# Heš indeksi i direktna organizacija

Mehanizmi za ubrzavanje operacija

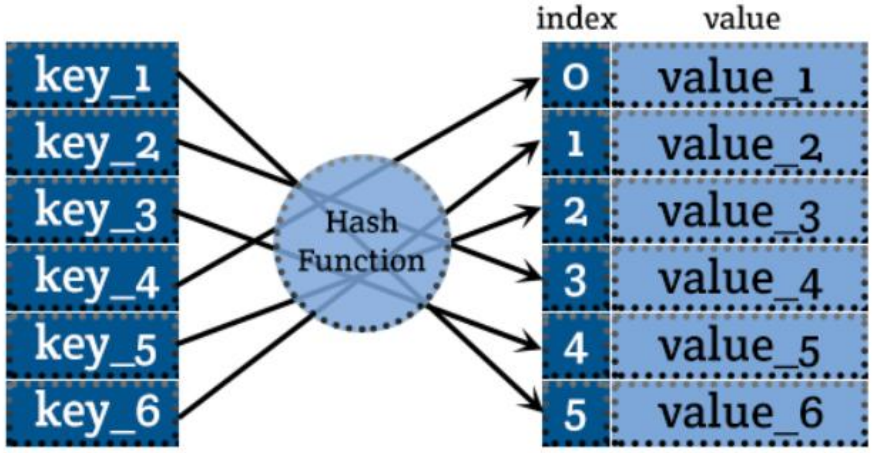
# Heširanje

- Heširanje – tehnika pretrage podataka (na osnovu vrednosti ključa naći podatak pridružen ključu) u kom se pozicija traženog para ključ-vrednost određuje direktno.
  - Parovi ključ-vrednost su smešteni u strukturi nazvanoj **heš tabela ili heš mapa**.
  - Za određivanje pozicije para ključ-vrednost se koristi **heš funkcija**.
  - Heš funkcijom se vrši transformisanje ključa u ceo broj (**heš vrednost**) koji pripada opsegu indeksa u tabeli i predstavlja poziciju u tabeli (indeks, ofset) na kojoj se nalazi traženi par.
  - Heš funkcija se koristi da locira zapise prilikom pristupa, dodavanja i brisanja

heš funkcija

ključ -> pozicija u tabeli -> <ključ, vrednost>

heš vrednost ili heš kod



# Heš tabela

- Kompleksnost (n – broj vrednosti ključa)
  - Memorijska kompleksnost - O(n)
  - Vremenska kompleksnost - prosečna O(1), najlošija O(n)
- Pri odabiru tehnike heširanja, tj. implementaciji heš tabela, potrebno je:
  - Odbrati efikasnu heš funkciju.
  - Ukoliko broj ključeva prevazilazi broj različitih heš vrednosti (ako dolazi do kolizije ključeva) onda treba odrediti hešing šemu (izabrati način razrešavanja kolizije ključeva).
    - Manja kolizija -> Veća heš tabela
    - Manja tabela -> veća kolizija -> sporije pronalaženje traženih slogova.

# Heš funkcija

## Uređeni indeksi

O heš funkciji

# Heš funkcija – osobine

- Za svaki ključ vraća njegovu integer reprezentaciju.
  - Mora vratiti broj od 0 do tablesize (veličina heš tabele).
- Idealno, heš funkcija bi trebalo da preslikava svaki mogući ključ u zaseban indeks, ali ovaj zadatak je u praksi najčešće neostvariv.

  Ako bismo hteli da vrednosti heš-funkcije budu jedinstvene za svaku nisku dužine do 14 karaktera koja se sastoji samo od malih slova engleske abecede, bilo bi nam potrebno

  $$1 + 26^1 + 26^2 + \cdots + 26^{14} = \frac{26^{15} - 1}{26 - 1} \approx 26^{14} \approx 2^{66} > 2^{64}$$

  Dakle, za potreban broj različitih heš vrednosti (indeksa/adrese podatka u heš tabeli) ne bi bio dovoljan opseg 64-bitnog celog broja, što je najširi primitivni celobrojni tip podataka.

# Kolizija

- Ako je veličina heš tabele manja od broja različitih ključeva kolizija ključeva je nužna.
  - Heš kolizije / sudari – različiti ključevi sa istim heš vrednostima.
  - Svakoj heš vrednosti je pridružen jedan skup parova ključ-vrednost.
  - Ključevi sa istom heš vrednošću se smeštaju i isti baket (*bucket*).

Najlošija moguća heš funkcija sa kolizijom

Sve vrednosti ključa se slikaju u istu heš vrednost.

Idealna heš funkcija sa kolizijom

Ravnomerno raspodeljuje ključeve tako da je svakoj heš vrednosti pridružen isti broj slogova.

# Hash funkcija

- U DBMSovima se izbegavaju kriptografske heš funkcije.
- Faktori na osnovu kojih se vrši odabir su brzina i mala stopa kolizije ključeva.
- Primeri
  - **CRC-64(1975)** – detekcija grešaka na mrežama.
  - **MurmurHash(2008) –** brza, opšta primerna.
  - **Google CityHash(2011)** - modifikovan Murmur, prilagođen kraćim ključevima (<64 bytes).
  - **Facebook XXHash(2012) -** From the creator of zstd compression.
  - **Google FarmHash(2014) – novija verzija** CityHash sa boljim stopama kolizije.

# Heš funkcija sa kolizijom – osobine

Poželjne osobine:

- Uniformna distribucija.
  - Svakoj heš vrednosti se pridružuje isti broj vrednosti ključa iz skupa svih mogućih vrednosti.
- Distribucija je slučajna.
  - U proseku, za svaku heš vrednost postoji približno isti broj parova ključ-vrednost koji su mu dodeljeni, bez obzira na trenutnu raspodelu vrednosti ključa.
  - Hash vrednost ne sme da bude u korelaciji sa bilo kojim kriterijumom uređivanja vrednosti ključa, npr. alfabetskim ili uređenjem prema dužini zapisa ključa.

Dobra heš funkcija ima prosečno vreme pronalaženja traženog sloga blisko maloj konstantnoj vrednosti, koja je nezavisna od broja trenutno upisanih vrednosti.

*Bucket skew* (neravnomerne korpe) – neravnomerna popunjenost korpi, može se desiti ako distribucija nije uniformna ili nije slučajna.

# Heš funkcija – osobine primer

`Plate (ogranak, prosecna_plata)`

Heš funkcija definisana nad poljem ogranak koja mapira nazive mesta ogranaka jedne firme sa i-tim početnim slovom u i-ti bucket.

- Neuniformna distribucija vrednosti ključa, jer ima više mesta sa početnim slovom B nego X.

Heš funkcija definisana nad poljem račun koja podrazumeva 10 opsega, 1–10,000, 10,001–20,000, ...

- Raspodela jeste uniformna, ali nije slučajna, jer je realno očekivati da će najveći broj plata biti u rasponu od 50001-60000.

# Heširanje u DBMS-ovima

## Heš indeksi i direktna organizacija

Primena heširanja u DBMS-u

# Heširanje u DBMS-ovima

- U DBMS-u se koristi:
  - Za direktnu organizaciju fajlova (heš fajl organizacija) – *secundary storage hash table*

    <key,value>  = ceo slog,  u korpama su celi slogovi
  - Za organizaciju indeksa (heš indeksi) – *secundary storage hash table*

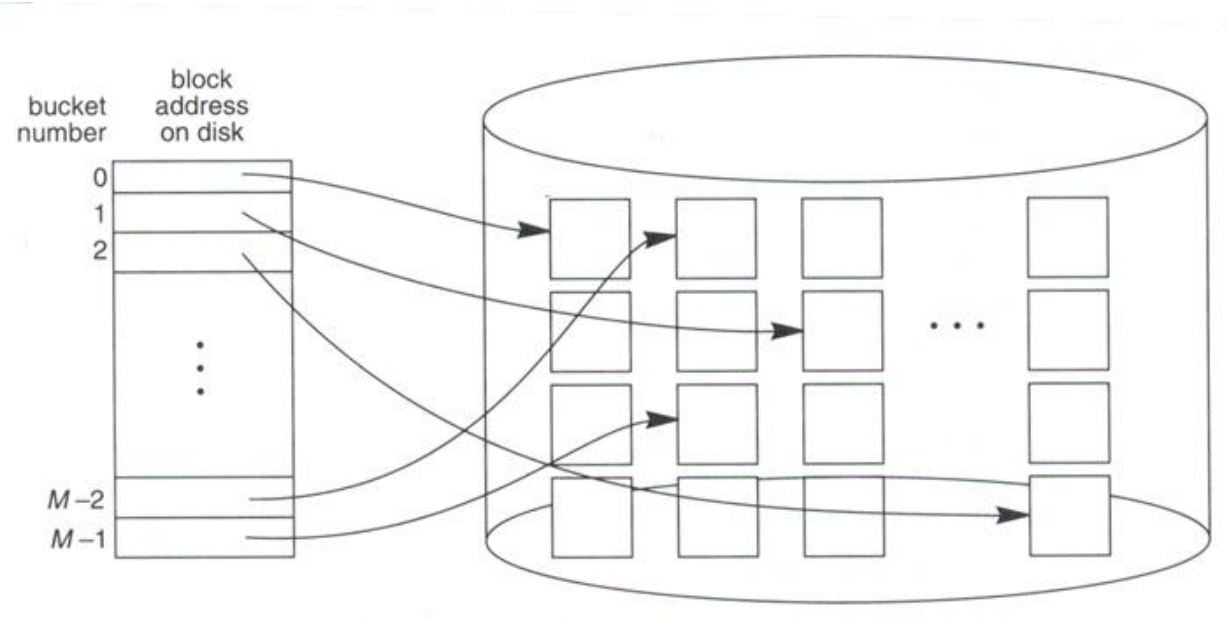
    <key,value>  = <ključ, rid>, u korpama su pokazivači na slogove
  - Za implementiranje privremenih struktura u radnoj memoriji tokom izvršavanja operacija (npr., join) – najčešći slučaj, *in-memory hash table*

# Unutrašnje i spoljašnje heširanje

*In-memory* vs. *secundary storage* heš tabela

- Unutrašnje heširanje – heš tabela sadrži cele slogove, u svakom slotu se nalazi jedan slog.
- Slogovi se nalaze na disku. Heš tabela sadrži samo ključ i pokazivač na lokaciju gde se slog može pronaći. Adresni prostor se sastoji od bucket-a. Svi slogovi sa istom heš vrednošću se nalaze u istom bucket (jedan bucket – minimalno jedan blok)

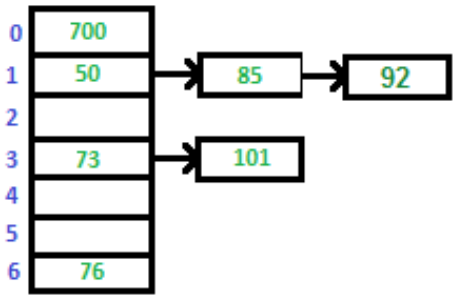
| | Name | Ssn | Job | Salary |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| | ⋮ | | | |
| $M-2$ | | | | |
| $M-1$ | | | | |

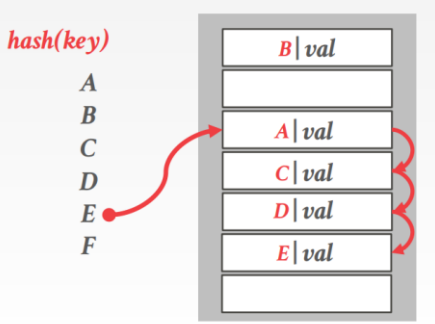


# *In-memory* vs. *secondary storage* heš tabela

- Ako je heš tabela kreirana u radnoj memoriji kao privremena struktura tada se implementira kao **niz (mapa) slogova**.

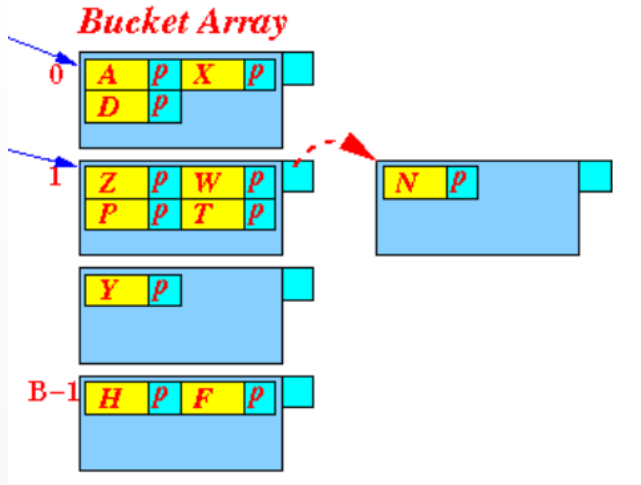
  U slučaju kolizija, slogovi se smeštaju u okviru niza ili se tabela implementira kao **niz lista slogova**, jedna lista za svaki baket.




- Ako je tabela kreirana za sekundarno skladište, kad se primenjuje na tabele i indekse, onda je uobičajeno da se implentira kao **kolekcija** (najčešće uzastopnih) **stranica**, primarnih stanica baketa.

  Slogovi čiji su ključevi heširani heš funkcijom h na određenu korpu dodaju se u stranicu te korpe. Ako je stranica korpe puna, najčešće se korpi dodaje stranica prekoračenja (overflow) ulančavanjem.



# Heš (direktna, rasuta) fajl organizacija

- Kod datoteke organizovane kao heš, baket čuva zapise
- Relacija je organizovana kao kolekcija strana. Heš funkcija određuje korpu, odnosno primarnu stranicu korpe u koju slog treba da bude smešten ili tražen.
- Jedna relacija može imati samo jedan atribut (ključ) direktnog pristupa, a više indeksiranih atributa.

<key,value>  = ceo slog

bucket 0

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

bucket 1

| | | | |
|---|---|---|---|
| 15151 | Mozart | Music | 40000 |
| | | | |
| | | | |
| | | | |

bucket 2

| | | | |
|---|---|---|---|
| 32343 | El Said | History | 80000 |
| 58583 | Califieri | History | 60000 |
| | | | |
| | | | |

bucket 3

| | | | |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |
| 33456 | Gold | Physics | 87000 |
| 98345 | Kim | Elec. Eng. | 80000 |
| | | | |

bucket 4

| | | | |
|---|---|---|---|
| 12121 | Wu | Finance | 90000 |
| 76543 | Singh | Finance | 80000 |
| | | | |
| | | | |

bucket 5

| | | | |
|---|---|---|---|
| 76766 | Crick | Biology | 72000 |
| | | | |
| | | | |
| | | | |

bucket 6

| | | | |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| | | | |

bucket 7

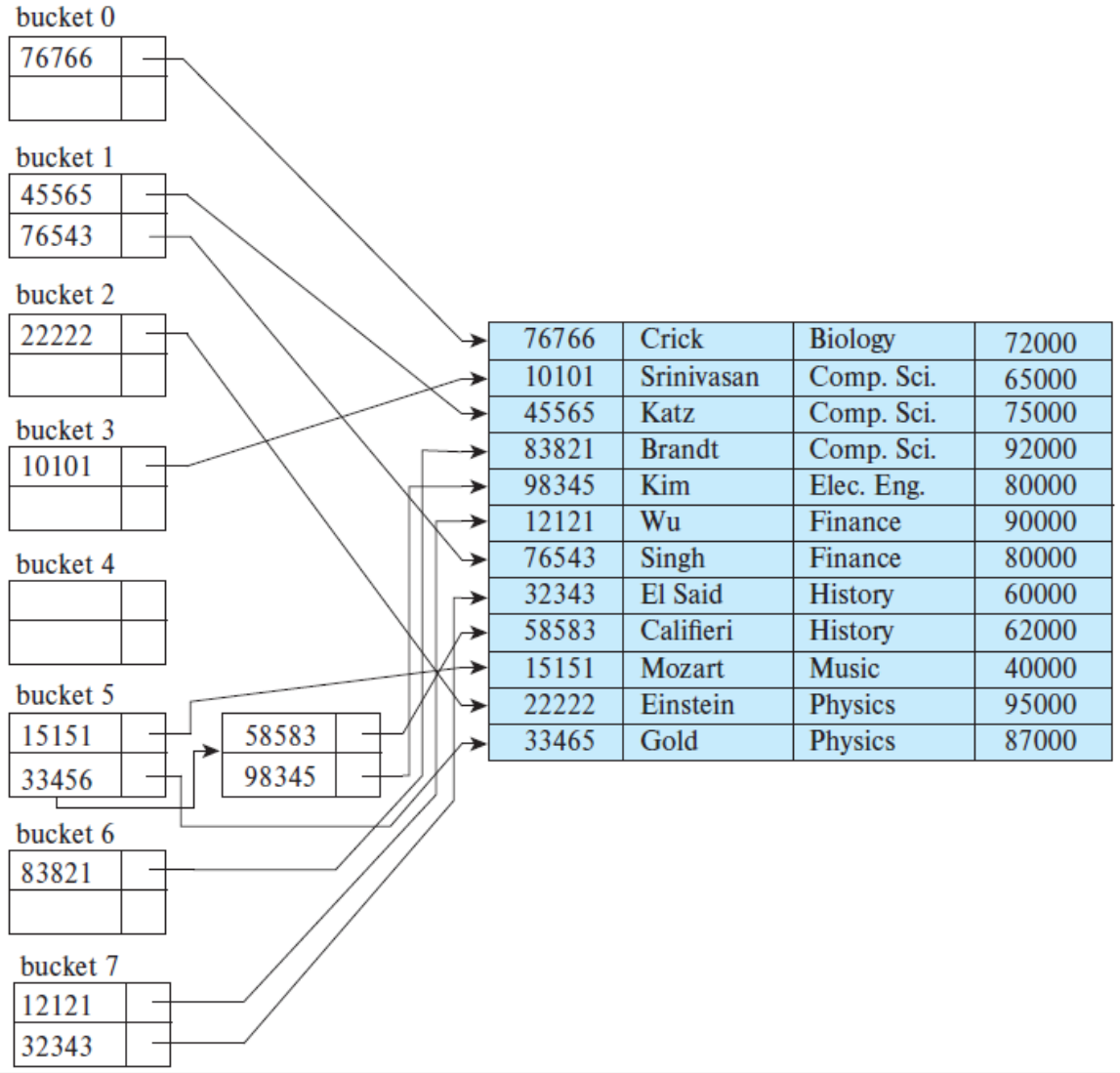
| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

# Heš (heš-bazirani) indeksi

- Heš fajl organizacija se može primeniti na fajlove sa podacima, ali i na indekse.
- Dodatna struktura, kao i B+ stablo
- Korpa čuva parove sa pokazivačima na zapise

  <key,value> = <ključ, RID>

- Heš indeks je sekundarna indeksna struktura. Heširanje je primenjeno na parove (key,RID), gde je key ključ pretrage sekundarnog indeksa.

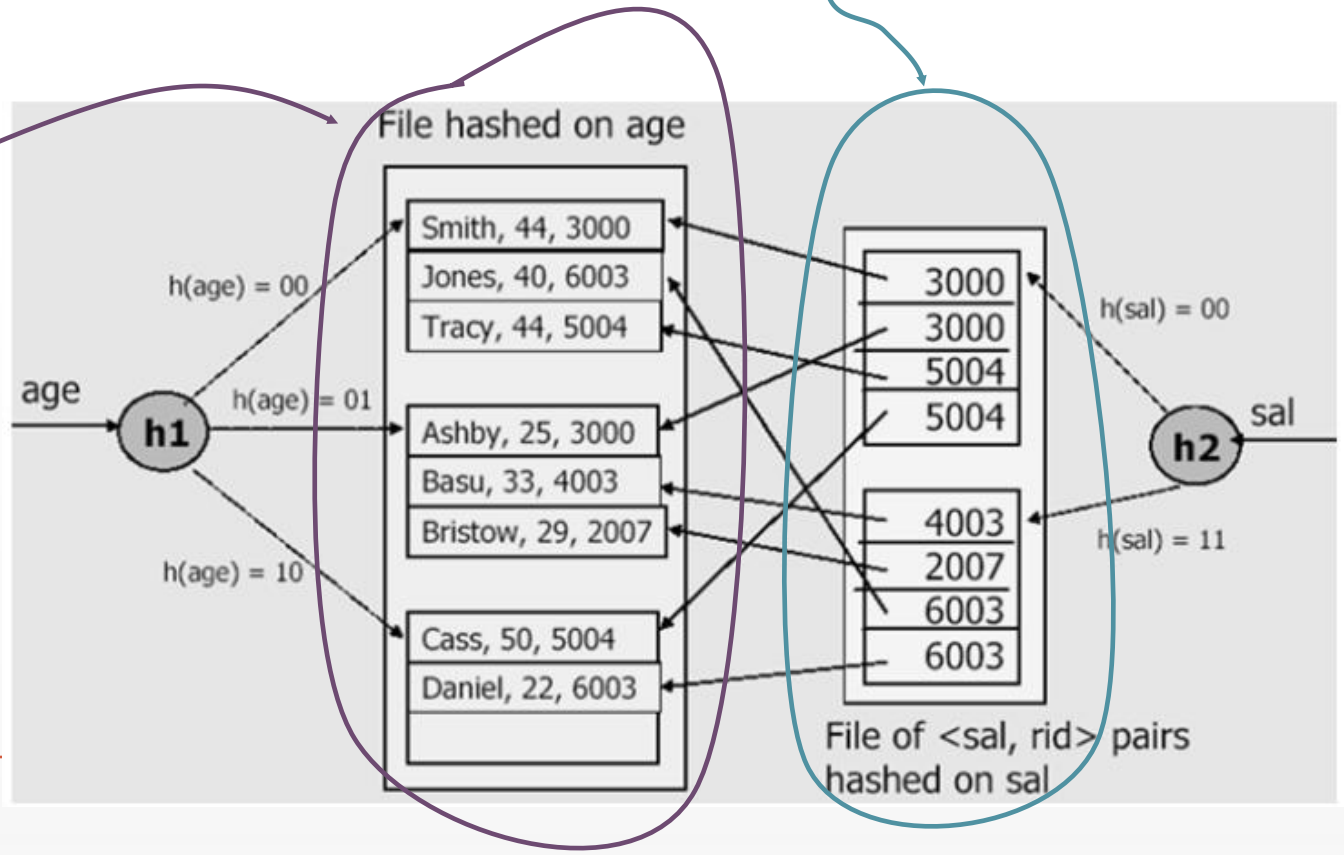


# Heš indeks vs. Heš fajl organizacija

**Heš Heš indeks**
Heap file Unclustered
Hashed Index on sal Hash function identifies appropriate bucket

**Heš fajl organizacija**
data stored in file hashed on age, no index used

Ovakva organizacija se može nazvati i klasterisanim hash indeksom



# Hešing šeme

## Uređeni indeksi

Razrešavanje kolizija

# Hešing šema

- Hešing šema – strategija razrešavanja kolizije (situacije u kojoj je lokacija na koju treba smestiti novi slog već zauzeta)
- Prvo pitanje – Da li je broj heš vrednosti (korpi) fiksan?
  - Fiksan broj heš vrednosti – statičko heširanje
  - Dinamičko heširanje – skup heš vrednosti koji se menja tokom vremena.

# Statičko heširanje

Gde smestiti novi slog ako je korpa kojoj pripada zauzeta/puna? Osnovna podela strategija za razrešavanje:

- Otvoreno heširanje - otvoreno adresiranje

  kolizije se razrešavaju u okviru postojećih korpi, ako je ciljana zauzeta, koristi se prostor slobodne
  nije koristi se za heširanje na sekundarnim skladištima

- Zatvoreno heširanje – posebno (odvojeno) ulančavanje

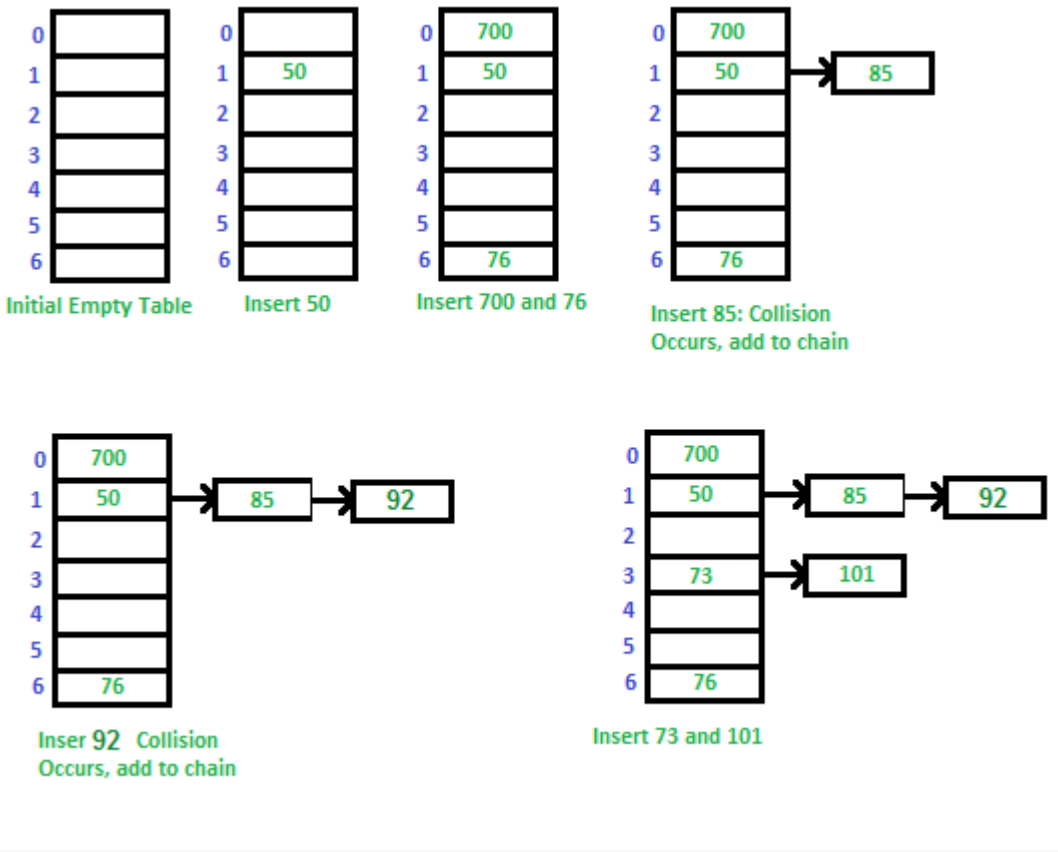
  kolizije se razrešavaju ulančavanjem dodatnih stranica prekoračenja, a sami slogovi dosledno ostaju u istoj korpi

# Otvoreno adresiranje

- Svi parovi ključ-vrednost su smešteni u heš tabeli. Dakle, tabela mora biti dovoljno velika.
- Svodi se na pronalaženje neke druge lokacije u heš tabeli različite od one dobijene heš funkcijom.
- Formulisati pravilo koje za svaki ključ određuje niz adresa u tabeli koje se proveravaju pri umetanju novog ključa.
- Neke od tehnika otvorenog adresiranja:
  - linearno pretrazivanje (linear probing)
  - slučajno pretraživanje
  - kvadratno pretraživanje
  - dvostruko heširanje

# Zatvoreno adresiranje - Odvojeno ulančavanje

Ako je in-memory hashing, za svaku vrednost heš ključa postoji posebna lista slogova.



# Upis/Brisanje – spoljašnje zatvoreno heširanje

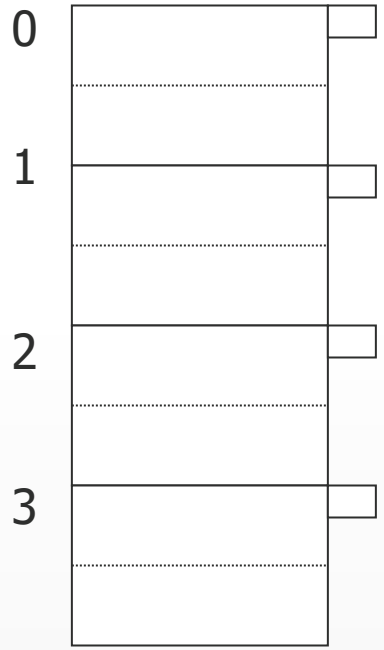
INSERT:

h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0



# Upis/Brisanje

INSERT:

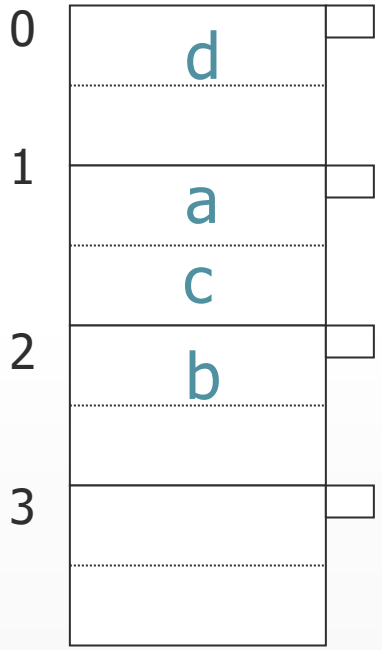
h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

h(e) = 1



# Upis/Brisanje

INSERT:

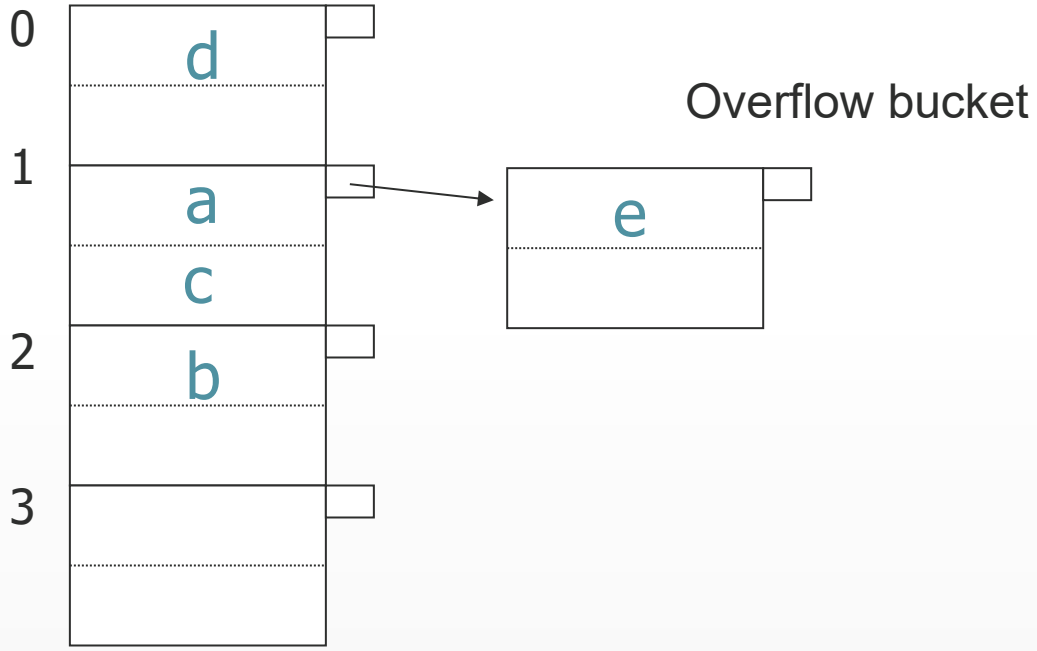
h(a) = 1

h(b) = 2

h(c) = 1

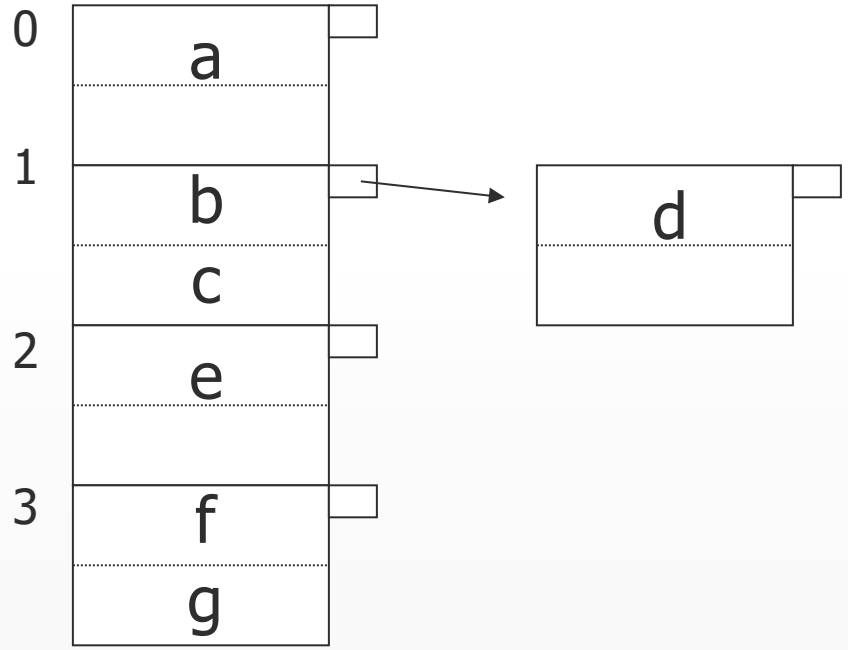
h(d) = 0

h(e) = 1



# Upis/Brisanje

Delete:
e
f

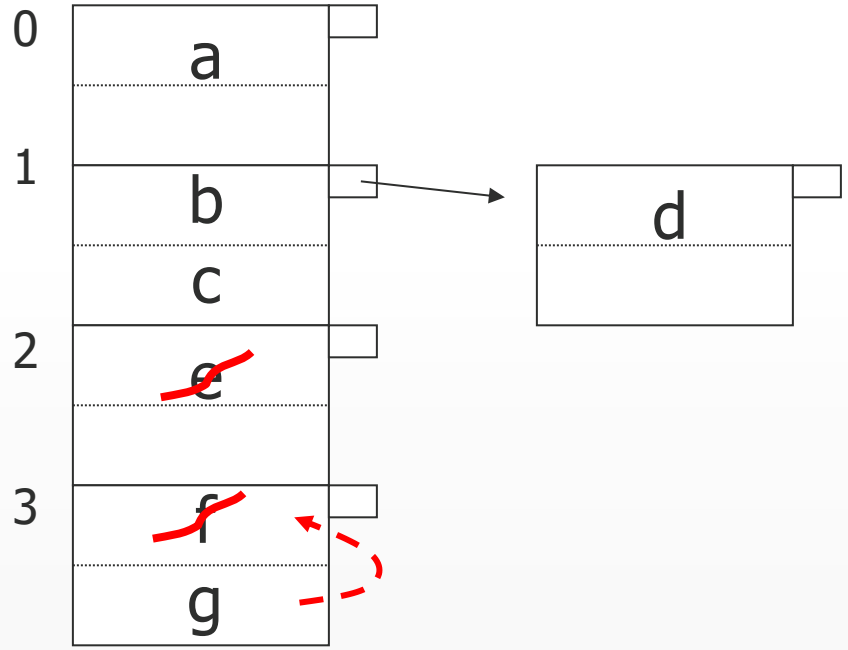


# Upis/Brisanje

Delete:

e

f



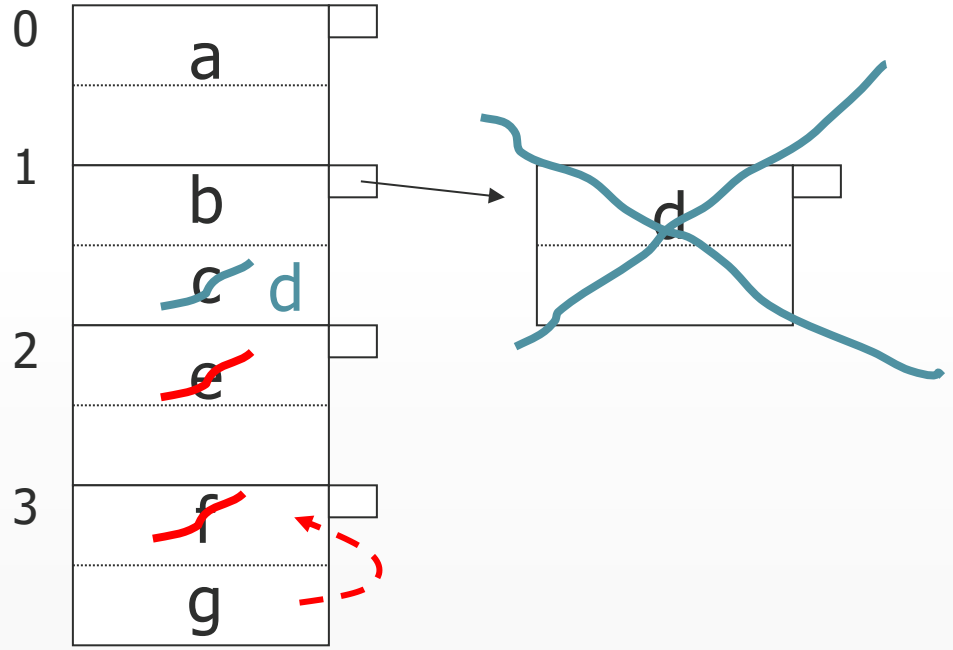
# Upis/Brisanje

Delete:

e

f

c



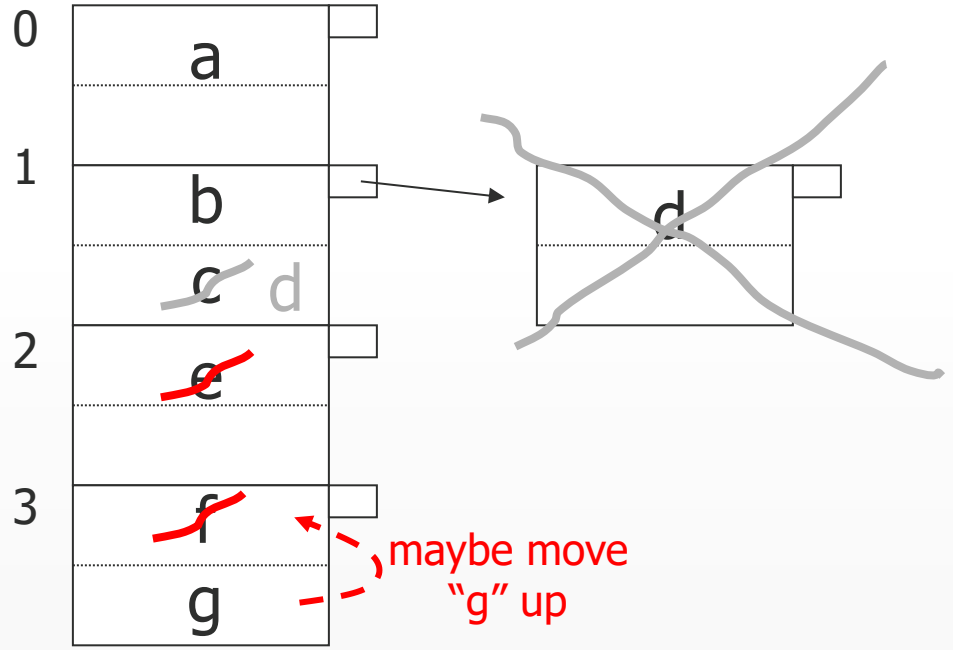
# Upis/Brisanje

Delete:

e

f

c



# Statičko heširanje – broj korpi

- Statičko spoljašnje heširanje - kada datoteka raste, raste dužina lanaca stranica prekoračenja, što degradira performanse rasute organizacije fajla.
- Broj korpi mora da zadovoljii uslov

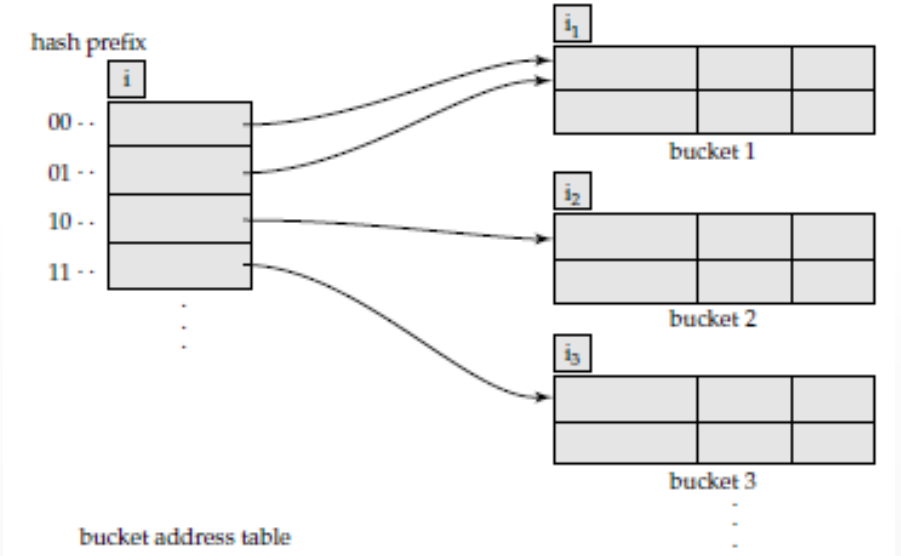
  nB > nr/fr, nB – broj korpi, nr - ukupno slogova, fr – kapacitet korpe
- Moguće je:
  - Definisati heš funkciju na osnovu trenutne veličine skupa slogova.
    Loš izbor, jer brzo dolazi do prekoračenja.
  - Definisati heš funkciju na osnovu pretpostavljene veličine fajla u budućnosti.
    Trošenje memorijskog prostora preko potrebnog.
  - Periodična reorganizacija heš strukture u skladu sa rastom fajla.
    Vremenski zahtevno. Potrebna zabrana pristupa tokom reorganizacije
- Rešenje: dinamičko heširanje.

Da bi se smanjilo nepotrebno trošenje stranica, a izbeglo prekoračenje bucket-a, obično se za smeštaj slogova ostavlja oko 20% više prostora u datoteci podataka i zoni prekoračenja, nego što je to potrebno s obzirom na broj i veličinu slogova.

(nr/fr)*(1+d)
gde je d obično 0.2

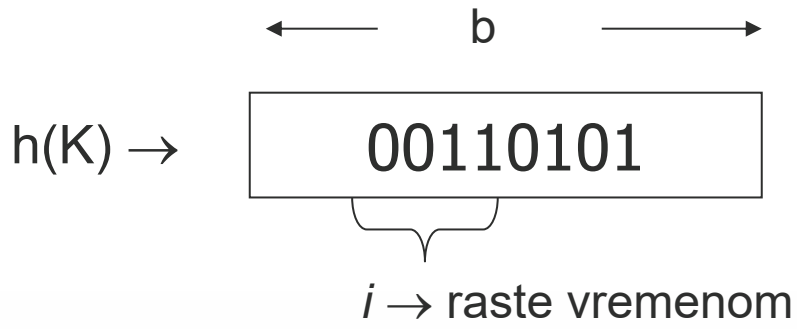
# Dinamičko heširanje

- Jedna od tehnika – proširivo heširanje (*Extendable hashing*)
- Adresna tabela sadrži zaglavlje u kojoj se nalazi dužina hash prefiksa.
- Funkcija heširanja prevodi ključ u binarni kod i uzima prvih i bitova kao takozvani pseudoključ.
- Svaki bucket ima svoju lokalnu dužinu prefiksa (lokalna dubina bloka) koja označava koliko prvih bitova pseudo ključa svih zapisa u bucket-u se poklapa.



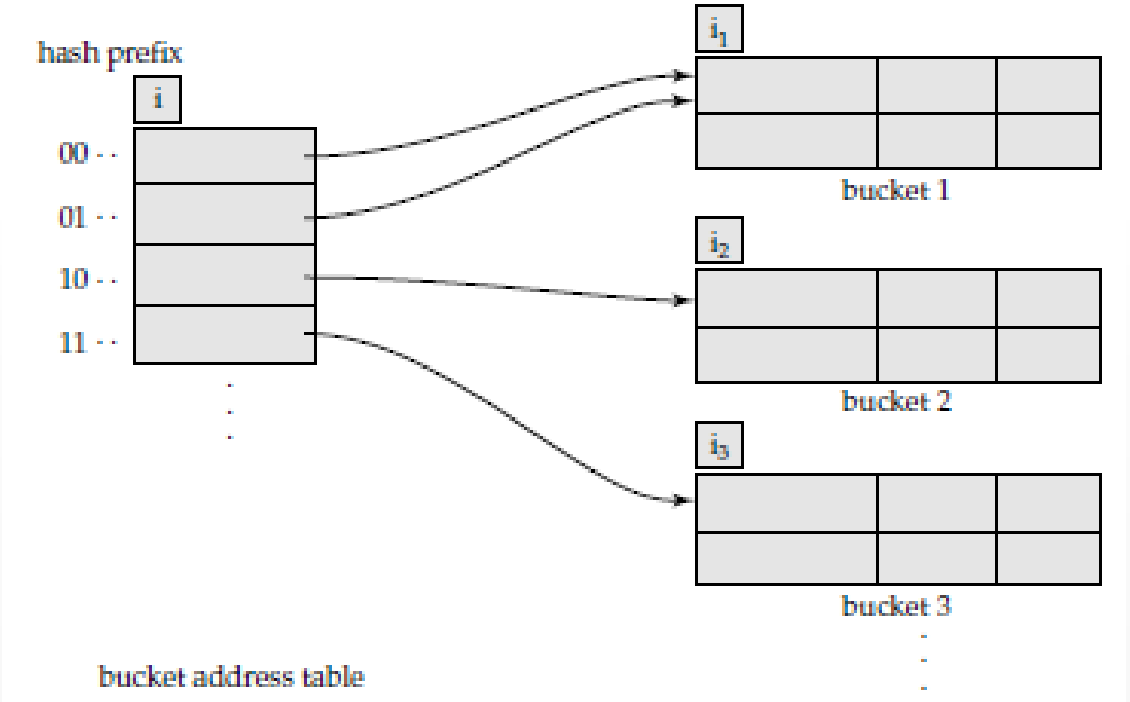
# Proširivo heširanje

(a) Upotreba *i* od *b* bitova vrednosti heš funkicje za datu vrednost ključa

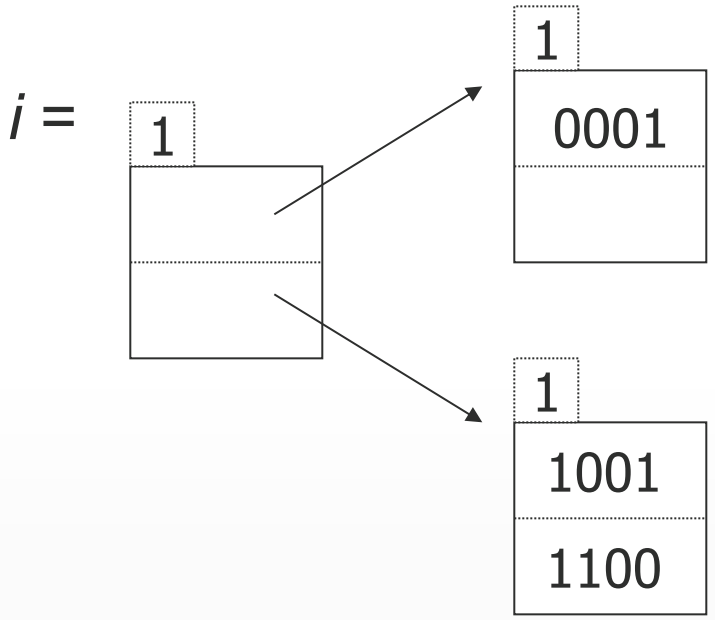



(b) Upotreba posebne strukture – direktorijum.

Strukturu čini zaglavlje u kome se daje dubina direktorijuma (d), a zatim $2^d$ pokazivača na adrese blokova.

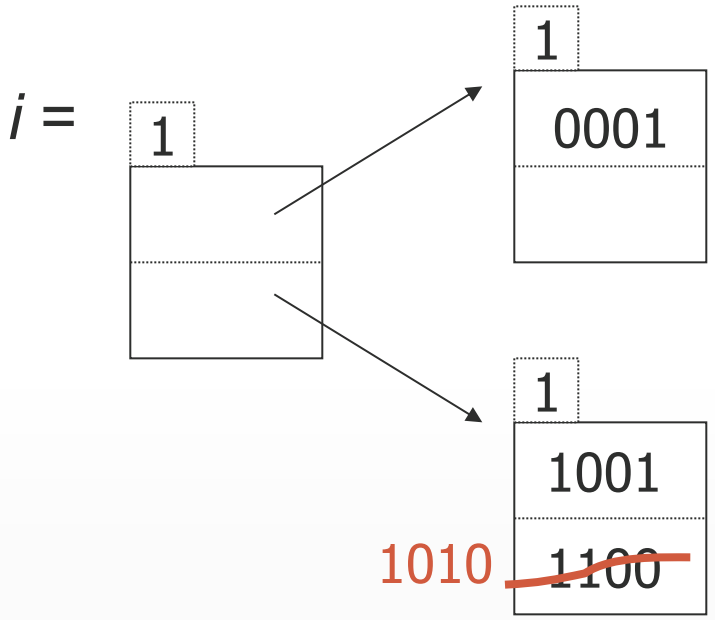


# Proširivo heširanje

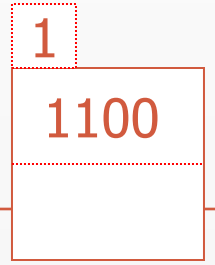



Insert 1010

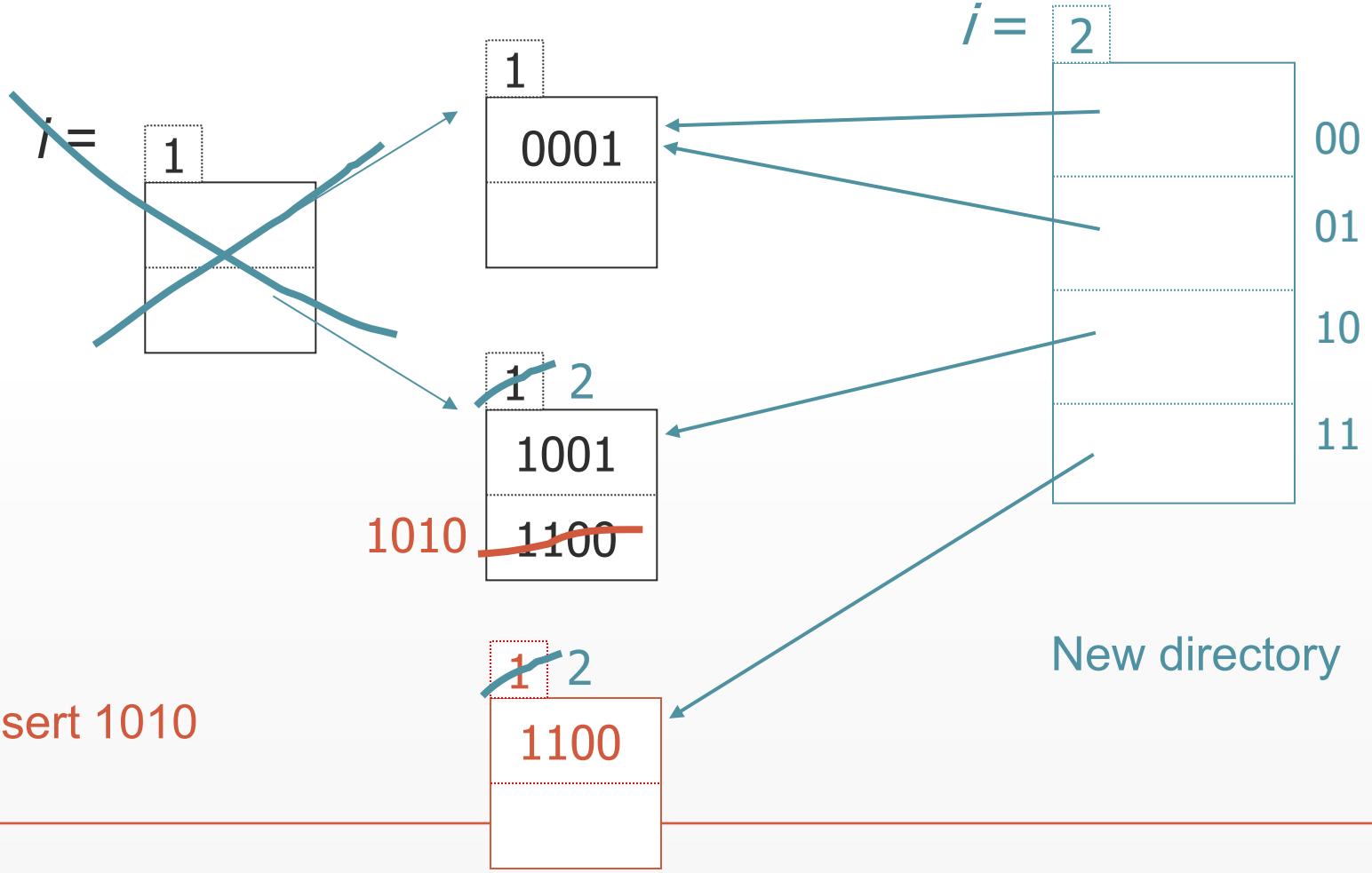
# Proširivo heširanje



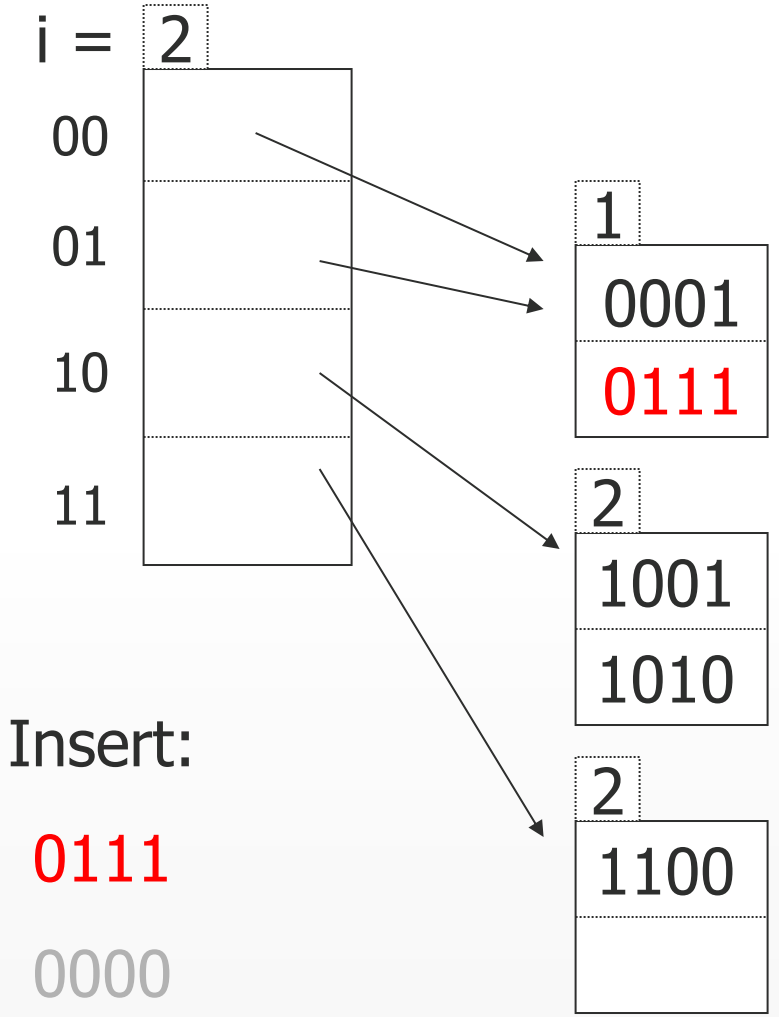

Insert 1010



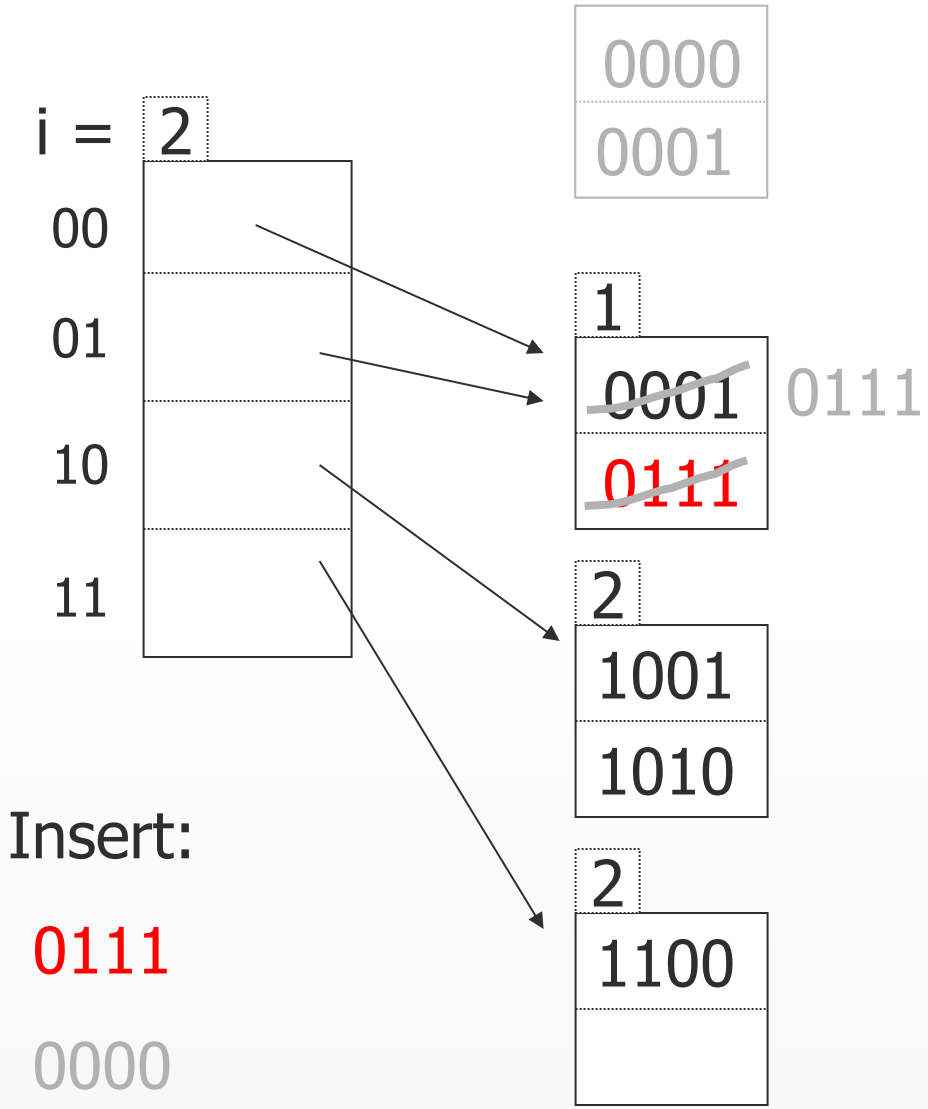
# Proširivo heširanje



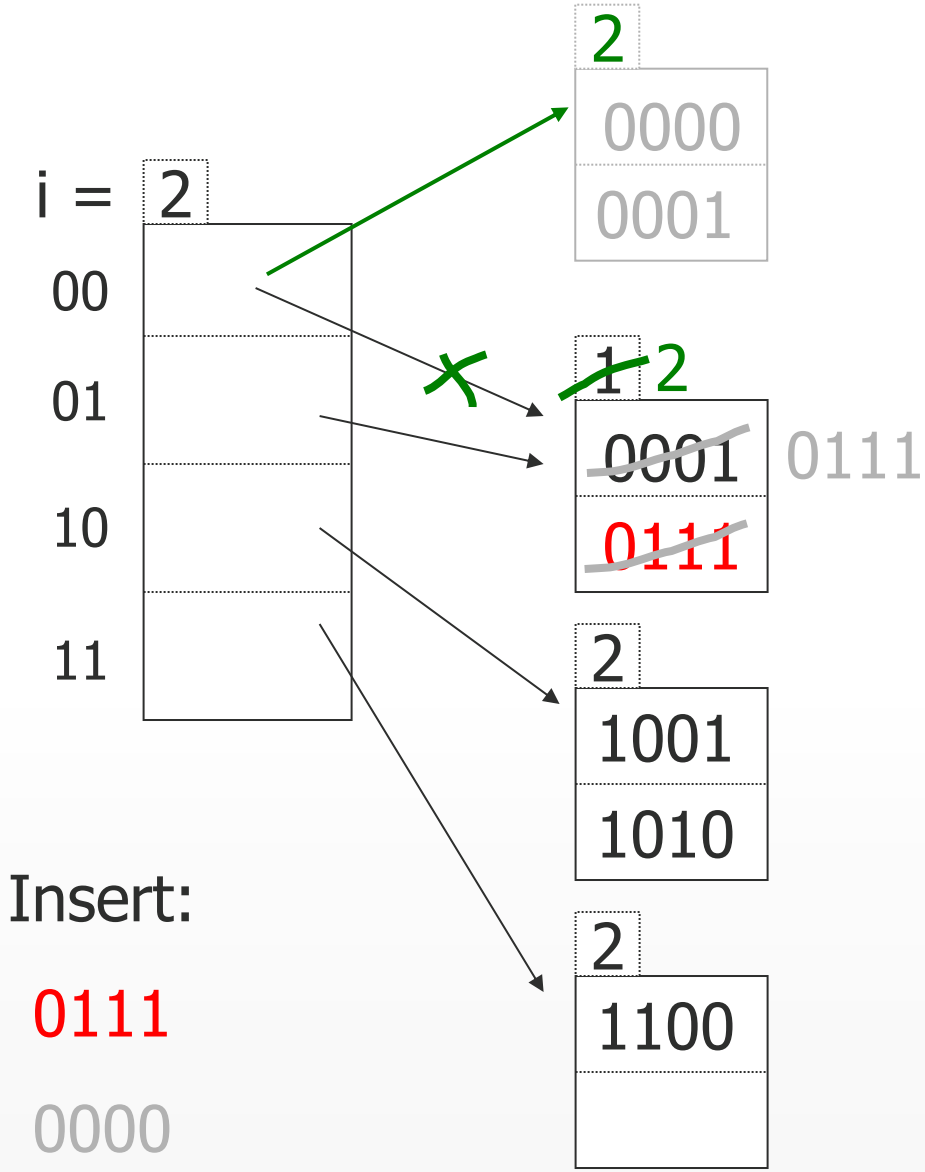
# Proširivo heširanje



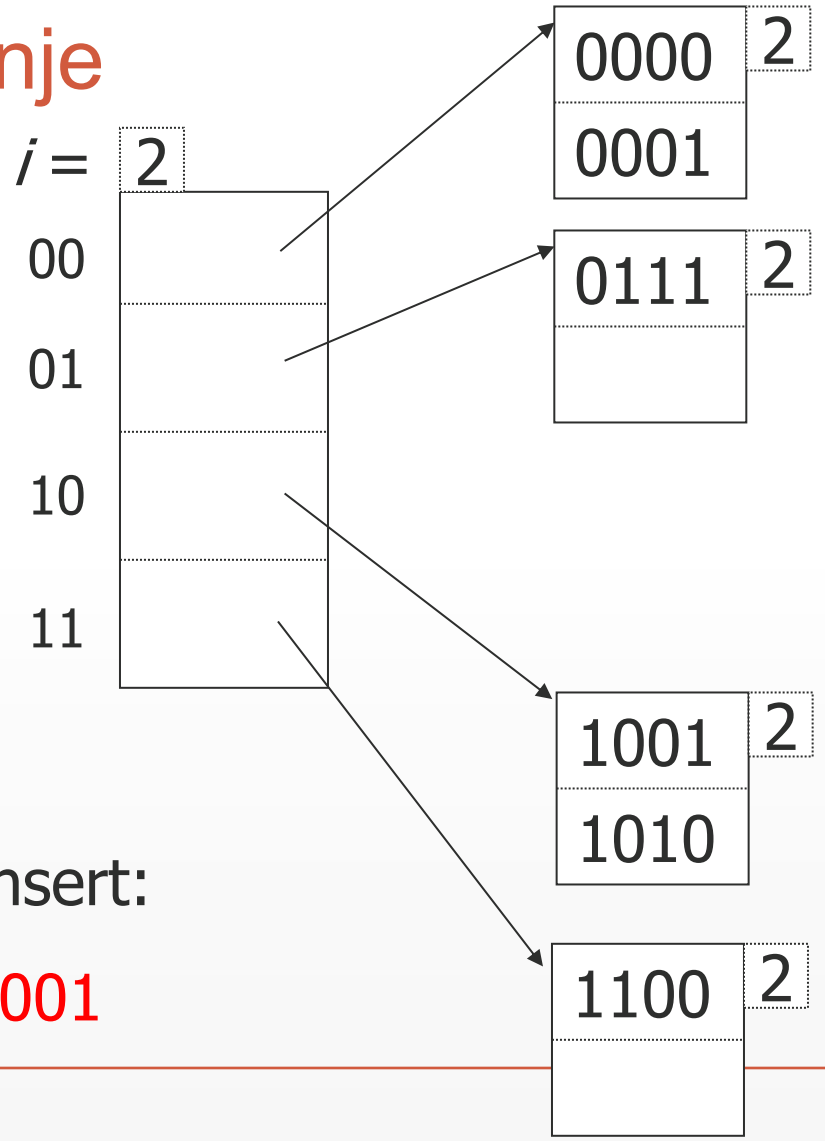
# Proširivo heširanje



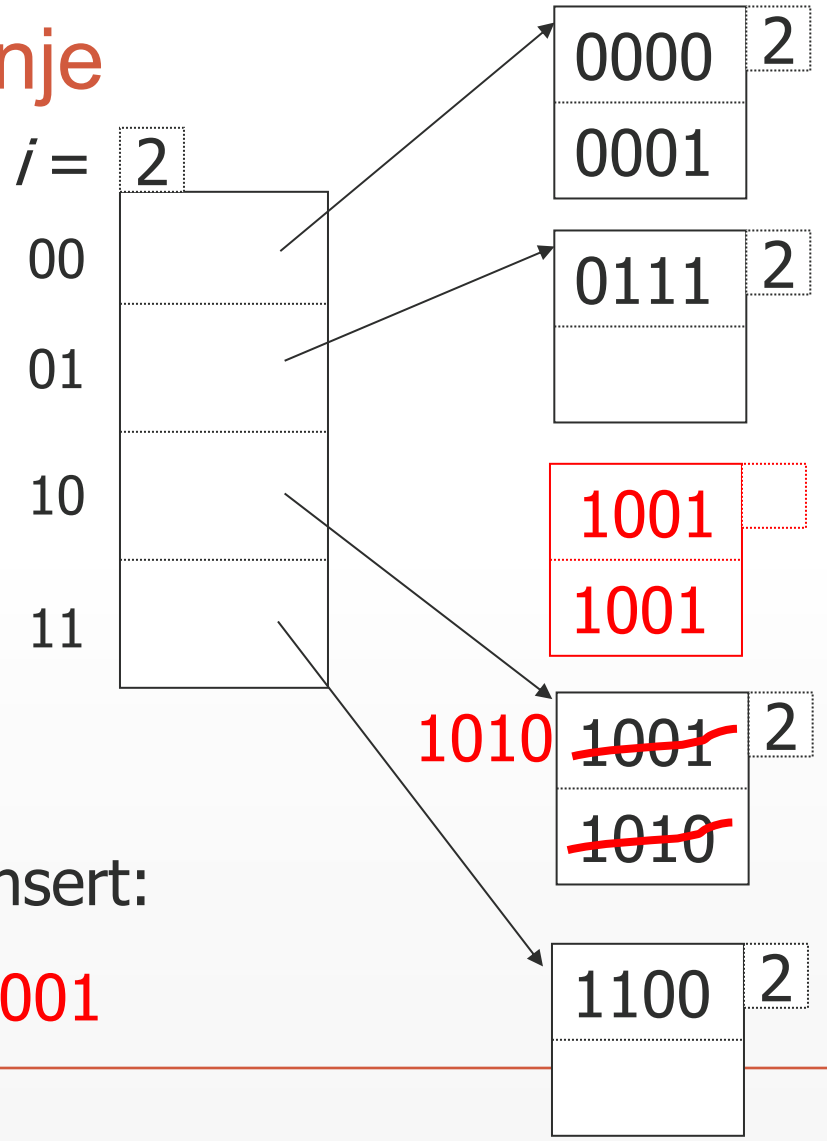
# Proširivo heširanje



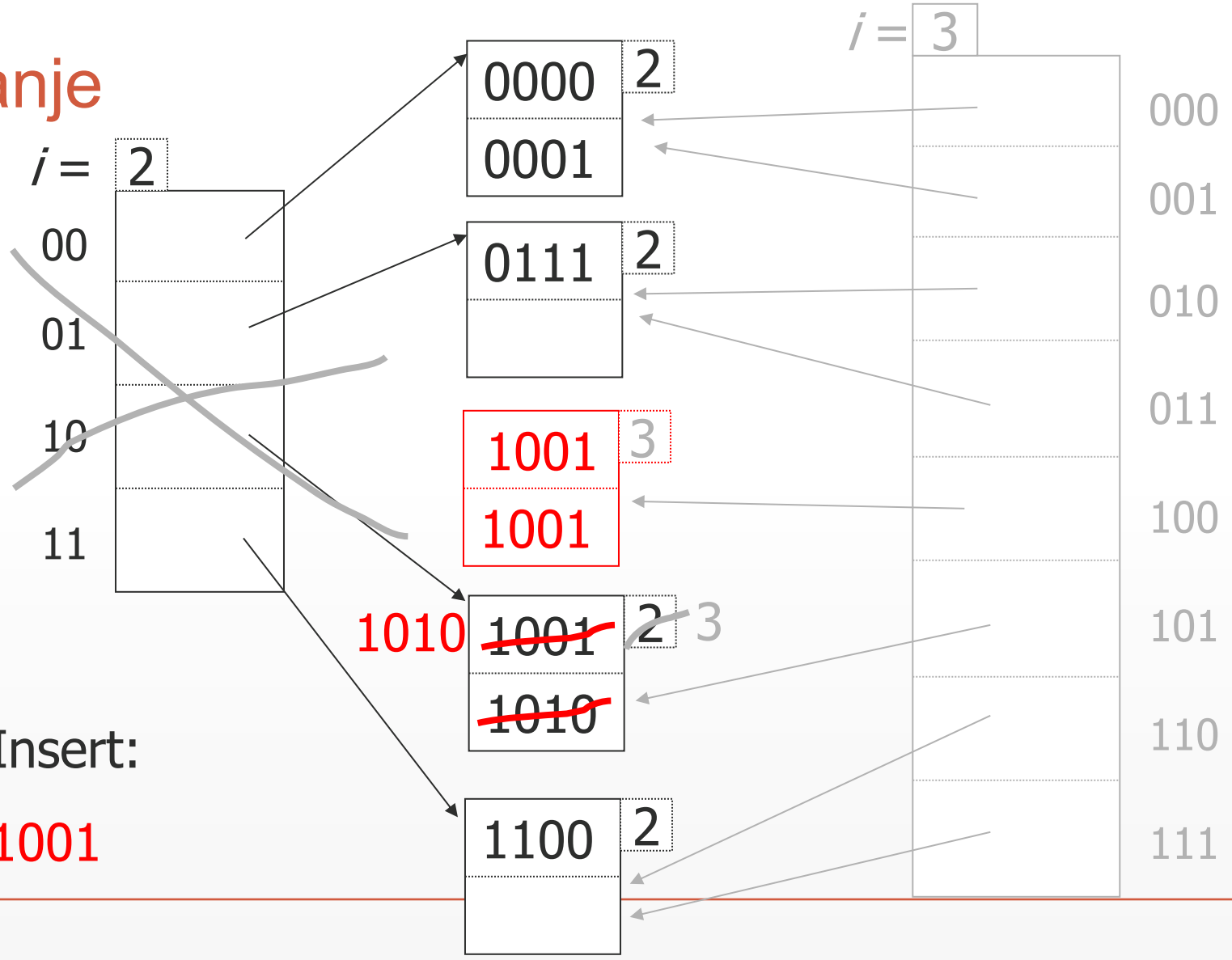
# Proširivo heširanje



# Proširivo heširanje

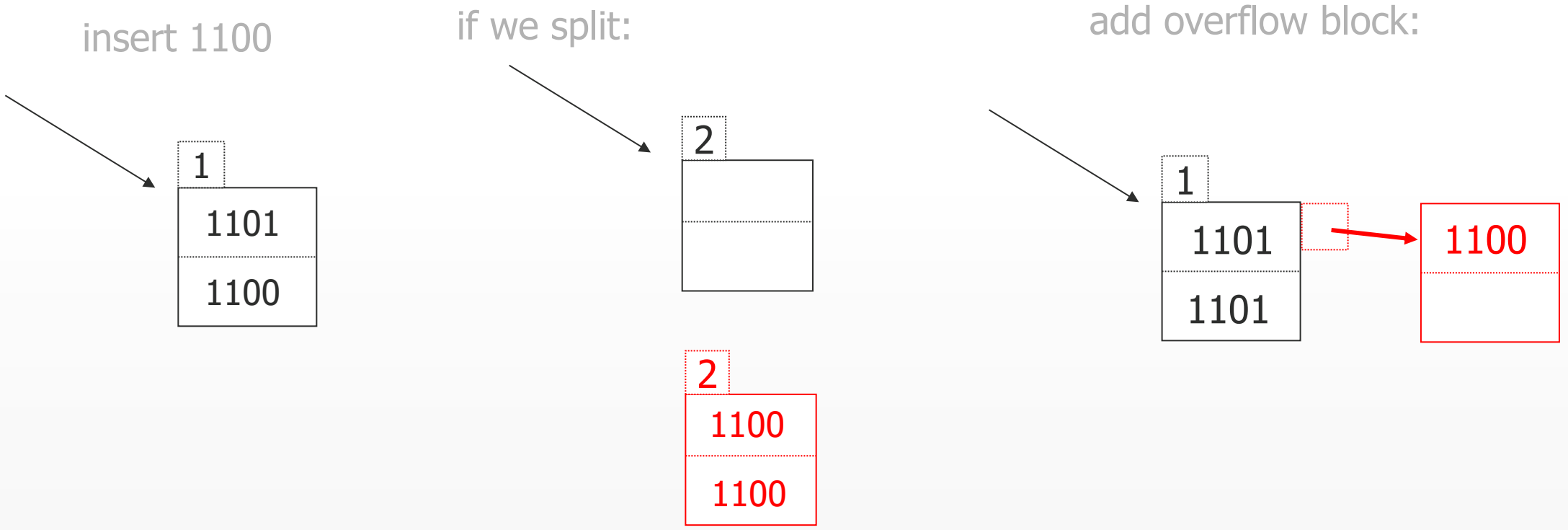


# Proširivo heširanje



# Proširivo heširanje

- U slučaju duplikata vrednosti ključa koristi se ulančavanje.



# B+ stablo vs. Heš indeksi

## Heš indeksi i direktna organizacija

Poređenje

# Poređenje

- Ključni faktor poređenja jeste vrsta upita.
  - Upiti jednakosti ključa pretraživanja sa nekom konstanom i upiti po oblasti vrednosti ključa indeksiranja i heširanja.

```
SELECT …
FROM R
WHERE R.A = 5
```

Broj pristupa pri indeksiranju je srazmeran logaritmu broja vrednosti za A, dok je pri heširanju nezavisan od veličine same relacije.

```
SELECT
FROM R
WHERE R.A > 5
```

Ovakav upit bi zahtevao heširanje funkcijom koja zadržava redosled ključeva u rasutoj datoteci. Indeksiranje je u ovom slučaju bolji izbor.

# Poređenje (2)

- Očekivani tipovi upita:
  - Heš je generalno bolji za pretragu za datu vrednost ključa
  - Za pretragu po opsegu (range queries), neophodni su uređeni indeksi
- U praksi:
  - PostgreSQL podržava heš indekse, ali obeshrabruje njihovu upotrebu (zbog loših performansi)
  - Oracle podržava datoteke sa heš organizacijom, ali ne i heš indekse
  - SQL Server podržava samo B+-stabla