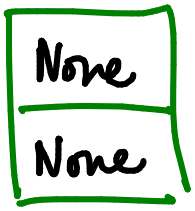
delete(v)

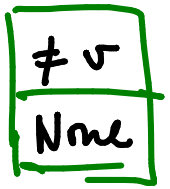
Case 1:

Do nothing

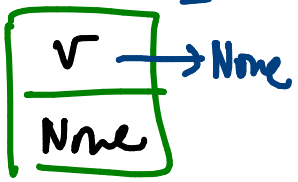


Case 2:

Do Nothing

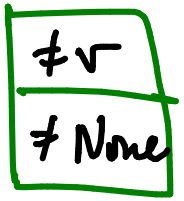


Case 3 [v] ↦ []

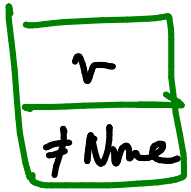


Case 4:

Recursive

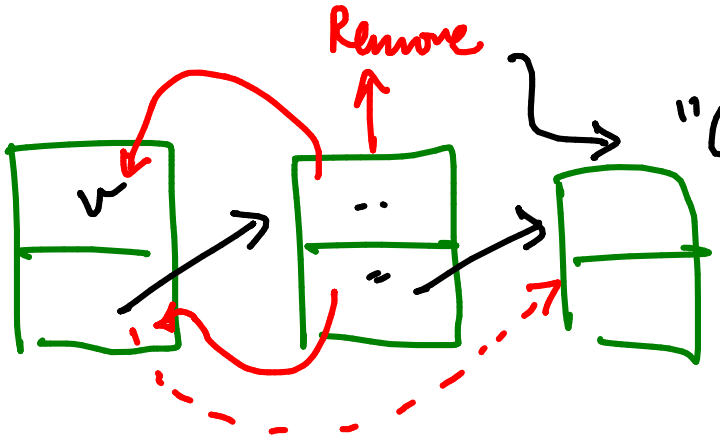


Case 5



Case 5:

self ⇒



"Garbage"

└ reclaimed by "garbage collection"

```
def delete(self, v):
    if self.isempty():          # Case 1
        return

    if self.next == None:
        if self.value == v:
            self.value = None   # Case 3
        return                  # Cases 2 & 3

    if self.value == v:         # Case 4
        self.value = self.next.value
        self.next = self.next.next
        return
    self.next.delete(v)         # Case 5
```

next is not None

Exercise: Replace this by a loop

Insert, append, delete

l[i] def valueatpos(self,i):

=

l[i:j]

...

find(v)

__init__ Constructor

Other special functions

__str__ Invoked implicitly by str(..)

Should return string

Exercise: Print a list as [v1,v2,...,vn]

Other special functions

__add__ , __mult__ +, *

Object o with __add__(v) defined

o+v ⇒ o.__add__(v)

self.value + v ✗

o1 + o2

o1.__add__(o2)

o1.value + o2

o1.value.__add__(o2)

v + self.value ✓

o1 + o2

o1.__add__(o2)

o2 + o1.value

o2.__add__(o1.value)

o2.value + o2.value

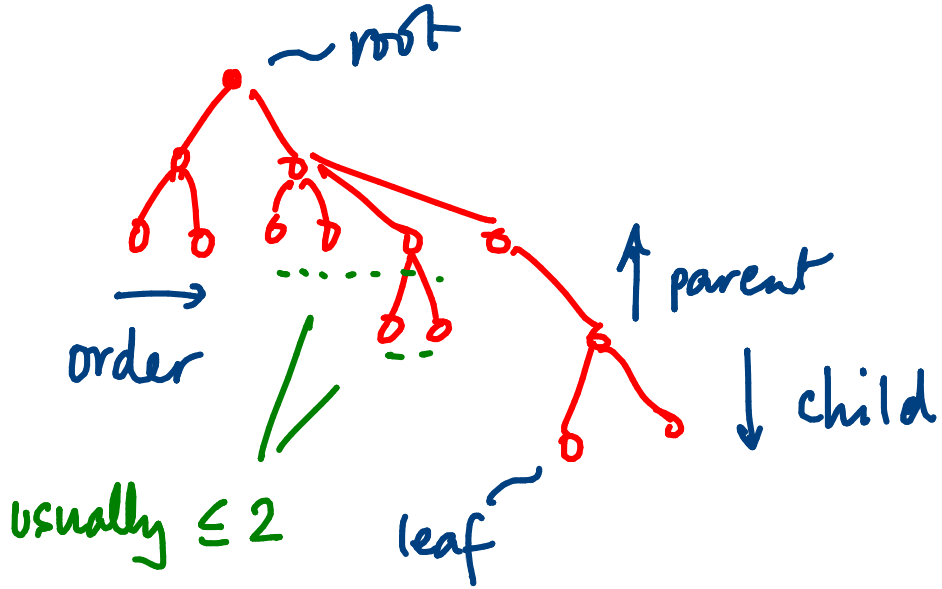
Similarly for comparisons

--eq--, --lt-- etc

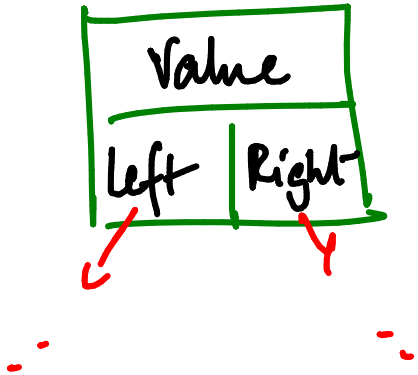
o1 == o2

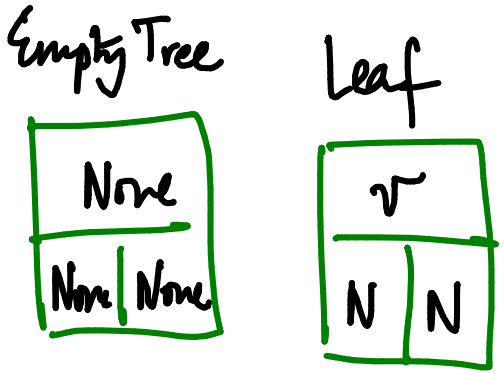
o1 < o2

From lists to trees



Binary Trees



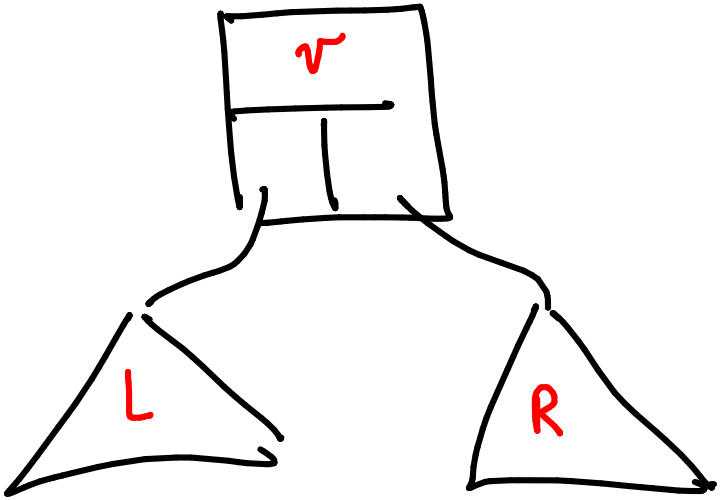



```
class TNode:
    def __init__(self, initval = None):
        self.value = initval
        self.left = None
        self.right = None
```

isEmpty()

isleaf()

Search Trees (over integers)



all values in L < v

all values in R > v

No duplicate values

find(v)

insert(v) - if v ∉ tree, add it

delete(v) - if v ∈ tree, remove it

```
def find (self, v):
    if self.isEmpty():
        return (False)
    if self.value == v:
        return (True)
    if v < self.value and self.left:
        return (self.left.find(v))
    elif: v > self.value and self.right:
        return (self.right.find(v))
    else:
        return (False)
```

Can also follow path iteratively

None is False

```
def insert(self, v)        # Try to find v
                           # If False, insert at failure point
    if self.isEmpty():
        self.value = v
        return
    if self.value == v:
        return
    if v < self.value:
        if self.left:
            self.left.insert(v)
        else:
            self.left = TNode(v)
    if v > self.value:
        if self.right:
            self.right.insert(v)
        else:
            self.right = TNode(v)
```

```
def delete(self, v):
    ↓
def deletemax(self):
    Locate rightmost value
    in tree
    Delete it & return it
```