Algoritmer og Datastrukturer

Hashing
[CLRS, kapitel 11.1-11.4]

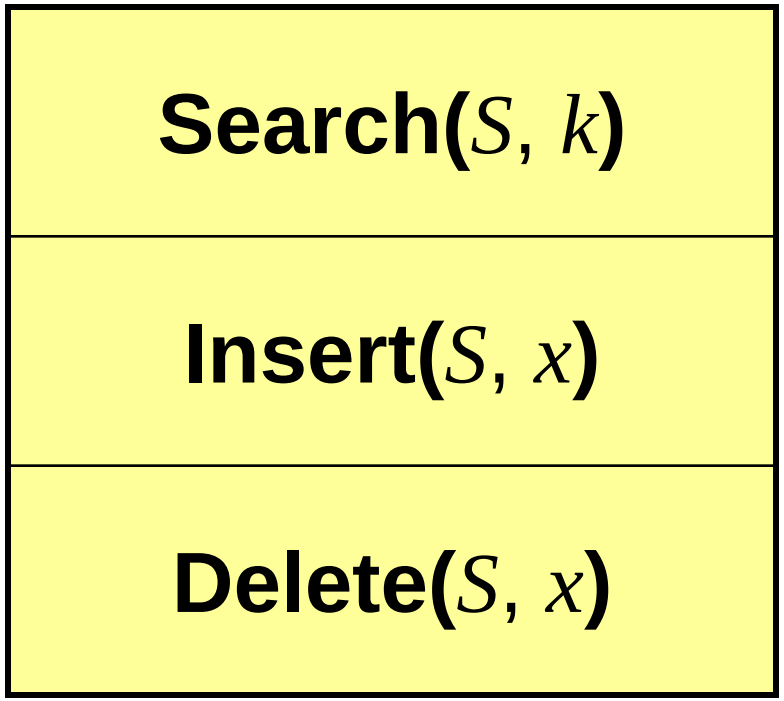
hash (Engelsk-Dansk)

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



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

Hvilket tal svarer ”AU” til ?

a) 4155_{10}
b) 6175_{10}

c) 16725_{10}
d) 24949_{10}
e) ved ikke

”AU” = 4155_{16} = 16725_{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…

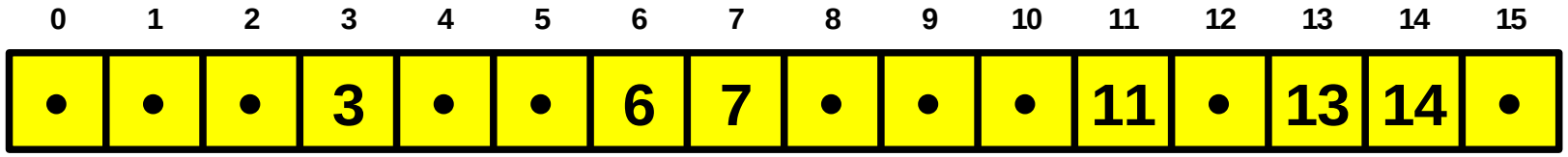
- Beskriv for hver bil og lastbil hvor meget den er flyttet til højre eller op
- Biler beskrives ved tal 0..4 dvs. 3 bits
- Lastbiler beskrives ved tal 0..3 dvs. 2 bits

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

$000.001.10.001.01.000.000.100.10.000.11_2$

12911171_{10}

Direkte Adressering :
Nøgler $\{0,1,2,\ldots,m\text{-}1\}$




```
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] = 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**

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

$$m << |U|$$

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

$h(k)=(5 \cdot k) \bmod 7$

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

Antal mulige (hash) funktioner

$h : U \rightarrow \{0,1,\ldots, m\text{-}1\}$?

a) $|U| \cdot m$

b) $|U|$!

c) $2^{|U|}$

d) $m^{|u|}$

e) $|U|^m$

f) ved ikke

Antal mulige (hash) funktioner der kan beskrives ved 6 tegn ?

	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

a) $|U|$

b) 128^6

c) $6^{|U|}$

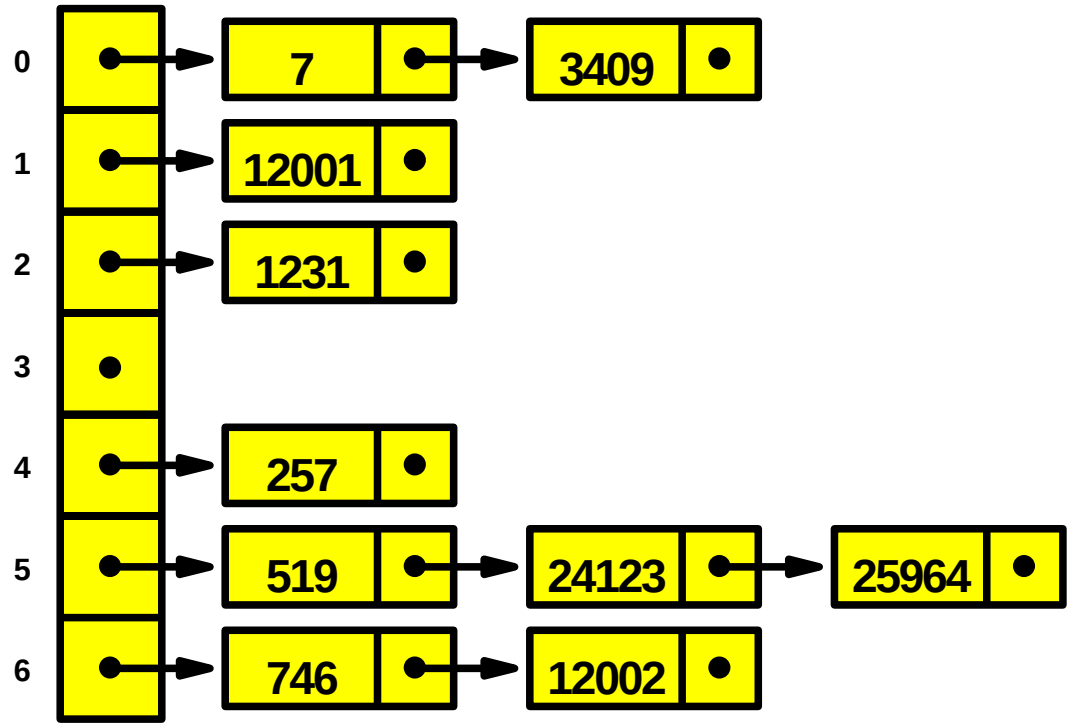
d) $6 \cdot m$

e) ved ikke

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

$$h(k) = 5k \bmod 7$$

Hash Tabel : Kollisionslister

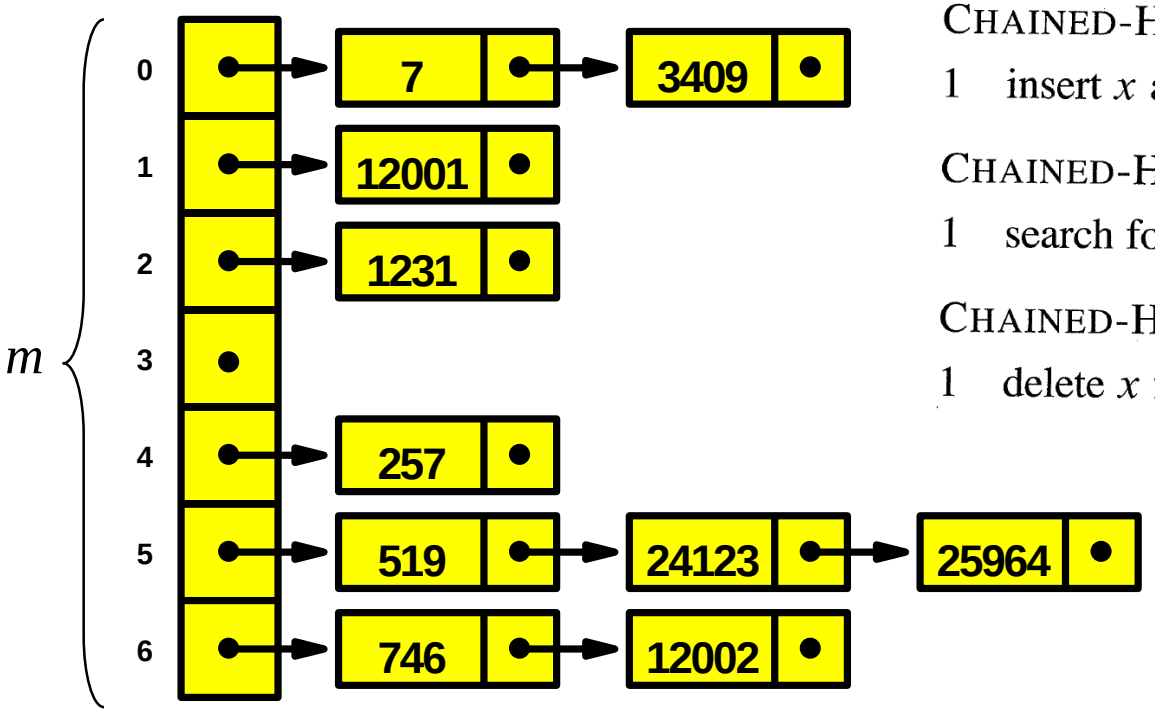


x	$h(x)$
7	0
257	4
519	5
746	6
1231	2
3409	0
12001	1
12002	6
24123	5
25964	5

$h(k)=(5 \cdot k) \bmod 7$

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

Hash Tabel : Kollisionslister



CHAINED-HASH-INSERT(T, x)

1 insert x at the head of list $T[h(x.key)]$

CHAINED-HASH-SEARCH(T, k)

1 search for an element with key k in list $T[h(k)]$

CHAINED-HASH-DELETE(T, x)

1 delete x from the list $T[h(x.key)]$

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 rimmelig 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:

$$h(...x_3x_2x_1 10101111) = h(...y_3y_2y_1 10101111)$$

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

$$h(”c_3c_2c_1”) = h(”c_1c_3c_2”)$$

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

$h(0101000010101010) = 01000$

$0101000010101010 \cdot 1001111000110111$

$= 00110001110110100100000010000110$

Universelle Hash Funktioner

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

Definer $p \cdot (p\text{-}1)$ 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

$$\Pr[h_{a.b}(x) = h_{a,b}(y)] \leq 1/m$$ **(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 : Universel Hashing

Search(S, k)	O(1) **Forventet**
Insert(S, x)	
Delete(S, x)	

Hashing af tal med mange bits...

(s bits) $x_{s/w-1}$ (w bits) x_1 (w bits) x_0 (w bits)

$$x = (\boxed{b_{s-1}b_{s-2}\ldots b_{s-w}}\ \boxed{b_{s-w-1}\ \ldots\ b_{2w}}\ \boxed{b_{2w-1}\ldots b_{w+1}b_w}\ \boxed{b_{w-1}..b_2b_1b_0})_2$$

$$= x_{s/w-1}\cdot 2^{w(s/w-1)} + x_{s/w-2}\cdot 2^{w(s/w-2)} + \cdots + x_1\cdot 2^w + x_0$$

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

$p > 2^w$ primtal

$0 < a < p$ tilfældig

Polynomium evaluering $h_a(x)$

$y = x_{s/w-1} \bmod p$

for $i = s/w - 2$ downto 0

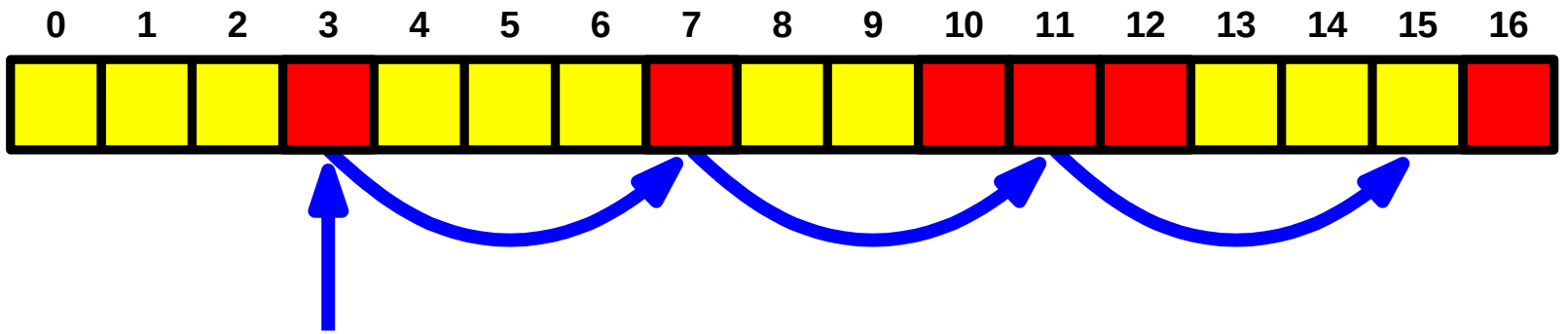
$y = (y \cdot a + x_i) \bmod p$

$h_a(x) = y$

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

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

Åben Adressering



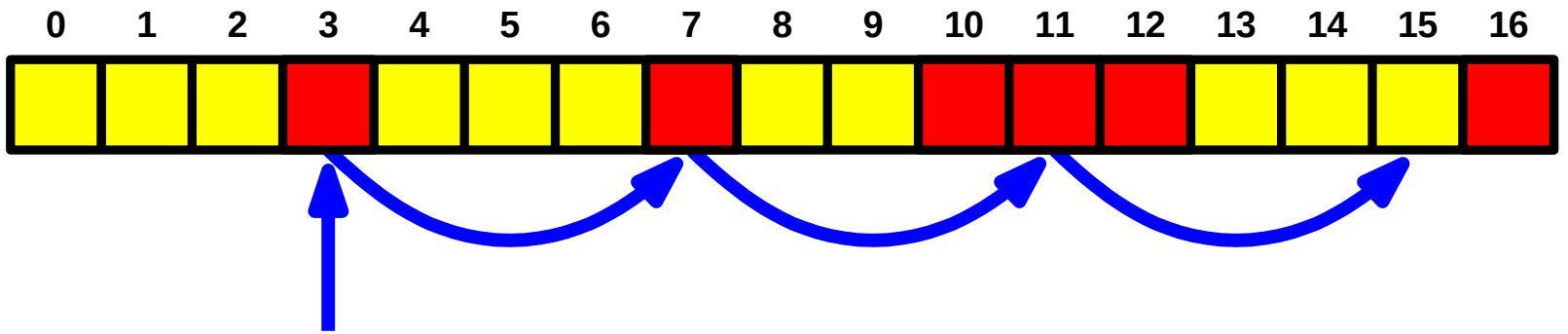
HASH-INSERT(T, k)

```
1  i = 0
2  repeat
3       j = h(k, i)
4       if T[j] == 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] == NIL or i == m
8  return NIL
```

Åben Adressering : Analyse



Uniform hashing

$h(k, 0), h(k, 1), h(k, 2)$,... 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

Lineær Probing

Indsæt k på første ledige plads

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

for $i = 0,1,2...$

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

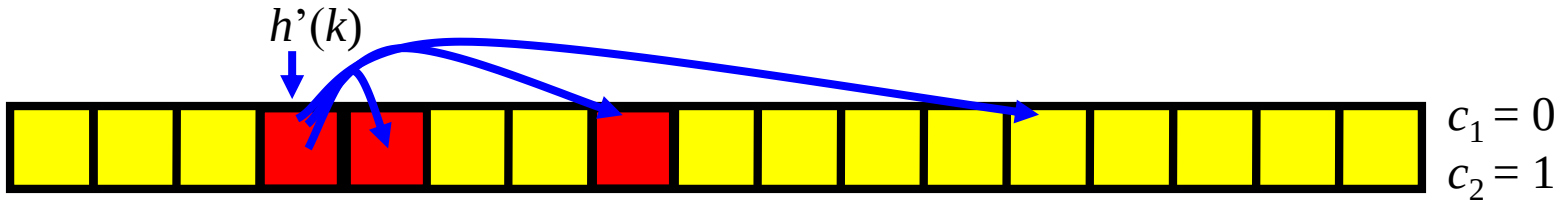
$h'(x)=(2\cdot x) \mod 17$

Kvadratisk Probing

Indsæt *k* på første ledige plads

$$h(k,i) = (h'(k) + c_1 \cdot i + c_2 \cdot i^2) \bmod m$$

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

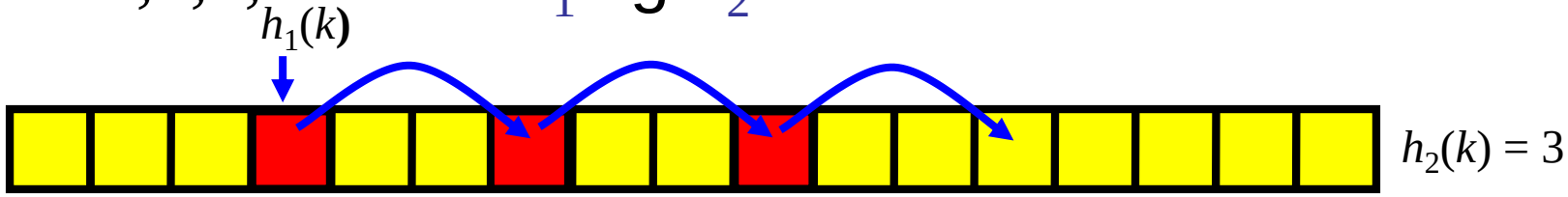


Dobbelt Hashing

Indsæt *k* på første ledige plads

$$h(k,i) = (h_1(k) + i \cdot h_2(k)) \bmod m$$

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



Dobbelt Hashing : Hvor skal 1 indsættes ?

a) 1
b) 5
c) 7
d) 9
e) ved ikke

$$h_1(k) = (2k) \bmod 10$$

$$h_2(k) = 1+((2k) \bmod 9)$$

0	1	2	3	4	5	6	7	8	9
5		6	3	2	1	8		7	

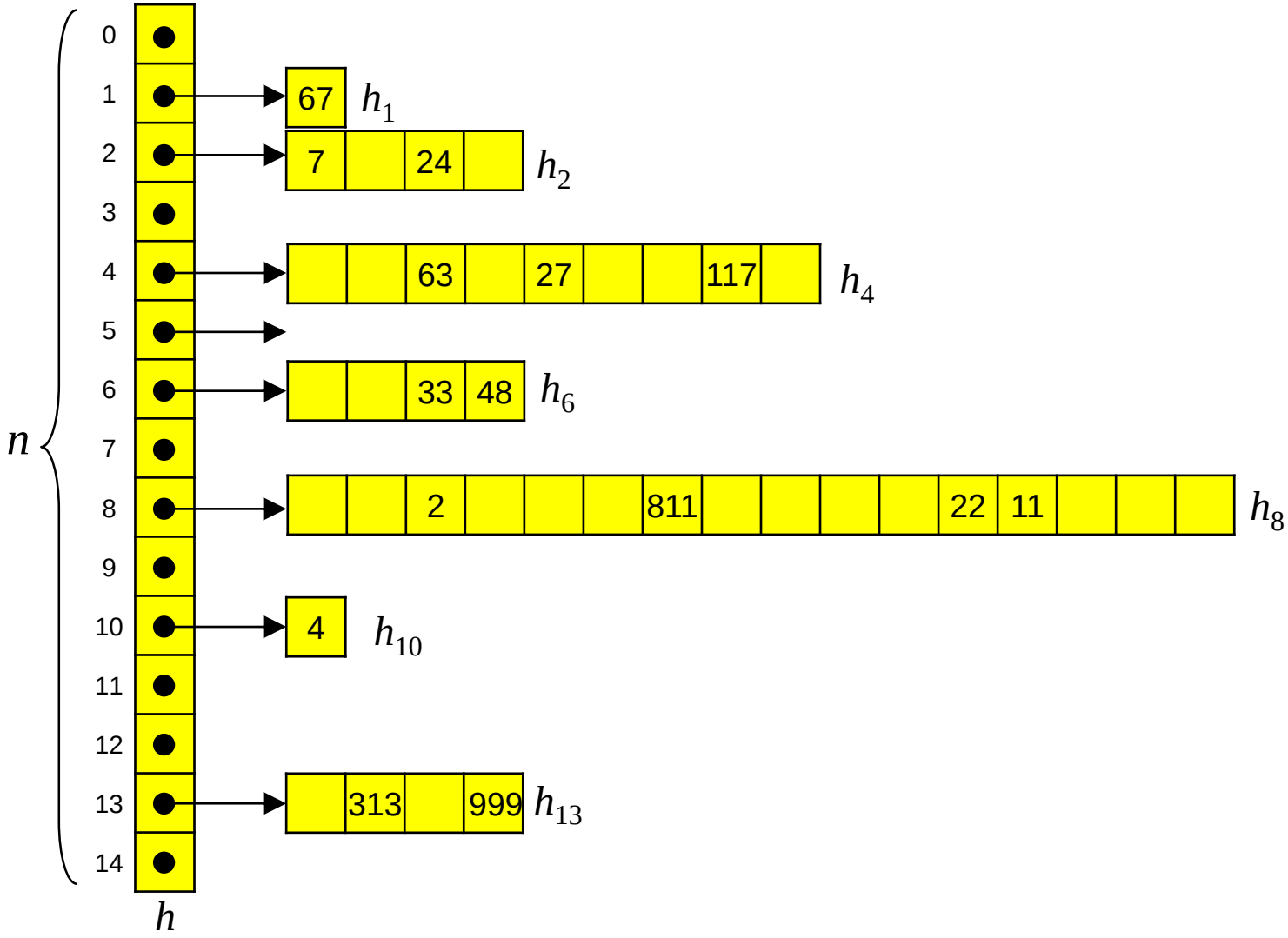
dADS1 eksamensopgave august 2010

CLRS Kapitel 11.5 – ikke pensum

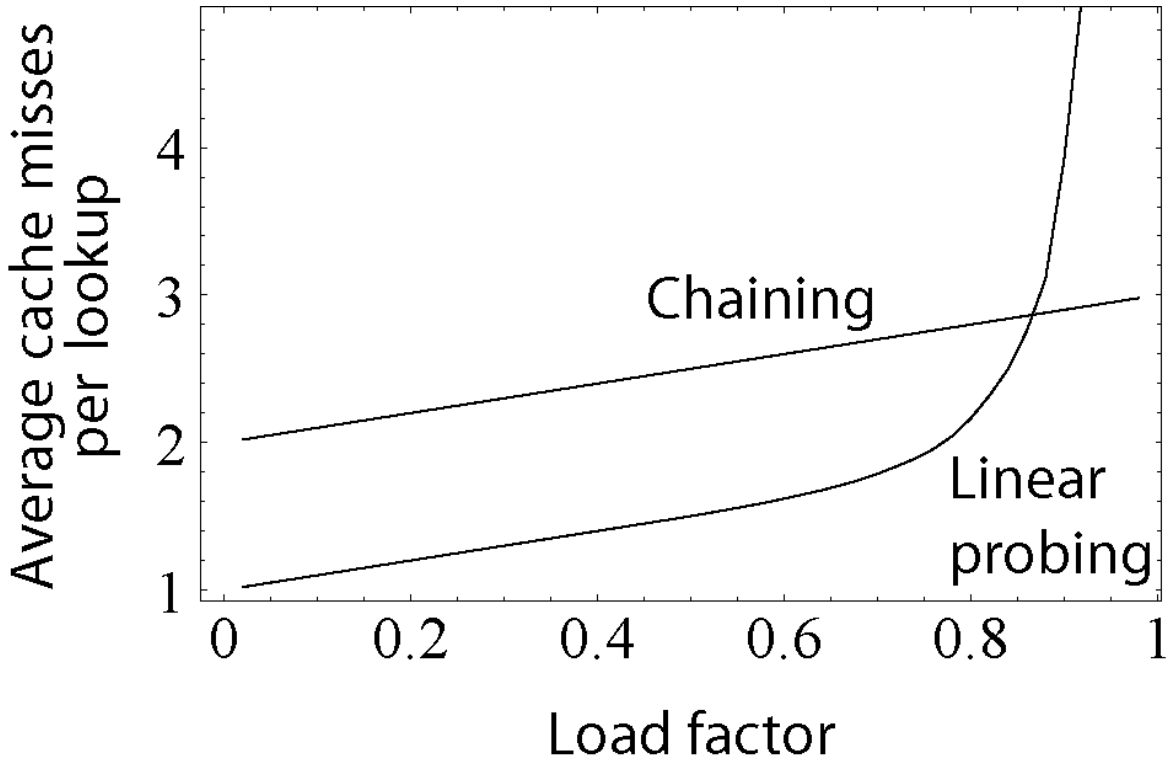
Perfekt hashing (statisk)

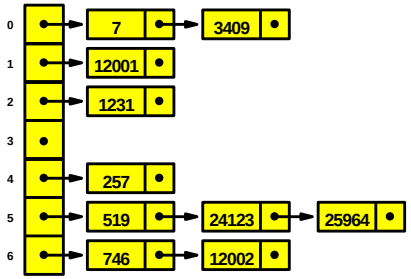
- n elementer
- 2 niveauer af hash funktioner
- hver spand egen hash funktion
- h hash tabel med n indgange
- Spand med n_i elementer størrelse n_i^2
- Universelle hash funktioner

E[kollisioner] = $\binom{n}{2}$ / # indgange



Eksperimentel Sammenligning





Chaining


Linear probing

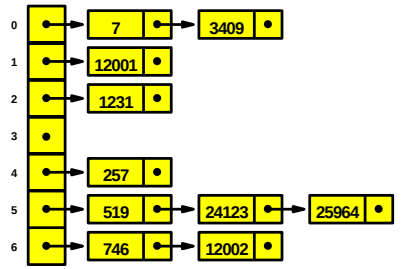
wikipedia.org

Hashing

- **Valg af hash funktion**
  - **Prøv sig frem...**
  - **Universelle hash funktioner**

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

- **Hash tabeller**
  - **Kollisionslister (kædede lister)**



- **Åben adressering**
  - **Lineær probing**
  - **Kvadratisk probing**
  - **Dobbelt hashing**