class DListNode:

"""

A node in a doubly-linked list.

"""

def \_\_init\_\_(self, data=None, prev=None, next=None):

self.data = data

self.prev = prev

self.next = next

def \_\_repr\_\_(self):

return repr(self.data)

class DoublyLinkedList:

def \_\_init\_\_(self):

"""

Create a new doubly linked list.

Takes O(1) time.

"""

self.head = None

def \_\_repr\_\_(self):

"""

Return a string representation of the list.

Takes O(n) time.

"""

nodes = []

curr = self.head

while curr:

nodes.append(repr(curr))

curr = curr.next

return '[' + ', '.join(nodes) + ']'

def prepend(self, data):

"""

Insert a new element at the beginning of the list.

Takes O(1) time.

"""

new\_head = DListNode(data=data, next=self.head)

if self.head:

self.head.prev = new\_head

self.head = new\_head

def append(self, data):

"""

Insert a new element at the end of the list.

Takes O(n) time.

"""

if not self.head:

self.head = DListNode(data=data)

return

curr = self.head

while curr.next:

curr = curr.next

curr.next = DListNode(data=data, prev=curr)

def find(self, key):

"""

Search for the first element with `data` matching

`key`. Return the element or `None` if not found.

Takes O(n) time.

"""

curr = self.head

while curr and curr.data != key:

curr = curr.next

return curr # Will be None if not found

def remove\_elem(self, node):

"""

Unlink an element from the list.

Takes O(1) time.

"""

if node.prev:

node.prev.next = node.next

if node.next:

node.next.prev = node.prev

if node is self.head:

self.head = node.next

node.prev = None

node.next = None

def remove(self, key):

"""

Remove the first occurrence of `key` in the list.

Takes O(n) time.

"""

elem = self.find(key)

if not elem:

return

self.remove\_elem(elem)

def reverse(self):

"""

Reverse the list in-place.

Takes O(n) time.

"""

curr = self.head

prev\_node = None

while curr:

prev\_node = curr.prev

curr.prev = curr.next

curr.next = prev\_node

curr = curr.prev

self.head = prev\_node.prev