

ListenBibliothek

Programming Partners   
Willi Hertel, Tristan Bernatzik, Marius Biegger und Iven Kopfsguter



# Linked List

Eine Liste besteht immer aus einzelnen Knoten und diese wiederrum bestehen aus Daten und einem Zeiger. Der Zeiger zeigt auf das nächste Element, deshalb spricht man auch häufig von verketteten Listen. Eine einfach verkettete Liste besteht aus einem Startpunkt, mehreren Knoten und einem Endpunkt Null. Bei einfach verketteten Listen zeigen die Knoten nur unmittelbar auf ihre Nachbarn und nicht auf weitere Elemente. Vorteil dieser Liste ist, dass man bei Beginn der Implementierung nicht die Größe wissen muss, sondern sie beliebig füllen kann. Eine solche Liste verfügt über die Operationen Einfügen, Suchen und Löschen. Beim Einfügen ist zu beachten das man erst den Nachfolger des neuen Elements bestimmt und dann die den Zeiger des vorherigen Knoten setzt. Es spielt keine Rolle, ob man den Knoten am Anfang, Ende oder mittendrin einsetzt. Beim Löschen ist dies umgekehrt, man setzt erst den Zeiger auf das Element, welches gezeigt wird, wenn das andere gelöscht wird und dann erst wird der Knoten gelöscht. Beim Suchen wird über die Liste iteriert, bis sie das gewünschte Element erreicht hat.

# Double Linked List

Die Double Linked List funktioniert vom Prinzip her gleich wie die einfach verkette Liste. Der wesentliche Unterschied ist, dass bei der doppelten Liste es in den Knoten immer zwei Zeiger gibt. Einer für den Vor- und einer für den Nachgänger. Bei dem Startpunkt gibt es kein Vorgänger sowie es beim Endpunkt kein Nachgänger gibt.

# Queue

Ist eine spezielle Form der Liste und gehört ebenfalls zu den dynamischen Datenstrukturen. Sie ist deshalb speziell, da die Objekte zwischengespeichert werden. Es kommt immer von der einen Seite ein Element rein und das letzte wird entfernt. Enqueue ist zum Hinzufügen der Objekte, hingegen die Operation dequeue Elemente entfernt. Dieses Prinzip funktioniert nach dem FIFO (First In First Out) Prinzip. Es wird also das Element, welches am längsten in der Queue ist, als erstes aus der Queue entfernt.

# Stack

Ein Stack ist bildlich gesehen nicht wie eine Queue horizontal, sondern vertikal aufgebaut. Man kann also immer nur auf das obere Element zugreifen. Ein Stack funktioniert also nach dem LIFO (Last In First Out) Prinzip. Das Element, welches zuletzt hinzugefügt wurde, wird auch als erste wieder entfernt. Die Push Operationen ermöglicht es, Elemente auf einem Stack draufzulegen. Die Pull Operation dient dazu, das obere Element des Stacks zu entfernen. Möchte man nicht das oberste Element des Stacks entfernen, sondern eines was sich weiter unten befindet, muss man alle Objekte, die darüber liegen entfernen, um an das gewünschte Element heranzukommen.

# Set

Ein Set ist eine ungeordnete Sammlung von Elementen eines Datentyps. In einem Set kann jedes Element nur einmal vorkommen. Da ein Set ungeordnet ist, kann man nicht über den Index auf die Elemente zugreifen. Daraus ergeben sich folgende Operationen. Man kann prüfen ob ein Element bereits vorhanden ist, aufzählen der Elemente in beliebiger Ordnung oder eine Menge aus den verfügbaren Elementen erzeugen.

# Suchbaum (Binärbaum)

Die Datenstruktur des Such- oder auch Binärbaums beginnt immer mit einem Wurzelknoten. Die weiteren Knoten, die daraus entstehen werden Elternknoten genannt und die aus den Elternknoten entstehen bezeichnet man als Kindsknoten. Das Entscheidende an dieser Datenstruktur ist, dass ein Knoten höchstens zwei weitere Knoten enthält und diese wieder höchstens zwei weitere. Der gesamte Binärbaum kann auch aufgeteilt werden in Teilbäume. Man unterscheidet auch zwischen gewissen Arten von Binärbäumen. Ein Binärbaum wird als geordnet bezeichnet, wenn der linke Teil des Baums kleiner ist als der rechte. So stehen also vom Wurzelknoten aus, links alle Elemente, die vom Wert kleiner sind als der Wurzelknoten und rechts die, die vom Wert größer sind als der Wurzelknoten. Ein Baum ist voll, wenn jeder Knoten zwei Kindsknoten besitzt oder gar keine.

# Sortieralgorithmen

Sortieralgorithmen werden dazu verwendet, gegebene Objekte in einer sinnvollen Reihenfolge anzuordnen. Sie werden in allen möglichen Programmen implementiert, um beispielsweise Dateien nach ihrem letzten Zugriffsdatum zu sortieren. Es gibt zahlreiche Sortierverfahren, von denen wir uns 2 genauer anschauen werden, wie diese funktionieren. Bubblesort, Quicksort und Simplesort . Vorab noch ein kurzer Einblick, wie Sortieralgorithmen unterschieden werden können. Sie können unter mehreren Aspekten unterschieden werden, so gibt es interne und externe Sortierverfahren. Bei den internen passen alle Daten in den Speicher, das heist es sind alle Objekte bekannt und bei externen immer nur ein Teil der Objekte. Ebenso kann zwischen stabil oder instabil unterschieden werden. Bei den Stabilen Algorithmen ändert sich bei gleichen Werten die Reihenfolge der vorigen Sortierung nicht, bei instabilen kann dies nicht garantiert werden. Der bekannteste Unterscheidungspunkt ist die Laufzeit der Algorithmen. Unter Laufzeit versteht man die Schnelligkeit des Algorithmus beim Sortieren.

## Bubblesort

Ist ein Stabiler Sortieralgorithmus, der aufsteigend und paarweise sortiert. So gehört Bubblesort auch zu den vergleichsbasierten Verfahren, da er die Objekte nebeneinander aufsteigend vertauscht. Es werden also alle Objekte solange vertauscht bis die richtige Reihenfolge zustande kommt. Es beginnt mit der ersten sogenannten Bubble Phase in dem die ersten beiden Elemente des Arrays verglichen werden. Sind diese in der richtigen Reihenfolge wird weiter über den Array iteriert, falls nicht werden diese vertauscht. Als nächstes wird das zweite mit dem dritten Element verglichen und wieder überprüft, ob es vertauscht werden muss. Dies geschieht solange, bis man am Ende des Arrays angekommen ist und von da an beginnt die zweite Bubble Phase und der Algorithmus iteriert wieder über alle Elemente.

## Quicksort

Ist ein rekursiver und nicht stabiler Sortieralgorithmus der ein Array quasi teilt, indem man durch auswählen auf ein beliebiges Element (Pivotelement) automatisch alle weiteren Elemente, die vor und nach dem Pivotelement anzeigt. Dies kann man dann weiter mit den beiden Teilen des Arrays machen, bis der Array vollständig sortiert ist. Es wird also in jeder Teilliste ein größeres und ein kleineres Element gesucht und vertauscht, die Suche des kleineren beginnt am Anfang und die des größeren am Ende des Arrays. (das Pivotelement wird dabei ausgeschlossen). Diese Funktion wird rekursiv aufgerufen bis der Array komplett sortiert ist.

## Simple Sort

Simplesort ist ein nicht stabiles Sortierverfahren, bei dem man im Array die Positionen nacheinander betrachtet. Es werden in der ersten Iteration die jeweiligen Elemente gesucht und in der zweiten einsortiert. Es kann also bei Simplesort mehrere Vertauschungen von zwei Elementen geben, bis das richtige Element einsortiert ist.

# Reflexion

Zum Ende eines Projektes ist die Analyse der gemachten Fehler natürlich denkbar schwer, da die Aufgabe ja bereits bewältigt wurde und man sich an die kleinen Hindernisse kaum mehr erinnern kann, dennoch lässt sich sagen, dass die größten und auch beinahe einzigen Schwierigkeiten durch eine zu geringe Planungsphase und einem daraus resultierendem fehlenden Überblick entstanden sind.

Da der erste Schritt, nachdem der Startschuss für das Projekt erfolgt war, die Zuteilung der Aufgabenbereiche war, machte sich niemand ernstere Gedanken über das Projekt im Ganzen. Dadurch war jeder auf seine eigenen Aufgaben fokussiert und erledigte diese. Hätte man sich etwas genauer mit der Materie auseinander gesetzt wäre es eventuell nicht dazu gekommen, dass man einen Teil des bereits erreichten Fortschritts überarbeiten muss. Grund für diesen Aufwand war eine falsch geplante Reihenfolge in der die Aufgaben abgearbeitet wurden. Denn dadurch dass die falschen Listenstrukturen zuerst programmiert wurden, wurde nicht auf die richtige Reihenfolge der Vererbung geachtet. Erst bei einem genaueren begutachten der Hierarchie des Java Collection Framework fiel dieser Fehler auf. Durch diese Unachtsamkeit machten wir uns in etwa eine Listenstruktur mehr Arbeit als nötig gewesen wäre.

Dadurch, dass sämtliches Wissen, welches für die Bearbeitung des Programmentwurfs benötigt wurde, in den Vorlesungen mitgegeben wurde, gab es neben minimalen Startschwierigkeiten mit GitHub und IntelliJ keine nennenswerte Schwierigkeiten mehr welche es galt zu beseitigen.

Abschließend lässt sich sagen, dass sowohl Umfang, als auch der Schwierigkeitsgrad der Aufgaben passend gewählt worden waren. Auch wenn sie fordernd waren, ließen sie einen nie aus purer Verzweiflung das Interesse an der Aufgabe verlieren.