

# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

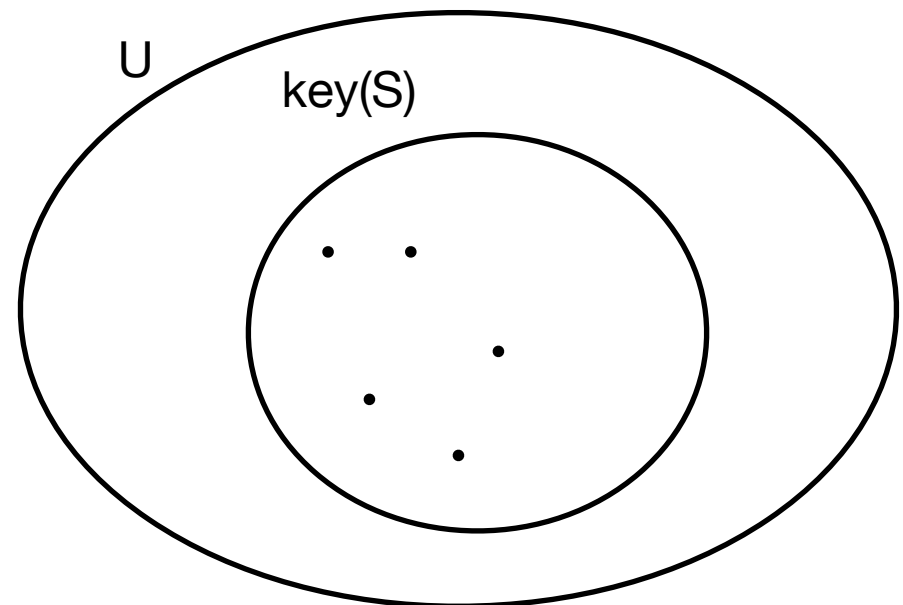
# Ordbøger

---

- **Ordbøger.** Vedligehold en dynamisk mængde  $S$  af elementer. Hvert element har en nøgle  $x.key$  fra et **univers** af nøgler  $U$  og satellitdata  $x.data$ .
- **Ordbogsoperationer.**
  - $SEARCH(k)$ : afgør om element med nøgle  $k$  findes i  $S$ , og returner elementet.
  - $INSERT(x)$ : tilføj  $x$  til  $S$  (vi antager  $x$  ikke findes i forvejen)
  - $DELETE(x)$ : fjern  $x$  fra  $S$ .

- **Eksempel.**

- $U = \{0, \dots, 99\}$
- $key(S) = \{1, 13, 16, 41, 54, 66, 96\}$



# Ordbøger

---

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

# Ordbøger

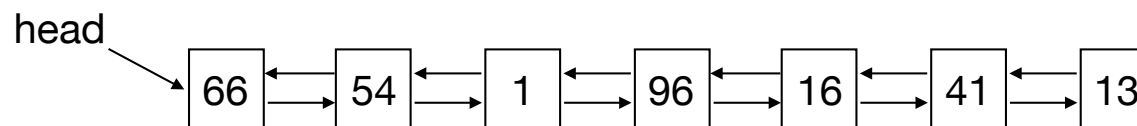
---

- **Udfordring.** Hvordan kan vi løse problemet med nuværende teknikker?

# Ordbøger

---

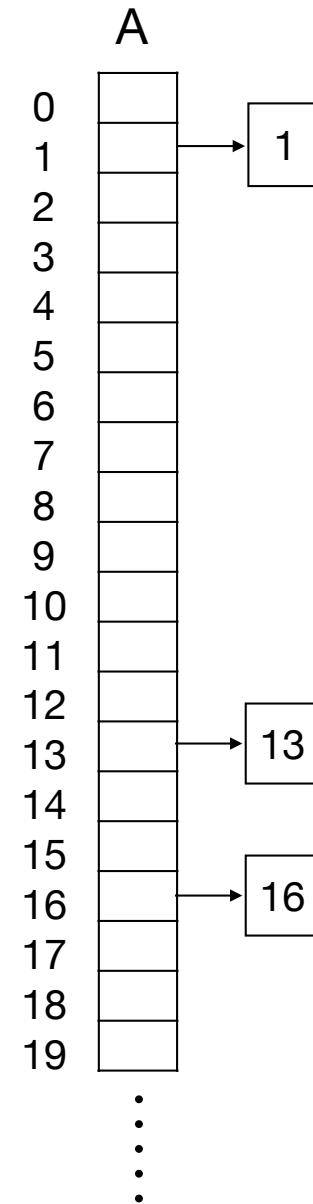
- **Løsning med hægtet liste.** Gem S i hægtet liste.



- SEARCH(k): lineær søgning i listen efter nøgle k.
- INSERT(x): Indsæt x i start af liste.
- DELETE(x): fjern x fra liste.
- **Tid.**
  - SEARCH i  $O(n)$  tid.
  - INSERT og DELETE i  $O(1)$  tid.
- **Plads.**
  - $O(n)$ .

# Ordbøger

- Løsning med direkte adressering (*direct addressing*).
  - Gem  $S$  i tabel  $A$  af størrelse  $U$ .
  - Gem element  $x$  på position  $A[\text{key}[x]]$
- SEARCH( $k$ ): returner  $A[x.\text{key}]$ .
- INSERT( $x$ ): Sæt  $A[x.\text{key}] = x$ .
- DELETE( $x$ ): Sæt  $A[x.\text{key}] = \text{null}$ .
- Tid.
  - SEARCH, INSERT og DELETE i  $O(1)$  tid.
- Plads.
  - $O(|U|)$



# Ordbøger

---

Datastruktur	SEARCH	INSERT	DELETE	Plads
hægtet liste	$O(n)$	$O(1)$	$O(1)$	$O(n)$
direkte adressering	$O(1)$	$O(1)$	$O(1)$	$O( U )$

- **Udfordring.** Kan vi gøre det betydeligt bedre?



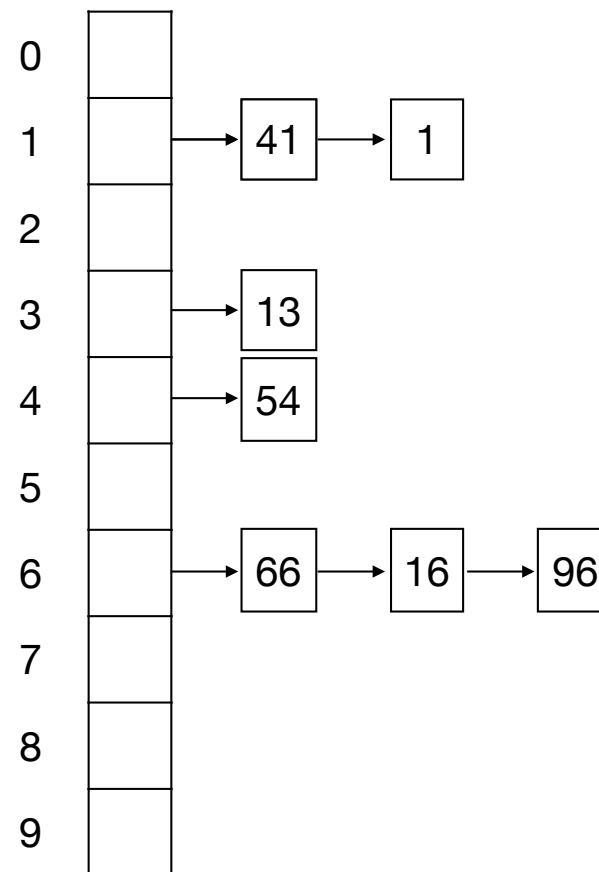
# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

# Hægtet hashing

- **Ide.** Find en **hashfunktion**  $h : U \rightarrow \{0, \dots, m-1\}$ , hvor  $m = \Theta(n)$ . Hashfunktion skal fordele nøglerne fra  $S$  **nogenlunde jævnt** over  $\{0, \dots, m-1\}$ .
- Hashing = forkludre, sprede, bikse.
- **Hægtet hashing (chained hashing).**
  - Vedligehold tabel  $A[0..m-1]$ .
  - Element  $x$  gemt i **hægtet liste** på  $A[h(x.key)]$ .
- **Kollision.**
  - $x$  og  $y$  **kolliderer** hvis  $h(x.key) = h(y.key)$ .
- **SEARCH(k):** lineær søgning i liste  $A[h(k)]$  efter nøgle  $k$ .
- **INSERT(x):** Indsæt  $x$  i start af liste  $A[h(x.key)]$ .
- **DELETE(x):** fjern  $x$  fra liste  $A[h(x.key)]$ .



$U = \{0, \dots, 99\}$

$key(S) = \{1, 13, 16, 41, 54, 66, 96\}$

$m = 10$

$h(k) = k \text{ mod } 10$

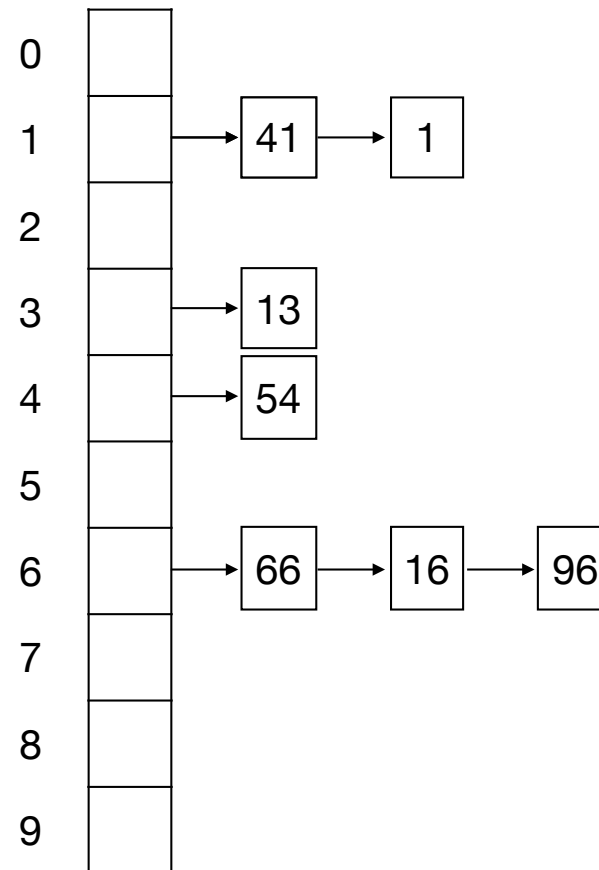
# Hægtet hashing

---

- SEARCH(k): lineær søgning i liste  $A[h(k)]$  efter nøgle k.
- INSERT(x): Indsæt x i start af liste  $A[h(x.key)]$ .
- DELETE(x): fjern x fra liste  $A[h(x.key)]$ .
  
- **Opgave.** Indsæt følgende nøglesekvens  $K = 5, 28, 19, 15, 20, 33, 12, 17, 10$  i en hashtabel af størrelse 9 vha. hægtet hashing med hashfunktionen  $h(k) = k \bmod 9$ .

# Hægtet hashing

- SEARCH(k): lineær søgning i liste  $A[h(k)]$  efter nøgle k.
- INSERT(x): Indsæt x i start af liste  $A[h(x.key)]$ .
- DELETE(x): fjern x fra liste  $A[h(x.key)]$ .
- Tid.
  - SEARCH i  $O(\text{længde af liste})$  tid.
  - INSERT og DELETE i  $O(1)$  tid.
  - Længde af lister er **afhængig** af hashfunktion.
- Plads.
  - $O(m + n) = O(n)$ .



$U = \{0, \dots, 99\}$

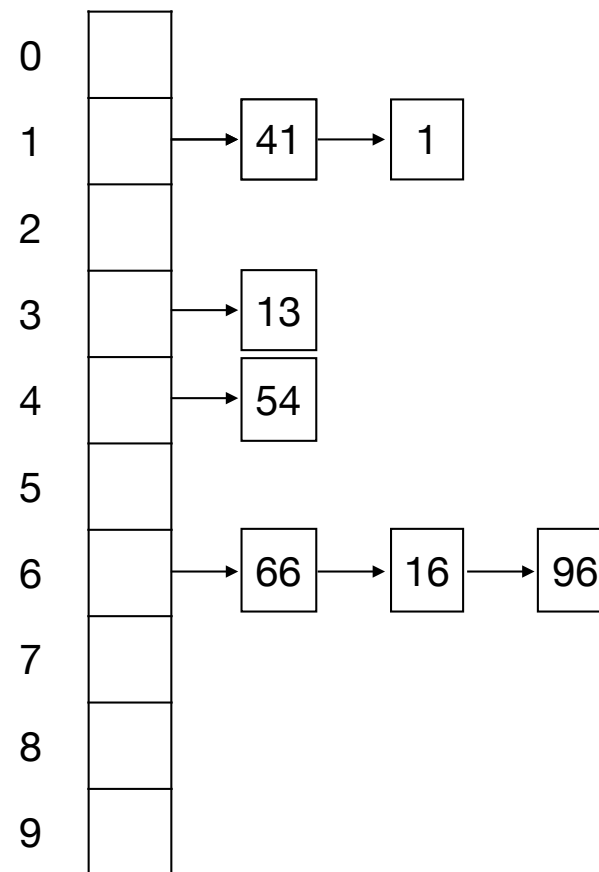
$\text{key}(S) = \{1, 13, 16, 41, 54, 66, 96\}$

$m = 10$

$h(k) = k \bmod 10$

# Uniform hashing

- **Def. Belastningsfaktor (load factor)**  $\alpha = n/m =$  gennemsnitlig længde af lister.  $m = \Theta(n) \Rightarrow \alpha = O(1)$ .
- **Simpel uniform hashing.** Antag at hvert element afbildes **uniformt tilfældigt** i A.
  - Forventet længde af liste =  $\alpha$ .
  - $\Rightarrow$  forventet tid for SEARCH er  $O(1)$
- **Tid.**
  - SEARCH i  $O(1)$  **forventet** tid.
  - INSERT og DELETE i  $O(1)$  tid.



$$U = \{0, \dots, 99\}$$

$$\text{key}(S) = \{1, 13, 16, 41, 54, 66, 96\}$$

$$m = 10$$

$$h(k) = k \bmod 10$$

# Ordbøger

---

Datastruktur	SEARCH	INSERT	DELETE	Plads
hægtet liste	$O(n)$	$O(1)$	$O(1)$	$O(n)$
direkte adressering	$O(1)$	$O(1)$	$O(1)$	$O( U )$
hægtet hashing	$O(1)^\dagger$	$O(1)$	$O(1)$	$O(n)$

† = **forventet** køretid med antagelse om simpel uniform hashing

- **Udfordring.** Hvad kan vi gøre uden at antage simpel uniform hashing? Findes der hashfunktioner der fordeler en mængde nøgler **nogenlunde jævnt**?

# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

# Hashfunktioner

---

- Divisionsmetoden.

- $h(k) = k \bmod m$ , hvor  $m$  er et primtal.
- Primtal  $m$  fordi fælles divisorer af nøgler og  $m$  kan reducere udnyttelse af tabel.
- $h(k) = ak \bmod m$  er lidt bedre.

- Multiplikationsmetoden.

- $h(k) = \lfloor m(kZ - \lfloor kZ \rfloor) \rfloor$ , hvor  $Z$  er en konstant  $0 < Z < 1$ .



# Hashfunktioner

---

- **Hashfunktioner for andet end heltal.** Alt er gemt som bits og kan derfor hashes.
- **Flydende tal.** Konverterer til bitrepræsentation.
- **Streng.** Bogstaver er heltal og så streng kan konverteres til sekvens af cifre. F. eks. "CLRS":
  - 256 forskellige ASCII-koder for bogstaver.
  - C = 67, L = 76, R = 82 og S = 83.
  - $\Rightarrow$  "CLRS" =  $67 \cdot 256^3 + 76 \cdot 256^2 + 82 \cdot 256^1 + 83 \cdot 256^0 = 1129075283$
- **Andre objekter.** Definer hashfunktion baseret på datafelter.

# Universel hashing

---

- Kan vi konstruere hashfunktioner med **beviselige garantier** uden antagelse om uniform hashing?
- **Ide (universelle hashfunktioner)**. Vælg en tilfældig hashfunktion  $h$  uniformt tilfældigt fra en mængde  $H$  af funktioner der kan beskrives **kompakt** og **beregnes** hurtigt og tilfredsstillende
  - For alle  $k_1 \neq k_2$  i  $U$  skal  $\Pr[h(k_1) = h(k_2)] \leq 1/m$ .
- **Eksempel**.
  - $h_{a,b}(k) = (ak + b \bmod p) \bmod m$  for primtal  $p$
  - $H = \{h_{a,b}(k) \mid a \in \{1, \dots, p-1\}, b \in \{0, \dots, p-1\}\}$
- Universel hashfunktion  $\Rightarrow$  Hægtet hashing i  $O(1)$  forventet tid uden antagelse om uniform hashing.
- Kun afhængig af tilfældigt valg fra  $H$ .

# Hashing

---

- [Anvendelser](#). Kodning, kryptografi, similaritet, geometri, ...

# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering

# Lineær probering

---

- Lineær probering (*linear probing*).
  - Gem S i tabel A af størrelse m.
  - Element x gemt i  $A[h(x.key)]$  eller i **klynge** (*cluster*) til højre for  $A[h(x.key)]$ .
  - Klynge = fortløbende (cyklisk) sekvens af fyldte indgange.

	41	1	11	13	54			98	
0	1	2	3	4	5	6	7	8	9

$h(k) = k \bmod 10$

- SEARCH(k): lineær søgning fra  $A[h(x.key)]$  i klynge til højre for  $A[h(x.key)]$
- INSERT(x): indsæt x på  $A[h(x.key)]$ . Hvis optaget, indsæt på næste tomme indgang til højre for x.
- DELETE(x): fjern x fra  $A[h(x.key)]$ . Genindsæt **alle** elementer til højre for i klyngen.

# Lineær probering

---

- **Caching.** Lineær probering er meget cache-effektivt.
- **Klumpning (*clustering*).** Nøgler har en tendens at "klumpe" sammen.

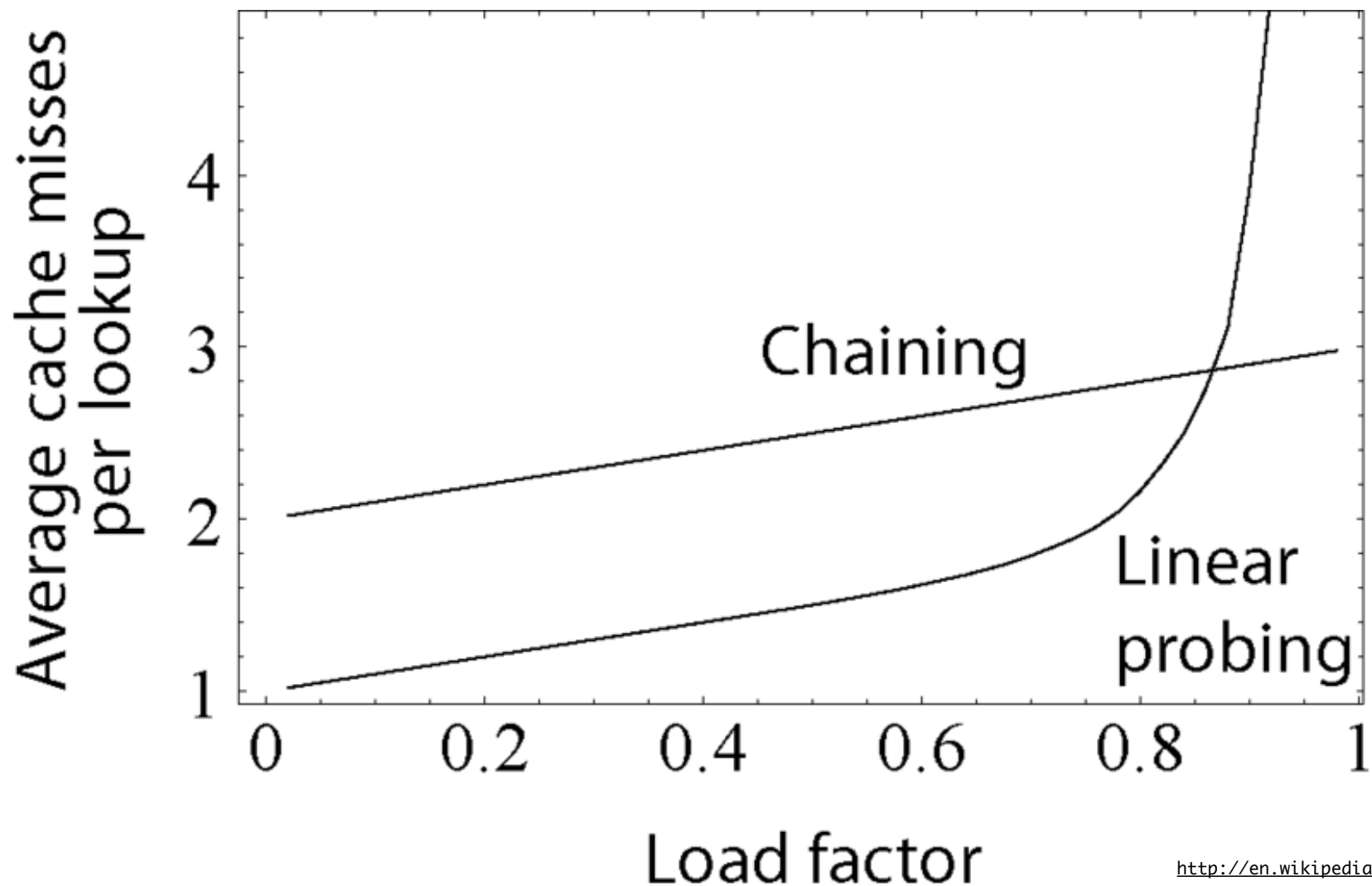
	41	1	11	13	54			98	
0	1	2	3	4	5	6	7	8	9

$$h(k) = k \bmod 10$$

- **Theorem.** Simpel uniform hashing  $\Rightarrow$  forventede antal proberinger =  $1/(1 - \alpha)$ .

# Lineær probering

- Lineær probering vs. hængtet hashing.



# Åben adressering

---

- Åben adressering (open addressing).
  - Lineær probering.
  - Kvadratisk probering.
  - Dobbelt hashing.



# Hashing

---

- Ordbøger
- Hægtet hashing
- Hashfunktioner
- Lineær probering