Principles of Computer Science II

Abstract Data Types

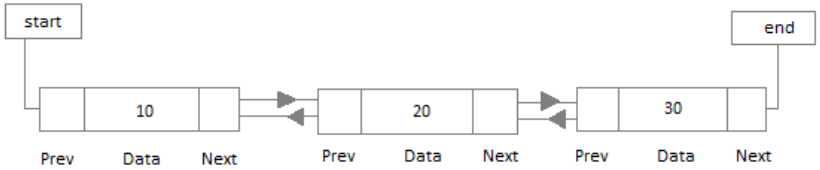
Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 15

Double Linked Lists

In a doubly linked list, each node contains two links the first link points to the previous node and the next link points to the next node in the sequence.



Double Linked List Nodes as Pointers

```
1 class Node(object):
2     def __init__(self, value=None, pointer=None,
          pointerPrev=None):
3         self.value = value
4         self.pointer = pointer
5         self.pointerPrev = pointerPrev
6
7     def getData(self):
8         return self.value
9
10    def getNext(self):
11        return self.pointer
12
13    def setData(self, newdata):
14        self.value = newdata
15
16    def setNext(self, newpointer):
17        self.pointer = newpointer
```

Double Linked List Code: Initialization

```
1 from node import Node
2
3 class DoubleLinkedList(object):
4
5     def __init__(self):
6         self.head = None
7         self.length = 0
8         self.tail = None # this is different from ll lifo
```

Double Linked List: Insertion at the Beginning

1. The first Node is the Head for any Double Linked List.
2. When a new Double Linked List is instantiated, it just has the Head, and Tail both Null.
3. Else, the Head holds the pointer to the first Node of the List.
4. When we want to add any Node at the front, we must make the head point to it.
5. And the Next pointer of the newly added Node, must point to the previous Head, whether it be NULL(in case of new List) or the pointer to the first Node of the List.
6. The previous Head Node is now the second Node of Linked List, because the new Node is added at the front.

Double Linked List: Insertion at the Beginning

```
1     def addFirst(self, value):
2         self.length = 1
3         node = Node(value)
4         self.head = node
5         self.tail = node
```

Double Linked List: Insertion at the End

1. If the Double Linked List is empty then we simply, add the new Node as the Head of the Double Linked List.
2. If the Double Linked List is not empty then we find the last node, and make it' next to the new Node, hence making the new node the last Node.

Double Linked List: Insertion at the Beginning

```
1     def add(self, value):
2         self.length += 1
3         node = Node(value)
4         if self.tail:
5             self.tail.pointer = node
6         self.tail = node
7
8     def addNode(self, value):
9         if not self.head:
10            self.addFirst(value)
11        else:
12            self.add(value)
```

Double Linked List: Printing the elements in reverse order

```
1     def printListReverse(self):
2         node = self.tail
3         while node:
4             print(node.value)
5             node = node.pointerPrev
```

Double Linked List: Deleting a Node

1. We first search the Node with data which we want to delete.
2. If the Node to be deleted is the first node, then simply set the Next pointer of the Head to point to the next element from the Node to be deleted.
3. If the Node is in the middle somewhere, then find the Node before it, and make the Node before it point to the Node next to it.

Double Linked List: Deleting a Node

```
1     def deleteNode(self, index):
2         if not self.head or not self.head.pointer:
3             self.deleteFirst()
4         else:
5             node, prev, i = self.find(index)
6             if i == index and node:
7                 self.length -= 1
8                 if i == 0 or not prev :
9                     self.head = node.pointer
10                else:
11                    prev.pointer = node.pointer
12                if not self.tail == node:
13                    self.tail = prev
14            else:
15                print('Node with index {} not found'.format
                      (index))
```

Double Linked Lists: Testing

```
1 if __name__ == '__main__':
2     ll = LinkedList()
3     for i in range(1, 5):
4         ll.addNode(i)
5     print('The list is:')
6     ll.printList()
7     print('The list after deleting node with index 2:')
8     ll.deleteNode(2)
9     ll.printList()
10    print('The list after adding node with value 15')
11    ll.add(15)
12    ll.printList()
13    print("The list after deleting everything...")
14    for i in range(ll.length-1, -1, -1):
15        ll.deleteNode(i)
16    ll.printList()
```

Circular Linked Lists

In the circular linked list the last node of the list contains the address of the first node and forms a circular chain.