Data Structures

DLL Deletion

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

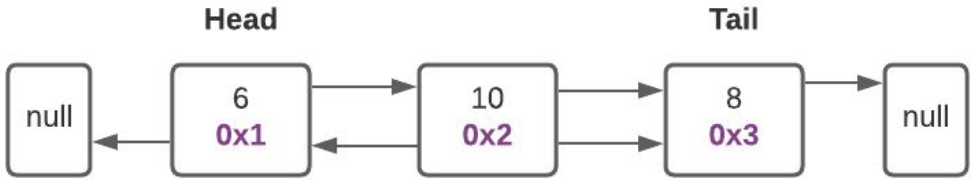
Delete Front

```
def delete_front(self):
    if not self.head:
        return

    next = self.head.next
    self._delete_node(self.head)
    self.head = next

    if self.head:
        self.head.prev = None

    if self.length <= 1:
        self.tail = self.head
```



Delete End

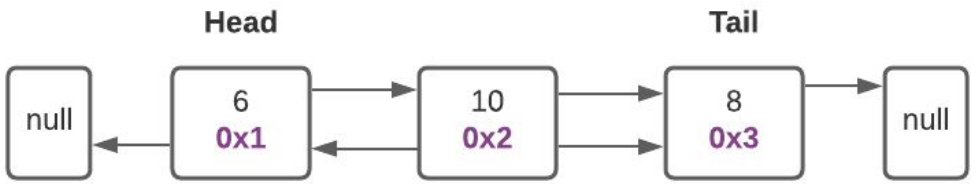
- In SLL, this code is O(n)
- Now, it is O(1)
- Your design choices can create big impact!

```
def delete_last(self):
    if self.length <= 1:
        self.delete_front()
        return

    #previous = self.get_nth(self.length - 1)
    previous = self.tail.prev

    self._delete_node(self.tail)
    self.tail = previous
    self.tail.next = None

    self.debug_verify_data_integrity()
```



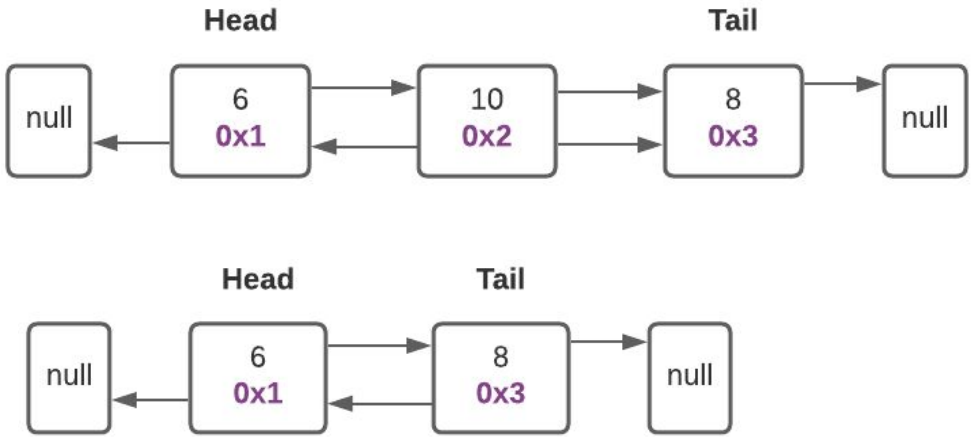
delete_and_link utility

- Previously, we implemented _delete_next_node. Do we need it? No
- Given a node, connect its ‘previous’ and ‘next’ nodes, and then delete it
  - Return the previous node
- Let’s delete node at 0x2 (value 10)

```
def _delete_link_node(self, node):
    if not node:
        return
    is_tail = node == self.tail
    prev = node.prev
    self._link(prev, node.next)
    self._delete_node(node)

    if is_tail:
        self.tail = prev

    return prev
```



Delete node with key

- Now, no need to keep the previous node!

```
def delete_node_with_key(self, key):
    if not self.length:
        return

    if self.head.data == key:
        self.delete_front()
    else:
        cur = self.head
        while cur:
            if cur.data == key:
                self._delete_link_node(cur)
                break
            cur = cur.next
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."