Insertion 2-3 tree Writeup

```
class TreeNode
{
 int *keys;
 Treenode **child;
 int n;
 bool leaf;
 friend class Tree;
};
class Tree : Insert (int k)
{
  TreeNode if (root==NULL)
  {
    root= new TreeNode (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 → insertNon full(k)
  }
}
void TreeNode :: Insert Nonfull (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++;
       }
    }
}
```

Deletion

```
void TreeNode :: remove(int k)
{
   int idx = findkey(k)
   if (idx <n  && key[idx]==k)
   {
      if (leaf)
         removefromLeaf (idx)
      else
         removefromNonLeaf(idx)
   }
   else
   {
     if(leaf)
     {
       return
     }
   }
```

```
void removeFromNonLeaf(int idx)
{
    int k = keys[idx];
    if (child[idx]->n >= 2)
    {
        int pred = getPred(idx);
        keys[idx] = pred;
        child[idx]->remove(pred);
    }
    else if (child[idx+1]->n >= 2)
    {
        int succ = getSucc(idx);
        keys[idx] = succ;
        child[idx+1]->remove(succ);
    }
    else
    {
        merge(idx);
        child[idx]->remove(k);
    }
    return;
}

void remove(int k)
{
    if (!root)
    {
        return;
    }
    root->remove(k);
    if (root->n == 0)
    {
        Forward *tmp = root;
        if (root->leaf)
            root = NULL;
        else
            root = root->child[0];
        delete tmp;
    }
    return;
}
```