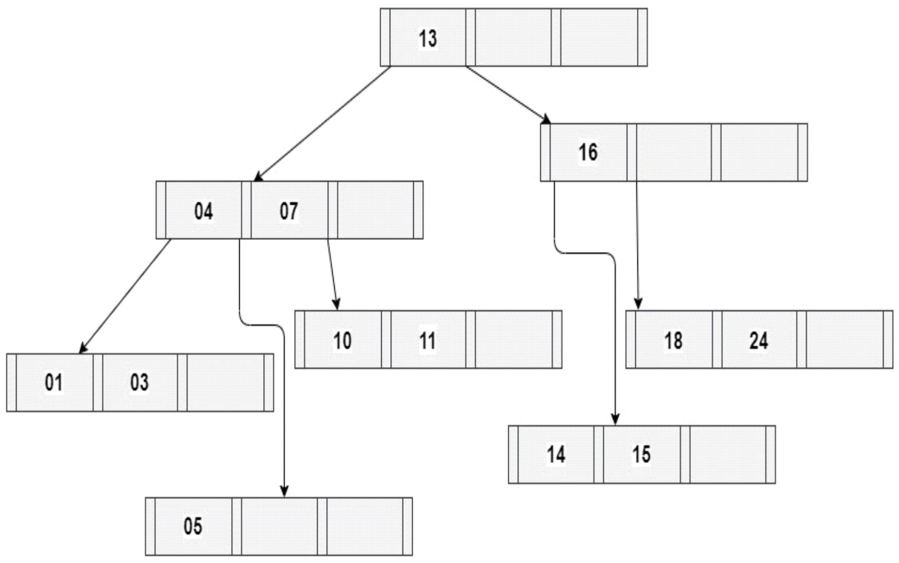
12.Insertion of 15:

Element `15` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `13`. By comparing it in that `15` is less than `16`.So, we reach a leaf node by using left node of `16`it has an empty position and it contains `14`and it is a leaf node. So, we can insert the value here at an empty position in sorted order.



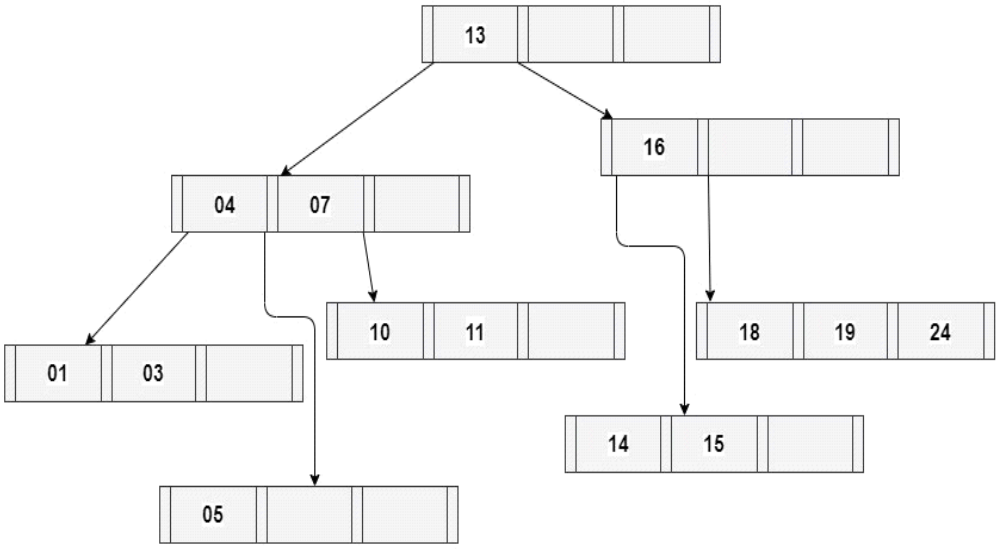
13.Insertion of '18':

Element `18` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `13`. By comparing it in that `18` is greater than `16`.So, we reach a leaf node by using right node of `16`it has an empty position and it contains `24`and it is a leaf node. So, we can insert the value here at an empty position in sorted order.



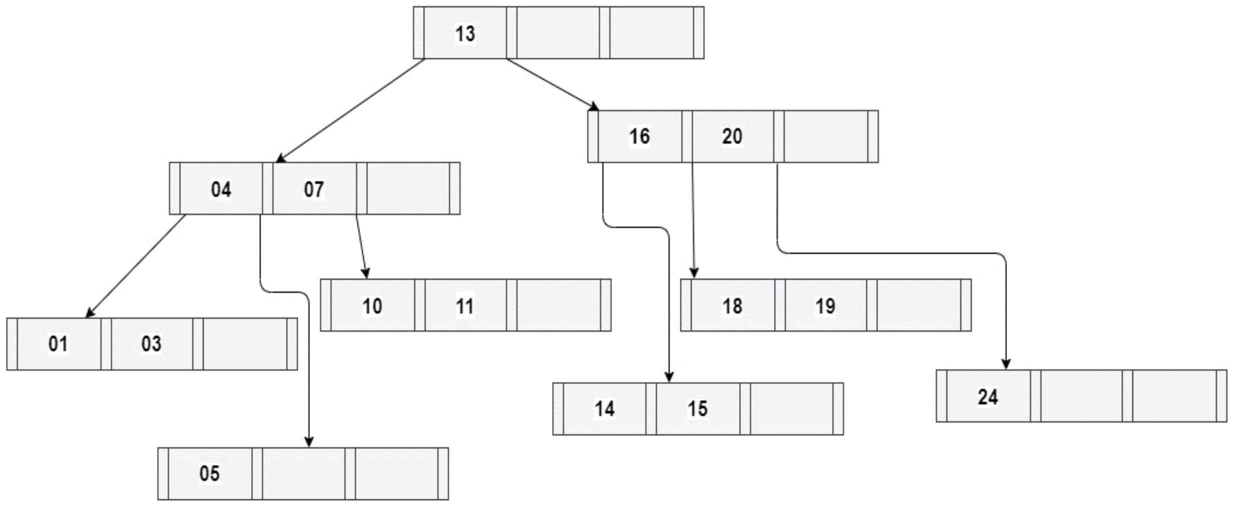
14.Insertion of '19':

Element `19` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `13`. By comparing it in that `19` is greater than `16`.So, we reach a leaf node by using right node of `16`it has an empty position and it contains `18 & 24`and it is a leaf node. So, we can insert the value here at an empty position in sorted order**.**



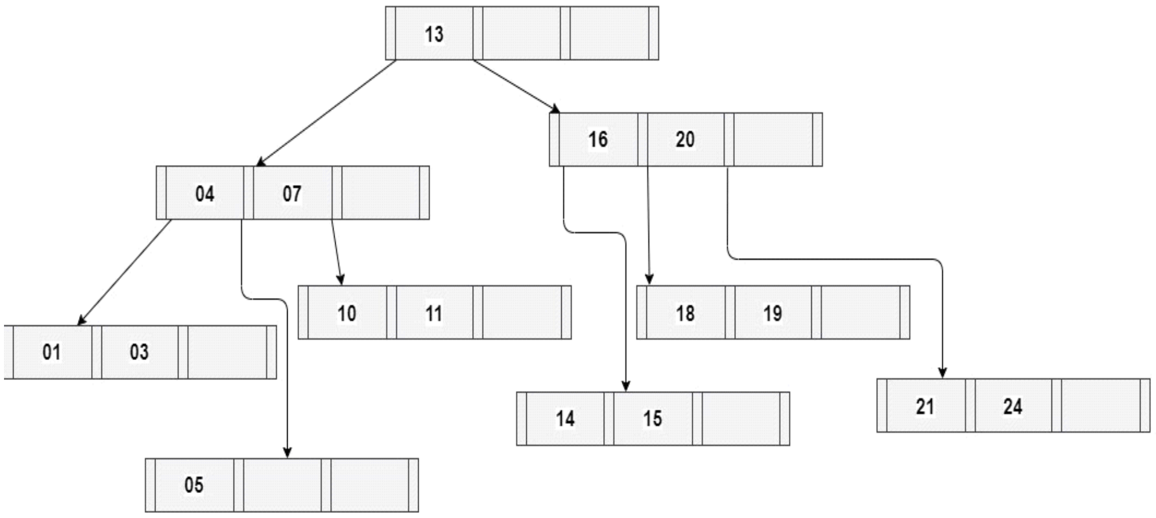
15.Insertion of '20':

First, the element `20` is compared, it’s greater than `13`,so the element goes to right node of `13`.Then the node contains `16` which is less than the given number.So,it goes to right node of it.As there is no space to fill the coming element we keep the middle element `20`after sorting into the parent node.Then split the node `18 & 19` to the left of `20` and `24` to the right of `20`.



16.Insertion of '21':

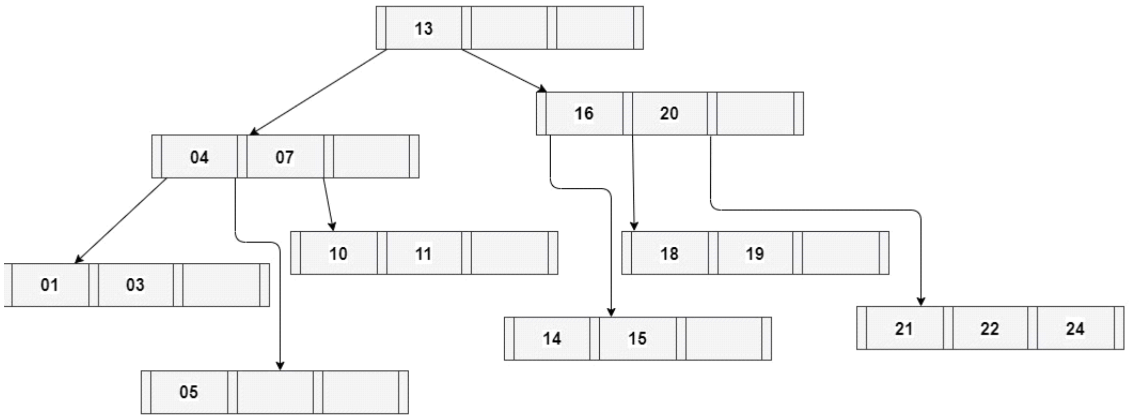
Element `21` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `13`. By comparing it in that `21` is greater than `16 &20`.So, we reach a leaf node by using the right node of `20`it has an empty position and it contains `24`and it is a leaf node. So, we can insert the value here at an empty position in sorted order.



17.Insertion of '22':

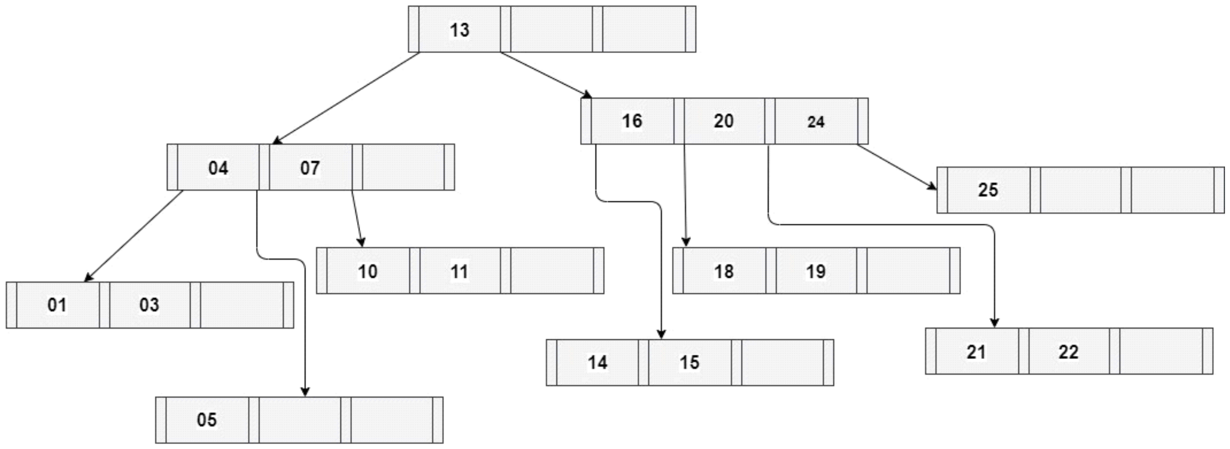
Element `22` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `13`. By comparing it in that `22` is greater than `16 &20`.So,

we reach a leaf node by using the right node of `20`it has an empty position and it contains `21 &24`and it is a leaf node. So, we can insert the value here at an empty position in sorted order.



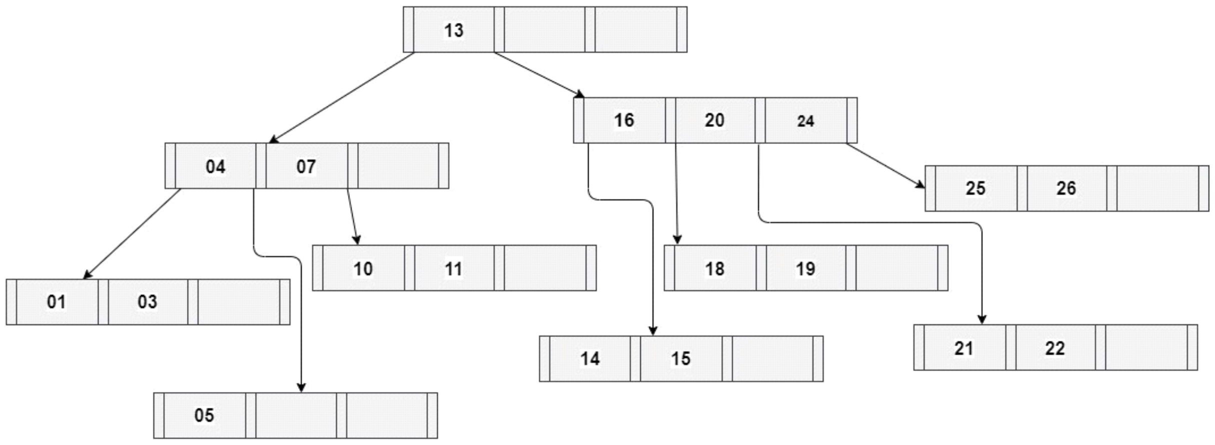
18.Insertion of '25':

First, the element `25` is compared, it's greater than `13`,so the element goes to right node of `13`.Then the node contains `16 &20` which is less than the given number.So,it goes to right node of `20`.As there is no space to fill the coming element we keep the middle element `24`after sorting into the parent node.Then split the node `21 & 22` to the right of `20` and 25` to the right of `24`.



19.Insertion of '26':

Element `26` is greater than `13`and it is not a leaf node. So, we move to the right of the root node of `16&29&24`. By comparing it in that `26` is greater than all the elements in the node.So, we reach a leaf node by using the right node of `24`it has an empty position and it contains `25`and it is a leaf node. So, we can insert the value here at an empty position in sorted order.

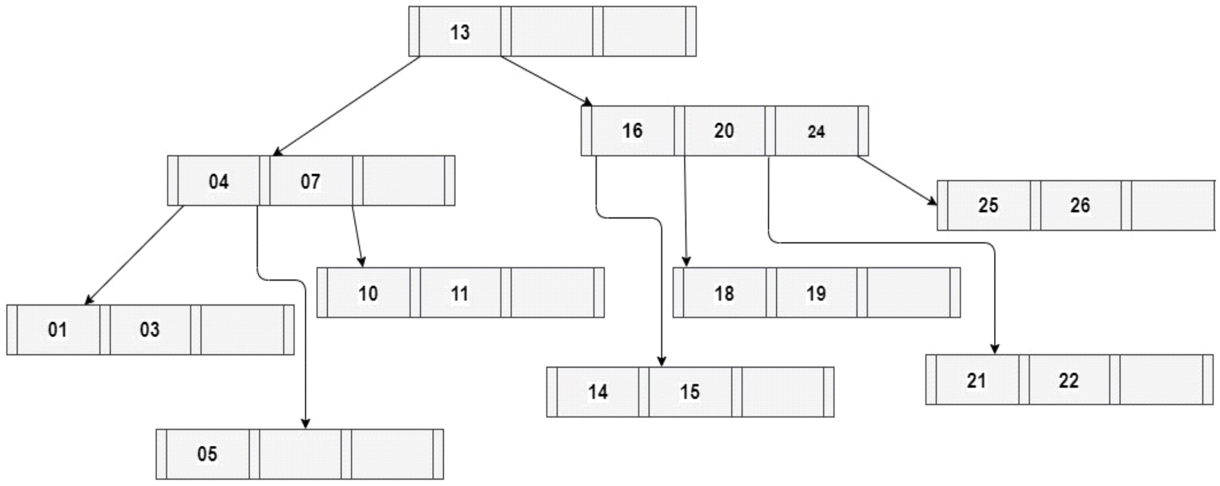


Note : The Elements which is shown in yellow color are responsible

for overflow elements.We split the node and link according to the

conditions

THE FINAL TREE IS:



THE END