Wiederholung: am Freitag wurde **Hashing** als **Implementierung** eines **Wörterbuchs** eingeführt.

Definition: Gegeben seien Objekte $o_1, \ldots, o_m$, die eindeutig über zugehörige Schlüssel $k_1, \ldots, k_m \in U$ identifiziert werden können, wobei $U = \{0, 1, \ldots, N\}$ das *Universum* sei und $m << |U|$ gelte. Ein *Wörterbuch* T ist eine Datenstruktur, welche folgende Operationen unterstützt:

- $\textsc{Find}(k)$: Teste, ob das Objekt zum Schlüssel k im Wörterbuch enthalten ist.
- $\textsc{Insert}(o, k)$: Füge das Objekt o mit Schlüssel k ins Wörterbuch ein.
- $\textsc{Delete}(k)$: Entferne das Objekt zum Schlüssel k aus dem Wörterbuch.

erste Idee: Man implementiert T durch ein Array $T[1 \dots N]$.

$\longrightarrow$ Die Operationen FIND, INSERT, DELETE brauchen $O(1)$ Zeit.

$\longrightarrow$ Das Array T benötigt $O(N)$ Speicherplatz.

ein Beispiel: Beim Compilieren will man wissen, welche Vaiablennamen bereits vergeben sind. Seien also die Objekte Zeichenketten aus $\{a, \dots, z, A, \dots, Z, \_\}^{|20|}$. Dann gilt $|N| \approx 10^{34}$...

$\longrightarrow$ Die erste Idee ist also ineffizient.

Ausweg: Man implementiert T als Array der Größe n (*Kapazität*) und führt es Funktion $h : U \rightarrow \{0, \dots, n-1\}$ (HASHFUNKTION) ein. Der Wert $h(k)$ heißt dann *Position* des Objektes mit Schlüssel k.

- **Problem:** Da $N >> n$ gilt, kann h *nicht injektiv* sein.
- Somit gibt es nach dem *Schubfachprinzip* Schlüssel $k_1, k_2 \in U$ mit $k_1 \neq k_2$ und $h(k_1) = h(k_2)$. In so einem Fall spricht man von einer **Kollision**.

Satz: In einer Hashtabelle der Größe n, in der m Schlüssel gespeichert sind, sei für jeden Schlüssel jede Position gleich wahrscheinlich. Dann gibt es mit Wahrscheinlichkeit

$$\geq 1 - e^{-\frac{m(m-1)}{2n}}$$

eine Kollision.

Korrolar: Sind in einer Hashtabelle der Größe n mindestens $\omega(\sqrt{n})$ Schlüssel gespeichert und ist für jeden Schlüssel jede Position gleich wahrscheinlich, so tritt mit Wahrscheinlichkeit $1-o(2)$ eine Kollision auf.

Satz: Sei für eine Hashfunktion für jeden Schlüssel jede Position in einer Hashtabelle gleichwahrscheinlich. Dann gilt: bei konstantem Füllfaktor α benötigt eine Operation auf der Hashtabelle unter Verwendung der CHAINING-Strategie zur Kollisionsauflösung im Mittel $\Theta(1)$ Laufzeit.

	A^-	A^+
Lineares Sondieren	$\frac{1}{2}(1 + \frac{1}{1-\alpha})$	$\frac{1}{2}(1 + \frac{1}{(1-\alpha)^2})$
Double Hashing	$\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$	$\frac{1}{1-\alpha}$