Tampereen yliopisto

Hajautustaulut (hash tables)

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

Hajautustaulun tarve

- Vektorin indeksointi nopea
- Assosiatiivinen säiliö: hakuavain
- Vektorin indeksi on hakuavain!
- Miten mielivaltainen hakuavain?
- *Ämpärit (buckets)*
- *Hajautusfunktio (hash function)*

- Vektori nopea
- Vektori myös (nopea) hakurakenne
- Vektorin hakurajoitukset (int, peräkkäiset arvot)
- Hajautustaulun tavoite
- Hajautusfunktio avain->int
- Sitten int -> % n
- Useat osumat

Tampereen yliopisto

Hajautustaulujen toteutus

COMP.CS.300 Tietorakenteet ja algoritmit 1

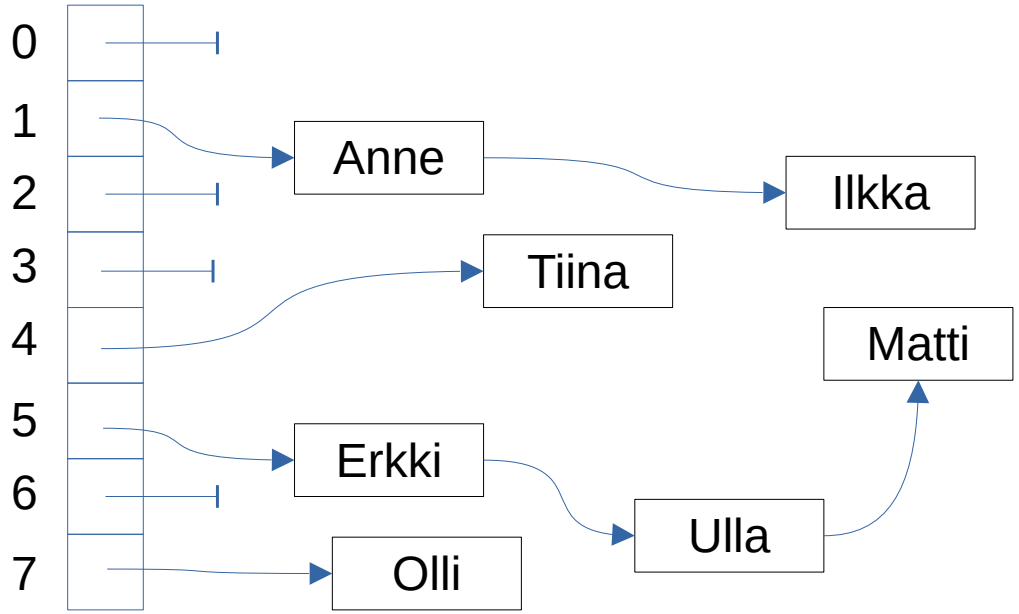
Matti Rintala (matti.rintala@tuni.fi)

Tampereen yliopisto

Ketjutettu hajautus (chained hashing)

Tampereen yliopisto

Esimerkki



hash	alkukirjain			
0	_	H	P	X
1	A	I	Q	Y
2	B	J	R	Z
3	C	K	S	Å
4	D	L	T	Ä
5	E	M	U	Ö
6	F	N	V	
7	G	O	W	

Tampereen yliopisto

Suljettu hajautus (closed hashing)

- Ämpäri terminä, sen idea
- Vaihtoehtoja samaan ämpäriin osumiselle
  - Ketjutettu hajautus
  - Suljettu hajautus
- Esimerkki (huonosta) hajautuksesta (vanhasta prujusta)
- C++:n ämpärirajapinta

Tampereen yliopisto

Hajautusfunktiot

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

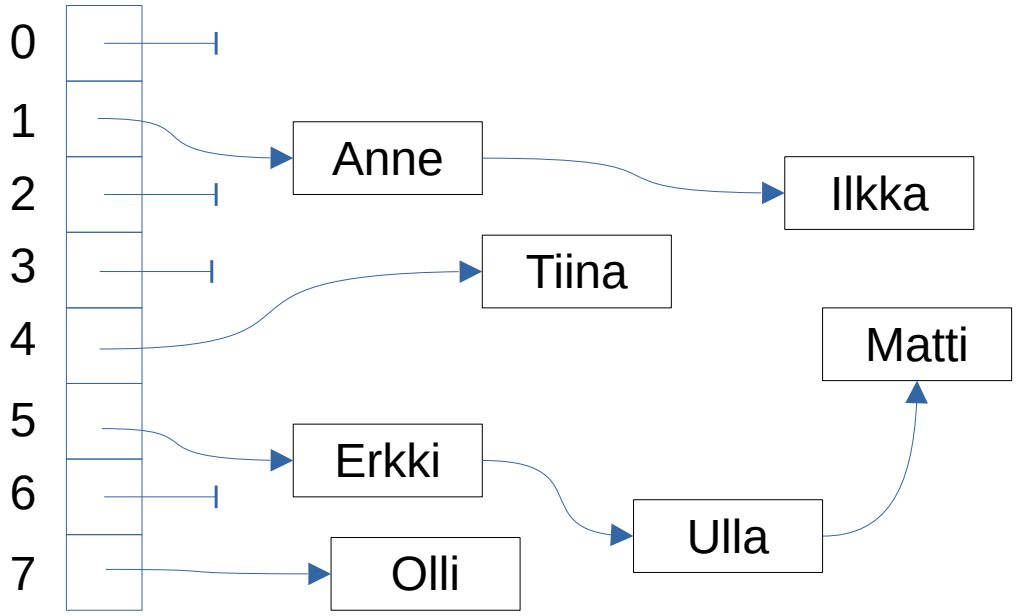
Tampereen yliopisto

Hajautusfunktiot

- Hyvä hajautusfunktio
  - Nopea
  - Deterministinen (aina sama tulos samalla avaimella)!
  - Tulos mahd. tasaisesti jakautunut kokonaisluku
  - Käyttää kaikkea hakuavaimen dataa
  - Tasoittaa avaimissa esiintyvät säännönmukaisuudet

Tampereen yliopisto

Hajautusfunktiot



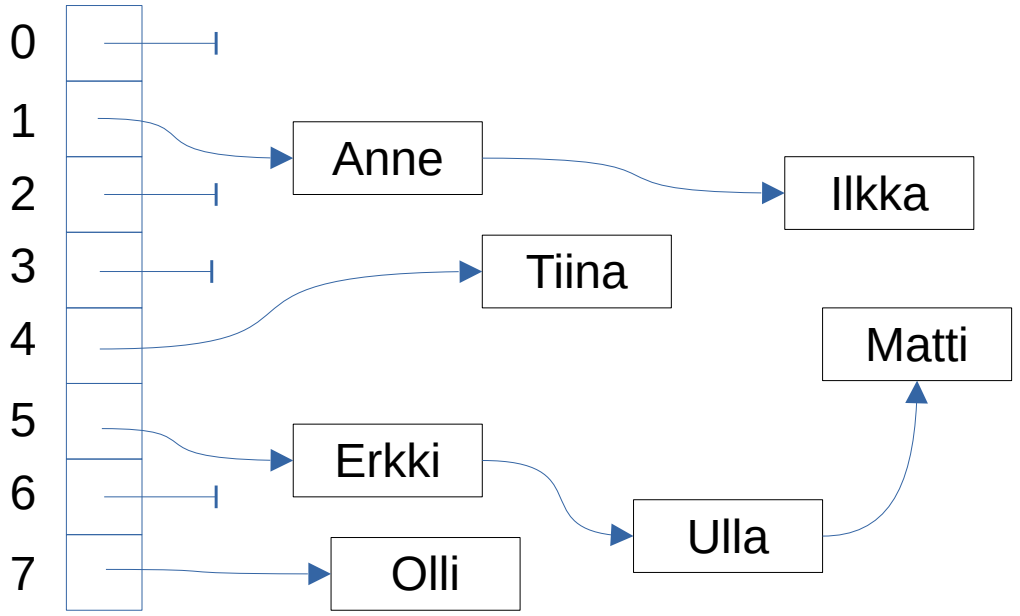
Hajautusfunktiot

- Käytännössä
  - Hyvän hajautusfunktion itse keksiminen vaikeaa!
  - (Huonon keksiminen turhankin helppoa)
  - **Älä keksi itse**, etsi hyväksi todettu funktio hakuavaimellesi
  - Ohjelmointikielissä ja kirjastoissa usein valmiit (toivottavasti hyvät) hajautusfunktiot perustyypeille, esim. `std::hash<tyyppi>`
  - Jos avaimessa monta osaa, voi joka osan hajauttaa erikseen ja yhdistää osat sopivalla kaavalla (älä keksi sitäkään itse!)

Tampereen yliopisto

Jakomenetelmä & ämpäreiden määrä

- Lasketaan datasta nopeasti & helposti yksi kok.luku *k* (esim. summa tms.)
- Jakojäännös (`k % ämpärilkm`)
- Jos tuloksessa säännöllisyyttä: riski epätasaisesta hajautumisesta ämpäreihin
- Jos ämpäreiden määrä (sopiva) **alkuluku**, riski pienempi


0
1
2
3
4
5
6
7
Anne
Ilkka
Tiina
Matti
Erkki
Ulla
Olli

Toinen esimerkki hajautusfunktiosta

- Kertomenetelmä
  - Yhdistetään data yhdeksi kokonaisluvuksi *k*
  - *m* on ämpäreiden määrä
  - Valitaan vakio A siten, että 0.0 < A < 1.0
  - $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$
  - Mikä on hyvä arvo A:lle?
  - Suurin osa arvoista ok
  - Kuulemma A ≈ $\frac{\sqrt{5}-1}{2}$ usein hyvä

- Millainen on hyvä hajautusfunktio
- Valmiit hajautusfunktiot
- Esimerkkejä hajautuksesta
- Hajautusten yhdistäminen
- Miksi ämpäreiden määrä alkuluku?
- C++ ja omat hajautusfunktiot unordered_map/set:ssä

Tampereen yliopisto

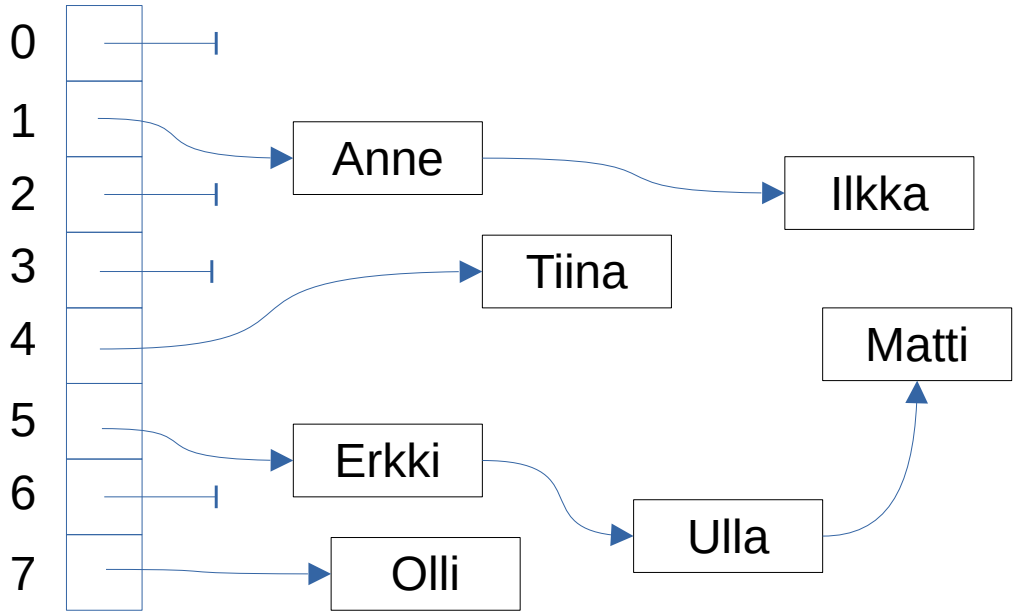
Hajautustaulun tehokkuus, uudelleenhajautus (rehashing)

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

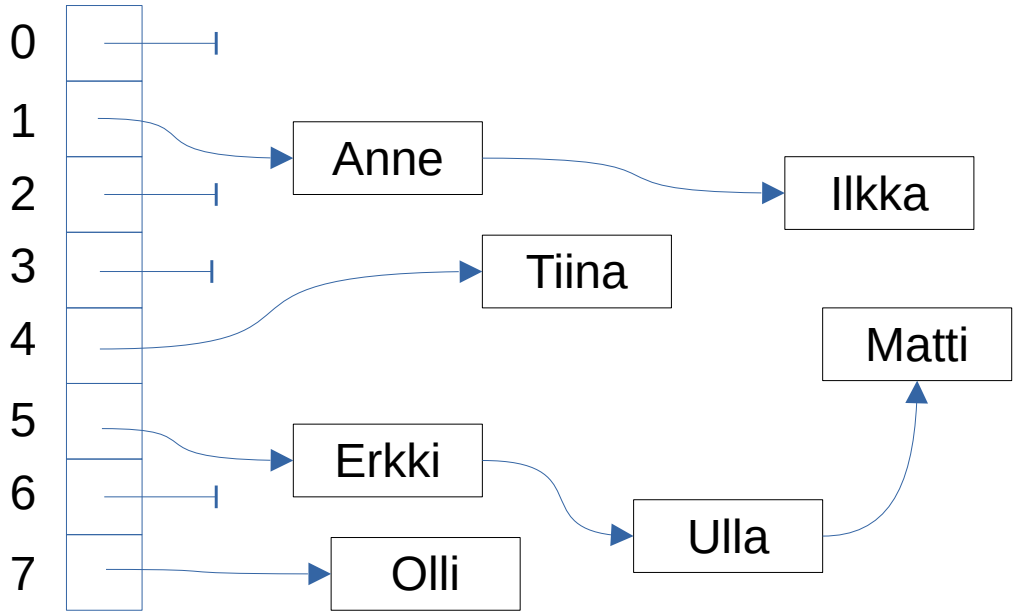
Tampereen yliopisto

Hajautustaulun tehokkuus



Uudelleenhajautus

- H:taulun *täyttöaste (load factor):* n / ämpärilkm
- Asetetaan tälle yläraja
- Yläraja ylitys → varataan isompi vektori, hajautetaan alkiot sinne
- Esim. ~ tuplataan ämpärit → lisäys *amortisoidusti* keskim. vakio
- Huom! Yksittäinen ämpäri voi edelleen olla iso!



- Hajautustaulun tehokkuus
- Uudelleenhajautuksen tarve, vaikutus asympt. tehokkuuteen
- Täyttöaste (load factor)
- Uudelleenhajautus
- Pahin tapaus uudelleenhajautuksesta riippumatta
- C++:n uudelleenhajautusrajapinta

Tampereen yliopisto

Algoritmi(*par*)

```
1 koodia                    (kommentti)
2 koodia
```

Tampereen yliopisto

(kommentti)

Tampereen yliopisto