Lösungen zu Blatt-02

Aufgabe 2

Für Reverse:

Python Code (Code gekürzt und von Kommentaren bereinigt):

```
class LinkedList:
    ...
        reversed_list = LinkedList()
        next_item = self.first
        next_item_before = None
        LinkedListItem.num_items = 0
        while next_item is not None:
            new_item = LinkedListItem(next_item.value)
            reversed_list.insert_before(new_item, next_item_before)
            next_item_before, next_item = [new_item, next_item.next]
        self.first, self.last = [reversed_list.first reversed_list.last]
    ...
```

Da wir durch alle Items iterieren müssen, müssen wir n pointern folgen. Alle anderen Operationen sind Konstant und können vernachlässigt werden. Somit benötigen wir im schlechtesten Falle $\Theta(n)$ Blockoperationen, müssen also jeden block in dem das nächste Item liegt, in den Cache laden.

Für Splice:

Python Code:

```
class LinkedList:
    ...
    def splice(self, l1, l2):
        if l1.prev is not None:
            l1.prev.next = l2.next
        if l2.next is not None:
            l2.next.prev = l1.prev
        if self.first == l1:
            self.first = l2.next
        if self.last == l2:
            self.last = None

        l1.prev = l2.next = None
```

```
        part_list = LinkedList()
        part_list.first, part_list.last = [l1, l2]
        return part_list
...
```

Egal wie n gewählt ist, wir müssen stets die gleichen operationen ausführen (wenn wir die if zweige ignorieren und immer ausführen). Das bedeuted wir bekommen eine Konstante anzahl an Block Operationen und somit müssen wir im schlechtesten Fall $\Theta(1)$, also genau jede Block Operation ausführen.