Cuckoo filter

praktisk bedre enn bloom

Hva løser Cuckoo filter?

Samme som Bloom filter: High speed set membership

Hva trenger vi?

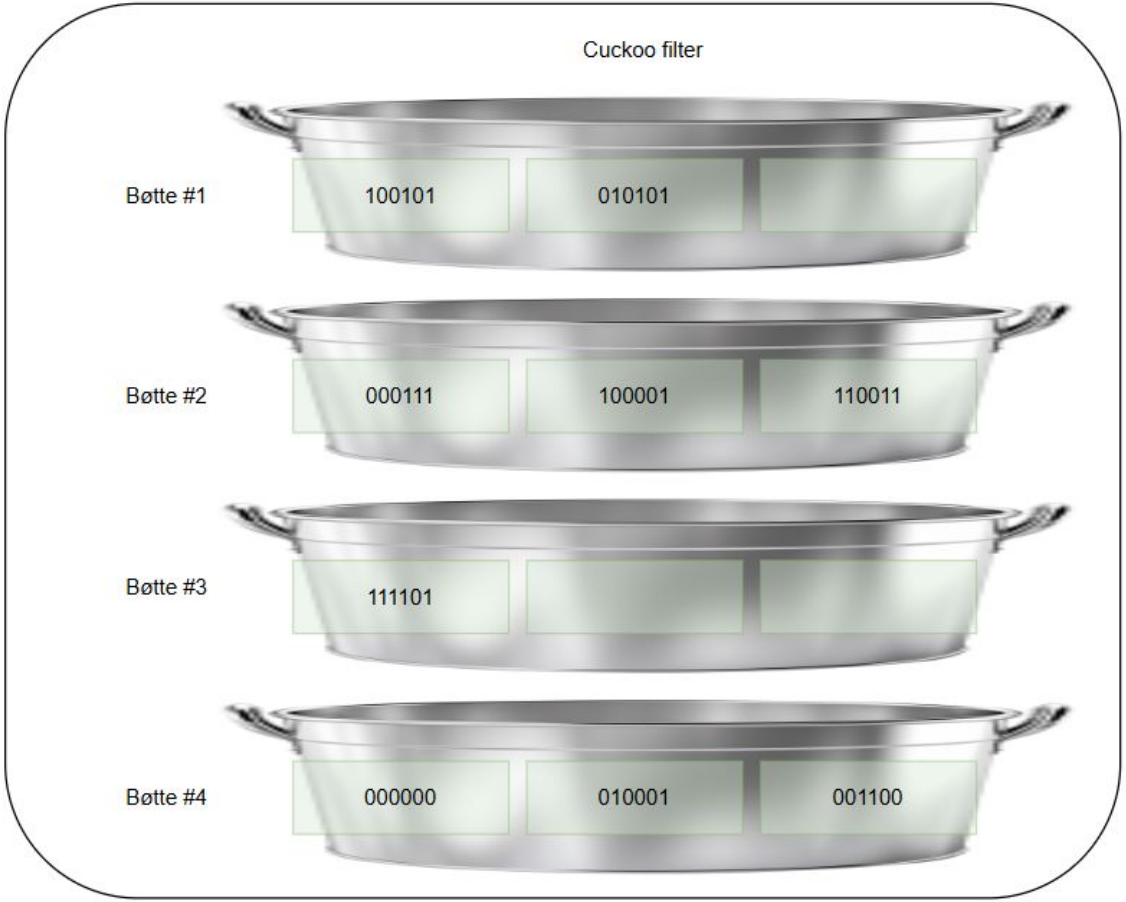
Et array av buckets

hashfunksjon for fingerprint

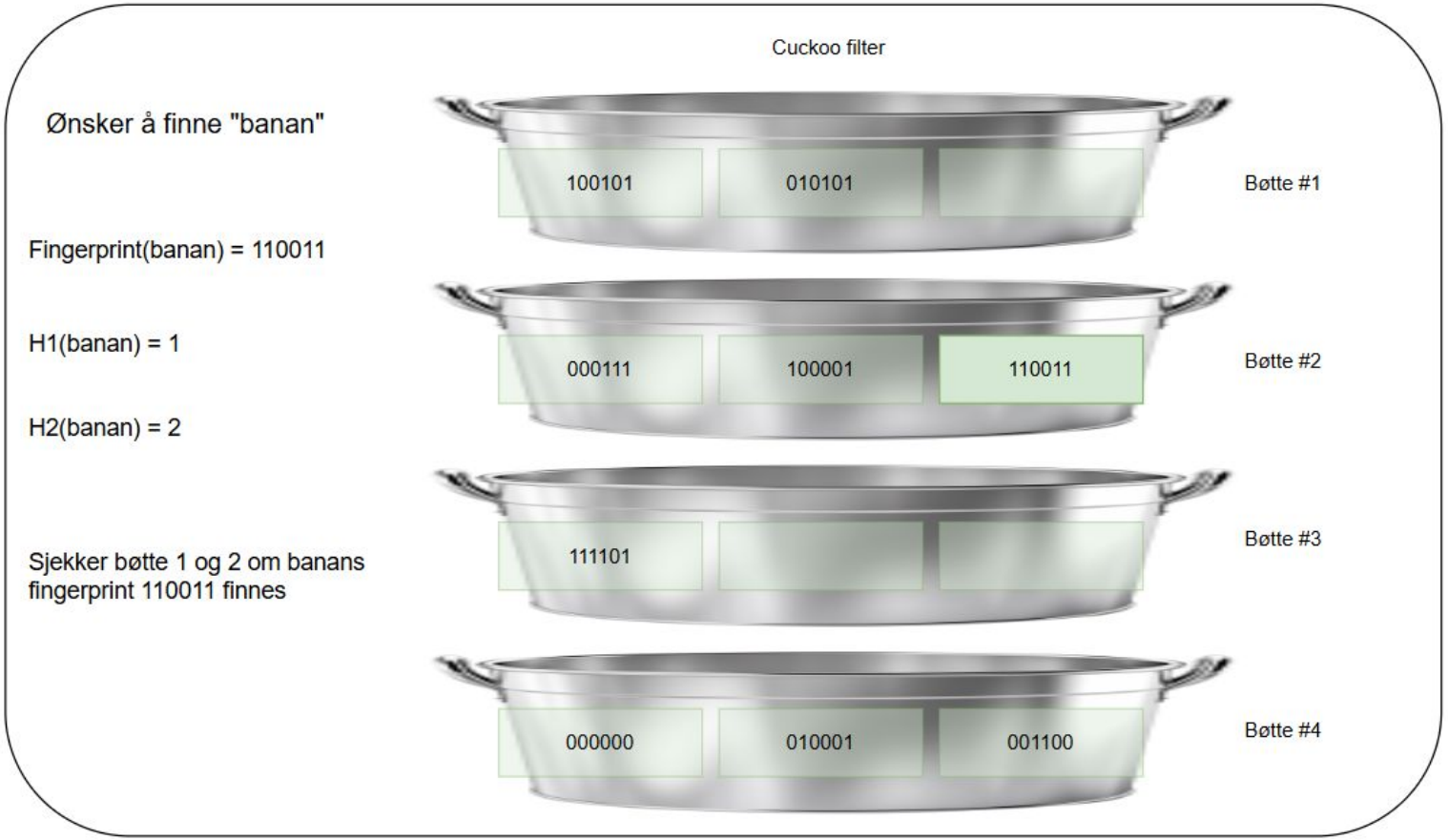
to hash funksjoner h1() og h2():

h1(x) = hash(x)

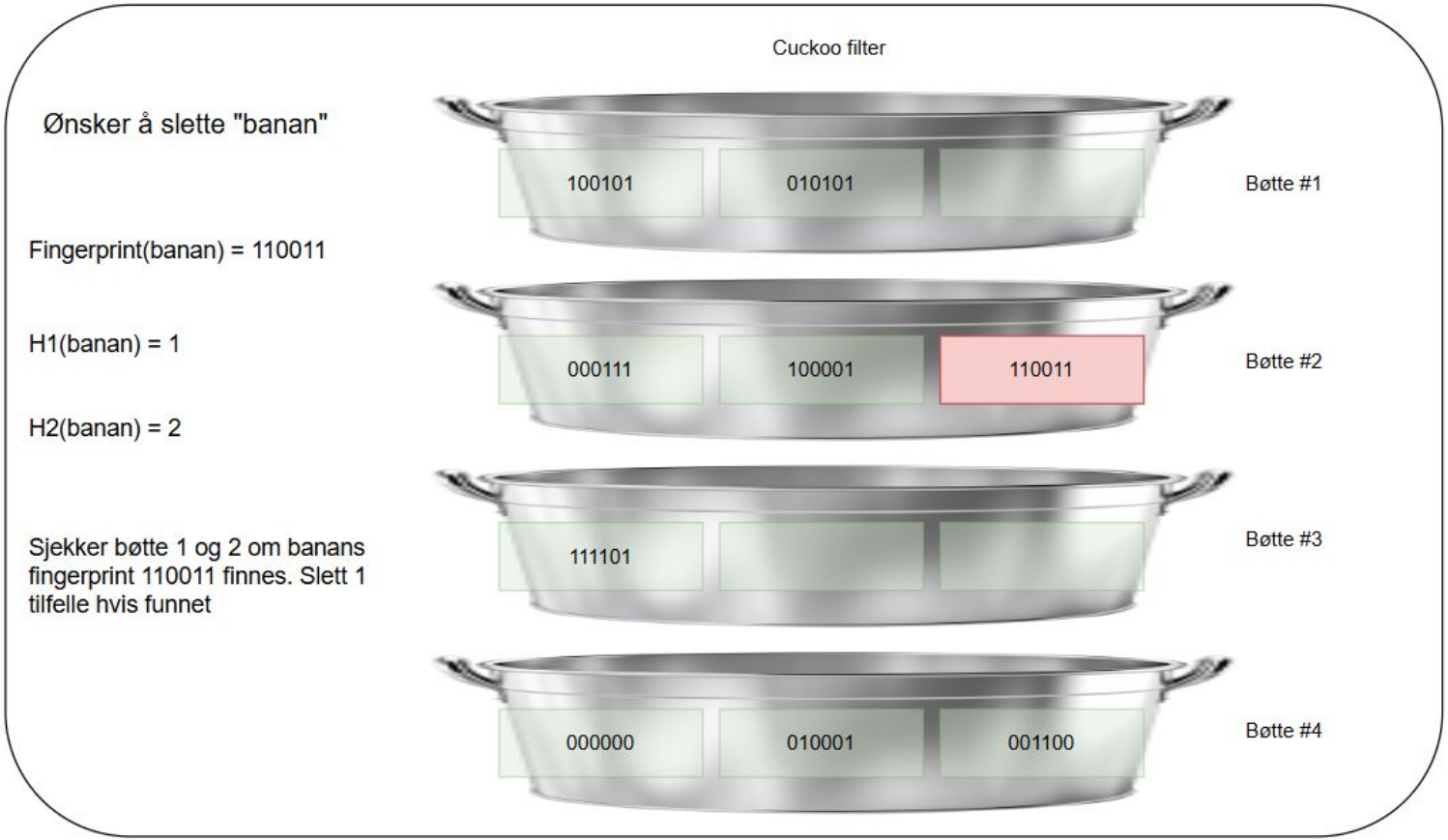
h2(x) = h1(x) XOR hash(fingerprint(x))



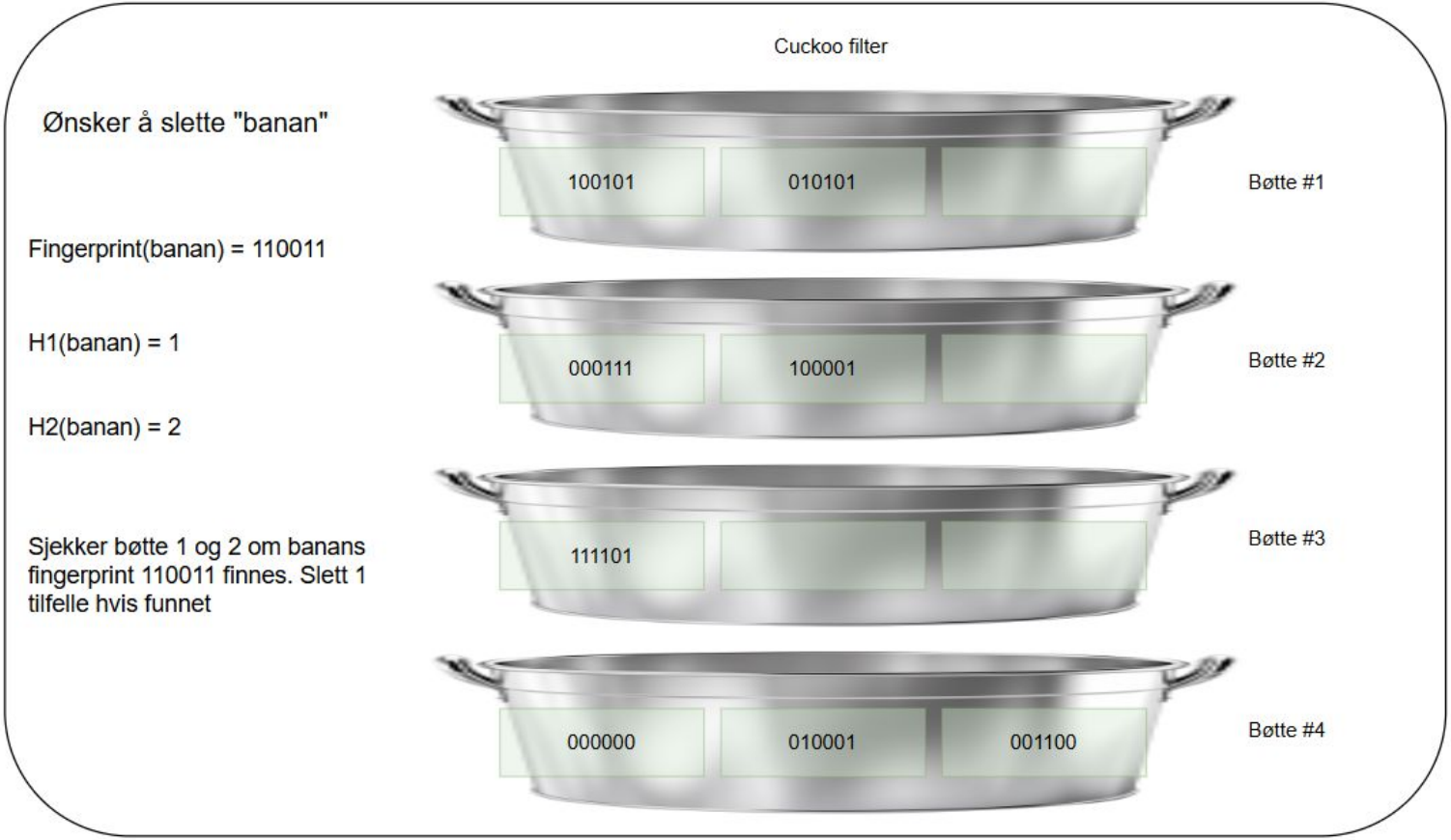
Lookup


Cuckoo filter
Ønsker å finne "banan"
Fingerprint(banan) = 110011
H1(banan) = 1
H2(banan) = 2
Sjekker bøtte 1 og 2 om banans fingerprint 110011 finnes
100101
010101
Bøtte #1
000111
100001
110011
Bøtte #2
111101
Bøtte #3
000000
010001
001100
Bøtte #4

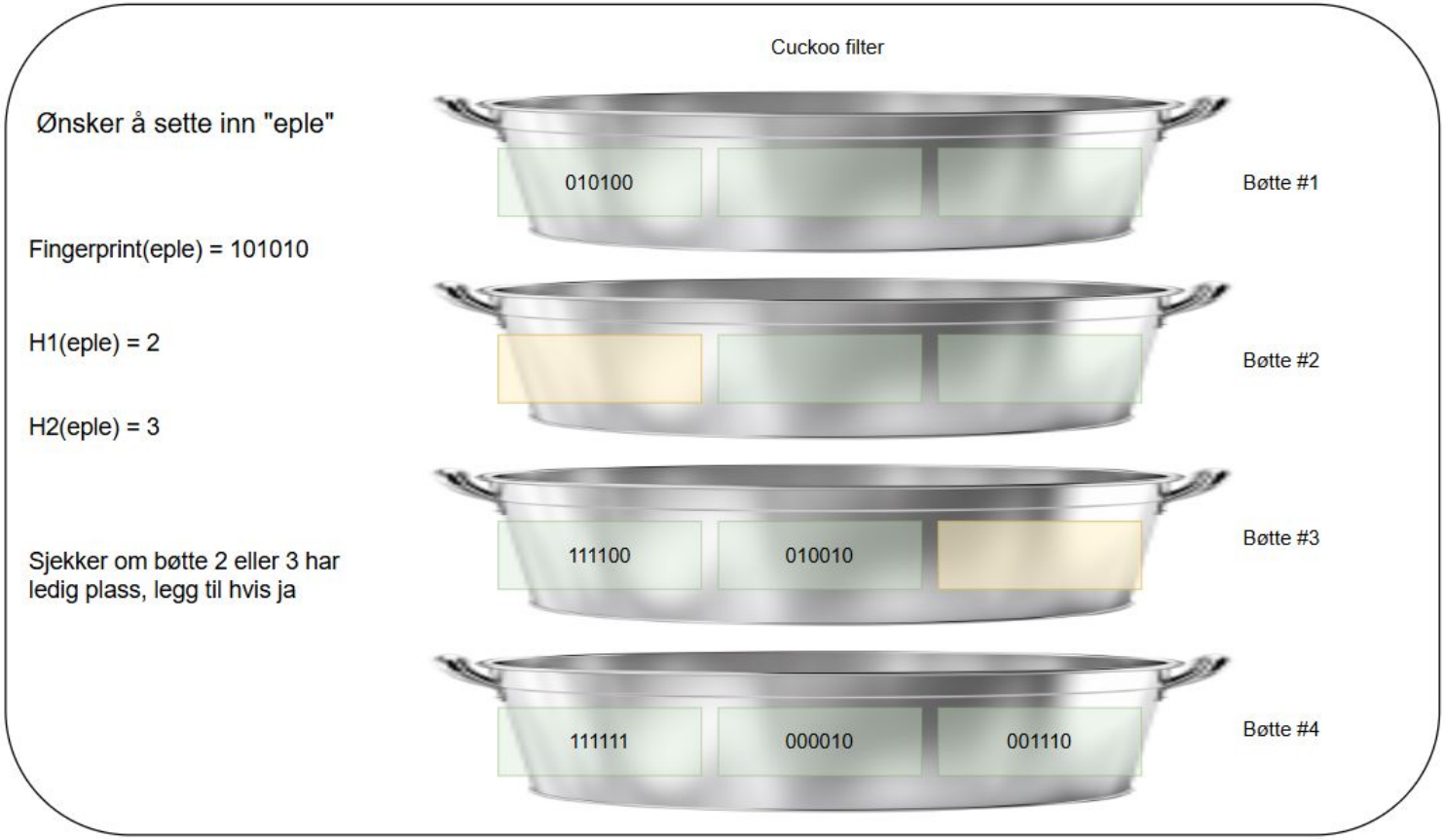
Delete (før)


Cuckoo filter
Ønsker å slette "banan"
Fingerprint(banan) = 110011
H1(banan) = 1
H2(banan) = 2
Sjekker bøtte 1 og 2 om banans fingerprint 110011 finnes. Slett 1 tilfelle hvis funnet
100101
010101
Bøtte #1
000111
100001
110011
Bøtte #2
111101
Bøtte #3
000000
010001
001100
Bøtte #4

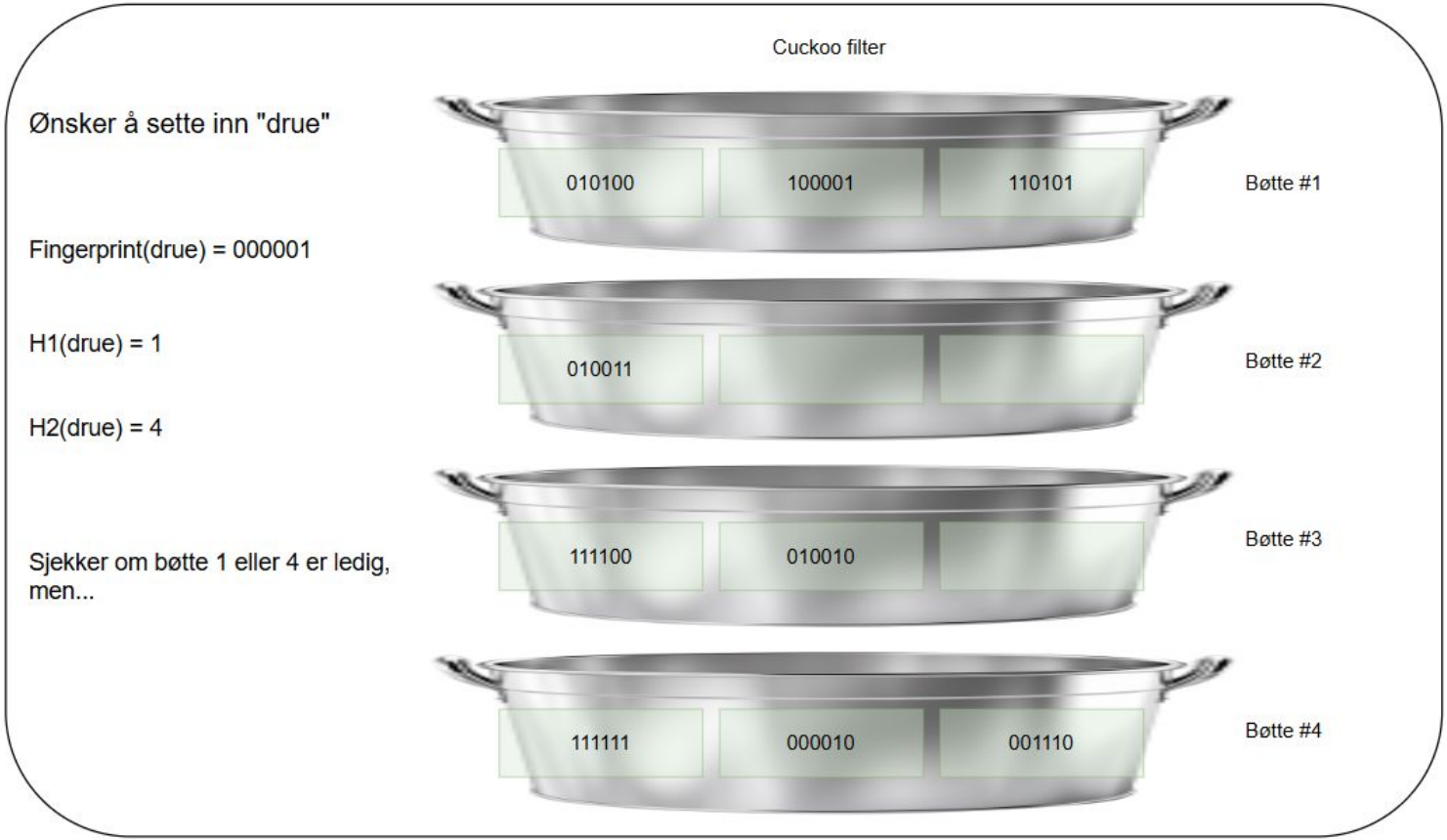
Delete (etter)



Insert (ledig plass)



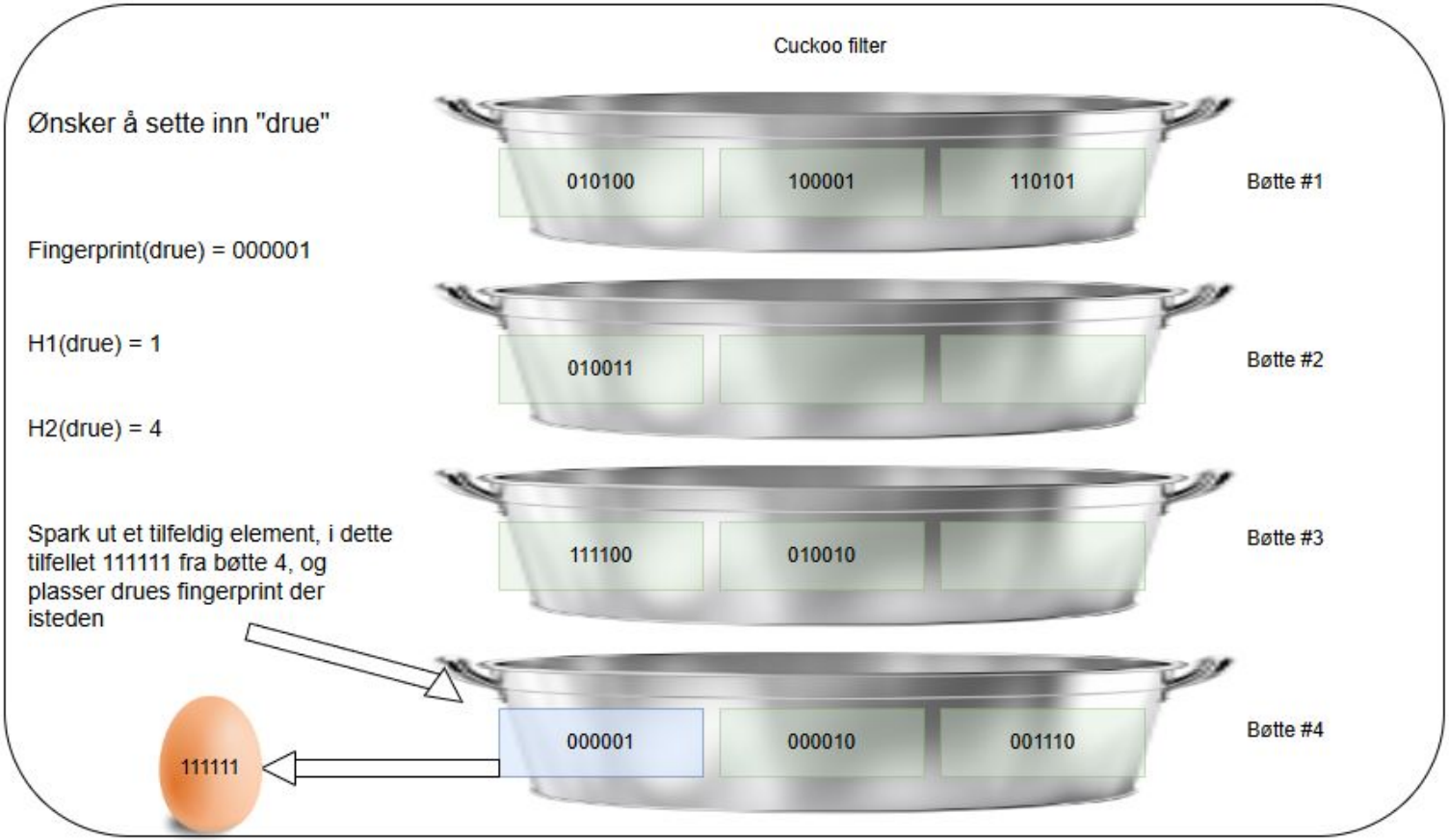
Insert (konflikt)



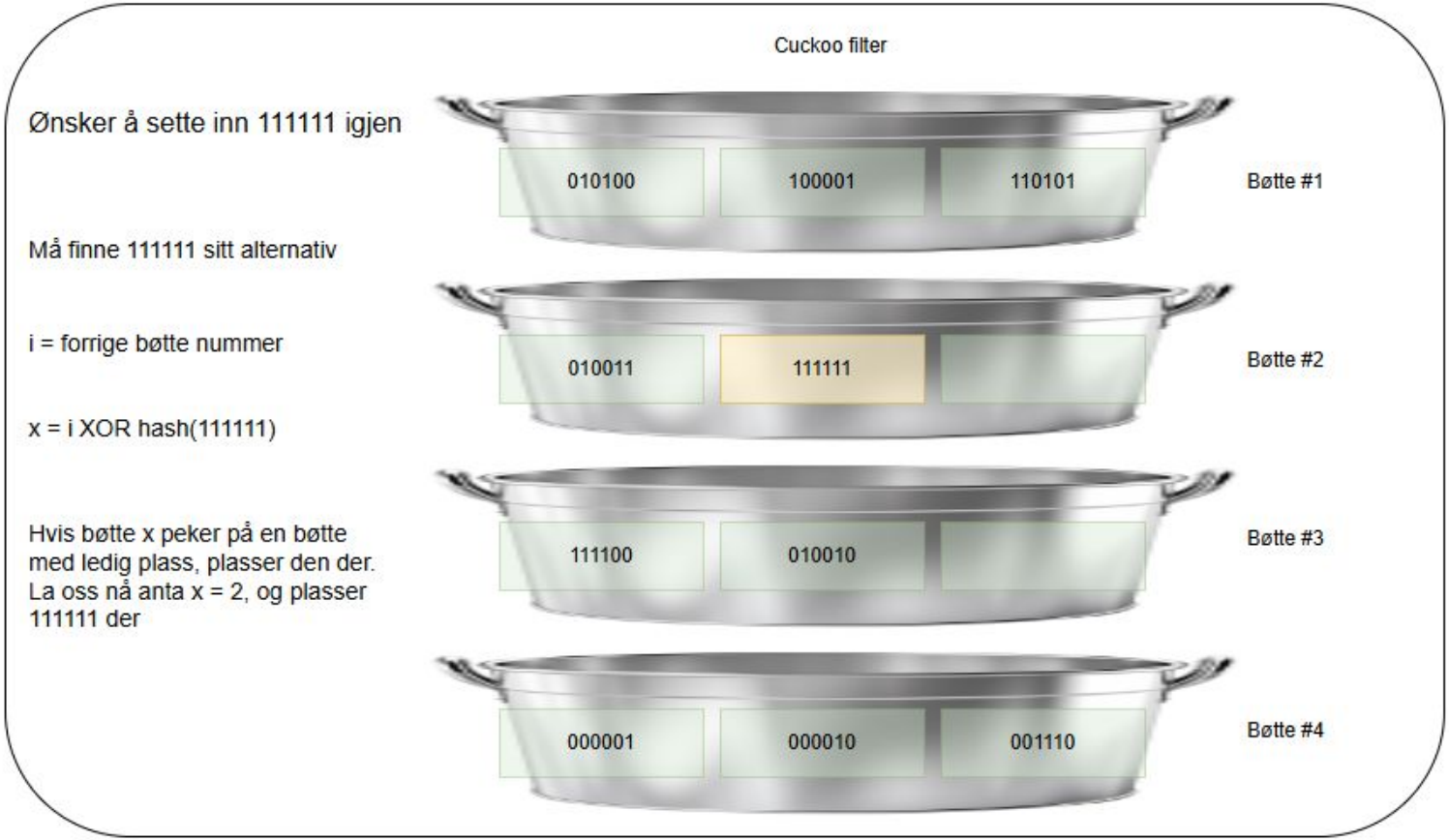
Hva gjør vi da?

Vi tar inspirasjon fra “Cuckoo”-fuglen (gjøk)

Insert (konflikt)



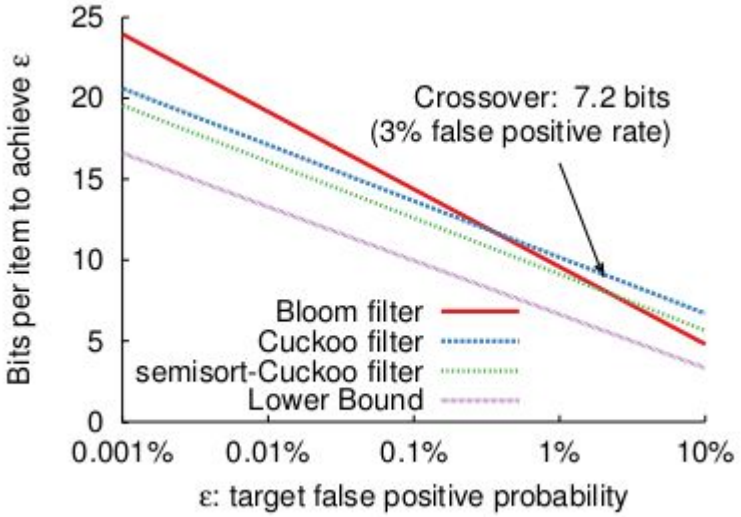
Insert (konflikt, plasser på nytt)



Space efficiency

1.44 *log (1/ϵ) bits optimal Bloom filter

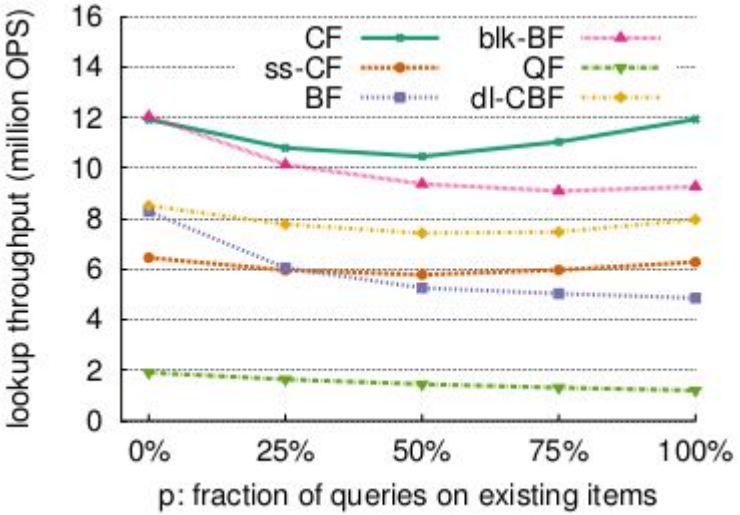
□*log(n) Cuckoo filter



Lookup performance

x akse- forhold mellom positive og negative queries.

y akse - million operations per second



Conclusions

1. Dynamisk sletting
2. Bedre lookup performance
3. Bedre space efficiency