

F08 - Tabell, hashtabell, relation, lexikon

5DV149 Datastrukturer och algoritmer

Kapitel 13, 16

Niclas Börlin

niclas.borlin@cs.umu.se

Anna Jonsson

aj@cs.umu.se

2020-02-13 Thu

Innehåll

- ▶ Tabell
- ▶ Hashtabell
- ▶ Relation
- ▶ Lexikon

Tabell

- ▶ Modell:
 - ▶ Uppslagsbok.
- ▶ Organisation:
 - ▶ Ändlig avbildning av argument på värden.
- ▶ Dynamisk datatyp.



Tabell, exempel

- ▶ Postadress:
 - ▶ Argument/nyckel: heltal eller sträng
 - ▶ Värde: sträng
 - ▶ Postadress(90187) → "Umeå"
- ▶ Bilregister:
 - ▶ Argument/nyckel: sträng
 - ▶ Värde: post som beskriver bil
 - ▶ Bilregister("CBY328") → (Toyota, Avensis, 2013, diesel)
- ▶ Artikelregister:
 - ▶ Argument/nyckel: artikelnummer
 - ▶ Värde: post som beskriver artikeln
 - ▶ Artikelregister("32-2837") → (Nätverkskabel, CAT 5, 79)

Gränsyta till Tabell

```
abstract datatype Table(arg, val)
  Empty() → Table(arg, val)
  Insert(x:arg, y:val, t:Table(arg, val))
    → Table(arg, val)
  Isempty (t:Table(arg, val)) → Bool
  Lookup (x:arg, t:Table(arg, val))
    → (Bool, val)
  Remove (x:arg, t:Table(arg, val))
    → Table(arg, val)
```

Fält kontra Tabell

- ▶ Likheter:
 - ▶ Bägge lagrar *värden*.
 - ▶ Fältets *index* svarar mot tabellens *argument/nyckel*.
- ▶ Skillnader:
 - ▶ Fält kräver att argumenttypen är diskret linjärt ordnad.
 - ▶ Tabell kräver endast att *likhet* ska vara definierat för argumentet/nyckeltypen.
 - ▶ Tabell är en *dynamisk* datatyp, fält oftast en *statisk*.

Konstruktion (1)

- ▶ Fält
- ▶ Lista av par
- ▶ Hashtabell
- ▶ Binärt sökträd (senare)

Konstruktion (2)

- ▶ Tabell kan konstrueras som Fält om:
 1. argumenttypen är *diskret linjärt ordnad*,
 2. det går att hitta en *ODEF*-konstant, dvs. en konstant som betyder att tabellvärdet är "ej definierat",
 3. argumenten är relativt väl samlade.
- ▶ Exempel:
 - ▶ Postadress.
 - ▶ Argument: heltal
 - ▶ ODEF: NULL eller tomma strängen
 - ▶ Intervall: 10000-99999
 - ▶ Bilregister (nja).
 - ▶ Artikelregister.

Lista av par: Insättningar

- ▶ Två huvudalternativ:
 1. Sätt in det nya paret först i listan.
 2. Kolla först om det finns par med samma argument, byt ut.
- ▶ Dubletthantering?

Tabell, Prestanda

- ▶ Jämför konstruktionen som Fält respektive Lista av par med avseende på kostnad för:
 - ▶ Insättningskostnad.
 - ▶ Avläsning.
 - ▶ Borttagning.
- ▶ När väljer man vad?

Tillämpningar av Tabell (1)

- ▶ Benämna objekt, t.ex.
 - ▶ (Artikel-nr) 32-2837: Nätverkskabel,
 - ▶ (Registration) SE-GVS: Piper Cherokee.
- ▶ Associera egenskaper hos ett objekt med motsvarande värden, t.ex.
 - ▶ 32-2837:

Namn	Nätverkskabel
Pris	SEK 79
Hyllplacering	E14
Lagerstatus	23 st
 - ▶ SE-GVS:

Type	Piper PA-28
Seats	4
Fuel capacity	182L
Gear type	Fixed tricycle

Tillämpningar av Tabell (2)

- ▶ Representera *samband* mellan objekt, t.ex.
 - ▶ *personer som äger en fastighet* eller
 - ▶ *personer som är medlemmar av en klubb*.
- ▶ Kompilatorer (symboltabeller).
 - ▶ Existens (*är symbolen definierad?*).
 - ▶ Attribut (*värdetyp, räckvidd*).
- ▶ Fält som Tabell (glesa matriser).
 - ▶ Nyckel: (rad, kolumn)
 - ▶ Ex.

0	0	3
8	0	0
2.5	0	0

skulle lagras som

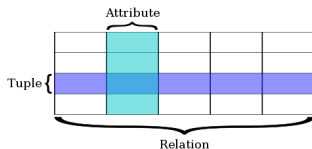
nyckel	värde
(2,1)	8
(3,1)	2.5
(1,3)	3

Tippel, Post

- ▶ Bägge liknar Tabell: Associerar *argument* med *värden*.
- ▶ Tippel (*tuple*) består av *n* *ordnade* element, ex. för koordinater $a=(29, 4, 3)$.
 - ▶ `Inspect-first(a)` \rightarrow 29,
 - ▶ `Inspect-second(a)` \rightarrow 4.
- ▶ Post (*record, struct*) är som abstrakt datatyp sett samma sak som Tippel, t.ex. för $a=(32-2837, \text{Nätverkskabel}, 79)$.
 - ▶ `Inspect-first(a)` \rightarrow 32-2837,
 - ▶ `Inspect-second(a)` \rightarrow Nätverkskabel.

Relation

- ▶ En *Relation* är en *egenskap* definierad för en *grupp av objekt*.



- ▶ Exempel: relationen *personer som äger fastigheter* (fastighetsägare):
 - ▶ (NB, Innertavle 10:31, 0.5),
 - ▶ (AL, Innertavle 10:31, 0.5),eller relationen *personer som är medlem i en flygklubb*
 - ▶ (NB, Umeå flygklubb, normal-medlem, <tom>),
 - ▶ (EN, Umeå flygklubb, normal-medlem, flyglärare),
 - ▶ (NB, Sturup flygklubb, elev-medlem, <tom>).
 - ▶ (HP, Sturup flygklubb, normal-medlem, <tom>).
- ▶ Används i *relationsdatabaser*.

Relationstabeller, exempel (1)

- ▶ Relationen *personer som äger fastigheter* (fastighetsägare)
 - ▶ (NB, Innertavle 10:31, 0.5),
 - ▶ (AL, Innertavle 10:31, 0.5),

Fastighetstabell:

Nyckel	Namn
1	Innertavle 1:1
...	
42	Innertavle 10:31
...	

Fastighetsägartabell:

Fastighet	Person	Andel
...		
42	3512	0.5
42	3513	0.5
...		

Persontabell:

Nyckel	Namn
...	
3512	NB
3513	AL
...	

Relationstabeller, exempel (2)

► Relationen *personer som är medlem i en flygklubb*:

- (NB, Umeå flygklubb, normal-medlem, <tom>),
- (EN, Umeå flygklubb, normal-medlem, flyglärare),
- (NB, Sturup flygklubb, elev-medlem, <tom>).
- (HP, Sturup flygklubb, normal-medlem, <tom>).

Flygklubbstabel:

Nyckel	Namn
12	Umeå flygklubb
18	Sturup flygklubb

Persontabell:

Nyckel	Namn	Flyglärare
416	NB	Ja
781	EN	
881	HP	

Klubbmedlemstabel:

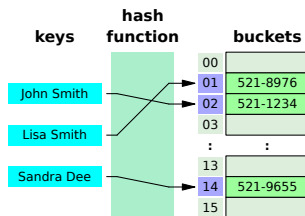
Person	Klubb	Medlemstyp
881	18	normal-medlem
781	12	normal-medlem
416	18	elev-medlem
416	12	normal-medlem

Tabell, att jämföra argument

- ▶ Om argumenttypen inte är inbyggd (t.ex. `int`, `double`) utan t.ex. `Post` måste man definiera (implementera) en *jämförelseoperation*.
- ▶ T.ex. om argumentet är en `Post` (artikel-nr, namn), definiera *likhet* som att *artikel-numren* är lika.

Hashtabell ¹

- ▶ Problemet med att implementera en tabell som fält är att grundmängden inte får bli för stor.
- ▶ Det går att avhjälpa med en *hashfunktion* *h* som avbildar den större indextypen A till en mindre indextyp B, t.ex. strängar till heltal.



¹*hash — chop and mix*

Hashfunktion (1)

- Funktionen $h(x)$ avbildar en stor mängd A av nycklar, på en mindre mängd B av tal, t. ex. för A, B heltal:

$$h(x) = x \bmod 13$$

eller för A datum, B heltal:

$$h(x) = \text{month}(x).$$

Hashfunktion (2)

- ▶ Operatoren mod beräknar heltalsrest vid division.
- ▶ För x heltal så avbildar

$$x \bmod b,$$

alla heltalen x på heltalen $[0, 1, \dots, b - 1]$.

- ▶ Ex.

$0 \bmod 4 = 0,$	$4 \bmod 4 = 0,$
$1 \bmod 4 = 1,$	$5 \bmod 4 = 1,$
$2 \bmod 4 = 2,$	$6 \bmod 4 = 2,$
$3 \bmod 4 = 3,$	$7 \bmod 4 = 3.$

- ▶ I programspråket C: operatoren $\%$, t.ex.
 - ▶ `hash % TABLE_SIZE`
- ▶ Används i de flesta hash-funktioner.

Hashfunktion (3)

- För A sträng så är det vanligt att loopa över alla element i strängen, t.ex.

```
Algorithm StringHash(s)
seed ← 131 // Magic number
hash ← 0
for i=1,2,...,length(s) do
    hash ← (hash * seed) + s[i]
return hash mod TABLE_SIZE
```

Kollisioner

- ▶ En kollision är när två argument avbildas på samma hash-värde.
- ▶ Kollisioner går ej att undvika helt.
- ▶ En bra hashfunktion kännetecknas av:
 - ▶ Litet förväntat antal kollisioner.
 - ▶ Elementen sprids jämnt över den mindre indexmängden B.
 - ▶ Funktionsvärdet påverkas av alla delar av nyckeln.
- ▶ Kollisioner kan hanteras med:
 1. Sluten hashning.
 - 1.1 Linjär teknik.
 - 1.2 Kvadratisk teknik.
 2. Öppen hashning.

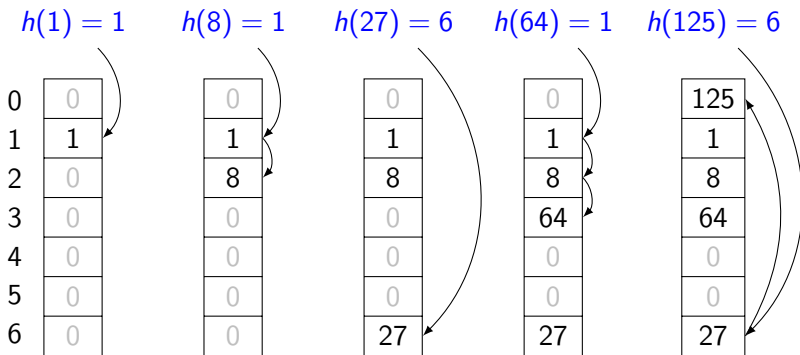
Sluten hashning

- ▶ Låt B vara en *cirkulär vektor*, dvs. det första elementet följer efter det sista.
- ▶ *Statisk* tabell, fast antal platser.
- ▶ *Linjär* teknik för kollisioner:
 - ▶ Om ett element kolliderar (platsen redan upptagen), sätt in det nya elementet på den första lediga platsen *efter* den upptagna.
 - ▶ Ger upphov till *klustring* (flockning?) i tabellen.
- ▶ *Sökning* efter ett element börjar på den plats dess hashvärde anger och fortsätter eventuellt framåt.
 - ▶ Om det inte påträffats före nästa lediga plats så finns det inte i tabellen.
- ▶ Vid *borttagning* i tabellen kan inte platsen lämnas tom — då kan senare sökningar misslyckas.
 - ▶ Därför måste i stället en “borttagen”-markör sättas in i tabellen.

Sluten hashning, linjär teknik, exempel

- ▶ Grundmängden $1, 2, \dots, 125$ ska avbilda på värdena $0, 1, \dots, 6$.
- ▶ Låt 0 betyda "ledig" och 126 betyda "borttagen".
- ▶ Given mängden $\{1, 8, 27, 64, 125\}$ och hashfunktionen

$$h(x) = x \bmod 7$$



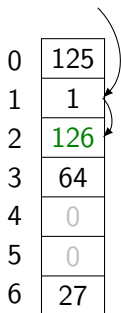
Sluten hashning, linjär teknik, sökning

- ▶ Slå upp 8
- ▶ Slå upp 13

Sluten hashning, linjär teknik, exempel

- Efter borttagning av 8:

$$h(8) = 1$$



0	125
1	1
2	126
3	64
4	0
5	0
6	27

Sluten hashning, linjär teknik, sökning

- ▶ Slå upp 8
- ▶ Slå upp 13
- ▶ Slå upp 64

Sluten hashning, linjär teknik, komplexitet

- ▶ Värstafallskomplexiteten för samtliga operationer är $O(n)$, där n är antalet element som finns insatta i tabellen.
- ▶ Är dock ytterst osannolikt. Alla element måste ligga i en följd.
- ▶ Under förutsättning att tabellen inte fylls mer än till en viss del får man i medeltal $O(1)$ för operationerna.

Hashtabeller, fyllnadsgrad

- ▶ En hashtabells *fyllnadsgrad* (λ) definieras som *kvoten mellan antalet insatta element och antalet platser i tabellen*.
- ▶ En tom tabell har $\lambda = 0$ och en full $\lambda = 1$.
- ▶ För linjär teknik vet man att:
 - ▶ Medelantalet platser som måste provas vid en insättning och misslyckad sökning är uppåt begränsad av

$$\frac{1}{2} \left(1 + \frac{1}{(1 - \lambda)^2} \right).$$

- ▶ Medelantalet platser för en lyckad sökning är uppåt begränsad av

$$\frac{1}{2} \left(1 + \frac{1}{1 - \lambda} \right).$$

- ▶ En halvfull tabell ($\lambda = 0.5$) ger 2.5 resp. 1.5.
- ▶ När fyllnadsgraden blir hög ($\lambda > 0.75$) blir operationerna för den linjära tekniken långsam (8.5 resp. 2.5).

Sluten hashning, kvadratisk teknik

- ▶ Låt H vara ett elements hash-värde.
- ▶ Vid linjär teknik testas positionerna

$$H, H + 1, H + 2, H + 3, \dots \mod m$$

tills en ledig plats hittas.

- ▶ Vid *kvadratisk* teknik testas i stället positionerna

$$\begin{aligned} &H, H + 1^2, H + 2^2, H + 3^2, \dots \mod m \\ &= H, H + 1, H + 4, H + 9, \dots \mod m \end{aligned}$$

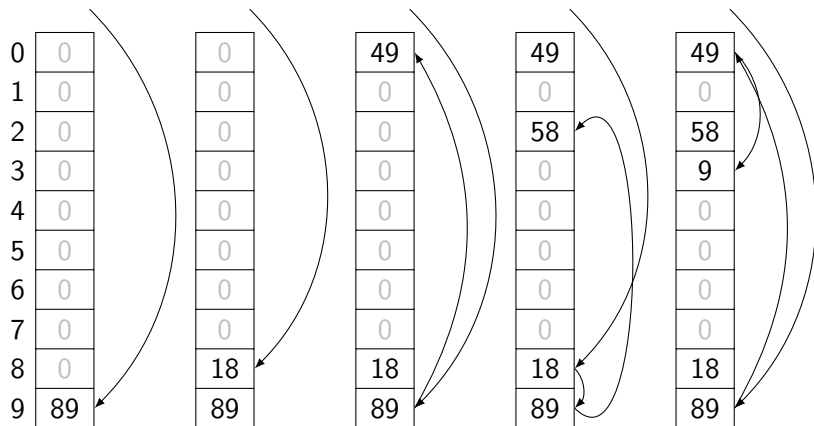
tills en ledig plats hittas.

Sluten hashning, kvadratisk teknik, exempel

- ▶ Sätt in talen 89, 18, 49, 58, 9 i en hashtabell med 10 platser.

- ▶ $h(x) = x \bmod 10$

$$h(89) = 9 \quad h(18) = 8 \quad h(49) = 9 \quad h(58) = 8 \quad h(9) = 9$$



Sluten hashning, kvadratisk teknik, problem

- ▶ Med olyckligt vald hashfunktion och tabellängd finns risk att man inte hittar en ledig plats även om den finns!
- ▶ Exempel:
 - ▶ Tabellstorlek = 16, hashfunktion $h(x) = x \bmod 16$.
 - ▶ Efter att ha stoppat in elementen 0, 16, 32, och 64 finns ingen mer plats för tal med $h(x) = 0$.
 - ▶ De enda platser som kommer att provas är de upptagna 0, 1, 4, 9, 0, 9, 4, 1, 0, 1, 4, 9, ...
- ▶ Men:
 - ▶ Om kvadratisk teknik används och tabellens storlek är ett **primtal** så kan ett nytt element alltid stoppas in om $\lambda < 0.5$.
 - ▶ Fullständig komplexitetsanalys saknas men i praktiken ger den upphov till mindre klustring än linjär hashning.

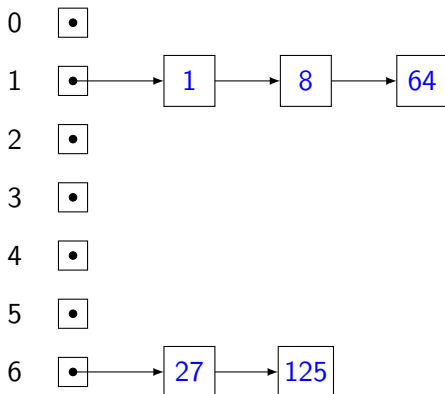
Öppen hashning (1)

- ▶ I stället för en cirkulär vektor används en k -Vektor av Lista, $h(x) = x \bmod k$.
- ▶ *Dynamisk* tabell, ingen begränsning på antalet element.
- ▶ I lista nummer i ligger alla element med hashvärde i .

Öppen hashning (2), exempel

- ▶ Grundmängden $1, 2, \dots, 125$ ska avbildas på värdena $0, 1, \dots, 6$.
- ▶ Given mängden $\{1, 8, 27, 64, 125\}$ och hashfunktionen

$$h(x) = x \bmod 7.$$



Öppen hashning (3)

- ▶ Värstafallskomplexitet:
 - ▶ $O(n)$ för alla operationer (alla element i samma lista), där n är antalet insatta element.
- ▶ Medelfallskomplexitet:
 - ▶ Insättning och misslyckad sökning blir n/k .
 - ▶ Lyckad sökning blir $(n-1)/(2k+1)$.
- ▶ Tumregel: Inte fler än $2k$ element bör sättas in.

Teoretiska resultat

Från <http://www.ida.liu.se/labs/logpro/ulfni/dalg/resources/f8.pdf>

- Antal sonderingar vid olika fyllnadsgrad:

Fyllnadsgrad (%)						
	10	25	50	75	90	99
Lyckad sökning						
Sluten, linjär	1.06	1.17	1.50	2.50	5.50	50.5
Sluten, kvadratisk	1.05	1.15	1.39	1.85	2.56	4.7
Öppen	1.05	1.12	1.25	1.37	1.45	1.5
Misslyckad sökning						
Sluten, linjär	1.12	1.39	2.50	8.50	50.5	5000
Sluten, kvadratisk	1.11	1.33	2.00	4.00	10.0	100
Öppen	0.10	0.25	0.50	0.75	0.90	0.99

Mer avancerade hashfunktioner

- ▶ Vi har använt

$$h(x) = x \mod m$$

i exemplen.

- ▶ Fungerar bra om m är ett primtal.
- ▶ Annars kan periodicitet uppstå.
- ▶ Finns mer generella metoder, t.ex.

$$h(x) = ((c_1x + c_2) \mod p) \mod m,$$

där

- ▶ divisorn p är ett stort primtal $> m$, t.ex. 1048583,
- ▶ konstanterna c_1 och c_2 är heltal > 0 och $< p$.
- ▶ c_2 är ofta satt till 0.
- ▶ Slumptalsgeneratorer, krypteringsalgoritmer.
- ▶ md5 (<https://en.wikipedia.org/wiki/MD5>)

Exempel, Month-to-days (1)

- ▶ Antag vi vill bygga en hashtabell som för varje månadsnamn lagrar antalet dagar i månaden.
- ▶ A=sträng av längden 3.
- ▶ B=heltal.

Month	Jan	Feb	Mar	Apr	May	Jun
Days	31	28	31	30	31	30
Hash	1831883	1763751	1883370	1679403	1883377	1834503

Month	Jul	Aug	Sep	Oct	Nov	Dec
Days	31	31	30	31	30	31
Hash	1834501	1680047	1986858	1917956	1902369	1729430

Exempel, Month-to-days (2)

► Alternativa tabellstorlekar

	Hash	Jan	Feb	Mar	Apr	May	Jun
Size	Mod	1831883	1763751	1883370	1679403	1883377	1834503
5		3	1	0	3	2	3
7		4	3	6	5	6	6
11		9	0	5	0	1	0
13		1	2	8	11	2	8
17		14	1	8	7	15	16
19		17	0	14	12	2	15
23		2	19	15	12	22	0
29		11	0	23	13	1	21

	Hash	Jul	Aug	Sep	Oct	Nov	Dec
Size	Mod	1834501	1680047	1986858	1917956	1902369	1729430
5		1	2	3	1	4	0
7		4	5	6	5	0	3
11		9	6	5	7	7	10
13		6	5	3	1	1	1
17		14	5	0	16	1	3
19		13	10	9	1	13	12
23		21	12	3	9	16	14
29		19	19	10	12	27	15

Exempel, Month-to-days (3)

- ▶ Sluten hashning, linjär teknik, storlek 17

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Jul	Feb	Sep	Oct	Nov	Aug	Dec	Apr	Mar						Jan	May	Jun

- ▶ Sluten hashning, kvadratisk teknik, storlek 17

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Sep	Feb		Oct	Nov	Aug	Jul	Apr	Mar				Dec		Jan	May	Jun

- ▶ Öppen hashning, storlek 5

0	Mar	Dec		
1	Feb	Jul	Oct	
2	May	Aug		
3	Jan	Apr	Jun	Sep
4	Nov			

Lexikon

- ▶ Ett *Lexikon* är som en tabell utan tabellvärden.
- ▶ Lexikon är en *solitär* datatyp.
- ▶ Man kan göra *modifieringar av isolerade dataobjekt*, ex. lägga till, ta bort eller slå upp element, men *inte kombinera/bearbeta två eller flera* Lexikon.
- ▶ Jämför den icke-solitära datatypen *Mängd*:
 - ▶ Det går att lägga till och ta bort element till både *Mängd* och *Lexikon*.
 - ▶ Både en *Mängd* och ett *Lexikon* är oordnade datatyper.
 - ▶ Det går att bilda *unionen* eller *snittet* av två mängder.
 - ▶ Motsvarande för två *Lexikon* är inte definierat. ("Det finns bara *ett* Lexikon av svenska ord.")