insert(74)

k = 2

k* = 2
insert(74) -> Leaf Split

- Old leaf: 11 15 34 56
- New leaf: 73 74 83

$k = 2$
$k^* = 2$

↑ old leaf  ↑ new leaf
Summary: insert(74) -> Leaf Split

before:

after:
BTree.insert(key, value)

Node.insert(key, value):
    AbstractNode subtree = this.choose_subtree(key);
    (newChildNode, newChildPivot) = subtree.insert(key, value);

    //find subtree to follow
    //route insert-operation to subtree
Node.insert(key, value):

AbstractNode subtree = this.choose_subtree(key);
(newChildNode, newChildPivot) = subtree.insert(key, value);
If newChildNode != null:
   If this.keys ≥ 2k:
      (newNode, newNodePivot) = this.split();
      If newChildPivot < newNodePivot:
         this.insertEntry(newChildNode, newChildPivot);
      Else:
         newNode.insertEntry(newChildNode, newChildPivot);
      return (newNode, newNodePivot);
   Else:
      this.insertEntry(newChildNode, newChildPivot);
      return (null, null);
Else:
   return (null, null);
BTree.insert(key, value)

Leaf.insert (key, value):  
If this.keys ≥ 2k*:  
(newLeaf, newLeafPivot) = split();  
If key < newLeafPivot:  
    this.insertEntry(key, value);  
Else:  
    newLeaf.insertEntry(key, value);  
return (newLeaf, newLeafPivot);  
Else:  
    this.insert(key, value);  
return (null, null);  

//check overflow condition  
//split this Leaf  
//check into which Leaf to insert  
//insert into this (old) Leaf  
//i.e., key ≥ newLeafPivot  
//insert into newly created Leaf  
//return split information to parent  
//no split necessary for this Leaf  
//insert into newly created Leaf  
//signal no split to parent
delete(74)

before:

after:
delete(73) -> merge()

before:

after:
delete() and merge()

delete():

=inverse of insert()

first: find_key()

then: delete entry from leave

merge():

=inverse of split()

first: merge two nodes into one

then: remove one pivot from parent