Hashing

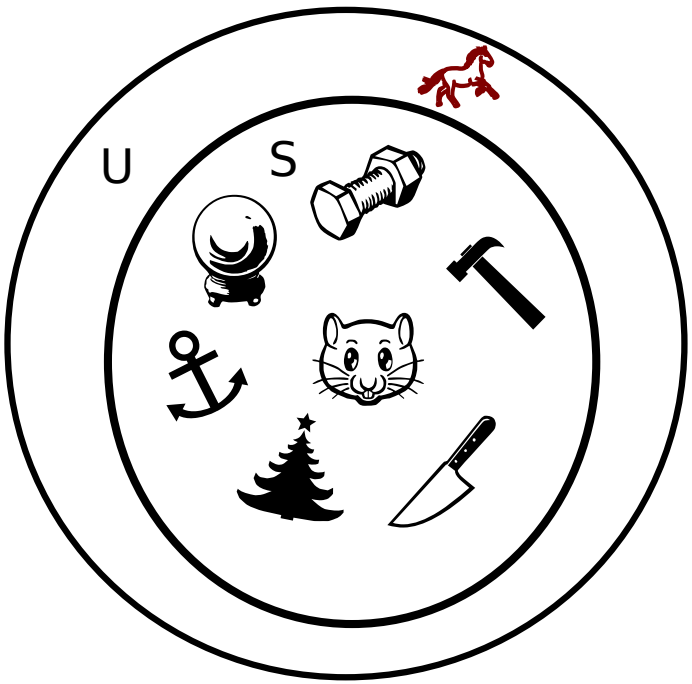
- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

Hashing

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

Ordbøger

Algoritmisk problem: vedligehold en *dynamisk mængde* $S \subseteq U$.



insert(🔨).

search(🌲)? JA: i S.

search(🐎)? NEJ: Ikke i S.

(Eksempel: Univers bestående af unicode-tegn.)

Ordbøger

Ordbog: Vedligehold en dynamisk mængde S. Hvert element x har en nøgle $x.\text{key}$ fra et univers U, og noget satellitdata $x.\text{data}$.
Operationer:

- search(k) afgør, om et element x with $x.\text{key} = k$ findes, og returner det.
- insert(x) tilføj x til mængden S.
- delete(x) fjern x fra mængden S.

Eksempel. Telefonnumre. U = alle ottecifrede tal.
$x \in S$ er på formen $(x.\text{key}, x.\text{data})$ dvs. (nummer, navn).

(22773456, Alice)
(33881357, Bob)
(25178809, Don)
(26670007, Carl)

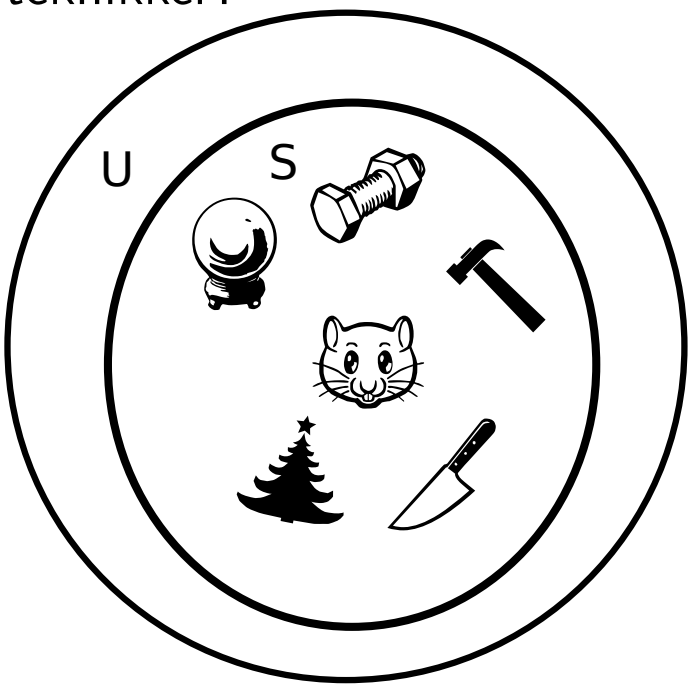
search(22773456)? Alice.
search(98436512)? Ikke i mængde.
insert(25178809, Carl)
delete(33881357)

Ordbøger

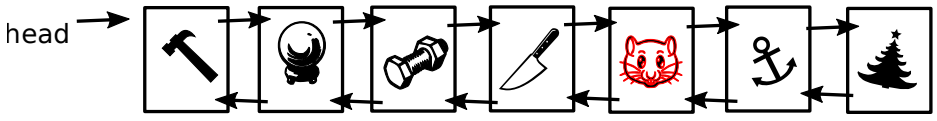
Anvendelser

- Grundlæggende datastruktur til at repræsentere en mængde,
- Bruges i mange algoritmer og datastrukturer.

Udfordring: hvordan kan vi implementere ordbøger med nuværende teknikker?



Ordbøger - løsning med hægtet liste - for langsom!



search()

Time:

- search(k) - Lineær søgning. $O(|S|)$ tid.
- insert(x) - Indsæt i starten af listen. $O(1)$ tid.
- delete(x) - Ændre hægter. $O(1)$ tid.

Plads: $O(|S|)$ plads.

Dictionaries - løsning med et array - for stor!

- A er et array af længde U
- Gem x på position $A[x.key]$ i A.

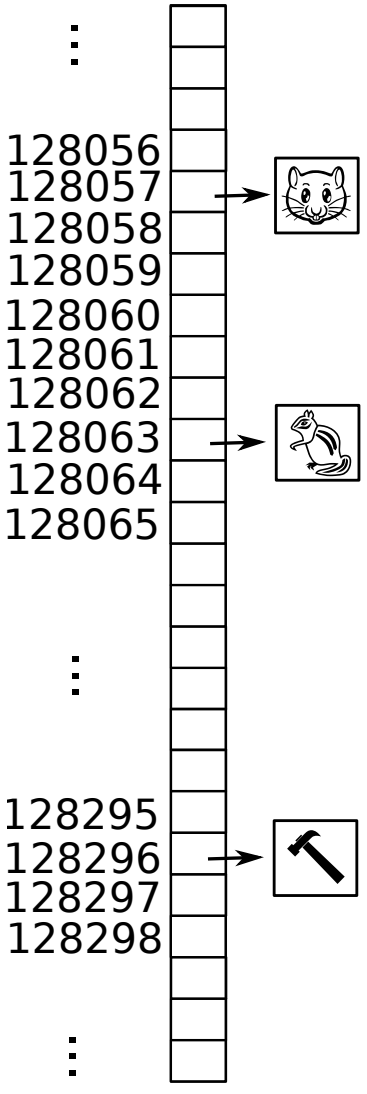
search(k) - returner $A[k]$ – $O(1)$ tid.

insert(x) - sæt $A[x.\text{key}] = x$ – $O(1)$ tid.

delete(x) - sæt $A[x.\text{key}] = null$. – $O(1)$ tid.

Plads: $O(|U|)$

Spørgsmål: Hvornår er dette et problem?
Særligt når U ikke kan være i RAM, men S kan!



Ordbøger – to utilfredsstillende løsninger

Datastruktur	`Search`	`Insert`	`Delete`	Plads
Hægtet liste	$O(\|S\|)$	$O(1)$	$O(1)$	$O(\|S\|)$
Array	$O(1)$	$O(1)$	$O(1)$	$O(\|U\|)$

Udfordring: Kan vi gøre det bedre?

Hashing

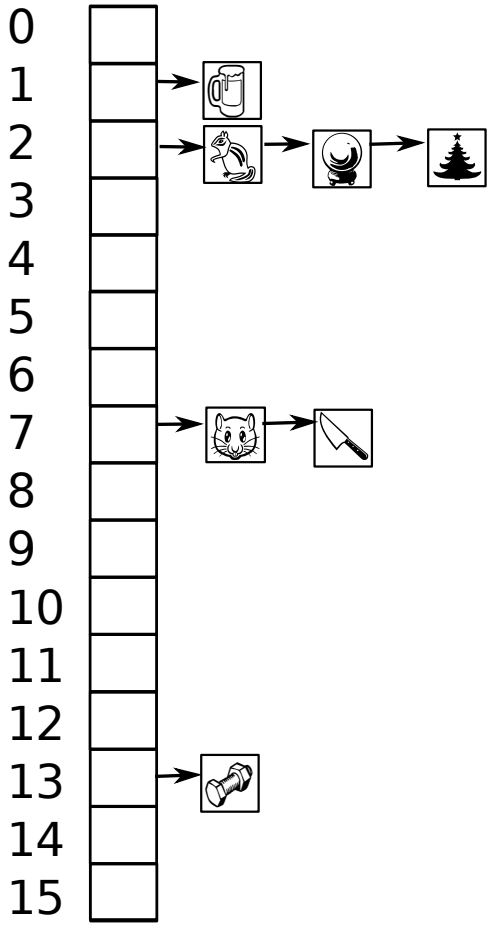
- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

Hægtet hashing

Ide: Brug en hashfunktion $h : U \to \{0, \ldots, m-1\}$ hvor $m = O(|S|)$.

- Vedligehold et array A af længde m,
- Hver indgang i arrayet peger på en hægtet liste.
- Elementet x er gemt et sted i den hægtede liste ved $A[h(x.key)]$.

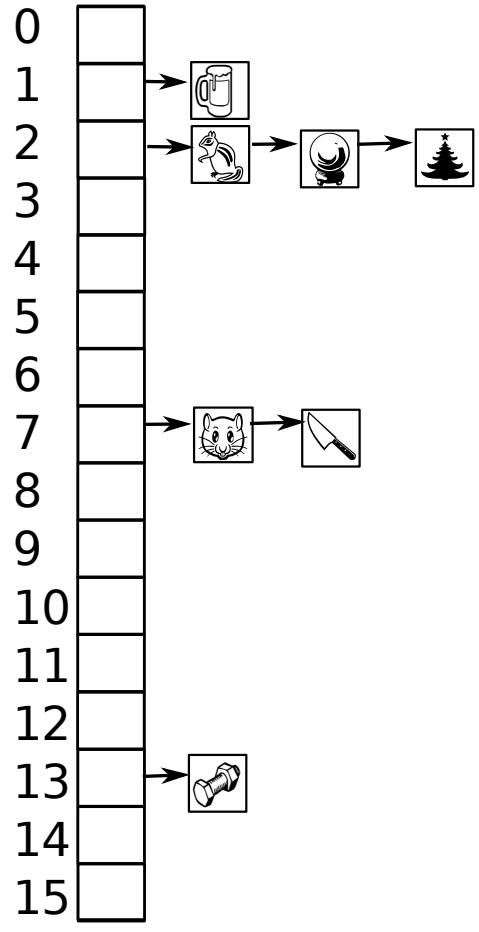
Kollision: Når $m < |U|$, så må der findes $x.key \neq y.key$, hvor $h(x.key) = h(y.key)$.
Dette kaldes en *kollision*.
Ønsker funktion h, så der er få kollisioner.
Hash (vb tr) "to confuse, muddle, or mess up".



Hægtet hashing

Hvordan det virker.

- insert(⚓)
  h(⚓) = 0
- insert(🔨)
  h(🔨) = 7
- search(🐎)
  h(🐎) = 15
- search(🛼)
  h(🛼) = 2



Hægtet hashing

Hvordan det virker.

search(k) - gennemsøg $A[h(k)]$'s liste efter k.

insert(x) - indsæt x i $A[h(x.key)]$'s liste.

delete(x) - slet x fra liste.

Time:

search(k) - $O(|\text{listens længde}|)$ tid

insert(x) - $O(1)$ tid (indsæt forrest)

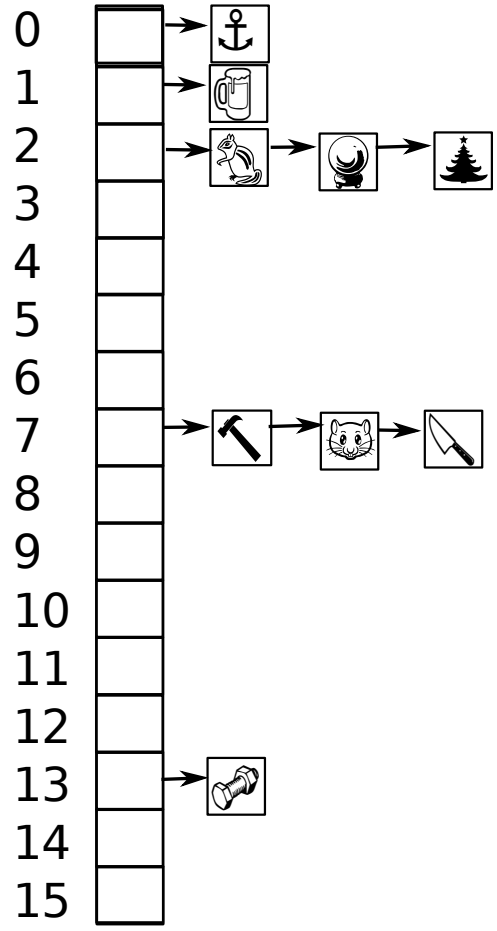
delete(x) - $O(1)$ tid (ændre hægter).

Plus den tid det tager at beregne $h(x.key)$

Plads:

$O(m + |S|) = O(|S|)$

Plus pladsen for at gemme h



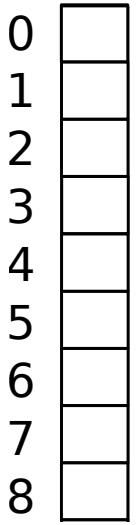
Hægtet hashing - Øvelse

Indsæt de følgende nøgler K i en hashtabel af størrelse 9 med hægtet hashing ved brug af den følgende hashfunktion:

$$h(k) = k \mod 9$$

$K = 5, 28, 19, 15, 20, 33, 12, 17, 10$

Hvor lang bliver den længste hægtede liste?



Hægtet hashing

How it works.

search(k) - gennemsøg $A[k]$'s liste efter k.

insert(x) - indsæt x i $A\,[h(x.key)]$'s liste.

delete(x) - slet x fra liste.

Time:

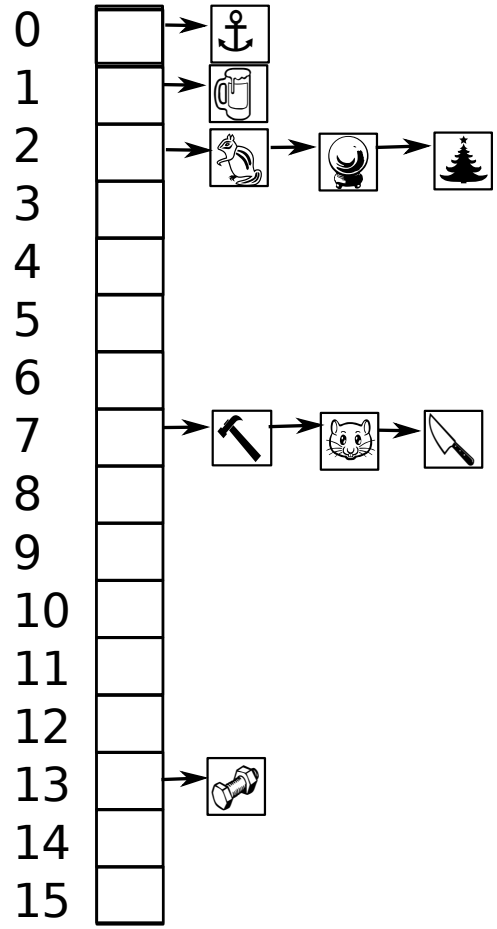
search(k) - $O(|\text{listens længde}|)$ tid

insert(x) - $O(1)$ tid (indsæt forrest)

delete(x) - $O(1)$ tid (ændre hægter).

Plads:
$O(m + |S|) = O(|S|)$

Så hvor lang er den liste? – Kommer an på h.



Hashing

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Linear probering

Eksempel – dårlig hash function.

(22773456, Alice)	2
(33881357, Bob)	3
(25178809, Don)	2
(26670007, Carl)	2
(45251199, DTU Study Guidance)	4
(45257246, SU Office)	4
(45873585, MizzPizza)	4
(45257300, DTU Study Administration)	4

Ide: $|S| \leq 10$, så lad os hashe et nummer til dets første ciffer.
Er det en god ide? (nej.)

Uniform hashing

Imagine there's a
uniform hash function $h : U \to \{0, \ldots, m-1\}$.

Uniform hashing

Definition (Belastningsfaktor (load factor))

$\alpha = |S|/m$. Gennemsnitslængden af listerne.

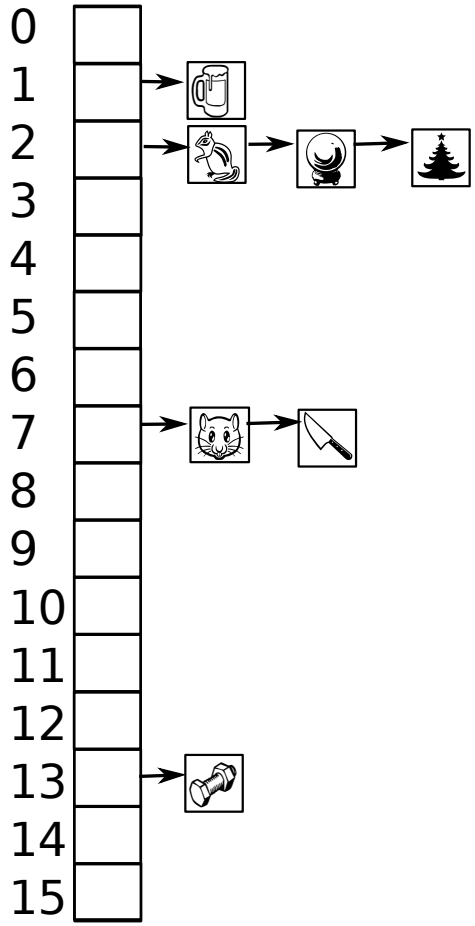
$m = \Theta(n) \Rightarrow \alpha = \Theta(1)$.

Drømmescenarie: Forestil dig at

- For hvert $x \in U$ er $h(x)$ uafhængigt uniformt tilfældig i $\{0, \ldots, m-1\}$.

Så:

- Forventet længde af hver liste $= \alpha$.
- $\Rightarrow$ search(k) tager $O(\alpha) = O(1)$ tid.
- Search, Insert, Delete: $O(1)$ tid.
  $O(|S|)$ plads. (Stadigvæk)



Ordbøger – to utilfredsstillende og en fantasiløsning

Datastruktur	`Search`	`Insert`	`Delete`	Plads
Hægtet liste	$O(\|S\|)$	$O(1)$	$O(1)$	$O(\|S\|)$
Array	$O(1)$	$O(1)$	$O(1)$	$O(\|U\|)$
Hægtet hashing	$O(1)^\dagger$	$O(1)$	$O(1)$	$O(\|S\|)$

†: Forventet køretid. Antager uniform hashing.

Udfordring: Find en hashfunktion som virker og som rent faktisk findes.

Universal hashing

Mål: Undgå kollisioner $h(y) = h(x)$ for $x \neq y$.
Hvis $h(x)$ og $h(y)$ er uafhængig uniform tilfældige, så er

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

Definition (Universal hashfunktion)

h er universal hvis for alle $x, y \in U$ hvor $x \neq y$, $\Pr[h(x) = h(y)] \leq 1/m$

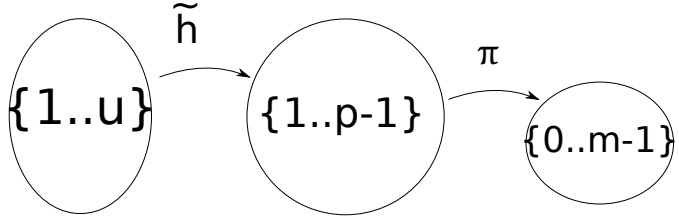
Alle operationer i (forventet) $O(1)$ tid!

Universal hashing - eksempel

multiply-mod-prime

$$h_{a,b}(x) = ((ax + b) \mod p) \mod m$$

Vælg h tilfældigt ved at sætte $a \in \{1, \ldots, p-1\}$ og $b \in \{0, \ldots, p-1\}$ uafhængigt uniformt tilfældigt.



Hashfunktioner

Ikke-heltal? Alt er gemt som bits og kan derfor fortolkes som heltal.
E.g. strenge: Hvert bogstav svarer til et tal. Tag "CLRS":

- 256 ASCII-koder for bogstaver.
- C $= 67$, L $= 76$, R $= 82$ og S $= 83$.
- $\Rightarrow$ "CLRS"
  $= 67 \cdot 256^3 + 76 \cdot 256^2 + 82 \cdot 256^1 + 83 \cdot 256^0 = 1129075283$

Hashing

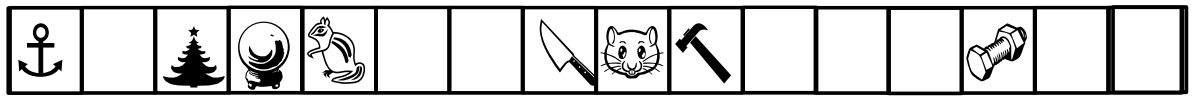
Datastruktur	`Search`	`Insert`	`Delete`	Plads
Hægtet liste	$O(\|S\|)$	$O(1)$	$O(1)$	$O(\|S\|)$
Array	$O(1)$	$O(1)$	$O(1)$	$O(\|U\|)$
Hægtet hashing	$O(1)^\dagger$	$O(1)$	$O(1)$	$O(\|S\|)$

†: Forventet køretid.

Hashing

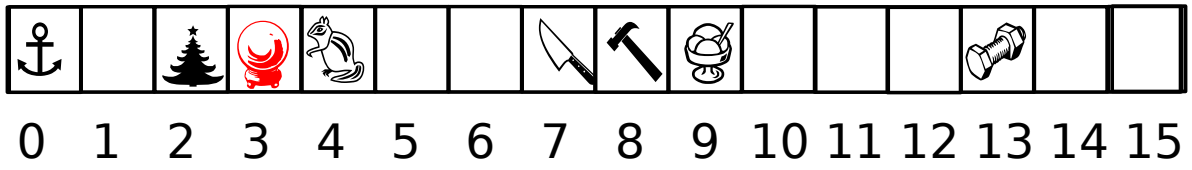
- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

Lineær probering - ide



- Vedligehold et array af længde m
- Idea: Gem x på plads $A[h(x.key)]$
- Udfordring: Kollisioner.

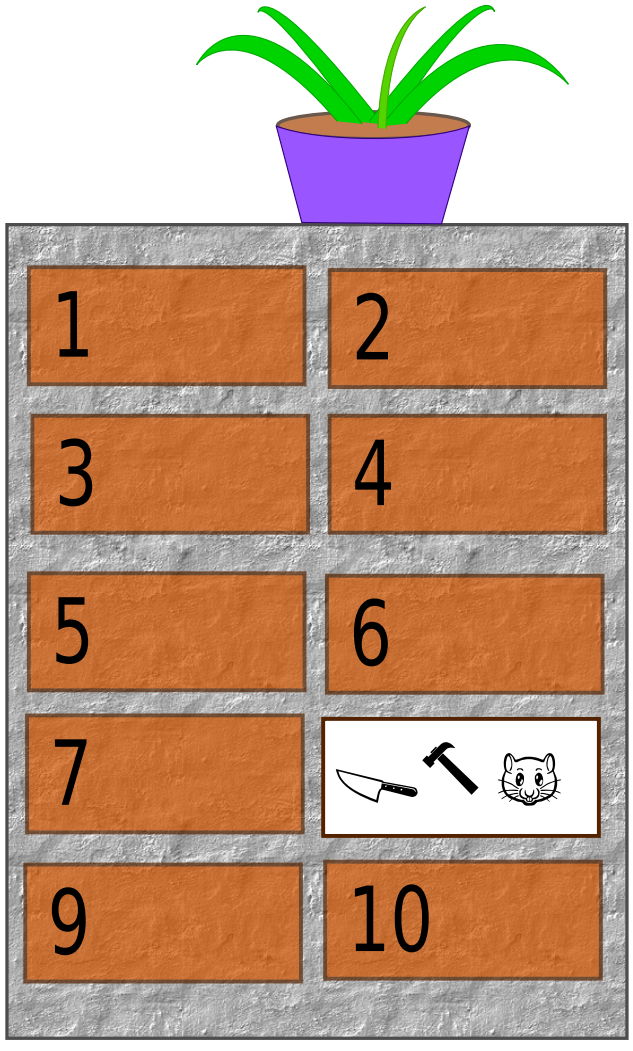
Lineær probering



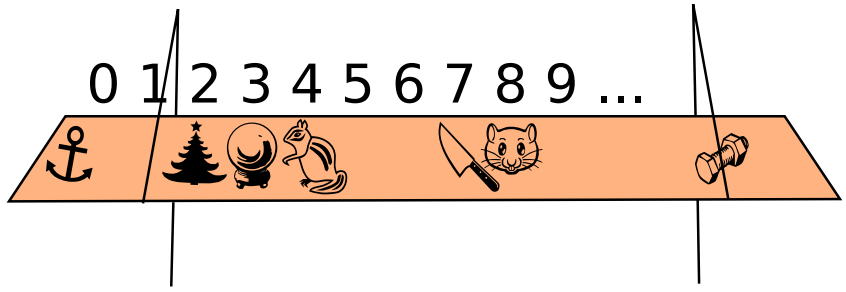
- Vedligehold et array af længde m
- En *klynge* (cluster) er en sekvens af konsekutive ikke-tomme pladser.
- Gem x på plads $A[h(x.key)]$, eller et sted i den klynge, der indeholder $h(x.key)$, til højre for $h(x.key)$.

Example:
Insert(). h() = 8.
Delete(). h() = 7. h()=7.
Search(). h() = 2.
Plads: $m = O(|S|)$. Tid: $O(|\text{cluster}|)$. $\leftarrow O(1)$ for velvalgte hashfunktioner h.

Analogier



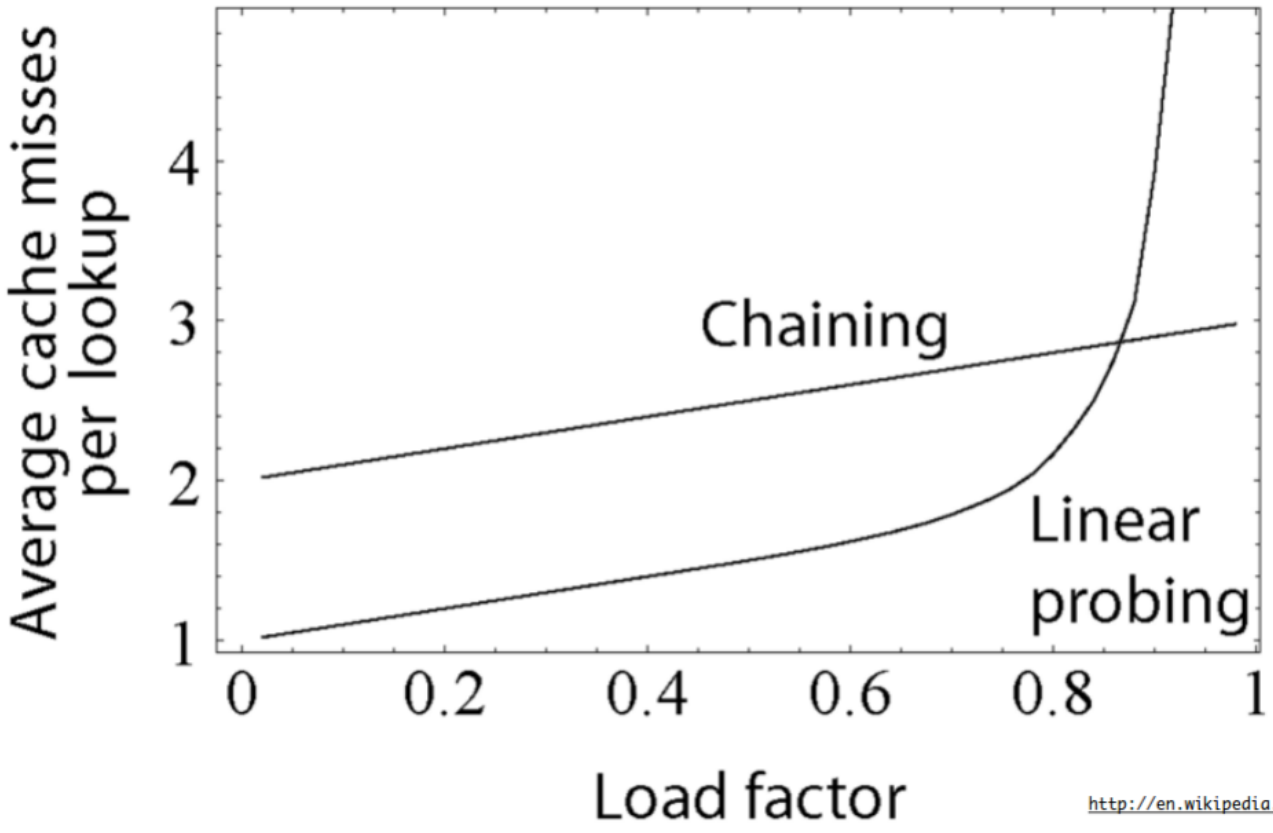
Hægtet hashing: Som en kommode.
Nødt til at søge lineært gennem skuffe nr. 8 for at finde 



Lineær probering: Som en hylde.
Ikke plads til  på $h($$) = 7$,
Så indsæt  på den nærmeste ledige plads til højre for.

Lineær probering

Kæmpe fordel: Lineær probering er cacheeffektivt.



http://en.wikipedia.org/wiki/Hash_table

Hashing

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

Lineær probering - Øvelse

Indsæt følgende nøgler K i et array af længde 9 ved lineær probering med hashfunktionen

$$h(k) = k \mod 9$$

$K = 5, 28, 19, 15, 20, 33$

Derefter, slet 28.

Hvor mange indgange er besøgt hvis man leder efter nøglen 1?