AD-Aufgabe05-Gruppe-SawatzkiGlake 08.11.2013

Daniel Glake, Fabian Sawatzki

Praktikumsgruppe\_2

Übungsaufgabe 5.1

|  |  |
| --- | --- |
| Aufgabe 1: Implementierung eines rekursiven Ansatzes, zum Einfügen von Elementen. | Zur Implementation des Einfügens, über einen rekursiven Ansatz wurde, wie in der Definition aus der Info in Aufgabenblatt\_5 eine getTail() Methode geschrieben, die den Zeiger der den Anfang der Liste zeigt um ein Element nach rechts verschiebt, sodass der neue Kopf lediglich das zweite Element ist. Da hierdurch die Referenz auf das eigentliche erste Element in der Liste verloren gehen würde, wurde dieser zunächst in einer Elem Variable gespeichert. Damit nun das neue Element an die korrekte Position eingefügt werden konnte, musste die Rekursion n-mal aufgerufen werden. Innerhalb der Parameterübergabe wurde das n dann um 1 dekrementiert. Die Abbruchbedingung lautete somit: „Wenn n <= 0, dann füge das neue Element einfach an die vorderste Stelle der aktuellen Liste an. Wurde das neue Element nun an die vorderste Stelle angefügt wurde aus der Rekursionsmethode herausgesprungen und es wurde der gespeicherte ehemalige Head wieder vorne an die neue Liste angefügt. Das wiederholt sich nun n-mal, wie die Rekursion aufgerufen wurde. Dadurch werden die alten Werte wieder an die neue Liste gehängt. Durch die Tatsache das die Methode cons() innerhalb der O-Klasse von O(1) liegt, findet keine Laufzeitbeeinträchtigung statt, da lediglich ein Element an die vorderste Stelle gehängt wird. Die Laufzeit dieser Methode liegt in O(n), da im schlimmsten Fall die gesamte Liste durchgelaufen wird, um ein neues Element anzufügen. Durch getTail() verschiebt sich der Fokus des Kopfes immer nur um eine Stelle nach rechts. |
| Aufgabe 2: Vergleich der iterativen Lösung mit der rekursiven Lösung, zum Einfügen von Elementen | In unserem Vergleich zeigte sich deutlich, dass der rekursive Ansatz eine höhere Laufzeit hat, als das iterative Vorgehen. Über unsere Hilfsvariable ermittelten wir für die rekursive Lösung einen Wert von 1501501 Aufrufzählungen, während die iterative Lösung lediglich 999002 Aufrufzählungen meldete. Für beide Versuche wurde der Versuchsaufbau von Aufgabe 1.2.7, mit n = 1000 verwendet. |
| Aufgabe 3: Versuchsnachbildung von Aufgabe 1.2.6; Prüfung auf Gleichheit. | Es wurden für diese Aufgabe zwei verschiedene Versuche angefertigt. Die Versuchsnachbildung von Aufgabe 1.2.6 war erfolgreich hatte für uns jedoch wenig über die tatsächliche Funktionalität des rekursiven Ansatzes ausgesagt, daher haben wir zusätzlich einen weiteren Versuch gestartet, bei dem jedes neue Element an das Ende der Liste angefügt wird. Dadurch wird zumindest gezeigt, dass keine fehlerhaften Einfügeoperationen entstehen, wie das etwas an eine falsche Stelle eingefügt wird. |
| Zusätzliche Quellen: | Es wurde für die Implementation rekursiven Methode auf die Precondition Methode, aus der Google Guava Bibliothek zurückgegriffen. |

Übungsaufgabe 5.2

|  |  |
| --- | --- |
| Aufgabe 1: Berechnung der Werte der unten stehenden Funktion für n = [0..9]. | |
|  |  |
| 0 | 1 |
| 1 | 1 |
| 2 | 1 |
| 3 | 6 |
| 4 | 11 |
| 5 | 26 |
| 6 | 66 |
| 7 | 151 |
| 8 | 361 |
| 9 | 861 |
| Aufgabe 2: Implementierung der obigen Funktion. | Die obige Funktion wurde einfach in die passende Syntax konvertiert und übernommen. Die obigen Werte aus der Tabelle wurde, mit den Ergebnissen der Implementation verglichen und stimmte überein. |
| Aufgabe 3: Bestimmung des maximalen Wertes für n, sodass die Methode f(n) einen korrekten Wert berechnet. | Für die Bestimmung von n haben wir den Wert soweit inkrementiert bis der Wert des MAX\_VALUE für einen Integer überschritten wurde. Der Wert für n liegt somit bei 26. Das Ergebnis dafür lautet: 2080960156 |
| Aufgabe 4: Anlegen eines Arrays zur Bestimmung der Aufrufhäufigkeit, für einen bestimmten n- Wert. | Um zu ermitteln, wie oft die rekursive Methode, mit einem bestimmten n Wert aufgerufen wurde, wird ein Array zu Beginn der Methode, am Arrayfeld[n] um 1 inkrementiert. Die Initialisierung erfolgte im vornherein, mit einer Feldgröße von 26, also dem maximalen Wert bis zum Overflow. |
| Aufgabe 5: Alternative Methode, mit iterativem Ansatz. | Die iterative Lösung basiert darauf, dass jeweiligen Einzelwert, als Variable, mit Initialwert 1 gespeichert werden und anschließend, nach dem Funktionsschema verrechnet werden. Die Einzelwerte, kriegen dann mit dem Wert ihres linken Nachbars zugewiesen. Ausnahme hier bildet der Wert f1, der in jedem Iterationsschritt neuberechnet wird. |
| Aufgabe 6: Alternative mit nur einem rekursivem Aufruf. | Das Vorgehen bei dieser Methode verläuft ähnlich zur iterativen Lösung. Einziger Unterschied besteht darin, dass die verwendete While- Schleife, aus der iterativen Lösung entfällt und durch einen Rekursionsaufruf ersetzt wird. Innerhalb des Aufrufs wird der Wert für n anschließend dekrementiert. |