11/10

# Hashing

**Universal hash function family**

$m, n \in \mathbb{Z}^+$

$H$ is set of hash function $\ni$ for random $h \in H$ where $h: \{0,1\}^n \to \{0,1\}^m$

$Pr(h(x) = h(y)) \leq 1/m$

Example: $h_{k,q}(x) = ((kx + q) \bmod p) \bmod m$

w/ $k, q \in \mathbb{Z}_p$, $k \neq 0$

$p$ prime $\geq m$

(Note also common

$(kx \bmod p) \bmod n$

... not universal but collision prob is $2/m$ in expectation)

## Non-Cryptographic hash function

No need to defend against collision resistance

Goal:

- fast to compute
- low probability of collisions

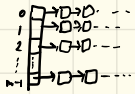
Eg

- FNV
- MurmurHash
- City & Farm Hash

Dictionary data structure
of m buckets labeled $\{0, \ldots, m-1\}$

For table of m buckets
w/ n used keys
Load factor, denoted $\alpha$, is defined as
$\alpha = n/m$

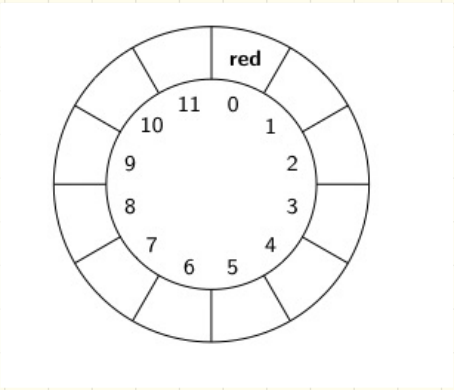
Some props
- $\alpha \in [0, 1]$
- $\alpha$ close to 1 high prob of collisions
- $\alpha$ close to 0 low prob of collisions



Handling collisions
- Closed addressing: $2^{nd}$ary data structure for elemts w/ collision
- Open addressing: store in position that differs from preferred hash
  Eg: Linear probing
  Cuckoo hashing

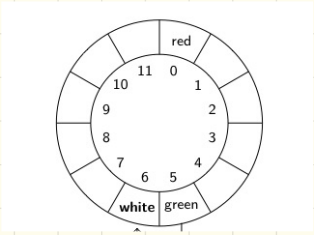
Linear Probing
- Store bins in circular buffer

Insert elt a
- $x = hash(a)$
- if x is empty insert a into bucket x
  else insert a into first bucket after x that is empty




Example insert green w/ hash(green) = 5
insert white w/ hash(white) = 5

Result:



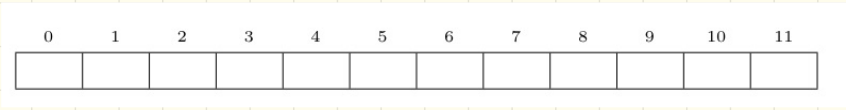

Lookup (a)
- $x = hash(a)$
- start at x, look ccw until find x or empty bin

## Cuckoo hashing

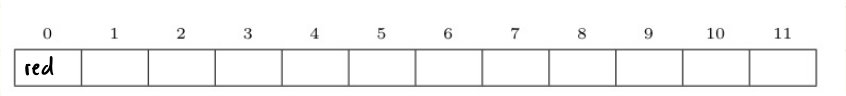
Idea: 2 candidate buckets determined by 2 hash functions $h_1$ & $h_2$

$x_1 = h_1(a)$
$x_2 = h_2(a)$

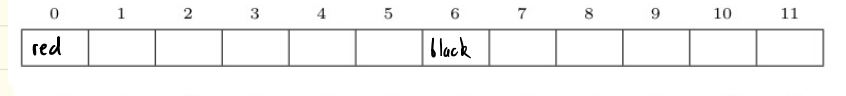
if $x_1$ is empty insert in $x_1$
  $x_2$ is empty insert into $x_2$
else randomly pick $x_1$ or $x_2$ (call $X$)
  move elt from $X$ to alt bucket
  repeat until alt bucket is found (or max # of iters ... considered full)



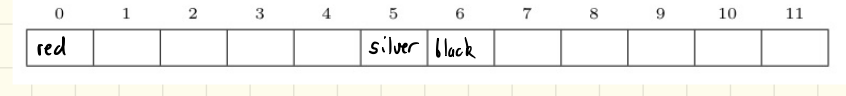

$h_1(\text{red}) = 0$
$h_2(\text{red}) = 8$



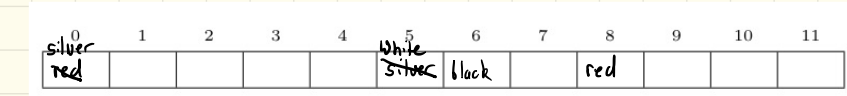

$h_1(\text{black}) = 6$
$h_2(\text{black}) = 0$




$h_1(\text{silver}) = 5$
$h_2(\text{silver}) = 0$




$h_1(\text{white}) = 5$
$h_2(\text{white}) = 6$




lookup (a)
- checks $h_1(a)$ and $h_2(a)$

ext time: Set membership