Look-up tables

CMPT 145

Table ADT

- An application of Binary Search Trees
- A Table provides search, insertion, and deletion of keyed data
- A `key` is a value that is unique to the data being stored, e.g.,
  - Student number
  - International Standard Book Number (ISBN)
  - Social Insurance Number
- A key tells us which data we're looking for, but the data may be much more than the key, e.g.,
  - Student record, employee record
  - Data about a book, or the entire contents
  - A health record, or a taxation record

Table ADT

- An application of Binary Search Trees
- A Table provides search, insertion, and deletion of keyed data
- A `key` is a value that is unique to the data being stored, e.g.,
  - Student number
  - International Standard Book Number (ISBN)
  - Social Insurance Number
- A key tells us which data we're looking for, but the data may be much more than the key, e.g.,
  - Student record, employee record
  - Data about a book, or the entire contents
  - A health record, or a taxation record

KVTreeNode Class

```python
class KVTreeNode(object):
    def __init__(self, key, value, left=None, right=None):
        """
        Create a new KVTreeNode for the given data.
        Pre-conditions:
            key:    A key used to identify the node
            value:  Any data value to be stored in the KVTreeNode
            left:   Another KVTreeNode (or None, by default)
            right:  Another KVTreeNode (or None, by default)
        """
        self.value = value
        self.left = left
        self.right = right
        self.key = key
```

Table ADT

- Purpose:
    - Manage a keyed data
- Implementation:
    - Data:
        - keys and values
    - Essential Operations:
        - Create empty table
        - Query if table is empty
        - Query size of table
        - Insert key, value into table
        - Delete key, value from table
        - Retrieve the data for a given key

Table Class: An Object Oriented ADT

```
class Table(object):
    def __init__(self):
        self.__root = None
        self.__size = 0
```

Table Class: An Object Oriented ADT

```
    def retrieve(self, key):
        """
        Return the value associated with the given key.
        Preconditions:
            :param key: a key
        Postconditions:
            none
        Return
            :return: True, value if the key appears in the table
                     False, None otherwise
        """
```

Table Class: An Object Oriented ADT

```
        def retrieve_prim(tnode):
            if tnode is None:
                return False, None
            else:
                ckey = tnode.key
                if ckey == key:
                    return True, tnode.value
                elif key < ckey:
                    return retrieve_prim(tnode.left)
                else:
                    return retrieve_prim(tnode.right)

        return retrieve_prim(self.__root)
```