Lernzettel

Verkettete Listen: Einfach- und Zweiseitig verkettete Listen

Universität: Technische Universität Berlin
Kurs/Modul: Einführung in die Informatik - Vertiefung
Erstellungsdatum: September 20, 2025

Lernzettel: Verkettete Listen: Einfach- und Zweiseitig verkettete Listen

(1) Grundbegriffe. Eine einfach verkettete Liste besteht aus einer Folge von Knoten, wobei jeder Knoten zwei Felder besitzt: ein Datenfeld und einen Verweis auf den nächsten Knoten. Der letzte Knoten verweist auf NULL. In Java-ähnlicher Darstellung entspricht ein Knoten meist der Struktur

$$\text{Knoten} = (\text{Daten}, \text{Next})$$

(2) Aufbau einer Liste. Die Liste wird durch einen Head-Knoten beschrieben. Der Head verweist auf den ersten Knoten bzw. auf NULL, wenn die Liste leer ist. In einer zweiseitig verketteten Liste hat jeder Knoten zusätzlich einen Verweis auf den vorherigen Knoten (Prev). Formal:

$$\text{Singly: Knoten} = (\text{Daten, Next})$$

$$\text{Doubly: Knoten} = (\text{Daten, Next, Prev})$$

(3) Einfach verkettete Liste – Typische Operationen. - Einfügen am Anfang:

$$\text{Neu.Next} = \text{Head}, \quad \text{Head} = \text{Neu}$$

- Einfügen am Ende (ohne Tail-Zeiger):

$$\text{Letzt} = \text{Head}; \quad \text{while Letzt.Next} \neq \text{ do Letzt} = \text{Letzt.Next}; \quad \text{Letzt.Next} = \text{Neu}; \quad \text{Neu.Next} =$$

- Suchen eines Elements erfordert meist lineare Zeit. - Entfernen eines Knotens in der Liste erfordert das Finden des Vorgängers (bis O(n)) und Anpassung der Verweise.

(4) Zweiseitig verkettete Liste – Vorteile. Jeder Knoten besitzt neben Next auch Prev. Dadurch lassen sich folgende Operationen effizienter durchführen, insbesondere das Entfernen eines Knotens oder das Durchblättern in beide Richtungen:

$$\text{Knoten} = (\text{Daten, Next, Prev})$$

Vorteile: - Rückwärtsnavigation möglich (Backward Traversal). - Entfernen eines Knotens kann leichter erfolgen, wenn man Zeiger auf den Knoten selbst hat (O(1) Zeit bei vorhandenen Next/Prev-Verweisen).

(5) Traversierung. - Singly: Start bei Head; solange aktueller Knoten != NULL, verarbeite Daten und gehe weiter zu Next. - Doubly: Gleiches Vorzeichen, zusätzlich ermöglicht Prev eine Rückwärtsrichtung.

$$\text{aktueller} \leftarrow \text{Head}; \quad \text{while aktueller} \neq \text{ tun aktueller} = \text{aktueller.Next}$$

(6) Speicher- und Zeitkomplexität. - Zugriff auf das i-te Element: $O(i)$ in der durchschnittlichen Situation. - Suchen eines Elements: $O(n)$. - Einfügen am Anfang: $O(1)$. - Einfügen am Ende: $O(n)$ (ohne Tail-Zeiger) bzw. $O(1)$ mit Tail-Pointer. - Entfernen eines Knotens: $O(n)$ in der Regel, da oft die Suche notwendig ist; in einer Doubly-List oft besser, wenn der Zielknoten bekannt ist (mit Prev-Verweisen). Zusätzlicher Speicherbedarf: - Singly: ein Next-Verweis pro

Knoten. - Doubly: zusätzlich ein Prev-Verweis pro Knoten; doppelter Speicherbedarf pro Knoten, aber bessere Navigationsmöglichkeiten.

(7) Beispiel-Diagramme. Singly: Head → [Daten1 | Next] → [Daten2 | Next] → NULL
Doubly: Head ↔ [Daten1 | Next | Prev] ↔ [Daten2 | Next | Prev] ↔ NULL

(8) Typische Anwendungsszenarien. Verkettete Listen eignen sich gut, wenn häufige Einfüge- und Löschoperationen an beliebigen Positionen stattfinden und der Speicherbedarf bekannt bleibt. Sie sparen Speicher für Arrays (kein fester Kapazitäts-Puffer) und ermöglichen dynamische Größenanpassungen.

(9) Übungsideen (ohne Lösung hier). - Implementiere eine generische singly und eine generische doubly Liste in Java mit Methoden: insertFirst, insertLast, remove, find, traverseForward, traverseBackward (nur Methoden-Signaturen sinnvoll skizzieren). - Ergänze eine Tail-Referenz für die einfache Liste, um das Ablegen am Ende in O(1) zu ermöglichen. - Vergleiche in einer kurzen Tabelle die Worst-Case-Komplexitäten beider Liste-Typen.

(10) Fazit. Verkettete Listen bieten flexible Speicherstrukturen mit schnellen Einfüge- und Löschoperationen am Anfang bzw. in der Nähe des aktuellen Knotens, liefern jedoch im Allgemeinen schlechteren Zugriffszeiten als Arrays. Die zweiseitig verkettete Variante erleichtert Traversierungen in beide Richtungen und ermöglicht effizienteres Entfernen eines bekannten Knotens.