Algoritmos T.P. N° 4 – TAD Pilas y Colas - Juan Cruz Ambrosini

mystack:

from algo1 import \*

from mylinkedlist import \*

from MyArray import length

"""----------------------------------------------------------------------------------"""

def push(list, element):

    add(list, element)

"""----------------------------------------------------------------------------------"""

def pop(list):

    if list.head == None :

        return None

    else:

        popedElement = list.head.value

        list.head = list.head.nextNode

        return popedElement

myqueue:

from mylinkedlist import \*

from algo1 import \*

from MyArray import length

"""----------------------------------------------------------------------------------"""

def enqueue (list, element):

    add(list, element)

"""----------------------------------------------------------------------------------"""

def dequeue(list):

    current\_node = list.head

    if list.head == None:

        return None

    elif( list.head.nextNode == None ):

        dequeued\_element = list.head.value

        list.head = None

        return dequeued\_element

    else:

        while(current\_node.nextNode.nextNode != None):

            current\_node = current\_node.nextNode

        dequeued\_element = current\_node.nextNode.value

        current\_node.nextNode = None

        return dequeued\_element

mypriorityqueue:

from mylinkedlist import \*

from algo1 import \*

from MyArray import length

from random import randint

"""---------------------------------------------------------------------------------------------"""

class PriorityQueue:

    head=None

class PriorityNode:

    value=None

    nextNode=None

    priority=None

"""---------------------------------------------------------------------------------------------"""

def enqueue\_priority(list, element, priority):

    new\_priority\_node = PriorityNode()

    new\_priority\_node.value = element

    new\_priority\_node.priority = priority

    if list.head == None :

        list.head = new\_priority\_node

        return 0

    else:

        new\_priority\_node.nextNode = list.head

        list.head = new\_priority\_node

        return 0

"""---------------------------------------------------------------------------------------------"""

def dequeue\_priority(list):

    current = list.head

    rep\_counter = 0

    counter = 0

    dequeue\_value = 0

    prev\_node = None

    if list.head == None:

        return None

    elif list.head.nextNode == None :

        dequeue\_value = list.head.value

        list.head = None

        return dequeue\_value

    else:

        max\_priority = list.head.priority

        while current != None:

            if max\_priority < current.priority :

                max\_priority = current.priority

            current = current.nextNode

        current = list.head

        while current != None:

            if current.priority == max\_priority:

                rep\_counter = rep\_counter + 1

            current = current.nextNode

        current = list.head

        while current != None:

            if current.priority == max\_priority:

                counter = counter + 1

            if counter == rep\_counter:

                dequeue\_value = current.value

                if prev\_node == None:

                    list.head = current.nextNode

                    break

                else:

                    prev\_node.nextNode = current.nextNode

                    break

            prev\_node = current

            current = current.nextNode

        return dequeue\_value

Texto

El contenido generado por IA puede ser incorrecto.