14/10/2020

AVL tree

Pavan A.G.
1BM18CS068

```
Node * insert(Node *node, int key) {
if (!node) return new node(k);
 if (key < node -> key)
   node -> left = insert(Node -> left, key)
  if (key > node -> key)
     node -> right = insert (node -> right, key)
   node -> height = 1 + max(node -> left -> height,
                    node -> right -> height);
    if (gt balance(node) > 1) <
    if (key < node -> left -> key)
   return right rotate(node);
    else <
        leftrotate(node -> left)
   return right rotate (node);
     }
   else if (gt balance(node) < -1) {
      if (key > node -> right -> key,
   return leftrotate (node);
   else
       right rotate (node -> right)
   return left rotate (node);
   }
    return node;
}
```

Pavan. A.G.
1BM19CS088

AVL tree

```
Node * delete (node *root, int key) {
  if (! root) return root;
  if (key < root->kg)
    root -> left = delete (root->left, key)
  else if (key > root -> key)
    root -> right = delete (root-> right, key)
    else {
      if (! root -> left (! root->right) {
    Node * temp = root -> left) root * eft; root *right;
    if (! temp)
      delete (root)
    else { return root; // null
      * root = *temp;
      delete temp;
    }
    else {
      node * temp = inorder succesor (root -> right)
      root -> key = temp -> key;
      root -> right = delete (root -> left, temp -> key);
      int bf = get balance (root);
        if (bf > 1) {
        if (get balance (root -> left) >= 0)
      return right rotate (root);
      else left rotate (root);
      else
```

```
left rotate (root->left);
  return right rotate (root)
}
 if (bf < -1);
    if (get balance (root->right) <= 0)
        left rotate (root)
    else < right rotate (root->right)
        left rotate (root)
      }
    }
  return root;
  }
```