1. Data definition for binary tree of floating point values.

4. find\_nearest (array or linked list) find nearest value to specified value.

find\_nearest(lst1, value)

def find\_nearest(lst,value):

if lst.lst == None:

raise IndexError()

else:

result = []

for i in range(length(lst)):

test = lst.lst[i] - value

result.append(abs(test))

return value - min(result)

def index\_of\_smallest(list2):

if len(list2) == 0:

return -1

smallest = 0

for i in range(1,len(list2)): #you said smallest is zero so start checking at 1

if list2[i] < list2[smallest]: #list2[i] < list2[0]

smallest = i

return smallest

class BSTNode:

def \_\_init\_\_(self, value, left, right):

self.value = value

self.left = left

self.right = right

def \_\_eq\_\_(self, other):

return ((type(other) == BSTNode)

and self.value == other.value

and self.left == other.left

and self.right == other.right

)

def \_\_repr\_\_(self):

return ("BSTNode({!r}, {!r}, {!r})".format(self.value, self.left, self.right))

#BinarySearchTree is of:

# -BSTNode

# -function comes\_before

class BinarySearchTree:

def \_\_init\_\_(self, comes\_before, node=None):

self.node = node

self.comes\_before = comes\_before

def \_\_eq\_\_(self, other):

return ((type(other) == BinarySearchTree)

8. find\_sum\_some. Finds sum of the smallest "n" values in a binary search tree.

Assume there is a function "count" that finds the amount of values in a bst.

If not enough values, raise indexerror()

find\_sum\_some(bst, n):

def sum\_some(tree,n):

if(count(tree)<n):

raise IndexError

else:

helper(tree,n,n)

def helper(tree,n,count,sum=0):

if(tree.left==None and count!=0):

sum+=tree.value

count-=1

if(tree.right!==None):

helper(tree.right,n,count,sum)

else:

if(tree.left!=None and count!=0):

return helper(tree.left,n,count,sum)

elif(count==0):

return sum

and self.node == other.node

and self.comes\_before == other.comes\_before

)

def \_\_repr\_\_(self):

return ("BinarySearchTree({!r}, {!r})".format(self.node, self.comes\_before))

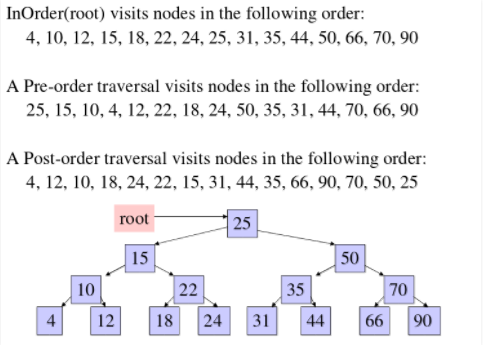
2. Describe order for binary search tree.

Binary search tree is a tree where nodes store a key. At your current node, the left child key is less than your current node,

And the right child key is larger than the current node.

everything to the left of the tree is less than the root value and everything to the right is greater

3. Given tree find preorder, inorder, postorder

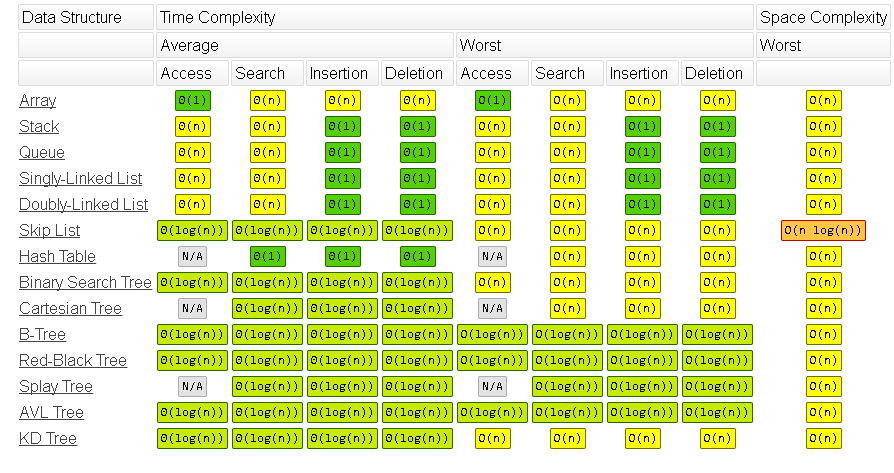


5. What is an iterator? Example of why its useful

Iterator is the encapsulation of a partially completed traversal.

6. Worst case time complexity for enqueue, dequeue, peek for an unordered list and seeking for highest-priority elements

Best case time complexity for a better design (prob heap)



enque: O(1)

dequeue: O(n)

peek: O(n)

seeking: O(n)

for binary heap: it's O(logn) for all except peek likely o(l)

7. takes 2.3 milliseconds to add to front of 1000 size list. how long does it to add to list of size 5 million?

2.3/1000 = x/5mill = 11500 🡪 11.5 secs