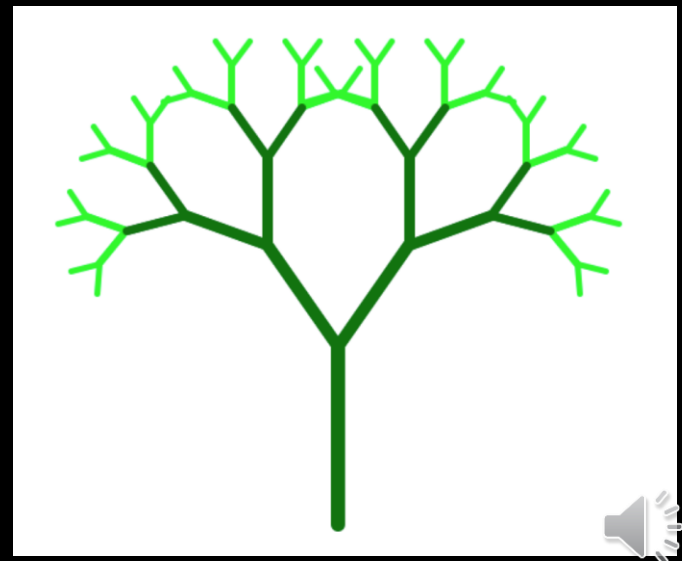
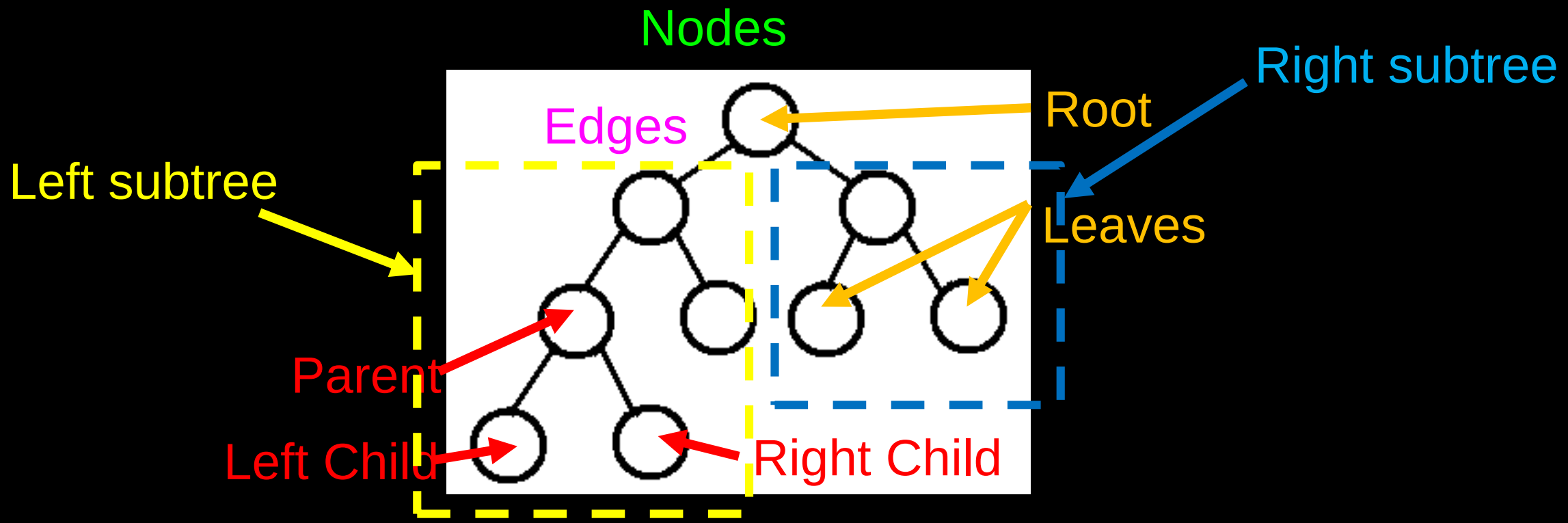
Binary Tree using OOP

LT14c



Recap (Binary Tree)

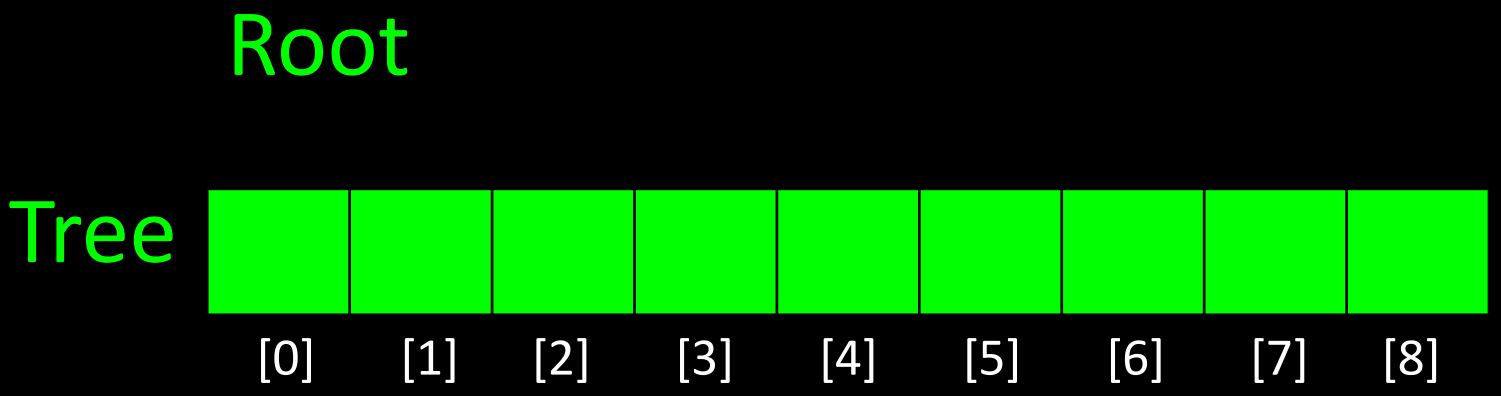
- A non-linear data structure whose entries follow a hierarchical organization.



Binary Tree using OOP

1) The Tree object has an attribute Root which is the pointer to the root node and an attribute tree which is an array that stores the nodes added to the binary tree.

2) Each node of a binary tree must contain the data to be stored and two pointers: one to the left child, one to the right child.



Node object

Node Class

- Tree Node objects can be instantiated from the Node class.

Node Class
data leftPtr: INTEGER rightPtr: INTEGER
Constructor(data) get_data() get_leftPtr() get_rightPtr() set_data(new_data) set_leftPtr(new_left) set_rightPtr(new_right)

Node object

Tree Class

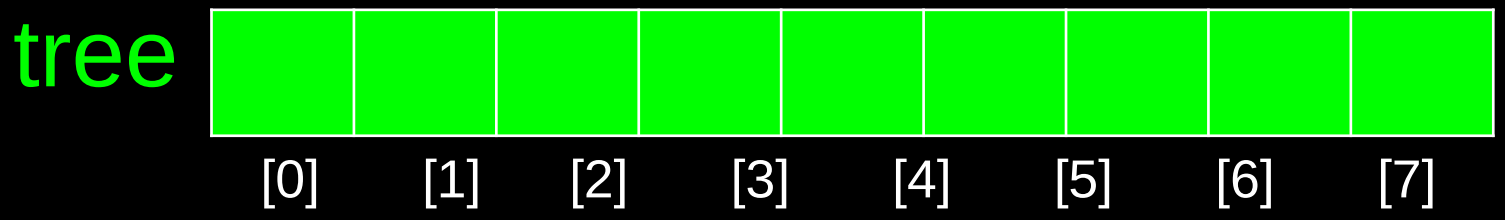
- Binary Tree objects can be instantiated from the Tree class.

Tree Class

```
tree : ARRAY OF Nodes
Root: INTEGER
NextFreeChild: INTEGER
```

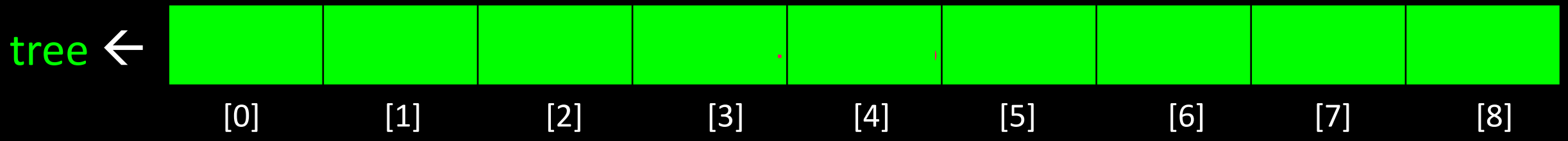
```
Constructor()
add(newItem)
inOrder()
display()
```

Root ← -1



NextFreeChild ← 0

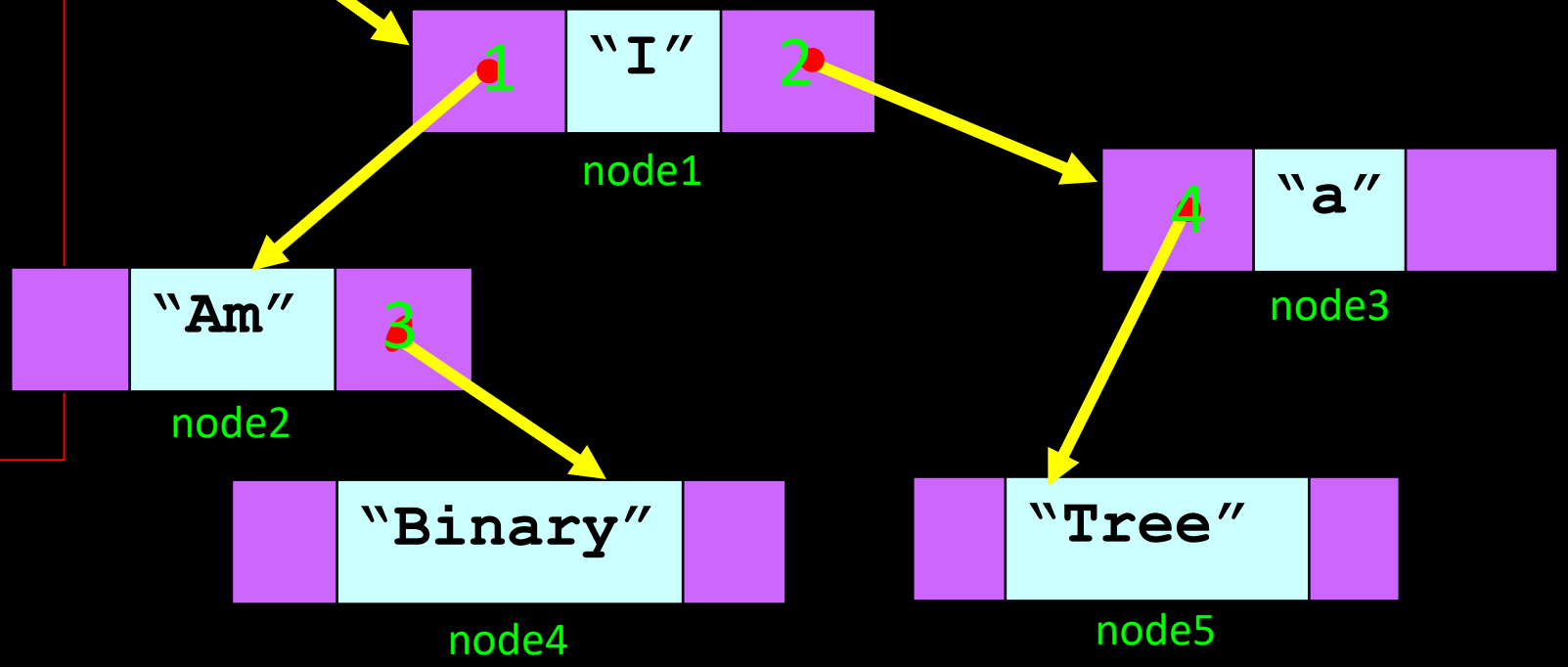
Binary (Search) Tree – Adding Nodes to Tree




```
>> t1= Tree()
>> t1.add('I')
>> t1.add('Am')
>> t1.add('a')
>> t1.add('Binary')
>> t1.add('Tree')
```

NextFreeChild ← 1

Root ← 0



Binary Tree – Traversing through the binary tree

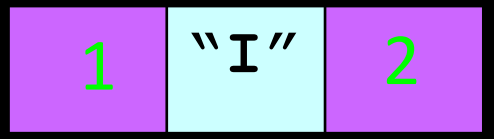
Starting at the root, we traverse in the following order recursively:

For in-order traversal:

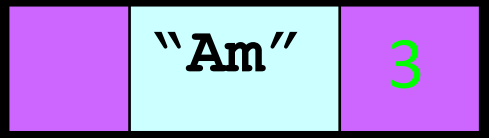
1. First visit the left subtree of the node.
2. Then visit the node itself.
3. Then visit the right subtree of the node.

Root ← 0

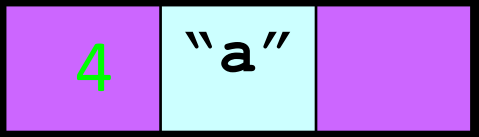
```
>> t1.inorder()
>> 'Am'
>> 'Binary'
>> 'I'
>> 'Tree'
>> 'a'
```



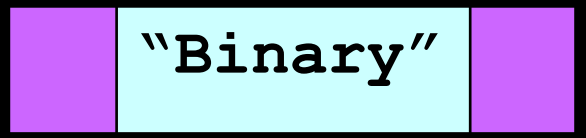
tree[0]



tree[1]



tree[2]

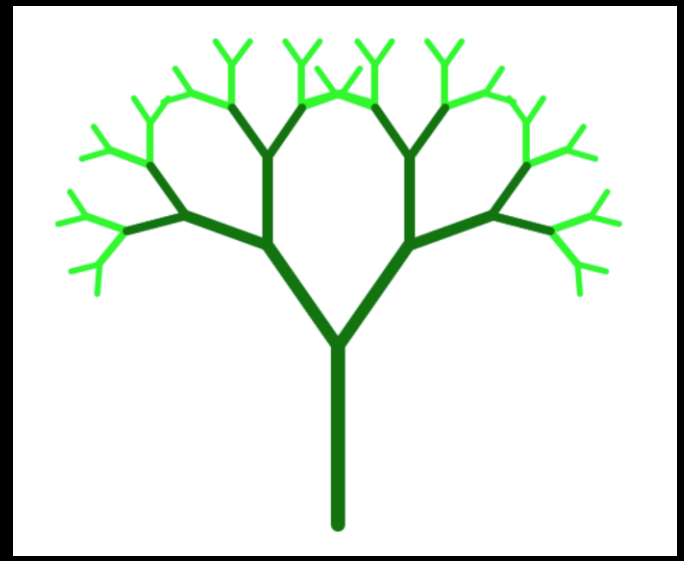


tree[3]

tree[4]

Binary Tree using OOP (with linked list)

LT14c



Tree Class

- tree: Array with N nodes that are linked together.

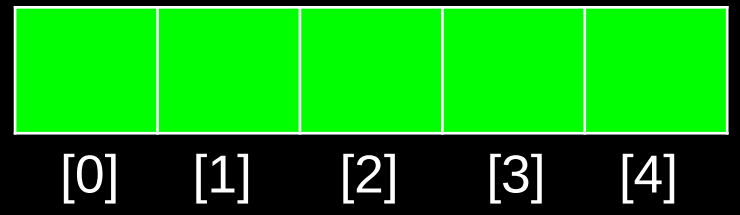
Root ← -1

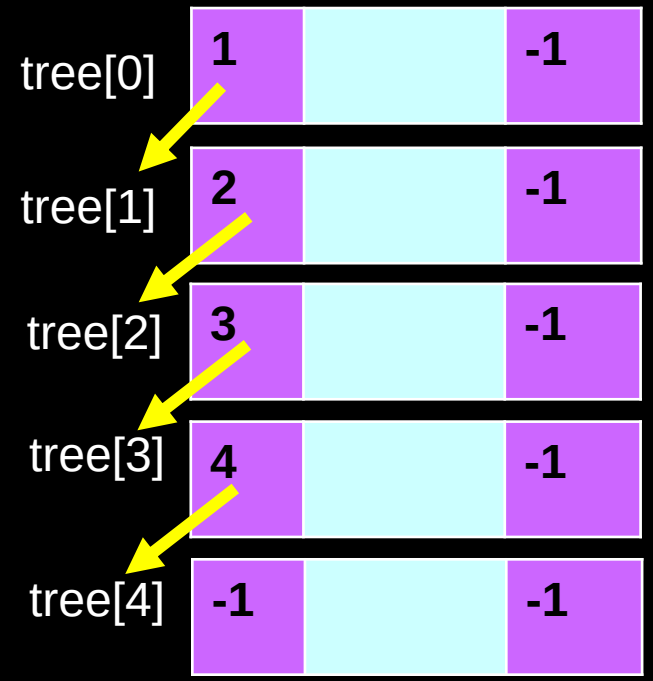
NextFreeChild ← 0

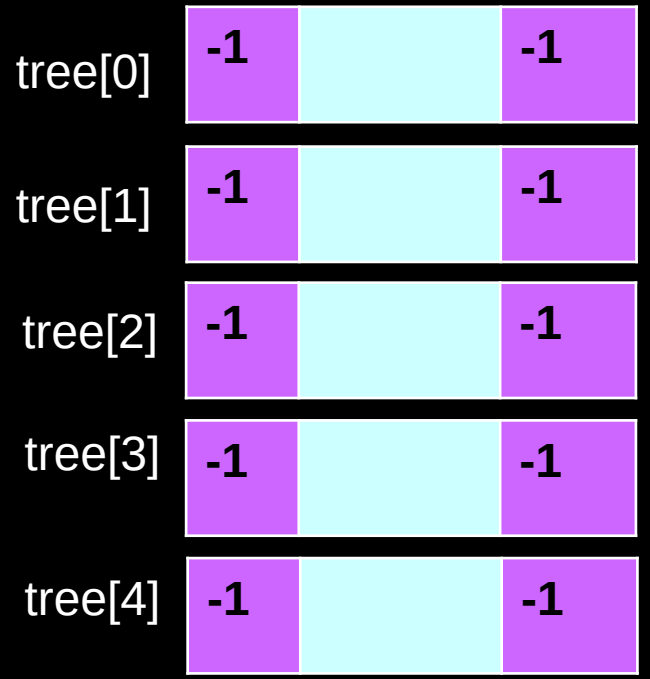
Tree Class

```
Tree : ARRAY OF N Nodes
Root: INTEGER
NextFreeChild: INTEGER
```

```
Constructor()
add(newItem)
inOrder()
display()
```





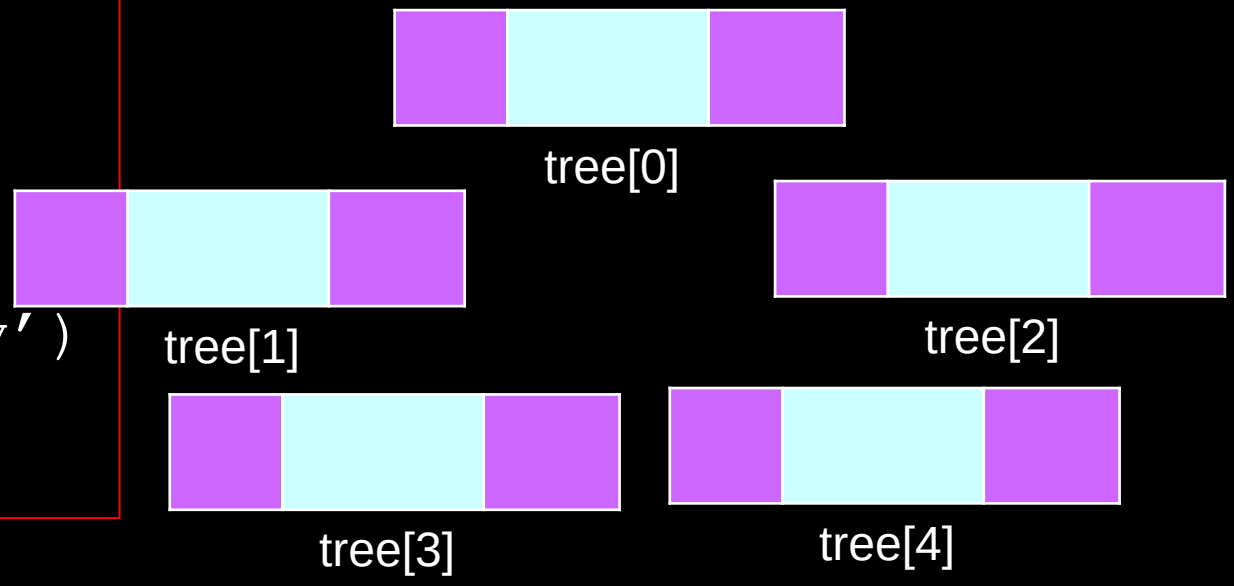


Binary (Search) Tree – Adding Nodes to Tree

Root ← ___

NextFreeChild ← __

```
>> t1= Tree()
>> t1.add('I')
>> t1.add('Am')
>> t1.add('a')
>> t1.add('Binary')
>> t1.add('Tree')
```

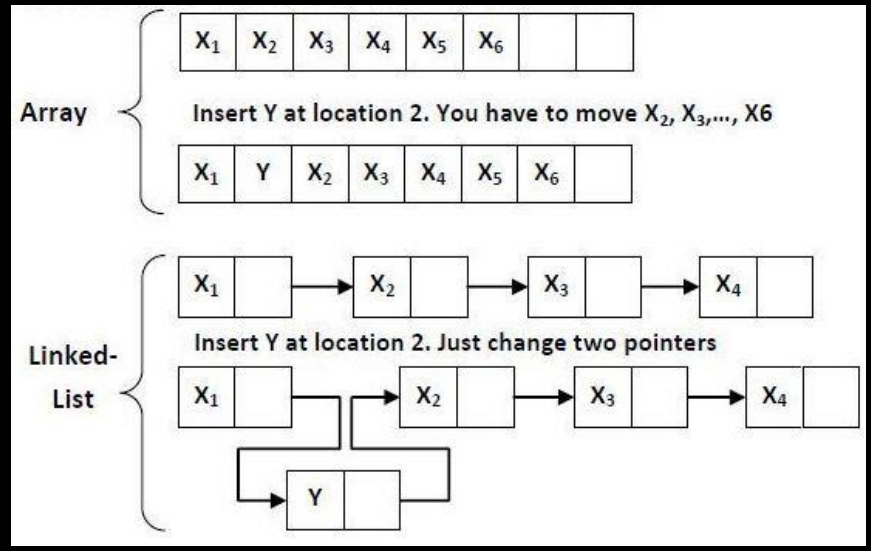


tree

tree[0]	1	'I'	2
tree[1]	-1	'AM'	3
tree[2]	4	'a'	-1
tree[3]	-1	'Bin	-1
tree[4]	-1	'Tre	-1

Why learn Linked List?

- Advantages:
- Insertion/deletion operations on linked lists are less costly and easier as compared to lists/arrays
- Dynamic data structure



- Drawbacks:
- Additional memory needed to store pointer
- Does not provide random access to elements (i.e. need to traverse through the linked list)