Lecture 37 BST

1. <https://leetcode.com/problems/search-in-a-binary-search-tree/>

class Solution {

public:

TreeNode\* searchBST(TreeNode\* root, int val) {

if(root==NULL)

{

return root;

}

if(root->val == val)

{

return root;

}

else if(root->val > val)

{

return searchBST(root->left, val);

}

else

{

return searchBST(root->right, val);

}

}

};

1. <https://leetcode.com/problems/delete-node-in-a-bst/>

class Solution {

public:

TreeNode\* deleteNode(TreeNode\* root, int key) {

if(root==NULL)

{

return NULL;

}

else if(root->val > key)

{

root->left = deleteNode(root->left, key);

return root;

}

else if(key == root->val)

{

// leaf node

if(root->left == NULL && root->right==NULL)

{

delete root;

return NULL;

}

// only 1 child

if(root->left == NULL && root->right!=NULL)

{

TreeNode\* temp = root->right;

delete root;

return temp;

}

if(root->right==NULL && root->left!=NULL)

{

TreeNode\* temp = root->left;

delete root;

return temp;

}

// 2 children

TreeNode\* replace = root->right;

while(replace->left!=NULL)

{

replace = replace->left;

}

root->val = replace->val;

root->right = deleteNode(root->right, replace->val);

return root;

}

else

{

root->right = deleteNode(root->right, key);

return root;

}

}

};

1. <https://leetcode.com/problems/validate-binary-search-tree/>

class Solution {

bool isBST(TreeNode\* root, long minV, long maxV)

{

if(root==NULL)

{

return true;

}

if(root->val > minV && root->val < maxV && isBST(root->left, minV, root->val) && isBST(root->right, root->val, maxV))

{

return true;

}

return false;

}

public:

bool isValidBST(TreeNode\* root) {

long minV = LONG\_MIN;

long maxV = LONG\_MAX;

if(isBST(root, minV, maxV))

{

return true;

}

else

{

return false;

}

}

};