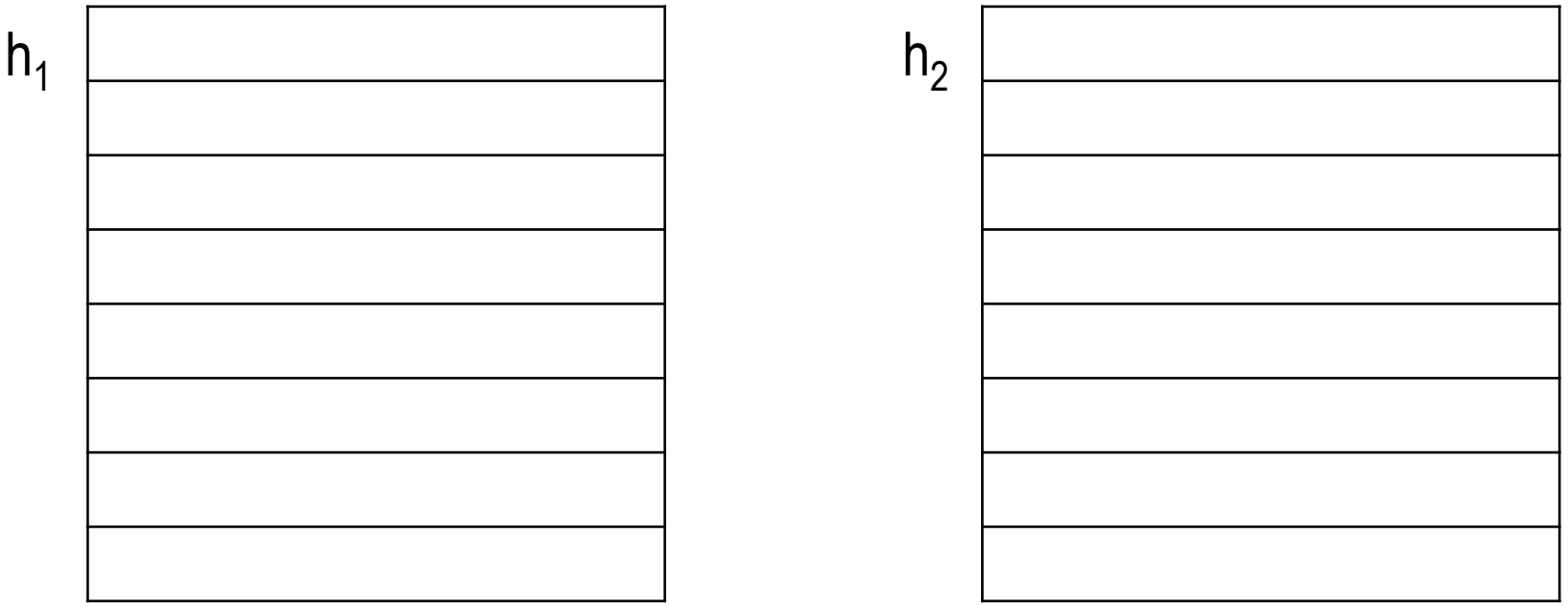
Cuckoo Hashing

Eine praktische Umsetzung des mathematischen Konzepts „universal hashing“

Zwei Arrays (in unserem Kontext gleich groß. Es gibt auch Varianten, in denen eines der Arrays größer ist.)

Zwei zufällig gewählte Hashfunktionen.

Element wird zunächst in erste Tabelle eingefügt. Wenn der Platz besetzt ist, dann in die zweite



Trivial

Suche zunächst in der ersten Tabelle mit der ersten Hashfunktion. Falls nicht erfolgreich, mit der zweiten Hashfunktion in der zweiten Tabelle. Falls das auch nicht erfolgreich ist, ist das Element nicht in der Hashtabelle enthalten

Trivial

Führe Suche durch. Falls das Element gefunden wird, entsprechende Position als frei markieren.

Was passiert, wenn beide für das Element ermittelten Positionen belegt sind?

Das blockierende Element wird in die erste Tabelle eingefügt, und verdrängt ein eventuell dort vorhandenes Element, das dann in die zweite Tabelle eingefügt wird usw.

In Pseudocode (x sei das Element das einzufügen ist):

Falls x schon in der Tabelle, retourniere
Einzufügen ist x;
So lange (Anzahl der Schleifendurchläufe nicht zu groß) {
  Betrachte Position h1(Einzufügen) in Tabelle 1.
  Falls leer, schreibe Einzufügen dort hin und retourniere.
  Sonst vertausche Einzufügen und das an der besetzten Position gespeicherte Element
  Betrachte Position in h2(Einzufügen) in Tabelle 2.
  Falls leer, schreibe Einzufügen dort hin und retourniere.
  Sonst vertausche Einzufügen und das an der besetzten Position gespeicherte Element
}

Zyklus wäre erkennbar, wenn das ursprünglich einzufügende Element zum dritten Mal auftritt, Aufwand zahlt sich aber nicht aus.
Die hier zu wählende Größe muss für die Speicherung von n Datenwerten von der Ordnung O(n) sein.

Einfügen (2)

Falls die Anzahl der maximalen Schleifendurchläufe erreicht wird, werden zufällig zwei neue Hashfunktionen gewählt und alle Elemente rehasht.

Das kann man in situ machen, indem man einfach beide Tabellen durchläuft und jedes Element, das nicht an der richtigen Stelle steht, (nach den neuen Hashfunktionen) wieder einfügt.

Danach wird wieder versucht, das ursprünglich einzufügende Element einzufügen.

Erwartungswert für die Laufzeit beim Einfügen ist O(1) (wenn die Tabelle zu weniger als 50% belegt ist)

Universelle Familie von Hashfunktionen

Für Adressen mit q Bits (Größe der Tabellen ist dann eine 2er-Potenz)
und w Bits für die Berechnung (z.B. 64 bei 64 Bit-Zahlen),
wähle zufällig ein ungerades a mit $0 < a < 2^w$ und
berechne

$$h_a(x) = (a * x \; mod \; 2^w) \; div \; 2^{w-q}$$

(mod ist die Modulo-Operation und div die ganzzahlige Division).
Das lässt sich extrem effizient berechnen!

Diese Familie von Hashfunktionen ist zwar mathematisch nur fast universell, reicht aber für unsere Zwecke aus.

Eine simple Änderung der Berechnung führt zu einer echt universellen Familie

$$h_{a,b}(x) = ((a * x + b) \; mod \; 2^w) \; div \; 2^{w-q}$$

mit einer weiteren zufällig gewählten Zahl b mit $0 \leq b < 2^{w-q}$