Lab 7

Minimum Priority Queue

Objective

Implement Minimum Priority Queue ADT by completing **min_pq.py file** included in the zipped starter code package. Complete the class definition of MinPQ class. You can copy some of your code, such as enlarge, and shrink functions, from previous labs and projects. Do not forget to implement shift_up and shift_down operations. Review the lecture slides on Min Heap. You are free to create any helper functions.

Implementation

Download a starter file, min_pq.py, from Polylearn.

Implement MinPQ class with following attributes and methods:

```
class MinPQ:
    """Minimum Priority Queue
    Attributes:
        capacity (int): the capacity of the queue. The default capacity is 2, but will be
increased automatically.
        num_items (int): the number of items in the queue. This also points to the position
where a new item will be addded.
        arr (list): an array which contains the items in the queue.
    """
    def __init__(self, arr=None):
        """initializes an object of MinPQ
        If the arr is None (default), it initializes its capacity as 2,
         and creates an array: [None] * self.capacity. The num_items shall be initialized as
0.
        Otherwise, self.arr = self.heapify(arr).  The arr shall be transformed into a min heap
with heapify method.
```

```
        In this case, the capacity and num_items are set to the size of the arr, which is
len(arr).

        Args:
            arr (list): the default value is None
        """

    def heapify(self, arr):
        """initialize the queue with a given array and convert the array into a min heap
        Args:
            arr (list): an array
        Returns:
            None : it returns nothing
        """

    def insert(self, item):
        """inserts an item to the queue. Before inserting an item it checks if the array is full,
if so, it enlarges the array by doubling the capacity.
        Args:
            item (any): an item to be inserted to the queue. It is of any data type.
        Returns:
            None : it returns nothing
        """

    def del_min(self):
        """deletes the minimum item in the queue. After the deletion and just before
returning the removed item, it checks if the array needs to be shrinked. If so, it
downsizes the array by halving the capacity:
        if self.capacity > 2 and self.num_items and self.capacity >= self.num_items * 4:
            self.shrink()

        YOU ARE NOT ALLOWED TO USE PYTHON LIST's BUILT-IN FUNCTIONS.

        Returns:
            any : it returns the minimum item, which has just been deleted
        Raises:
            IndexError : Raises IndexError when the queue is empty
        """
```

```
def min(self):
    """returns the minimum item in the queue without deleting the item
    Returns:
        any : it returns the minimum item
    Raises:
        IndexError : Raises IndexError when the queue is empty
    """

def is_empty(self):
    """checks if the queue is empty
    Returns:
        bool : True if empty, False otherwise.
    """

def size(self):
    """returns the number of items in the queue
    Returns:
        int : it returns the number of items in the queue
    """

def shift_up(self, idx):
    """shifts up an item in the queue using tail recursion.
    Use only < operator to compare two items: do not use <=, >, >=.

    Args:
        idx (int): the index of the item to be shifted up in the array.
    Returns:
        None : it returns nothing
    """

def shift_down(self, idx):
    """shifts down an item in the queue using tail recursion.
    Use only < operator to compare two items: do not use <=, >, >=.
    YOU NEED TO DETERMINE WHERE THE END OF THE HEAP IS.
    YOU CAN USE self.num_items FOR DOING SO.
    Args:
        idx (int): the index of the item to be shifted down in the array.
    Returns:
        None : it returns nothing
    """

def enlarge(self):
    """enlarges the array.
    """

def shrink(self):
    """shrinks the array.
    """

def index_left(self, idx):
    """returns the index of the left child
    Args:
        idx (int): the index of the node
    Returns:
        int : it returns the index of the left child
    """

def index_right(self, idx):
    """returns the index of the right child
    Args:
        idx (int): the index of the node
    Returns:
        int : it returns the index of the right child
    """

def index_parent(self, idx):
    """returns the index of the parent
    Args:
        idx (int): the index of the node
    Returns:
        int : it returns the index of the parent
    """

def index_minchild(self, left, right, end):
    """returns the index of the min child
    Args:
        left (int): the index of the left child
        right (int): the index of the right child
        end (int): the end index of the heap
    Returns:
        int : it returns the index of the min child
    """
```

Submission

You must submit the following files to the grader and polylearn:

- min_pq.py
    - The functions specified and any helper functions necessary.
- min_pq_tests.py
    - Include test cases you used in developing your programs.