Practice 5. Search & Tree Traversals

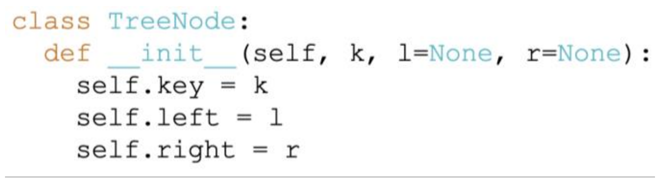
[CSE2010] Data Structures

Department of Data Science

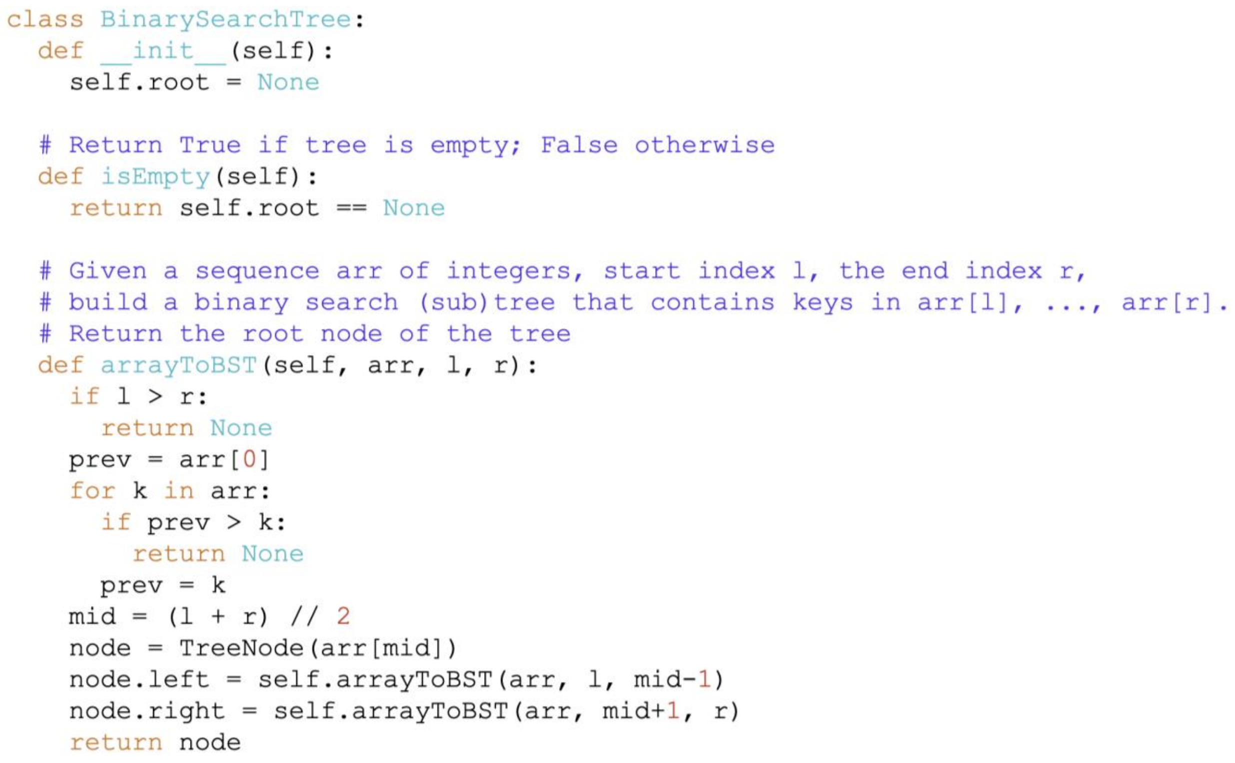
Practice 4. Build BST

- First, define a node of a BST with members:
  - key
  - left
  - right

- For building a balanced BST, **recursively** partition a given input sequence into two subarrays with the same size.



```
class TreeNode:
  def __init__(self, k, l=None, r=None):
    self.key = k
    self.left = l
    self.right = r
```



```
class BinarySearchTree:
  def __init__(self):
    self.root = None

  # Return True if tree is empty; False otherwise
  def isEmpty(self):
    return self.root == None

  # Given a sequence arr of integers, start index l, the end index r,
  # build a binary search (sub)tree that contains keys in arr[l], ..., arr[r].
  # Return the root node of the tree
  def arrayToBST(self, arr, l, r):
    if l > r:
      return None
    prev = arr[0]
    for k in arr:
      if prev > k:
        return None
      prev = k
    mid = (l + r) // 2
    node = TreeNode(arr[mid])
    node.left = self.arrayToBST(arr, l, mid-1)
    node.right = self.arrayToBST(arr, mid+1, r)
    return node
```

Practice 4. Find Min/Max

- FindMin
    - Keep moving to the left until we encounter the node whose left child is NULL
- FindMax
    - Keep moving to the right until we encounter the node whose right child is NULL

```
# Return the node with the minimum value
def findMin(self):
  if self.isEmpty():
    return None
  p = self.root
  while p.left:
    p = p.left
  return p

# Return the node with the maximum value
def findMax(self):
  if self.isEmpty():
    return None
  p = self.root
  while p.right:
    p = p.right
  return p
```

Overview

- Implement search and tree traversals in a **binary search tree** (BST).
- Functions
    1. Search for a given integer in the binary search tree: O(n) time
    2. Inorder traversal: O(n) time
    3. Preorder traversal: O(n) time
    4. Postorder traversal: O(n) time

Input of Tree Traversals

- Each line represents a single operation.
  1. **S<space>[int]**
     - If the integer exists, write that integer to output file.
     - Otherwise, immediately terminate the program with an error.
  2. **N**
     - Write the values of every node visited in inorder traversal.
  3. **R**
     - Write the values of every node visited in preorder traversal.
  4. **O**
     - Write the values of every node visited in postorder traversal.
- You can start from the last practice code that builds a BST.

Input and Output

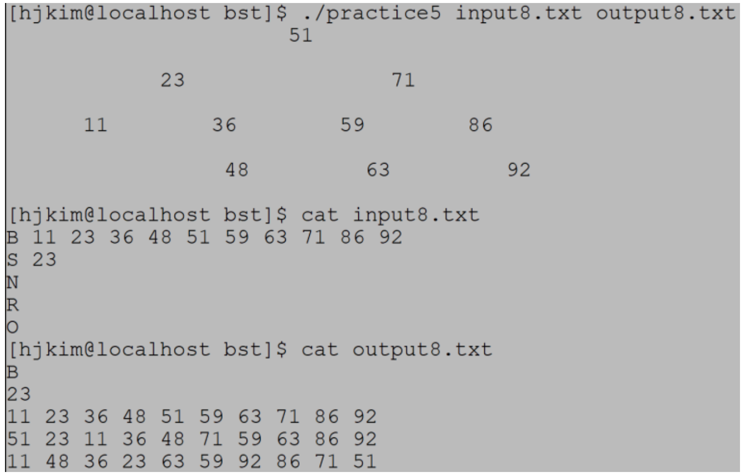
- Each line in output file represents the result of the corresponding line of the input file.

- Input File

```
B 11 23 36 48 51 59 63 71 86 92
S 23
N
R
O
```

Output File

```
B
23
11 23 36 48 51 59 63 71 86 92
51 23 11 36 48 71 59 63 86 92
11 48 36 23 63 59 92 86 71 51
```



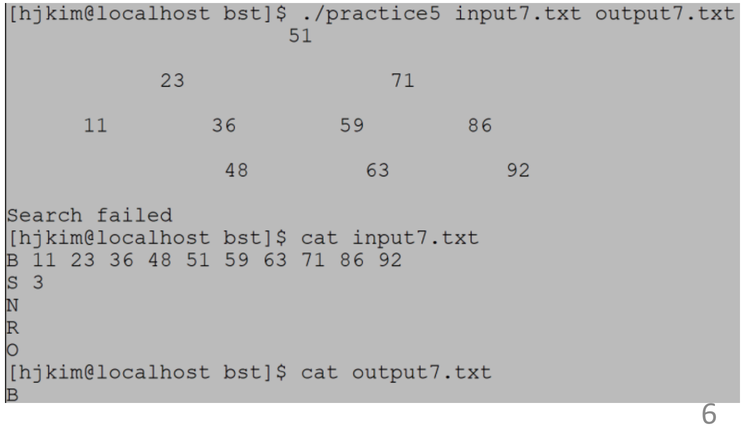
```
[hjkim@localhost bst]$ ./practice5 input8.txt output8.txt
                      51

            23                71

      11        36        59        86

                 48         63          92

[hjkim@localhost bst]$ cat input8.txt
B 11 23 36 48 51 59 63 71 86 92
S 23
N
R
O
[hjkim@localhost bst]$ cat output8.txt
B
23
11 23 36 48 51 59 63 71 86 92
51 23 11 36 48 71 59 63 86 92
11 48 36 23 63 59 92 86 71 51
```

- Input File

```
B 11 23 36 48 51 59 63 71 86 92
S 3
N
R
O
```

Output File

```
B
```



```
[hjkim@localhost bst]$ ./practice5 input7.txt output7.txt
                      51

            23                71

      11        36        59        86

                 48         63          92

Search failed
[hjkim@localhost bst]$ cat input7.txt
B 11 23 36 48 51 59 63 71 86 92
S 3
N
R
O
[hjkim@localhost bst]$ cat output7.txt
B
```

Hints

- Implement recursive functions for search and traversals.
  - Assume that you are given **the root of a subtree** in the definition of a function that recursively traverse down the tree.
  - Call that function with **the root of the tree**.

- Make sure you defined a **base case** as well as a **recursive case** in a recursive function.
  - For the details on recursive functions, refer to the lecture slides on “abstract_data_type”.

```
int Fact(int n){
    if(n == 0) // Base case
        return 1;// Terminate
    else // Recursive case
        return n * Fact(n-1);
}
```

Submission Guideline

- Submission: **source code, makefile**
  - Where: Assignments in LMS
  - Deadline: **23:59, April. 10th (Sunday)**
- Extra points
  - **From April 11th (Monday)**
  - Share your **code, input & output** on Open Board in LMS.
  - Review classmates' code. Give questions or comments on his/her post.
  - Answer others' questions on your post.
  - Title: [Practice5]StudentID
    - e.g., [Practice5]2021000000

Next Practice

- **April 13th (Wednesday)**
- Main operations supported by a BST
    - Insertion
    - Deletion