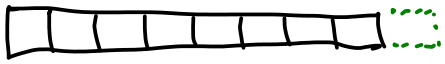
Dynamische Listen

	a[i]	EINF. LÖSCHEN	am Anfang/Ende
• Felder 	✓	✗	?
• verkettete Listen 	✗	✓	✓
• Bäume	✗	✓	✓
		$\log n$	

Liste, bei der am Ende angehängt wird. (oder gelöscht)

- Liste wird in einem Feld mit Länge L gespeichert.
- Wenn die Länge n der Liste L überschreitet, wird die ganze Liste in ein größeres Feld mit Länge L' umgespeichert. $O(n)$ Zeit

$L \to L+1$ $\quad$ $L \to 2L = L'$

Listenlängen sind Zweierpotenzen: $L = 2^k$

Insgesamt wurden n Elemente eingefügt. $2^{k-1} < n \leq 2^k = L$

Die Gesamtkosten der Umspeicheroperationen

$$O\left(1 + 2 + 4 + \dots + 2^{k-1}\right) = O\left(\underbrace{2^k - 1}_{\leq 2n}\right) = O(n)$$

Umgelegt auf die n Einfügeoperationen

$O(1)$ pro Einfügeoperation

Die amortisierte Laufzeit einer Einfügeoperation ist $O(1)$.

Benötigter Speicher $L \leq 2n = O(n)$. *

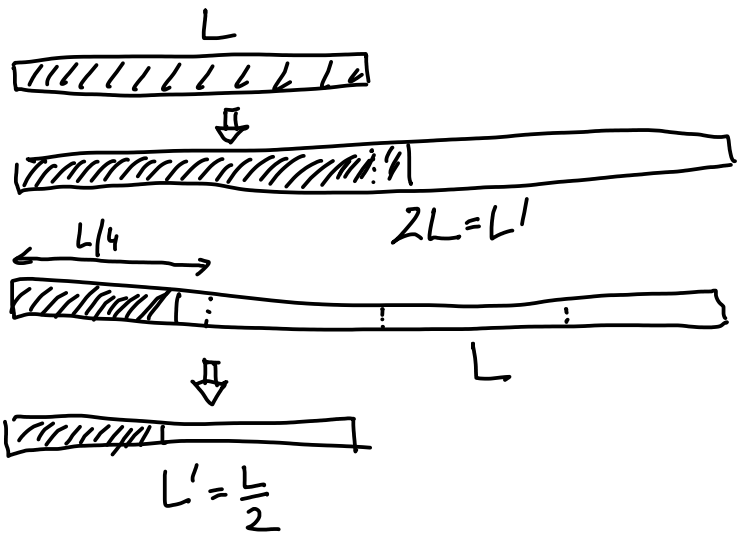
Löschen:

Wenn man beim Löschen die Bedingung * aufrecht-erhalten möchte, muss man gegebenen falls die Liste auf ein kleineres Feld umspeichern.

Einfügen: $n > L \Rightarrow L' := 2L$

Löschen: $n < \frac{L}{4} \Rightarrow L' := \frac{L}{2}$

$n \geq \frac{L}{4}$ $\quad L \leq 4n$



„kritische" Füllstände: 1 und $\frac{1}{4}$

nach dem Umspeichern: $\frac{1}{2}$ $\quad \frac{1}{2} \cdot L + 1 = n$

Bis zum nächsten Umspeichern sind mindestens $\frac{n}{2}$ Einfüge oder Lösch-Operationen möglich.

- Das gleiche gilt bei Hash-Tabellen !

Einfügen oder Löschen

a) am Ende einer dynamischen Liste

b) in einer Hashtabelle

geht in $O(1)$ amortisierter Zeit,

wobei zu jedem Zeitpunkt für n Elemente nur $O(n)$ Speicher benötigt wird.

Listen in Python sind als Felder gespeichert.

- `a[i] = ...` — $O(1)$
- ✓ `a.append(x)` — $O(1)$ amortisierte Zeit
- `a.pop()`
- `a + b` } $O(n)$
- `a + [x]`
- `del a[5]`

Mengen: `s = set()` } Hash-Tabellen
Wörterbücher: `d = dict()`

- `x in s ?` } $O(1)$ erwartete Laufzeit
- `s.add(x)`
- `d[x]`