DSC 40B - Discussion 03

Problem 1.

In this problem we will implement two methods on treaps.

a) Implement a `.min(x)` method which takes in a `TreapNode` `x` and returns the `TreapNode` in the subtree rooted at `x` which has the smallest key.

Solution:

```
def min(self, x):
    parent = x.parent
    while x is not None:
        parent = x
        x = x.left
    return parent
```

b) Implement a `.successor(x)` method which takes in a `TreapNode` `x` and returns the `TreapNode` with the next largest key.

Solution:

```
def successor(self, x: TreapNode):
    """Find a node's successor (the next largest node by key).

    Parameters
    ----------
    x : TreapNode
        The node whose successor will be found.

    Returns
    -------
    TreapNode
        The successor of x.

    Raises
    ------
    ValueError
        If x has no successor.

    Example
    -------
    >>> treap = Treap()
    >>> x = treap.insert(3, 10)
    >>> treap.insert(6, 2)
    TreapNode(key=6, priority=2)
    >>> treap.insert(5, 12)
    TreapNode(key=5, priority=12)
    >>> treap.successor(x)
    TreapNode(key=5, priority=12)
```

```
    """
    if x.right is not None:
        return self._min_in_subtree(x.right)
    else:
        # walk up the tree until you find a node that is a left child
        while x is not None and x is x.parent.right:
            x = x.parent
        return x.parent

    raise ValueError(f'There is no successor of {x}')
```