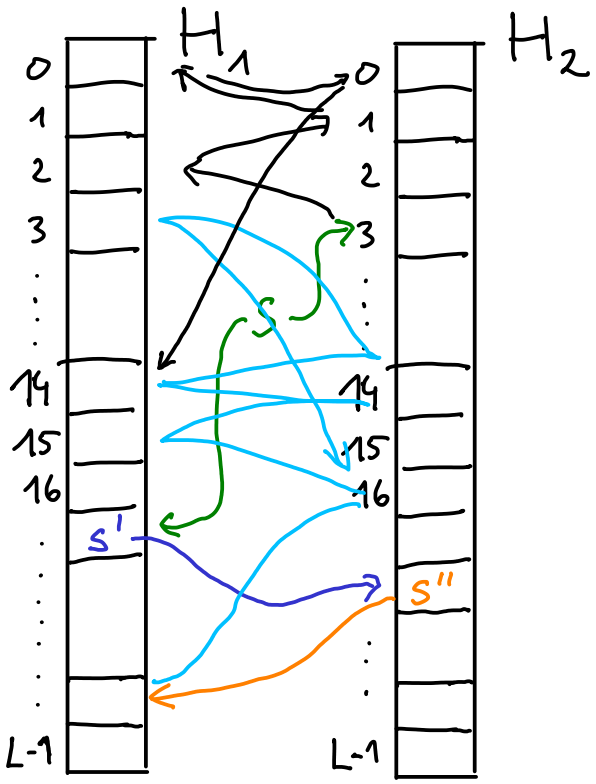
Kuckucks-Hashing: Analyse

2 Hashfunktionen h_1, h_2

> Schlüssel x ist in $H_1[h_1(x)]$ oder $H_2[h_2(x)]$ gespeichert.



$O(1)$ für Suchen (und Löschen) im schlimmsten Fall.

Einfügen: Kette von Vertauschungen,

- im Erwartungswert $O(1)$.
- Im schlimmsten Fall muss die gesamte Tabelle mit neuen Hashfunktionen h_1, h_2 aufgebaut werden.

---

	h_1	h_2
x_1		
⋮		
x_i	$h_1(x_i)$	$h_2(x_i)$
⋮		
x_n		

$\in 0, \dots, L-1$

IDEALISIERTE ANNAHME.

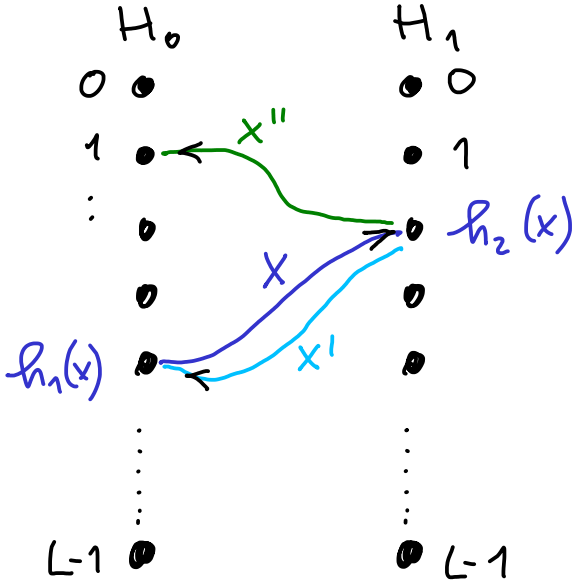
> Alle $(L^2)^n = L^{2n}$ Möglichkeiten sind gleich wahrscheinlich.

Belegungsfaktor $\alpha = \frac{n}{2L}$

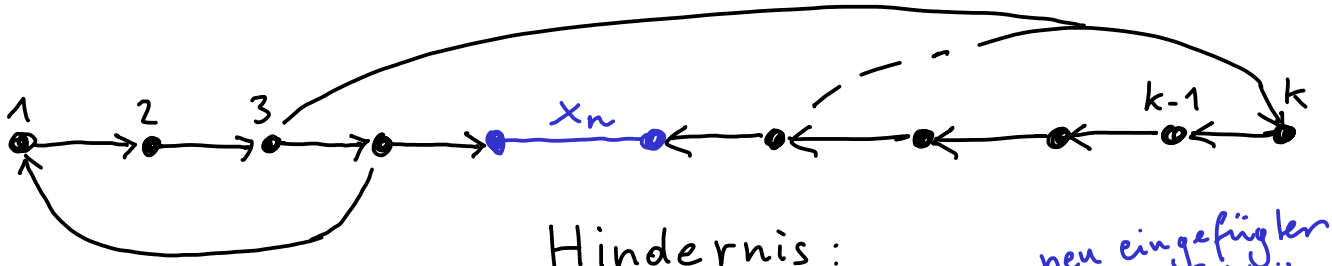
ANNAHME. $\alpha \leq \frac{1}{4}$

($\alpha < \frac{1}{2}$ möglich)

Kuckucksgraph



zulässige Orientierung des Kuckucksgraphen:
Jeder Knoten hat Innengrad ≤ 1.



Hindernis:

neu eingefügter Schlüssel ↓ x_n

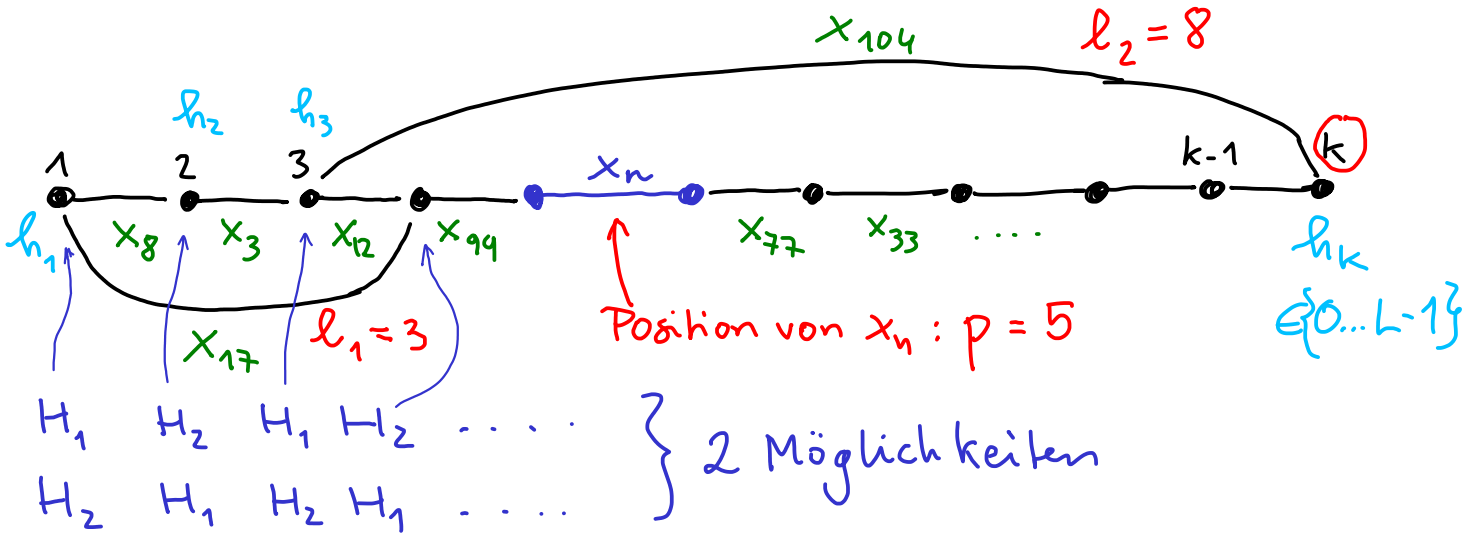
Ein Weg mit $k \geq 2$ Knoten durch die Kante x_n
+ eine Kante vom ersten Knoten
+ eine Kante vom letzten Knoten

- Wenn der Einfügealgorithmus steckenbleibt, gibt es ein Hindernis.
- Wenn es ein Hindernis gibt, hat der Kuckucksgraph keine zulässige Orientierung!

k Knoten $(k-1)+2 = k+1$ Kanten $>$ # Knoten.

P[Der Kuckucksgraph enthält ein Hindernis durch die Kante x_n]?

! Kuckucksgraphen mit Hindernis können effizienter codiert werden als durch die allgemeine Hashwertetabelle.



Gestalt: h_1, h_2 $\leq k$ Möglichkeiten

p $\leq k$ Möglichkeiten

x_i n^k Möglichkeiten

h_i L^k Möglichkeiten

Die Hashwerte für die Schlüssel x, die nicht im Hindernis vorkommen werden explizit codiert: $(h_1(x), h_2(x))$

$n \leq \frac{L}{2}$ $\quad (L^2)^{n-(k+1)}$ Möglichkeiten.

$$2k^3 n^k L^k L^{2n-2k-2} \leq \frac{2k^3 \cdot L^k \cdot L^k \cdot L^{2n-2k-2}}{2^k} = \frac{2k^3 L^{2n-2}}{2^k}$$

$$\Pr[\exists \text{ Hindernis} \ldots] \leq \frac{\sum_{k\geq 2} \frac{2k^3 L^{2n-2}}{2^k}}{L^{2n}} = \frac{1}{L^2} \underbrace{\sum_{k\geq 2} \frac{2k^3}{2^k}}_{O(1)} = O\left(\frac{1}{L^2}\right) = O(1/n^2)$$

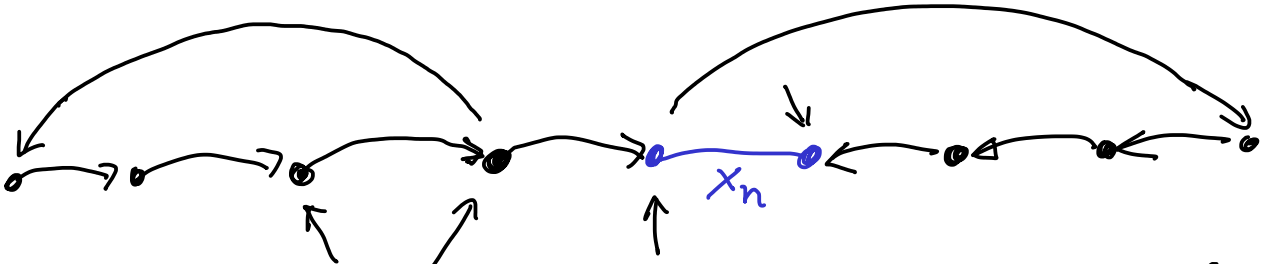
SATZ: Wenn $\alpha \leq 1/4$ ist, und die Hashwerte wirklich zufällig sind, dann ist ein Einfügen mit Wahrscheinlichkeit

$$1 - O(1/L^2)$$

erfolgreich.

Folgerung: Eine Folge von n Einfügeoperationen $(n \leq \frac{L}{2})$ scheitert nur mit Wahrscheinlichkeit $O(1/L)$.

Kuckucks-Hashing: Erfolgreiches Einfügen



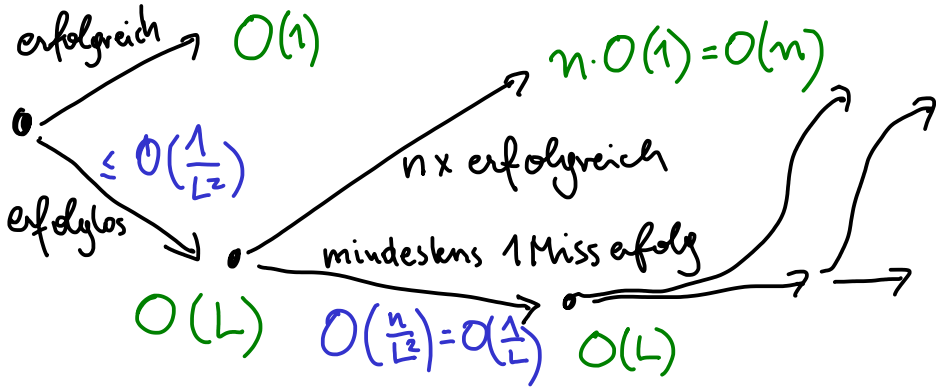
belegt sind $n-1$ Einträge.

$\Pr[\text{belegt}] = \frac{n-1}{L} < 2\alpha \leq \frac{1}{2}$.

Unter der Bedingung, dass das Einfügen erfolgreich ist, findet man in jedem Schritt mit Wahrscheinlichkeit $> 1/2$ eine freie Stelle.

$\Rightarrow$ Die Anzahl der Stellen, die angeschaut werden, ist im Erwartungswert höchstens $2 = O(1)$.

> SATZ: Wenn $\alpha \leq 1/4$ ist, und die Hashwerte zufällig sind, dann ist die erwartete Einfügezeit
>
> $O(1)$.



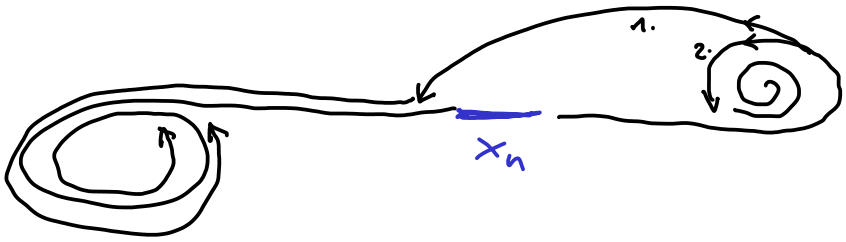
$$O(1) + O(\tfrac{1}{L^2}) \cdot \Big[O(L) + O(n) + O(\tfrac{1}{L}) \cdot \big[O(L) + O(n) + O(\tfrac{1}{L}) \cdot [\dots] \big] \Big]$$

$$\leq \underline{O(1) + O(\tfrac{1}{L}) + O(\tfrac{1}{L^3}) \cdot O(L) + O(\tfrac{1}{L^4}) \cdot O(L) + \dots} \qquad \square$$

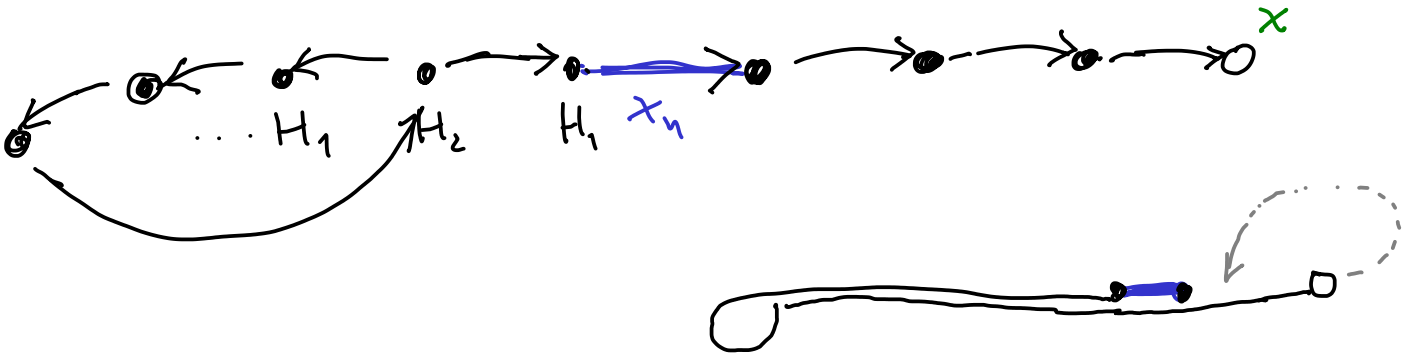
Implementieren des Einfügens

a) Markieren

b) ohne Markierungen, vgl. Aufgabe 80



c) ohne Markierungen, draufgängerisch.



Einfügen(x):

if x ist auf $H_1(h_1(x))$ oder $H_2(h_2(x))$ gespeichert: return

for $i = 1, \ldots, n$

$x \leftrightarrow H_1(h_1(x))$

if x = None: return

$x \leftrightarrow H_2(h_2(x))$

if x = None: return

Abbruch:
neue Hashfunktionen h_1 und h_2 bestimmen und Tabelle neu aufbauen.