Algorithms and Data Structures

Arnav Singh

March 2020

8 Sheet 8

8.1 Stacks and Queues

(a) Implemented in Stack folder

Time complexity commented in the code next to functions

(b) Implemented in Queue folder

8.2 Linked List and BST

(a) Reverse LL

```
void reverse()
    {
        Node* current = head;
        Node *prev = NULL, *next = NULL;

        while (current != NULL) {
            next = current->next;
            current->next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }
```

This is an insitu algorithm because no auxilary data structure is used to reverse the Linked list. In the implementation we are not storing the list into any other data-structure we simply initialize current, prev pointers and use them to reverse the list, no temp array or data-structure is being used. The time complexity is $\Theta(n)$ since we traverse the linked list once.

(b) Sorted BST to LL

Implemented in toLL folder
Since the insertion of elements happens recursively it takes constant time to insert the elements in the BST. SO for every insertion it takes contant time, we have n elements in order will take linear time. Thus time complexity is $\Theta(n)$.

(c) Sorted LL to BST

Bonus in bonus folder
For inserting an element into the tree we have a time complexity of O(log n) or O(h) where h is the height of the tree, so for n elements we insert n time, therefore we get a total time complexity of $\Theta(nlogn)$