

# Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

Hashing [CLRS, kapitel 11.1-11.4]



AARHUS UNIVERSITET

## hash (Engelsk-Dansk)

- 1. (sb) (ret med kød og kartofler) biksemad (fx a meat and potato hash); (fig.) kludder; noget værre rod;**  
⌘ **make a ~ of** forkludre; udføre på en elendig (el. kikset) måde; **settle somebody's ~** ordne nogen; få nogen ned med nakken;
- 2. (sb) (narko) hash (fx smoke hash);**
- 3. (vb) hakke; skære i stykker; (fig.) forkludre; slippe (rigtigt) dårligt fra;**  
⌘ **~ over** diskutere; drøfte (**fx we can hash it over later**); **~ up** forkludre; slippe dårligt fra.

# Abstrakt Datastruktur: Ordbog

**Search( $S, k$ )**

**Insert( $S, x$ )**

**Delete( $S, x$ )**

Kan vi udnytte at  $x$  for alle praktiske formål er en **sekvens af bits? JA !**

# Alle nøgler er tal...

"Sko" =  $01010011.01101011.01101111_2 = 5466991_{10}$

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	{	8	H	X	h	x
9	HT	EM	}	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

# RushHour tilstande...

- Beskriv for hver bil og lastbil hvor meget den er flyttet til højre eller op
- Biler beskrives ved tal 0..4 dvs. 3 bits
- Lastbiler beskrives ved tal 0..3 dvs. 2 bits



(0, 1, 2, 1, 1, 0, 0, 4, 2, 0, 3)

000.001.10.001.01.000.000.100<sub>2</sub>

100868<sub>10</sub>

# Direkte Adressering :

## Nøgler $\{0,1, 2,\dots, m-1\}$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
•	•	•	3	•	•	6	7	•	•	•	11	•	13	14	•

DIRECT-ADDRESS-SEARCH( $T, k$ )

1 return  $T[k]$

DIRECT-ADDRESS-INSERT( $T, x$ )

1  $T[x.key] = x$

DIRECT-ADDRESS-DELETE( $T, x$ )

1  $T[x.key] = \text{NIL}$

- + Godt ved små nøgle universer
- + Selv kan generere nøglerne som 1,2,3,...
- Stort plads overforbrug når kun få nøgler brugt

# Hash Funktion

- Nøgler  $U$
- Hash funktion

$$h : U \rightarrow \{0, 1, \dots, m-1\}$$

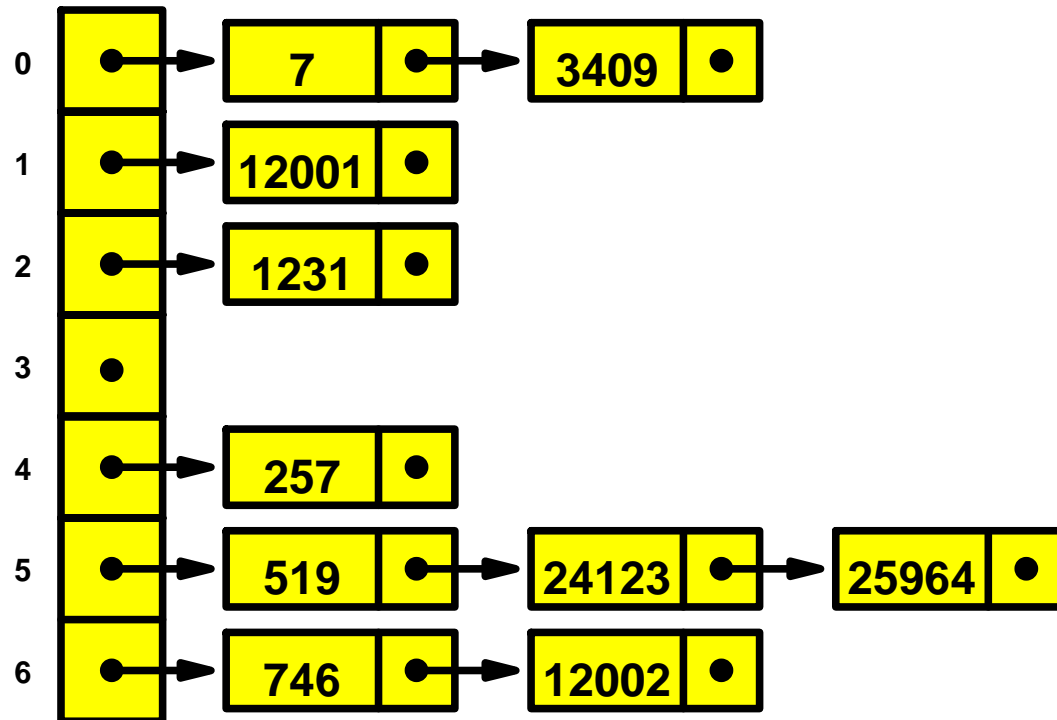
( $m \ll |U|$  og  $m^{|U|}$  mulige valg af  $h$ )

$x$	$h(x)$
7	0
257	4
519	5
746	6
1231	2
3409	0
12001	1
12002	6
24123	5
25964	5

$$h(k) = 5 \cdot k \text{ mod } 7$$

- + Nemt at jævne nøglerne jævnt ud
- Flere nøgler kan hashes til samme værdi
- Næsten ens nøgler kan være vilkårligt spredt

# Hash Tabel : Kollisionslister



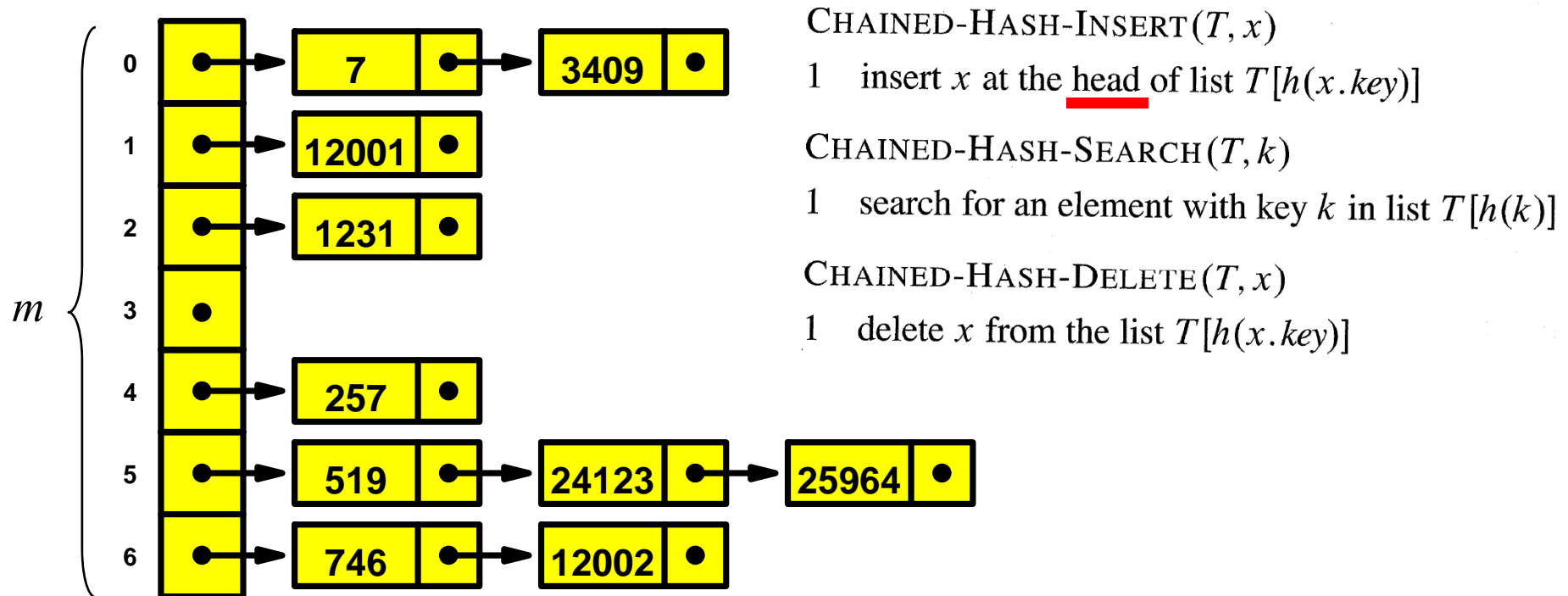
$x$	$h(x)$
7	0
257	4
519	5
746	6
1231	2
3409	0
12001	1
12002	6
24123	5
25964	5

$$h(k) = 5 \cdot k \pmod{7}$$

- Gem mængde af nøgler  $K$
- Vælg **tilfældig** hash funktion  $h : U \rightarrow \{0, 1, \dots, m-1\}$
- Gem nøglerne i tabel efter **hash værdi**
- **Kollisionslister** til nøgler med samme hashværdi



# Hash Tabel : Kollisionslister



Vælg en uniform tilfældig hash funktion:

- Forventet antal nøgler i en indgang i tabellen  $|K|/m$
- Insert, Delete, Search **forventet tid**  $O(|K|/m)$ ,  
dvs. tid  $O(1)$  hvis  $m = \Omega(|K|)$

# Hvad er en god Hash Funktion?

- For enhver funktion findes en dårlig mængde nøgler der hasher til samme værdi
- + For enhver mængde nøgler findes en god hash funktion der jævner godt ud (om den kan beskrives kompakt er et andet spørgsmål)

**Mål** Find en lille **mængde af hash funktioner** hvor en tilfældig funktion virker rimelig godt på en given mængde

# Hash Funktioner : Eksempler

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

(typisk  $m$  et primtal)

$m=2^8$  ignorerer alt på nær de 8 sidste bit:

$$h(\dots x_3 x_2 x_1 10101111) = h(\dots y_3 y_2 y_1 10101111)$$

$m=2^8-1$  ignorerer alle ombytninger af tegn:

$$h("c_3 c_2 c_1") = h("c_1 c_3 c_2")$$

$$h(k) = \lfloor s \cdot k / 2^{w-t} \rfloor \bmod 2^t$$

( $k = w$ -bit,  $h(k) = t$ -bit)

$$h(0101000010101010) = 01000$$

$$\begin{aligned} & 0101000010101010 \cdot 1001111000110111 \\ & = 00110001110110100100000010000110 \end{aligned}$$

# Universelle Hash Funktioner

Find **primtal**  $p \geq |U|$ .

Definer  $p \cdot (p-1)$  hash funktioner  $h_{a,b}$ , hvor  $1 \leq a < p$  og  $0 \leq b < p$

$$h_{a,b}(k) = (a \cdot k + b \bmod p) \bmod m$$

## Sætning

For to nøgler  $x \neq y$  og en tilfældig hash funktion  $h_{a,b}$  gælder

$$\Pr[h_{a,b}(x) = h_{a,b}(y)] \leq 1/m \quad \text{(Universel)}$$

## Korollar

For en hash tabel med en **tilfældig hash funktion**  $h_{a,b}$  tager Insert, Delete, Search **forventet tid**  $O(|K|/m)$

# Hash Tabel : Universal Hashing

<b>Search(<math>S, k</math>)</b>	<b><math>O(1)</math></b> <b>Forventet</b>
<b>Insert(<math>S, x</math>)</b>	
<b>Delete(<math>S, x</math>)</b>	

# Hashing af tal med mange bits...

$$\begin{aligned}x &= (b_{s-1}b_{s-2}\dots b_2b_1b_0)_2 \\ &= x_{s/w-1} \cdot 2^{w(s/w-1)} + x_{s/w-2} \cdot 2^{w(s/w-2)} + \dots + x_1 \cdot 2^w + x_0\end{aligned}$$

$$h_a(x) = x_{s/w-1} \cdot a^{s/w-1} + x_{s/w-2} \cdot a^{s/w-2} + \dots + x_1 \cdot a^1 + x_0 \pmod p$$

$$y_{s/w-1} := x_{s/w-1} \pmod p$$

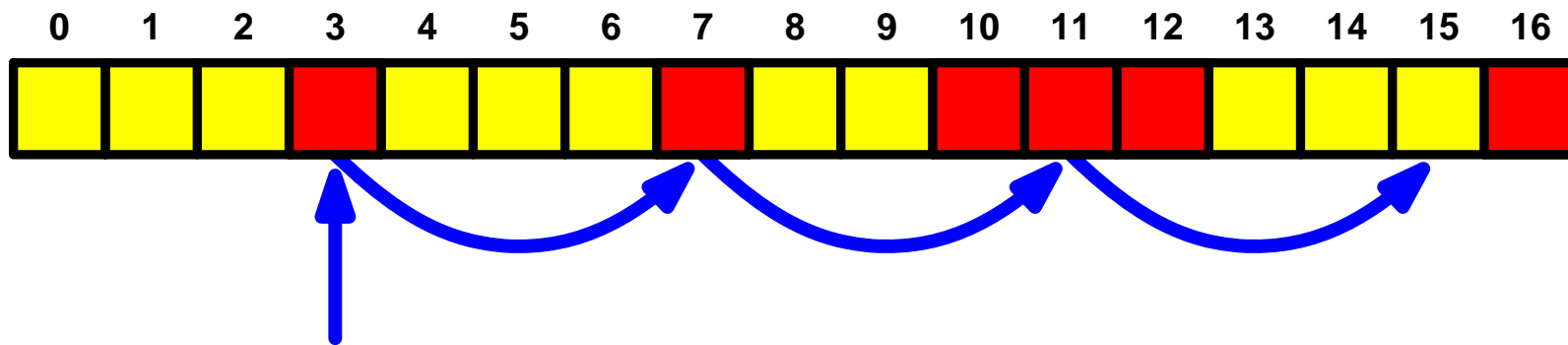
$$y_i := (y_{i+1} \cdot a + x_i) \pmod p \quad (\text{for } i = s/w-2 \dots 0)$$

$$h_a(x) = y_0$$

$$(a \cdot b) \pmod p = ((a \pmod p) \cdot b) \pmod p$$

$$(a + b) \pmod p = ((a \pmod p) + b) \pmod p$$

# Åben Adressering



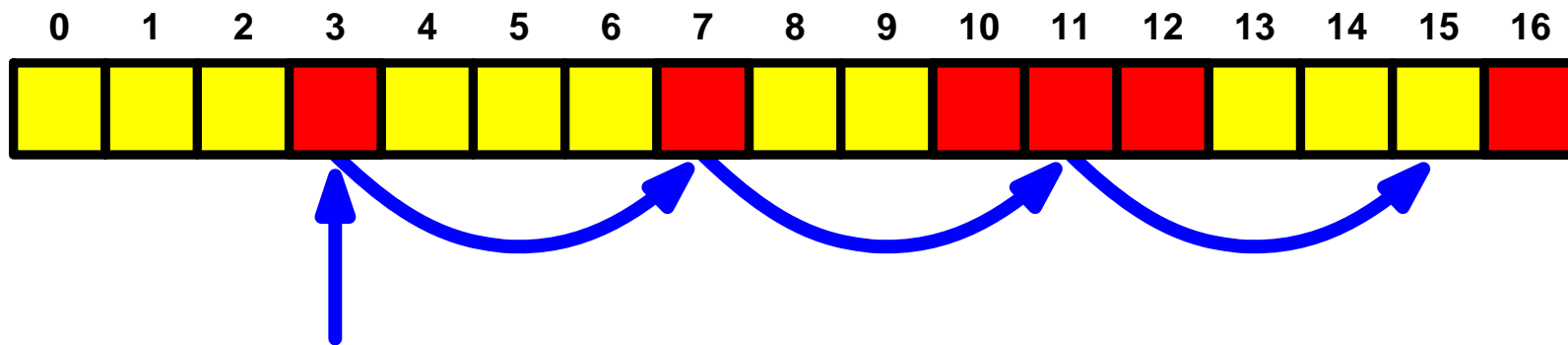
HASH-INSERT( $T, k$ )

```
1  $i = 0$ 
2 repeat
3      $j = h(k, i)$ 
4     if  $T[j] == \text{NIL}$ 
5          $T[j] = k$ 
6         return  $j$ 
7     else  $i = i + 1$ 
8 until  $i == m$ 
9 error "hash table overflow"
```

HASH-SEARCH( $T, k$ )

```
1  $i = 0$ 
2 repeat
3      $j = h(k, i)$ 
4     if  $T[j] == k$ 
5         return  $j$ 
6      $i = i + 1$ 
7 until  $T[j] == \text{NIL}$  or  $i == m$ 
8 return NIL
```

# Åben Adressering : Analyse



## Uniform hashing

$h(k,1), h(k,2), h(k,3), \dots$  er en **uniform tilfældig** rækkefølge

**(urealistisk)**

## Sætning

Ved uniform hashing er det forvente antal lookups  $1/(1-\alpha)$

hvor  $\alpha = |K|/m$  er belastningsfaktoren / fyldningsgraden



# Linjær Probing

Indsæt  $k$  på første ledige plads

$$h(k,i) = h'(k) + i \text{ mod } m$$

for  $i = 0, 1, 2, \dots$

**Eksempel :**

Indsæt 9, 3, 20, 6, 12, 2, 19, 11, 5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	9			2	19	3	20	12	11	5		6				

$x$	$h(x)$
2	4
3	6
5	10
6	12
9	1
11	5
12	7
19	4
20	6

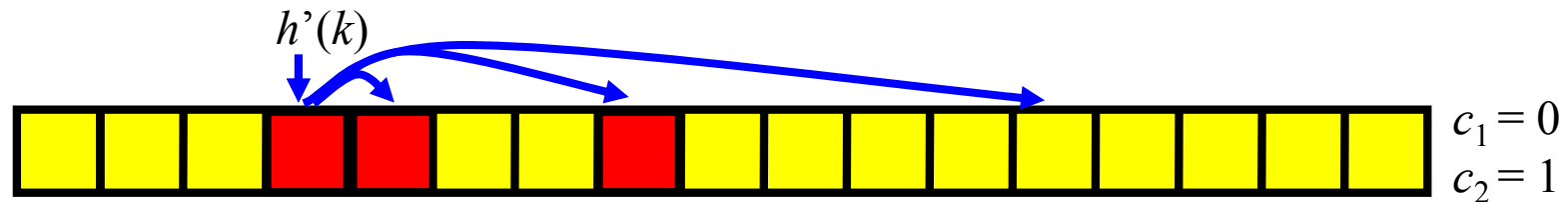
$$h'(x) = 2 \cdot x \text{ mod } 17$$

# Kvadratisk Probing

Indsæt  $k$  på første ledige plads

$$h(k,i) = h'(k) + c_1 \cdot i + c_2 \cdot i^2 \text{ mod } m$$

for  $i = 0, 1, 2, \dots$  hvor  $c_1$  og  $c_2 \neq 0$  er konstanter

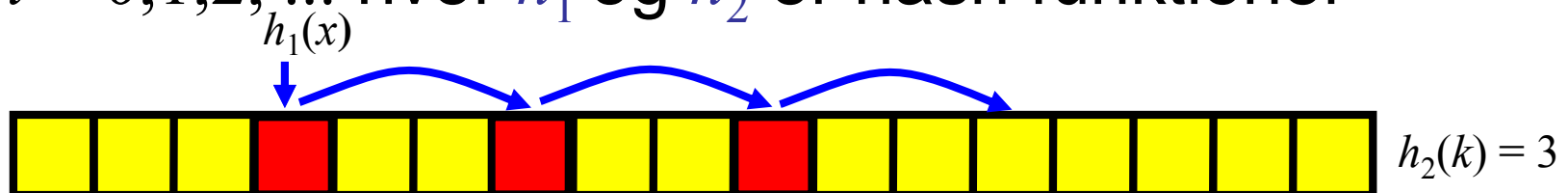


# Dobbelt Hashing

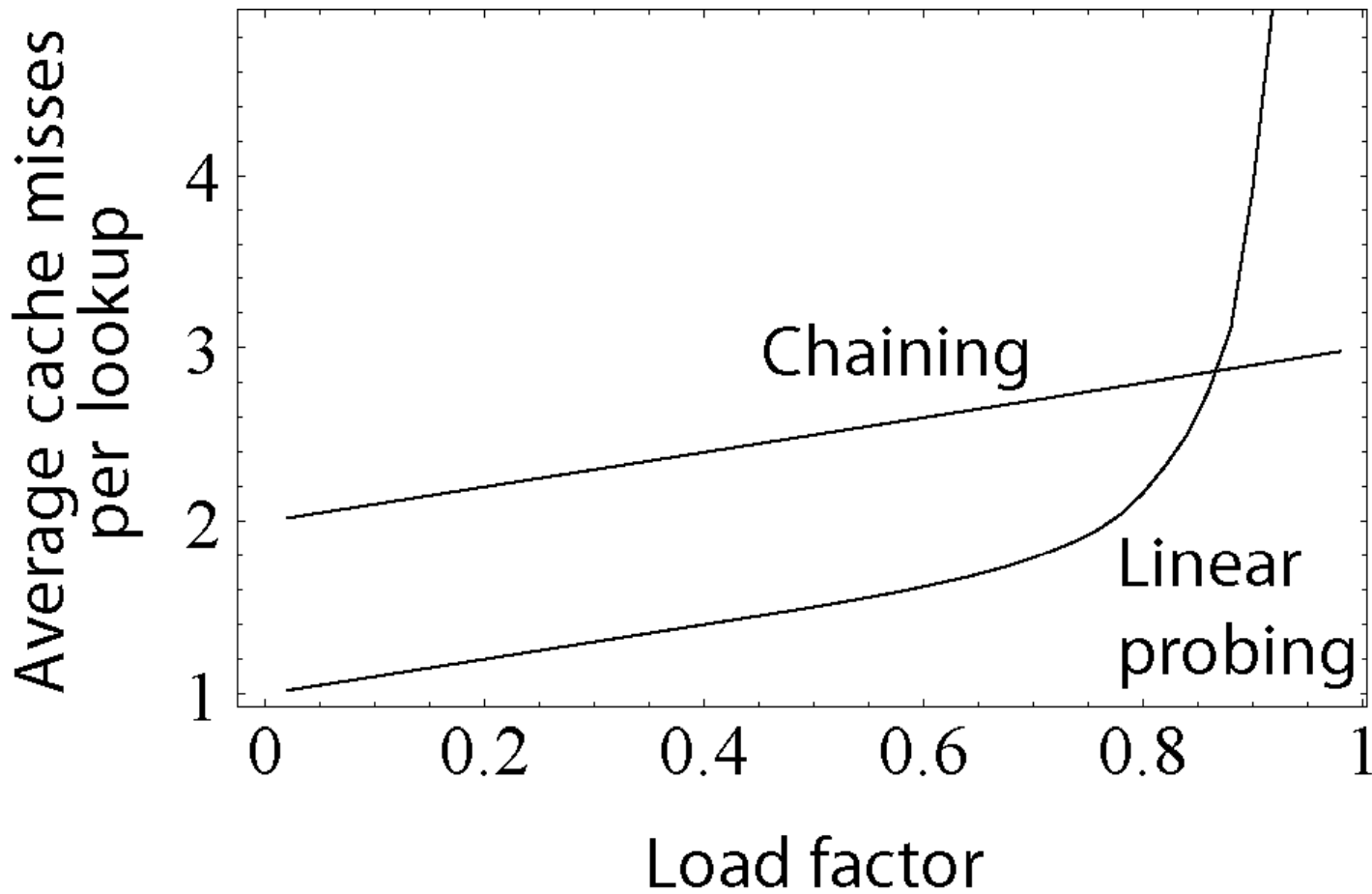
Indsæt  $k$  på første ledige plads

$$h(k,i) = h_1(k) + i \cdot h_2(k) \text{ mod } m$$

for  $i = 0, 1, 2, \dots$  hvor  $h_1$  og  $h_2$  er hash funktioner



# Eksperimentel Sammenligning



# Hashing

- **Valg af hash funktion**
  - Prøv sig frem...
  - Universelle hash funktioner
- **Hash tabeller**
  - Kollisionslister (kædede lister)
- **Åben adressering**
  - Linjær probing
  - Kvadratisk probing
  - Dobbelt hashing