

```

1 from queue import LinkedQueue
2
3 class BinarySearchTree:
4     class _Node:
5         __slots__ = '_element', '_left', '_right'
6
7         def __init__(self, element, left=None, right=None):
8             self._element = element
9             self._left = left
10            self._right = right
11
12    def __init__(self):
13        self._root = None
14        self._size = 0
15
16    def insert(self, e):
17        troot = self._root
18        ttroot = None
19        while troot:
20            ttroot = troot
21            if e < troot._element:
22                troot = troot._left
23            elif e > troot._element:
24                troot = troot._right
25        node = self._Node(e)
26        if self._root:
27            if e < ttroot._element:
28                ttroot._left = node
29            else:
30                ttroot._right = node
31        else:
32            self._root = node
33
34    def recurinsert(self, troot, e):
35        if troot == None:
36            node = self._Node(e)
37            return node
38
39        if e < troot._element:
40            troot._left = self.recurinsert(troot._left, e)
41        elif e > troot._element:
42            troot._right = self.recurinsert(troot._right,
43            e)
44
45        return troot
46

```

```
47
48     def search(self, k):
49         troot = self._root
50         while troot:
51             if k < troot._element:
52                 troot = troot._left
53             elif k > troot._element:
54                 troot = troot._right
55             else:
56                 return True
57         return False
58
59     def levelorder(self):
60         Q = LinkedQueue()
61         t = self._root
62         print(t._element, end='--')
63         Q.enqueue(t)
64
65         while not Q.is_empty():
66             t = Q.dequeue()
67             if t._left:
68                 print(t._left._element, end='--')
69                 Q.enqueue(t._left)
70             if t._right:
71                 print(t._right._element, end='--')
72                 Q.enqueue(t._right)
73
74     def inorder(self, troot):
75         if troot:
76             self.inorder(troot._left)
77             print(troot._element, end='--')
78             self.inorder(troot._right)
79
80     def preorder(self, troot):
81         if troot:
82             print(troot._element, end='--')
83             self.preorder(troot._left)
84             self.preorder(troot._right)
85
86     def postorder(self, troot):
87         if troot:
88             self.postorder(troot._left)
89             self.postorder(troot._right)
90             print(troot._element, end='--')
91
92
93 B = BinarySearchTree()
```

```
94 B._root = B.recurinsert(None,70)
95 B.recurinsert(B._root,30)
96 B.recurinsert(B._root,90)
97 B.recurinsert(B._root,40)
98 B.recurinsert(B._root,50)
99 B.recurinsert(B._root,110)
100 B.inorder(B._root)
101 print()
102 print(B.search(25))
103
```