ADS : lab 5. Akanksha Hadding

USN : 1BM18CS007. Batch B1. Date: P. No:

2-3 trees Implementation :

Insertion

- insert new leaf in appropriate place
- repeat until all non-leaf nodes have 2 or 3 children.
  - if there is a node with 4 children split parent into 2 parent node with 2 children each.
  - if split root : add a new root.
- adjust search values along insertion path.

Deletion

- delete x from 2-3 tree
  - let p - parent(x)
  - if x is root remove x
  - else if x has 3 children then remove x.
  - else
    - if p is root → remove x & p
    - else
      - let S be other child of p, let l = left sibling of p, r = right sibling of p
      - if l has 3 children, replace x by one of l's children.
      - else if r has 3 children, replace x by one of r's children
      - else

        remove x

Akanksha Hadding

Date:
P. No:

1BM18CS007.

combine 'p' with l (or r)
make s a children of l or r.
rename p to x.
recursively remove x.

Insert Algorithm:

insert (TwoThreeNode N, comparable value)
locate leaf in which to put value.
if leaf is a 2node, make it a 3 node
insert value at either small key or large key.

if leaft is a 3 node
insert value temporarily & split the node.

Split (TwoThreeNode N).
if N is root
make middle key into a 2 node
make small & large key into 2 node.
reattribute children
else : N has parent P → move middle key to P, make small, large key into 2 node, reattribute children.

Akanksha
Naddha.

1BM18CS007.

Date:
P. No:

```
insert (a,v)
{
    if (H consist of single leaf labeled b)
        create a new root 'r'
        create a new leaf v labeled a
        make l & v children of r'
        update L & H for r'
    else
        Set f to search (a,r)
        create a new leaf l labeled a
        if f has 2 children
            insert l into proper position
            update L & H.
        else
            create a transitory H node tree at f
            Addchild (f)
}

Addchild (v)
{
    create new node v'
    move 2 rightmost children of v to v'
    if (v has no parent)
        make new root r'
        make v: left child & v': right child
        update L & H
    else
        let f be parent of v.
        make v' child of f immediately to right of v
        if f now has H children
```

Date:
P. No:

IBM18CS007

Addchild cf)
else
update k & M
}

Delete :-

delete ()
{
let f be parent of node just deleted
while (f is an illegal internal node)
{
if (f has no parent)
make single child of f, new node
delete f
set cf to the root
else
let g be parent of f
if (one of f's sibling is a 3 node)
move one child from 3-node into
f, update K1 & K2 everywhere
else
give f's remaining child to one f's
sibling
delete f.
update K1 & K2 everywhere
set f to g.
}
}