Cuckoo Hashing

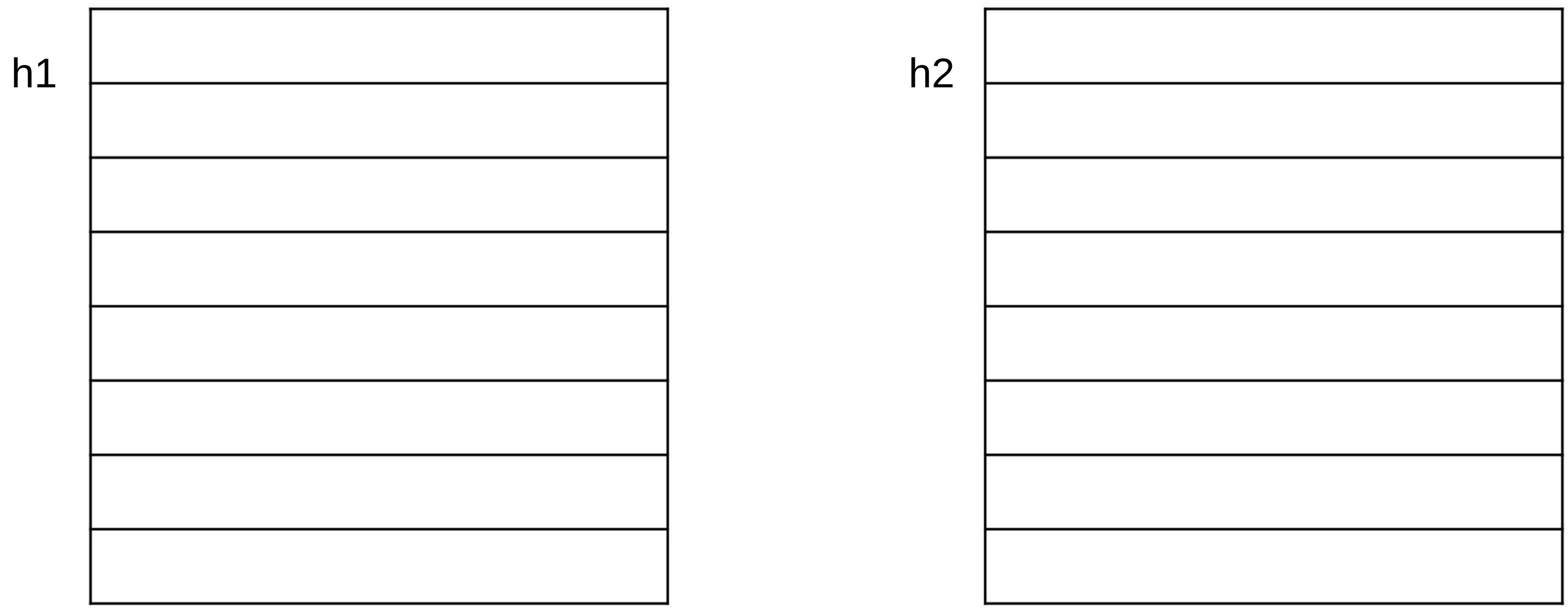
Cuckoo Hashing

A practical implementation of the mathematical concept „universal hashing“

Two arrays (same size in our context. There are also variants in which one of the arrays is larger.)

Two randomly chosen hash functions.

Element is first inserted into the first table. If the place is occupied, then go to the second



Search

universität wien

Trivial

First search in the first table with the first hash function. If unsuccessful, with the second hash function in the second table. If this is also unsuccessful, the element is not included in the hash table

Delete

Trivial

Perform search. If the element is found, mark the corresponding position as free.

Insert

universität wien

What happens if both positions determined for the element are occupied?

The blocking element is inserted into the first table, displacing any element present there, which is then inserted into the second table, and so on.

In pseudocode (let x be the element to be inserted):

If x is already in the table, return Insertion

is x; As long as

(number of loop iterations not too large) { Look at position

h1(To be inserted) in Table 1.

If empty, write Insert there and return.

Otherwise swap Insert and the element stored at the occupied position. Consider position

in h2(Insert) in Table 2.

If empty, write Insert there and return.

Otherwise swap Insert and the element stored in the occupied position

}

Cycle would be recognizable if the element to be inserted originally appears for the third time, but the effort is not worth it.
The size to be chosen here must be of order O(n) for storing n data values .

Insert (2)

If the number of maximum loop runs is reached

two new hash functions are randomly chosen and all elements are rehashed.

This can be done in situ by simply using both

It goes through tables and inserts every element that is not in the right place (according to the new hash functions) .

An attempt is then made again to insert what was originally inserted to insert element.

Expected running time for insertion is O(1) (if the table is less than 50% occupied)

Universal family of hash functions

For addresses with q bits (the size of the tables is then a power of 2)

and w bits for the calculation (e.g. 64 for 64 bit numbers),

randomly choose an odd a with 0 < < 2 and

calculate

ÿ =(ÿ) (2) 2 ÿ

(mod is the modulo operation and div is the integer division).

This can be calculated extremely efficiently!

This family of hash functions is mathematically only almost universal, but is sufficient for our purposes.

A simple change in calculation leads to a truly universal family

ÿ = (ÿ +) with another 2) 2 ÿ

randomly chosen number b with 0 ÿ < 2 ÿ