CS218 - Data Structures
FAST NUCES Peshawar Campus
Dr. Nauman (recluze.net)

August 27, 2019

1 Doubly Linked List in Python

Raster images of the notebook 05-doubly-linked-list

Doubly Linked List

A doubly-linked list is quite similar to a "singly-linked" list. Just need to add a `prev` pointer to all the nodes.

In [16]:
```
class Node:
    def __init__(self, data=None):
        self.val = data
        self.next = None
        self.prev = None   # (change)

class Doubly:
    def __init__(self):
        self.head = None
```

Push

The push operation is also the same mostly. The new node only needs to be linked to the second-to-last node.

In [17]:
```
def push(self, val):
    new_node = Node(val)

    # no node currently
    if self.head is None:
        self.head = new_node
        return

    # otherwise, reach the end and then insert
    last = self.head
    while last.next is not None:
        last = last.next

    last.next = new_node
    new_node.prev = last   # (change)

Doubly.push = push   ## We can add functions to classes even after definition
```

In [29]:
```
def __str__(self):
    ret_str = '['
    temp = self.head
    while temp is not None:    # or just  while temp:
#         print(hex(id(temp.prev)), hex(id(temp)), hex(id(temp.next)))

        ret_str += str(temp.val) + ', '
        temp = temp.next

    ret_str = ret_str.rstrip(', ')
    ret_str += ']'
    return ret_str

Doubly.__str__ = __str__
```

Pop

In [30]:
```
def pop(self):
    if self.head is None:
        raise Exception("Cannot pop. No value.")

    # case where there is only one node
    if self.head.next is None:
        val = self.head.val
        self.head = None  # automatic garbage collection
        return val

    # case where there is 2 or more nodes
    # reach the previous to last node
    temp = self.head
    while temp.next is not None:
        prev = temp
        temp = temp.next

    val = temp.val
    prev.next = None
    return val

Doubly.pop = pop
```

In [31]:
```
l = Doubly()
l.push(1)
l.push(2)
l.push(3)

print(l)
print("Pop:", l.pop())
print(l)
```

Insert

In [41]:
```
def insert(self, index, val):
    new_node = Node(val)

    # insertion at index 0 is different
    if index == 0:
        new_node.next = self.head

        if self.head is not None:  # (change)
            self.head.prev = new_node

        self.head = new_node

        return


    # for other indices
    temp = self.head

    counter = 0
    while temp is not None and counter < index:
        prev = temp
        temp = temp.next
        counter += 1
        # print(counter)

    # print("Will insert after: ", prev.val)
    prev.next = new_node
    new_node.prev = prev # (change)

    new_node.next = temp
    if temp is not None:
        temp.prev = new_node # (change)


Doubly.insert = insert
```

In [42]:
```
hex(id(None))
```

Out[42]:
```
'0x106b94f78'
```

In [43]:
```
l = Doubly()
l.push(1)
l.push(3)
l.insert(0, 10)
l.insert(1, 11)
print(l)
l.insert(4, 2324)
print(l)
```

```
[10, 11, 1, 3]
[10, 11, 1, 3, 2324]
```

In [ ]:
```
# Todo: len, remove
```