Übungsserie 7 – Algorithmen und Datenstrukturen

Aufgabe 1: Einfach verkettete Listen

Verwenden Sie folgende Implementierung um eine LinkedList selbst zu implementieren:

```
static class Node
{
    int val;
    Node next;
    Node(int v)
    {
        val = v;
        next = null;
    }
}
```

Erstellen Sie eine Methode static Node **insertAtBegin**(Node root, int value), welche die übergebene value an den Anfang einer Liste (vor root) fügt und root als return Wert liefert. Von welcher Ordnung ist die Laufzeit dieser Methode?

Erstellen Sie eine Methode static Node **append**(Node root, int value), welche die übergebene value an das Ende einer Liste (node Implementierung) fügt und root als return Wert liefert. Von welcher Ordnung ist die Laufzeit dieser Methode?

Erstellen Sie eine Methode static void **delete**(Node root, int value), welche die übergebene value aus einer sortierte Liste löscht und root als return Wert liefert. Von welcher Ordnung ist die Laufzeit dieser Methode?

Erstellen Sie eine Methode static Node **insertOrdered**(Node root, int value), welche die übergebene value in eine sortierte Liste einfügt und root als return Wert liefert. Von welcher Ordnung ist die Laufzeit dieser Methode?

Aufgabe 2: Bubblesort

Implementieren Sie den Bubblesort Algorithmus in einer verketteten Liste. Überlegen Sie sich einen Best-Case und einen Worst Case für das Verfahren und messen Sie mit beiden Cases die Laufzeit.

Aufgabe 3: Insertionsort

Implementieren Sie den Insertionsort Algorithmus in einer verketteten Liste (Node Implementierung aus Serie 4). Überlegen Sie sich einen Best-Case und einen Worst Case für das Verfahren und messen Sie mit beiden Cases die Laufzeit.