ADS LAB

Batch-3

Prajwal Patil
1BM17CS062

2-3 trees insertion & deletion

```
class treeNode
{
 int * keys;
TreeNode ** child;
 int n;
 bool leaf;
 friend class Tree;
};
 class Tree
{
 Tree Node *root = NULL;
 Public;
 void traverse() {
   if(root !=NULL)
     root -> traverse();
  }
 void insert (int k);
 void remove (int k);
};

void Tree :insert (int k)
{
  if (root==NULL)
  {
    root = new Tree Node(true);
   root -> keys[0] =k;
    root ->n =1;
}
 else{ if (root->n == 3)
     {
```

```
TreeNode *S =new TreeNode(false);
  S -> child[0] =root;
  S -> splitchild(0, root);
    int   i=0;
  if (S->keys[0]<K)
        i++;
  S-> child[i] -> insertNonfull(K);
    root = S;
}
  else root -> insert NonFull(K);
  }
}
void TreeNode:: 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;
}
 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] -> insert NonFull(k);

void TreeNode :: splitchild(int i, TreeNode* y)
{
 TreeNode *z = new TreeNode(y->leaf);
 z -> n=1
z -> keys[0] = y -> keys[2];

 if(y -> leaf == false)
 {
   for(int j=0; j<2; j++)

 z -> child[j] = y -> child[j+2];
}
  y -> n = 1;
for(int  j=n; j>=i+1; j--)
    child[j+1] = child[j];
 child[i+1] = z;
for(int  j = n-1; j >= i; j--)
     keys[j+1] = keys[j];
 keys[i] = y -> keys[1];
      n=n+1;
}
 void TreeNode :: remove(int k)
{

 int idx = find key(k)
 if (idx <n && keys[idx] == k)
{
  if(leaf)
          remove FromLeaf(idx);
else
     remove from non leaf(idx);
}
```

```
else{
   if(leaf)
{
cout << "key doesnt exist "<< endl;
return;

bool flag =(idx==n) ? true: false);

 if (child[idx]->n<2)
      fill(idx);
if( flag && idx >n)
    child[idx-1]→ remove(k);
else
 child (idx) → remove.[k];
}
return;
}
 void TreeNode :: remove fromleaf(int idx)
{
  for(int i=idx +1 ;i<n ;i++)
      keys[i-1]=keys(i);
 n--;
 return;
}
void TreeNode ::remove FromNonleaf(int idx)
{
 int K =keys[idx] ;
  if (child (idx]->n>=2)
{
 int pred=get pred(idx);
 keys[idx]=pred;
child[idx] -> remove(pred);
}
```

```
else if(child [idx+1] -> n>=2)
{
  int succ = get succ(idx);
   keys(idx) = succ;
child [idx]+1) -> remove(succ);
}
 else
 {
   merge(idx);
child (idx] -> remove(k);
}
return;
}
void Tree :: remove(int k)
 {
   if(!root)
{
  cout<< "Tree is empty "<< endl;
    return;
}
 root  -> remove(k);
 if(root -> n==0)
  {
   TreeNode *tmp  = root;
   if (root -> leaf)
        root =NULL;
   else
   root = root -> child [0];
  delete tmp;
}
return;
}
```