```
class Node:
    def __init__(self, is_leaf=False):
        self.keys = []
        self.children = []
        self.is_leaf = is_leaf


class BTree:
    def __init__(self, order):
        self.root = Node(is_leaf=True)
        self.order = order

    def insert(self, key):
        root = self.root
        if len(root.keys) == (2 * self.order) - 1:
            new_root = Node()
            self.root = new_root
            new_root.children.append(root)
            self.split_child(new_root, 0, root)
            self._insert_non_full(new_root, key)
        else:
            self._insert_non_full(root, key)

    def _insert_non_full(self, x, key):
        i = len(x.keys) - 1
        if x.is_leaf:
            x.keys.append(0)
            while i >= 0 and key < x.keys[i]:
                x.keys[i + 1] = x.keys[i]
                i -= 1
            x.keys[i + 1] = key
        else:
            while i >= 0 and key < x.keys[i]:
                i -= 1
            i += 1
            if len(x.children[i].keys) == (2 * self.order) - 1:
                    self.split_child(x, i, x.children[i])
                if key > x.keys[i]:
                    i += 1
             self._insert_non_full(x.children[i], key)

    def split_child(self, x, i, y):
        z = Node(is_leaf=y.is_leaf)
        x.children.insert(i + 1, z)
        x.keys.insert(i, y.keys[self.order - 1])
        z.keys = y.keys[self.order:]
        y.keys = y.keys[:self.order - 1]
        if not y.is_leaf:
            z.children = y.children[self.order:]
            y.children = y.children[:self.order]

    def search(self, key, x=None):
        if x is None:
            x = self.root
        i = 0
         hil i l ( k ) d k k [i]
        while i < len(x.keys) and key > x.keys[i]:
            i += 1
        if i < len(x.keys) and key == x.keys[i]:
            return (x, i)
        elif x.is_leaf:
            return None
        else:
             return self.search(key, x.children[i])
    def print_tree(self, x=None, level=0):
        if x is None:
            x = self.root
        print("Level ", level, ":", end=" ")
        for key in x.keys:
            print(key, end=" ")
        print()
        if not x.is_leaf:
            for child in x.children:
                self.print_tree(child, level + 1)


# Example usage:
btree = BTree(order=3)
keys = [10, 20, 30, 40, 50, 60, 70, 80, 90]
for key in keys:
    btree.insert(key)

btree.print_tree()

key_to_search = 60
result = btree.search(key_to_search)
if result:
    print(f"Key {key_to_search} found in the tree.")
else:
    print(f"Key {key_to_search} not found in the tree.")
```

```
Level 0 : 30 60
Level 1 : 10 20
Level 1 : 40 50
Level 1 : 70 80 90
Key 60 found in the tree.
```