classmate

Date

Page

M Ramaswamy
1BM18CS147

02/11/2020

EXPPROGRAM 05

Q) WAP to perform insertion and deletion on 2 3 trees

```
class TwoThreeTreeNode
{
  int *keys;
  TwoThreeTreeNode **child;
  int n;
  bool leaf;
  public:
    TwoThreeTreeNode(bool leaf);
    void traverse();
    void remove(int k);
    void findKey(int k);
    void insertNonFull(int k);
    void merge(int ind);
    // other function declarations
    friend class TwoThreeTree;
};

class TwoThreeTree
{
  TwoThreeTreeNode *root = NULL;
```

Date
Page

ARPANA M R
1BM18CS147

```
public:
    void traverse()
    {
        if (root != NULL)
            root -> traverse();
    }
    void insert(int k);
    void remove(int k);
};

void TwoThreeTree::insert(int k)
{
    if (root == NULL)
    {
        root = new TwoThreeTreeNode(true);
        root -> keys[0] = k;
        root -> n = 1;
    }
    else
    {
        if (root -> n == 3)
        {
            TwoThreeTreeNode *s = new TwoThreeTreeNode(false);
            s -> child[0] = root;
            s -> splitChild(0, root);
```

ARPANA M R
1BM18CS047
Arpana

```
int i=0;
if(s->keys[0]<k)
    i++;
s->child[i]->insertNonFull(k);
root=s;
}
else
root->insertNonFull(k);
}
}

void TwoThreeTreeNode::insertNonFull(int k)
{
int i=n-1;
if(leaf==true)
{
while(i>=0 && keys[i]>k)
{
keys[i+1]=keys[i];
i--;
}
keys[i+1]=k;
n=n+1;
}
```

ARPANA M R
1BM18CS014
Arpana

```
else{
    while (i>=0 && keys[i]>k)
        i--;
    if (child[i+1]->n==3)
    {
        splitchild(i+1, child[i+1]);
        if (keys[i+1]<k)
            i++;
    }
    child[i+1]->insertNonFull(k);
}}

void TwoThreeTreeNode::splitchild(int i, TwoThreeTreeNode *y)
{
TwoThreeTreeNode *z=new TwoThreeTreeNode(y->leaf);
z->n=1;
z->keys[0]=y->keys[2];
if(y->leaf==false)
{
  for(j=0;j<2;j++) z->child[j]=y->child[j+2];
}
y->n=1;
for(j=n;j>=i+1;j--)
    child[j+1]=child[j];
child[i+1]=z;
```

ARPANA M R
1BM18CS147
Aryana

```
for (j=n-1; j>=i; j--)
    Keys[j+1] = Keys[j];
Keys[i] = y->Keys[k];
n=n+1;
}

void TwoThreeTreeNode::traverse()
{
  for (i=0; i<n; i++)
  {
    if (leaf == false)
      child[i]->traverse();
    cout<<" "<<Keys[i];
  }
  if (leaf == false)
    child[i]->traverse();
  cout<<endl;
}

void TwoThreeTreeNode::remove(int k)
{
  int ind = findKey(k)
  if (ind<n && keys[ind]==k)
  {
    if(leaf) removeFromLeaf(ind);
```

ARPANA MR
1BM18CS047
Arpana

```
else
    removeFromNonLeaf(ind); }
else {
    if (leaf) {
    cout << "Key doesn't exist" << endl;
      return;
    }
    bool flag=((ind ==n) ? true : false);
    if (child[ind]->n<2)
        fill(ind);
    if (flag && ind >n)
        child[ind-1]->remove(k);
    else
        child[ind]->remove(k);
    }
    return; }

void TwoThreeTreeNode :: removeFromLeaf(int ind)
{
    for(i=ind+1; i<n; i++)
        keys[i-1]=keys[i];
    n--;
    return }

void TwoThreeTreeNode :: removeFromNonLeaf(int in
```

ARPANA MR
1BM18CS041
@Arjuna

```
}
int x = Keys[ind];
if (child[ind]->n >= 2)
{
  int pred = getPred(ind);
  Keys[ind] = pred; remove(pred); }
else if (child[ind+1]->n >= 2)
{
  int succ = getSucc(ind);
  Keys[ind] = succ;
  child[ind+1]->remove(succ) }
else
{ merge(ind);
  child[ind]->remove(k);
}
return; }

void TwoThreeTree::remove(int k)
{
  if(!root) {
  cout<< "Tree is empty\n"; return; }
  root->remove(k);
  if(root->n==0) { TwoThreeTreeNode *tmp=ro
  if(root->leaf) root=NULL;
  else root=root->child[0];
  delete tmp; }
  return; }
```