problem domain : binarytree

- Create a BinaryTree class
- Define a method for each of the depth first traversals called preOrder, inOrder, and postOrder which returns an array of the values, ordered appropriately.

- Any exceptions or errors that come from your code should be semantic, capturable errors. For example, rather than a default error thrown by your language, your code should raise/throw a custom, semantic error that describes what went wrong in calling the methods you wrote for this lab.

Input ← root
Output ← tree

Edge cases

- send null number
- send undefined number

problem domain : binarySearchtree

- Create a BinarySearchTree class
- Define a method named add that accepts a value, and adds a new node with that value in the correct location in the binary search tree.
- Define a method named contains that accepts a value, and returns a boolean indicating whether or not the value is in the tree at least once.

Input ← root
Output ← tree

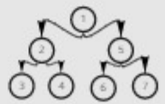
Edge cases

- send null number
- send undefined number

Algorithm : binarytree

- create class BinaryTree
- define method postorder
  -define tree as an array
  -define values as a function and check if node.left is null if not recall values
  -check if node.right is not null recall values
  -push node to the array
  -return tree
- define preorder as a function
  -define tree as an array
  -define values as a function
  - push node to the array
  -check if node.left is null if not recall values
  -check if node.right is not null recall values
  -return tree
- define preorder as a function
  -define tree as an array
  -define values as a function
  -check if node.left is null if not recall values
  - push node to the array
  -check if node.right is not null recall values
  -return tree

Visual

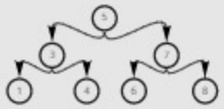


Algorithm : binarySearchTree

- create class BinarySearchTree
- define method add(value) wich take value as input
- check if root is null add the value to the root
- loop over the tree check if the root is equal the value dont add it
- if the root is greater than value so move to the left
- if the root is less than value move to the right
- check if the left / right is null add the value

- define contains which take value as input
- check if the root is null return that the tree is null
- else loop over the tree and check if the root is equal value if not check if is greater than value move to the left else move to the right
- if u found the value return true else return false

Visual



code : binary tree

```
class BinaryTree {
 constructor(root=null){
  this.root= root;
 }

 preorder(){
  let tree=[];
  let values = (node)=>{
   tree.push(node.value);
   if(node.left) values(node.left);
   if(node.right) values(node.right);
  };
  values(this.root);
  return tree;
 }

 postorder(){
  let tree=[];
  let values = (node)=>{
   if(node.left) values(node.left);
   if(node.right) values(node.right);
   tree.push(node.value);
  };
  values(this.root);
  return tree;
 }

 inorder(){
  let tree=[];
  let values = (node)=>{
   if(node.left) values(node.left);
   tree.push(node.value);
   if(node.right) values(node.right);
  };
  values(this.root);
  return tree;
 }
}
```

Code : binary search tree

```
class BinarySearchTree{
 constructor(){
  this.root=null;
 }
 add(value){
  let temp=this.root;
  let node = new Node(value);
  if(!temp) return this.root=node;

  while (temp) {
   if(value=== temp.value) return 'cant add exist value';
   if(temp.value>value){
    if(!temp.left) return temp.left=node;
    temp=temp.left;
   }else{
    if(temp.right) temp=temp.right;
    else return temp.right=node;
   }
  }
 }

 contains(value){
  let temp=this.root;
  if(!temp) return 'the tree is null';

  while (temp) {
   if(value<temp.value) temp=temp.left;
   else if (value>temp.value) temp=temp.right;
   else if(value === temp.value) return true;
  }
  return false;
 }
}
```

Verification

- Verify Big O of written Code.
- Verify Code Matches Algorithm.

Big O binarytree
time : O(n)
space : O(h)

Big O add
time : O(n)
space : O(h)

Big O contains
time : O(n)
space : O(h)