Listen

Für die Verarbeitung von Listen durch den Rechner ist relevant:

- **Anzahl der Listenelemente**
  - **beim erstmaligen Erstellen der Liste bekannt/unbekannt**
  - **stabil vs. wechselhaft**
  - **Existenz eines Maximalwerts**
- **Art der Listenelemente**
  - **gleiche vs. verschiedene Struktur**
  - **Ordnung (z.B. alphabetische Reihenfolge)**
- **Verwendungszweck**
  - **Häufigkeit von Such-/Erweiterungs-/Löschoperationen**

Beispiele:

- **alphabetische Liste aller deutschen Adjektive / Präpositionen**
- **alle Lexikoneinträge zu den Wörtern eines Satzes**
- **nach Häufigkeit geordnete Liste der Wortformen eines Textkorpus**

Aufgrund der oben genannten Eigenschaften wird festgelegt:

1. Datenstruktur für die Listenelemente

2. Darstellung:

- **lineare Liste mit wahlfreiem Zugriff**
- **verkettete Liste**
- **indexsequentielle Liste**

Aus 1. und 2. ergibt sich die benötigte Speicherplatzreservierung.

Beispiele zur Wahl der Datenstruktur für die Listenelemente:

1. Zeichenkette ohne weitere Struktur

```
Aal <m.; -(e)s, -e> 1 <i.e.S.> (Europäischer) ~ langer,
schlangenähnlicher Speisefisch: geräucherter ~

Aas1 <n.; -es; unz.> 1 verwesende Tierleiche; Sy Kadaver

abbauen <V.> 1 <500> Bodenschätze, Kohle, Eisenerz
```

2. Struktur mit festen Datenfeldern

Wortform	Aal	
Wortart	Nomen	
Genus	m	
Valenzklasse		
Genitivendung	(e)s	
Lesart1	**Definition**	langer,schlangenähnlicher Speise-fisch
	Beispiele	geräucherter Aal
	Synonym	
Lesart2	**Definition**	...
	Beispiele	
	Synonym	

...

Lesart-x	**Definition**	...
	Beispiele	
	Synonym	

3. Struktur mit variablen Datenfeldern

Wortform	Aal	
Wortart	Nomen	
Genus	m	
Genitivendung	(e)s	
Anzahl Lesarten	1	
Lesart1	**Länge der Definition**	37
	Definition	langer,schlangenähn-licher Speisefisch
	Beispiele	geräucherter Aal
	Synonym	

Lineare Liste mit wahlfreiem Zugriff

“wahlfreier Zugriff” = gleiche Zugriffsdauer bei allen Einträgen

(d.h. der erste und der hundertste Eintrag werden gleich schnell gefunden; dies bedeutet nicht die Suche nach einem Eintrag mit einem bestimmten Wortlaut)

wird erreicht durch Speicherbereich fester Größe pro Listen-element * laufende Nummer von 1 bis n

(entspricht den sog. Feldern (Arrays) in Programmiersprachen)

Aal	***in Süßwasser und Meer lebender Fisch mit ...***
aalartig	***wie ein Aal***
aalen	***sich behaglich ausgestreckt ausruhen***
Aalfang	***das Fangen von Aalen***
Aalfischer	***Fischer, der bes. auf Aalfang ausgeht***
Aalfischerei	***Aalfang***

Verkettete Liste

Jedes Listenelement enthält einen Verweis auf die Speicheradresse des nächsten Elements.

Vorteil: Alle Listenelemente können unterschiedlich groß sein und eine andere Struktur haben, und die Länge der Liste kann beliebig variieren.

Nachteil: kein wahlfreier Zugriff mehr

Spezialfall: doppelt (= vorwärts und rückwärts) verkettete Liste

Aal	***in Süßwasser und Meer lebender Fisch mit***

	schlüpfriger Haut		
aalartig	*wie ein Aa*l	**aalen**	*sich behaglich ausgestreckt ausruhen*
Aalfang	*das Fangen von Aalen*	**Aalfischer**	
Fischer, der bes. auf Aalfang ausgeht		**Aalfischerei**	*Aalfang*

Indexsequentielle Liste

verbindet Vorteile der linearen Liste und der verketteten Liste:

Listenelemente werden nach einem Sortierschlüssel in verschiedene verkettete Listen unterteilt, die über eine lineare Liste erreichbar sind

A	**Aal**	*in Süßwasser und Meer lebender Fisch mit schlüpfriger Haut*					
	aalartig	*wie ein Aal*	**aalen**	*sich behaglich ausgestreckt ausruhen*			
	Aalfang	*das Fangen von Aalen*	**Aalfischer**				
	Fischer, der bes. auf Aalfang ausgeht		**Aalfischerei**	*Aalfang*			
B	**babbeln**	*noch unverständlich sprechen*	**Babel**	*verworfener Ort*			
	Babusche	*bequemer, warmer, aus Stoff hergestellter Hausschuh*					
C	**Cab**	*einspännige englische Droschke*	**Caballero**	*spanischer Ritter*			
	Caban	*modischer kurzer Herrenmantel*					

Varianten:

- **Indexelemente sind ebenfalls als verkettete Liste organisiert**
- **Teillisten sind sortiert oder verfügen über wahlfreien Zugriff**

Unterschiede beim Einfügen und Löschen:

Lineare Liste:

Zusätzlich zur Liste wird eine Angabe über deren maximale und aktuelle Länge benötigt.

Einfügen: bei unsortierten Listen:

Erhöhen des Zählers für die aktuelle Länge;

falls die maximale Länge nicht überschritten wird:

Neueintrag in dieser Position

bei sortierten Listen:

Erhöhen des Zählers für die aktuelle Länge,

falls die maximale Länge nicht überschritten wird:

Ermitteln der korrekten Position,

Verschiebung aller Elemente ab dieser Position um ein Feld nach hinten (d.h. Richtung Listenende),

Neueintrag

Löschen eines Elements an Position X:

bei unsortierten Listen:

Verschiebung des letzten Listenelements nach X,

Verminderung des Zählers für die aktuelle Länge

bei sortierten Listen:

Verschiebung aller Elemente ab dieser Position um ein Feld nach vorne (Richtung Listenanfang),

Verminderung des Zählers für die aktuelle Länge

Verkettete Liste:

Benötigt werden Verweise (Zeiger) auf das erste, bei doppelt verketteten Listen auch auf das letzte Listenelement.

Einfügen: bei unsortierten Listen:

bei doppelt verketteten Listen:

Anhängen eines neuen Elements,
Anpassung des Zeigers auf das letzte Element

bei einfach verketteten Listen:

Durchlaufen der Liste bis ans Ende,
Anhängen des neuen Elements

bei sortierten Listen:

Durchlaufen der Liste bis zur passenden Stelle,
Einfügen eines neuen Elements durch entsprechende Änderung der Verkettung

Löschen:

Durchlaufen der Liste bis zur passenden Stelle,
Änderung der Verkettung,
Freigabe des Speicherbereichs, den das zu löschende Element bisher belegt hat

Verkettete Listen lohnen sich immer, wenn die Zahl der Listenelemente stark schwankt oder Elemente innerhalb der Liste hinzugefügt werden müssen (z.B. bei sortierten Listen).

Bei geringfügigeren Schwankungen oder häufigen Zugriffen auf Elemente am Ende der Liste sind lineare Listen (Felder) dagegen effizienter.

Spezialfälle:
Keller/Stapel (stack) und Schlange (queue)

spezielle Listen, bei denen das Lesen bzw. Löschen von Elementen einerseits und das Hinzufügen von Elementen andererseits jeweils nur an einer einzigen Stelle zulässig ist:

Keller:

Lesen/Löschen und Hinzufügen jeweils nur an demselben Ende

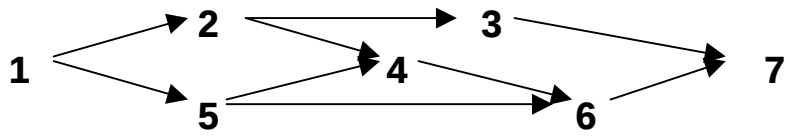
(“last in, first out - LIFO”)

Schlange:

Lesen/Löschen am einen, Hinzufügen am anderen Ende

(“first in - first out - FIFO”)

Beispiel: Suche nach einem Weg in einem Graphen mit Hilfe von Tiefen- und Breitensuche



Gesucht: alle Wege von 1 nach 7

Strategie: Ersetze einen Knoten in der Liste durch seine Nachfolger (unter Angabe des zurückgelegten Wegs)

Breitensuche: Schlange

1
2-1 5-1
5-1 3-2-1 4-2-1
3-2-1 4-2-1 4-5-1 6-5-1
4-2-1 4-5-1 6-5-1 **7-3-2-1**
4-5-1 6-5-1 7-3-2-1 6-4-2-1
6-5-1 7-3-2-1 6-4-2-1 6-4-5-1
7-3-2-1 6-4-2-1 6-4-5-1 **7-6-5-1**
6-4-2-1 6-4-5-1 7-6-5-1
6-4-5-1 7-6-5-1 **7-6-4-2-1**
7-6-5-1 7-6-4-2-1 **7-6-4-5-1**
7-6-4-2-1 7-6-4-5-1
7-6-4-5-1

Tiefensuche: Keller

1
2-1 5-1
3-2-1 4-2-1 5-1
7-3-2-1 4-2-1 5-1 (erster Weg)
4-2-1 5-1
6-4-2-1 5-1
7-6-4-2-1 5-1 (zweiter Weg)
5-1
4-5-1 6-5-1
6-4-5-1 6-5-1
7-6-4-5-1 6-5-1 (dritter Weg)
6-5-1
7-6-5-1 (vierter Weg)