A Search

def tree_contains(T, x):
    """True iff x is a label in T."""
    if x == T.label:
        return True
    else:
        for c in T:
            if tree_contains(c, x):
                return True
    return False

• This particular definition of trees lends itself to Noetherian induction
  with no explicit base case.

def tree_contains(T, x):
    """True iff x is a label in T."""
    return x == T.label or \n        any(map(lambda C: tree_contains(C, x),
                 T))

Printing Trees

• The __str__ method lends itself to recursion:

def tree_to_list_preorder(T):
    """The list of all labels in T, listing the labels of trees before those of their children, and listing their children left to right (preorder)."
    return sum(map(tree_to_list_preorder, T), (T.label,))
Search Tree Class

- To work on search trees, it is useful to have a few more methods on trees:

```python
class BinTree(Tree):
    @property
    def is_empty(self):
        """This tree contains no labels or children."""
        return self[0] == self[1] == None
    def left(self):
        return self[0]
    def right(self):
        return self[1]

    empty_tree = ...
```

Tree Search Program

```python
def tree_find(T, x):
    """True iff x is a label in set T, represented as a search tree. That is, T
    (a) Is an empty tree if T.is_empty(), or
    (b) Has two children, T.left and T.right, both search trees, and all labels in T.left are less than T.label, and all labels in T.right are greater than T.label."""
    if T.is_empty:
        return False
    if x == T.label:
        return True
    if x < T.label:
        return tree_find(T.left, x)
    else:
        return tree_find(T.right, x)
```

Iterative Tree Search Program

```python
def tree_find(T, x):
    """True iff x is a label in set T, represented as a search tree. That is, T
    (a) Is an empty tree if T.is_empty(), or
    (b) Has two children, T.left and T.right, both search trees, and all labels in T.left are less than T.label, and all labels in T.right are greater than T.label."""
    while not T.is_empty:
        if x == T.label:
            return True
        elif x < T.label:
            T = T.left
        else:
            T = T.right
    return False
```

Timing

- How long does the `tree_find` program (search binary tree) take in the worst case,
  1. As a function of \( H \), the height of the tree? \( A: \Theta(H) \)
  2. As a function of \( N \), the number of keys in the tree? \( A: \Theta(N) \)
  3. As a function of \( H \) if the tree is as shallow as possible for the amount of data? \( A: \Theta(H) \)
  4. As a function of \( N \) if the tree is as shallow as possible for the amount of data? \( A: \Theta(H) \)