

Sveučilište u Zagrebu  
Prirodoslovno Matematički Fakultet  
- Matematički odjel

Robert Manger  
**BAZE PODATAKA**  
Dodaci uz skripta

Prvo izdanje  
Zagreb, lipanj 2008.

# Sadržaj

<b>A</b>	<b>PRIMJERI VEZANI UZ RELACIJSKU ALGEBRU I RELACIJSKI RAČUN</b>	<b>3</b>
A.1	Rad s relacijskom algebrom . . . . .	3
A.1.1	Upiti o gurmanima . . . . .	3
A.1.2	Upiti o knjižnici . . . . .	4
A.1.3	Dokazivanje jednakosti algebarskih izraza . . . . .	6
A.2	Korištenje relacijskog računa . . . . .	6
A.2.1	Upiti o gurmanima, račun orijentiran na n-torke . . . . .	7
A.2.2	Upiti o gurmanima, račun orijentiran na domene . . . . .	7
<b>B</b>	<b>PRIMJERI VEZANI UZ FIZIČKU GRAĐU BAZE PODATAKA</b>	<b>9</b>
B.1	Korištenje raznih organizacija datoteki . . . . .	9
B.1.1	Jednostavna organizacija . . . . .	9
B.1.2	Datoteka u obliku hash tablice . . . . .	9
B.1.3	Indeks sekvencijalna datoteka . . . . .	12
B.1.4	Vezane liste zapisa . . . . .	12
B.1.5	Invertirana datoteka . . . . .	12
B.1.6	Organizacija s podijeljenom hash funkcijom . . . . .	15
B.1.7	Računanje veličine datoteke . . . . .	15
B.2	Rad s B-stablama . . . . .	19
B.2.1	Prikaz gustog indeksa pomoću B-stabla . . . . .	19
B.2.2	Ubacivanje u B-stablo . . . . .	19
B.2.3	Izbacivanje iz B-stabla . . . . .	20
B.2.4	Još jedno izbacivanje iz B-stabla . . . . .	20
B.2.5	Određivanje optimalnog reda i visine B-stabla . . . . .	22



## Dodatak A

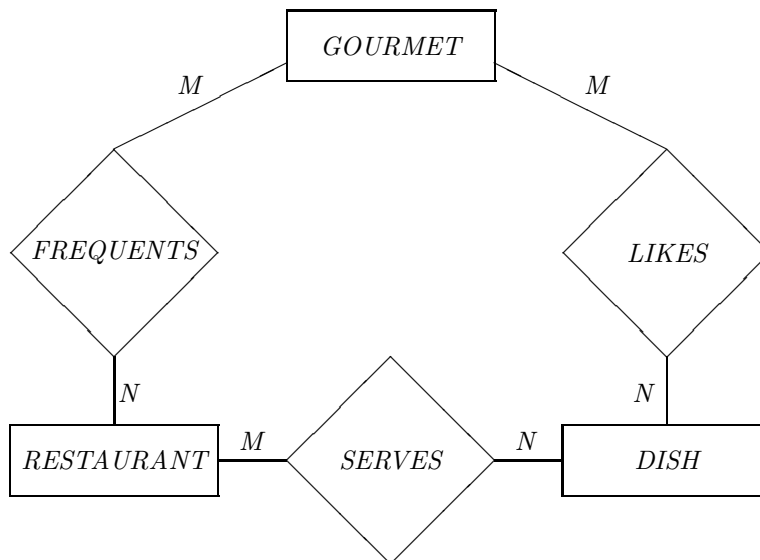
# PRIMJERI VEZANI UZ RELACIJSKU ALGEBRU I RELACIJSKI RAČUN

### A.1 Rad s relacijskom algebrom

Relacijska algebra je svojevrsni jezik za postavljanje upita u relacijskoj bazi podataka. Sastoji se od relacijskih operacija koje na osnovu jedne ili više zadanih relacija računaju novu relaciju. Uobičajene operacije su **union** (skupovna unija), **intersect** (skupovni presjek), **minus** (skupovna razlika), **where** (selekcija),  $[ ]$  (projekcija), **times** (Kartezijev produkt), **join** (prirodni spoj), **divideby** (dijeljenje). Upit se zapisuje kao algebarski izraz, a odgovor na upit dobiva se izvrednjavanjem tog izraza.

#### A.1.1 Upiti o gurmanima

Promatramo bazu podataka o gurmanima, jelima i restoranima čija ER-shema je prikazana na slici A.1. Svaki gurman posjećuje određene restorane i voli određena jela, a svaki restoran poslužuje neka od tih jela.



Slika A.1: ER-shema baze podataka o gurmanima

Zbog jednostavnosti, pretpostavljamo da se u toj bazi ne pohranjuju nikakvi podaci o entitetima osim njihovih imena. Uz tu pretpostavku, relacijska shema koja odgovara slici A.1 izgleda ovako:

$FREQUENTS ( \underline{GOURMET}, \underline{RESTAURANT} )$  ,  
 $SERVES ( \underline{RESTAURANT}, \underline{DISH} )$  ,  
 $LIKES ( \underline{GOURMET}, \underline{DISH} )$  .

*Zadatak.* Za promatranu bazu podataka o gurmanima sljedeće upite napišite u relacijskoj algebri.

Upit 1: Ispišite restorane koji poslužuju neko jelo koje voli gurman Joe.

Upit 2: Ispišite gurmane koji posjećuju bar jedan restoran koji poslužuje neko jelo koje oni vole.

Upit 3: Ispišite sve restorane koji poslužuju sva jela koje voli Joe.

*Rješenje.* Ispisujemo algebarske izraze koji daju odgovor na svaki od postavljenih upita.

Upit 1:

$JOES\_DISHES := (LIKES \textbf{ where } GOURMET = \text{'Joe'}) [DISH]$  ;  
 $JOES\_RESTS := (JOES\_DISHES \textbf{ join } SERVES) [RESTAURANT]$  .

Upit 2:

$RESULT := ( (LIKES \textbf{ join } SERVES ) \textbf{ join } FREQUENTS ) [GOURMET]$  .

Upit 3:

Najprije pod pretpostavkom da nemamo operaciju **divideby**.  
 $JOES\_DISHES := (LIKES \textbf{ where } GOURMET = \text{'Joe'}) [DISH]$  ;  
 $ALL\_RESTS := SERVES [RESTAURANT]$  ;  
 $ALL\_COMB := JOES\_DISHES \textbf{ times } ALL\_RESTS$  ;  
 $NOT\_JOES := (ALL\_COMB \textbf{ minus } SERVES) [RESTAURANT]$  ;  
 $JOES\_RESTS := ALL\_RESTS \textbf{ minus } NOT\_JOES$  .

Zatim pod pretpostavkom da imamo operaciju **divideby**.

$JOES\_DISHES := (LIKES \textbf{ where } GOURMET = \text{'Joe'}) [DISH]$  ;  
 $JOES\_RESTS := SERVES \textbf{ divideby } JOES\_DISHES$  .

### A.1.2 Upiti o knjižnici

Promatramo bazu podataka o knjižnici. Relacije govore o knjigama, članovima i o posudbama knjiga članovima. Relacijska shema izgleda ovako:

$BOOK ( \underline{CATNO}, \underline{TITLE}, \underline{AUTHOR}, \underline{PUBLISHER} )$  ,  
 $MEMBER ( \underline{MEMBERNO}, \underline{NAME}, \underline{ADDRESS} )$  ,  
 $LOAN ( \underline{CATNO}, \underline{MEMBERNO}, \underline{DATE\_LOANED} )$  .

Atribut *CATNO* je kataloški broj koji jednoznačno određuje svaku knjigu (to jest djelo i izdanje a ne primjerak), a *MEMBERNO* je članski broj koji jednoznačno određuje člana knjižnice.

*Zadatak.* Za promatranu bazu podataka o knjižnici sljedeće upite napišite u relacijskoj algebri.

Upit 1: Pronađite naslove i autore svih knjiga koje je izdao izdavač 'Prentice-Hall'.

Upit 2: Pronađite naslove svih knjiga koje su bile posuđene 15. veljače 2008.

Upit 3: Ispišite ime člana kojem je knjiga s naslovom 'For whom the bell tolls' bila posuđena 12. ožujka 2008.

Upit 4: Ispišite naslov i autora bilo koje knjige izdane od 'Prentice-Hall' koja je bila posuđena članu s imenom 'John Smith' prije 21. srpnja 2007.

Upit 5: Pronađite imena i adrese članova kojima su bile posuđene sve knjige napisane od autora 'E. Hemingway'-a.

Upit 6: Ispišite naslove i autore za sve knjige koje nikad nisu bile posuđene.

Upit 7: Ispišite sve podatke o knjigama koje nikad nisu bile posuđene članu s brojem 34216.

Upit 8: Ispišite imena članova koji su posudili sve one knjige koje je posudio član s brojem 67542.

*Rješenje.* Ispisujemo algebarske izraze koji daju odgovor na svaki od postavljenih upita.

Upit 1:

$RESULT := (BOOK \textbf{ where } PUBLISHER = \text{'Prentice-Hall'}) [TITLE, AUTHOR] .$

Upit 2:

$TEMP := (LOAN \textbf{ where } DATE\_LOANED = \text{'15-FEB-2008'});$   
 $RESULT := (BOOK \textbf{ join } TEMP) [TITLE] .$

Upit 3:

$TEMP := (BOOK \textbf{ where } TITLE = \text{'To whom the bell tolls'}) \textbf{ join}$   
 $(LOAN \textbf{ where } DATE\_LOANED = \text{'12-MAR-2008'});$   
 $RESULT := (TEMP \textbf{ join } MEMBER) [NAME] .$

Upit 4:

$TEMP1 := (MEMBER \textbf{ where } NAME = \text{'John Smith'}) [MEMBERNO] ;$   
 $TEMP2 := ( (TEMP1 \textbf{ join } LOAN) \textbf{ where } DATE\_LOANED < \text{'21-JUN-2007'}) [CATNO] ;$   
 $RESULT := (TEMP2 \textbf{ join } BOOK) [TITLE, AUTHOR] .$

Upit 5:

Najprije pod pretpostavkom da raspoložemo operatorom **divideby**:  
 $TEMP1 := (BOOK \textbf{ where } AUTHOR = \text{'E.Hemingway'}) [CATNO] ;$   
 $TEMP2 := LOAN[CATNO, MEMBERNO] \textbf{ divideby } TEMP1 ;$   
 $RESULT := (TEMP2 \textbf{ join } MEMBER) [NAME, ADDRESS] .$

Sada pod pretpostavkom da ne raspoložemo operatorom **divideby**:  
 $TEMP1 := (BOOK \textbf{ where } AUTHOR = \text{'E.Hemingway'}) [CATNO] ;$   
 $ALL\_COMB := TEMP1 \textbf{ times } MEMBER[MEMBERNO] ;$   
 $TEMP2 := (ALL\_COMB \textbf{ minus } LOAN[CATNO, MEMBERNO]) [MEMBERNO] ;$   
 $TEMP3 := MEMBER[MEMBERNO] \textbf{ minus } TEMP2 ;$   
 $RESULT := (TEMP3 \textbf{ join } MEMBER) [NAME, ADDRESS] .$

Ovdje  $TEMP2$  sadrži članske brojeve onih članova koji nisu posudili bar jednu Hemingwayevu knjigu.

Upit 6:

$TEMP := BOOK[CATNO] \textbf{ minus } LOAN[CATNO] ;$   
 $RESULT := (TEMP \textbf{ join } BOOK) [TITLE, AUTHOR] .$

Upit 7:

$TEMP1 := (LOAN \textbf{ where } MEMBERNO = 34216) [CATNO] ;$   
 $TEMP2 := BOOK[CATNO] \textbf{ minus } TEMP1 ;$   
 $RESULT := TEMP2 \textbf{ join } BOOK .$

Ovdje  $TEMP1$  sadrži katalogske brojeve knjiga koje su bile posuđene članu s brojem 34216.

Upit 8:

Najprije s korištenjem operatora **divideby**:  
 $TEMP1 := (LOAN \textbf{ where } MEMBERNO = 67542) [CATNO] ;$   
 $TEMP2 := LOAN[CATNO, MEMBERNO] \textbf{ divideby } TEMP1 ;$   
 $RESULT := (TEMP2 \textbf{ join } MEMBER) [NAME] .$

Sada bez korištenja operatora **divideby**:

```
TEMP1 := (LOAN where MEMBERNO = 67542) [CATNO];
ALL_MEMBERS := MEMBER[MEMBERNO];
ALL_COMB := TEMP1 times ALL_MEMBERS;
TEMP2 := (ALL_COMB minus LOAN[CATNO, MEMBERNO]) [MEMBERNO];
TEMP3 := ALL_MEMBERS minus TEMP2;
RESULT := (TEMP3 join MEMBER) [NAME].
```

Ovdje *TEMP1* sadrži kataloške brojeve knjiga koje je posudio član 67542, a *TEMP2* sadrži članove koji nisu posudili sve knjige iz *TEMP1*.

### A.1.3 Dokazivanje jednakosti algebarskih izraza

Često se dešava da se jedan te isti upit može zapisati pomoću raznih algebarskih izraza. Da bi dokazali da su dva izraza u relacijskoj algebri ustvari jednaki, služimo se činjenicom da oba izraza zapravo određuju skupove  $n$ -torki određenog tipa. Općenito, dva skupa su jednaka ako je bilo koji element prvog skupa ujedno element drugog skupa, te ako je bilo koji element drugog skupa ujedno element prvog skupa. Dakle, dokaz da su dva izraza u relacijskoj algebri jednaka provodi se ovako.

- Promatra se bilo koja  $n$ -torka koja pripada vrijednosti prvog izraza, te se pokaže da ta  $n$ -torka mora pripadati vrijednosti drugog izraza.
- Zatim se promatra bilo koja  $n$ -torka koja pripada vrijednosti drugog izraza, te se pokaže da ta  $n$ -torka mora pripadati vrijednosti prvog izraza.

*Zadatak.* Promatramo tri relacije:  $R(A,B)$ ,  $S(B,C)$ ,  $T(C,D)$ . Dokažite da vrijedi jednakost:

$$(R \text{ join } S) \text{ join } T = R \text{ join } (S \text{ join } T).$$

*Rješenje.* Prvo dokazujemo da je

$$(R \text{ join } S) \text{ join } T \subseteq R \text{ join } (S \text{ join } T).$$

Neka je  $(a, b, c, d)$  bilo koja  $n$ -torka iz skupa  $(R \text{ join } S) \text{ join } T$  - pokazat ćemo da je ona također iz skupa  $R \text{ join } (S \text{ join } T)$ . Zaista, u skladu s definicijom prirodnog spoja,  $(a, b, c, d)$  je nastala spajanjem manjih  $n$ -torki, pa slijedi da postoje odgovarajuće manje  $n$ -torke  $(a, b, c) \in R \text{ join } S$  i  $(c, d) \in T$ . Slično,  $(a, b, c)$  je nastala spajanjem još manjih  $n$ -torki, pa postoje i  $n$ -torke  $(a, b) \in R$  i  $(b, c) \in S$ . Tri male  $n$ -torke mogu se spojiti i na drugi način: najprije  $(b, c)$  i  $(c, d)$  daju  $(b, c, d) \in S \text{ join } T$ , a zatim  $(a, b)$  i  $(b, c, d)$  daju  $(a, b, c, d) \in R \text{ join } (S \text{ join } T)$ . Dakle opet smo dobili polaznu  $n$ -torku, no sada smo utvrdili da je ona element od  $R \text{ join } (S \text{ join } T)$ , što je i trebalo pokazati.

Dalje bi trebali dokazati obratnu inkluziju, dakle da je

$$(R \text{ join } S) \text{ join } T \supseteq R \text{ join } (S \text{ join } T).$$

Dokaz je posve analogan prethodnome, ustvari svodi se na čitanje prethodnog dokaza u obrnutom smjeru.

## A.2 Korištenje relacijskog računa

Relacijski račun je također jezik za postavljanje upita u relacijskoj bazi podataka. Zasniva se na notaciji iz matematičke logike, preciznije na predikatnom računu. Postoji verzija orijentirana na  $n$ -torke, gdje varijable predstavljaju cijele  $n$ -torke, te verzija orijentirana na domene, gdje varijable predstavljaju vrijednosti pojedinih atributa. Upit se izražava tako da se zapiše predikat kojeg  $n$ -torke odnosno atributi moraju zadovoljavati. U odnosu na relacijsku algebru, relacijski račun je u većoj mjeri "neproceduralan", naime on se svodi na definiranje rezultata kojeg želimo dobiti, bez ikakve naznake postupka dobivanja tog rezultata.

### A.2.1 Upiti o gurmanima, račun orijentiran na n-torke

*Zadatak.* Ponovo promatramo bazu podataka o gurmanima iz odjeljka A.1.1. Sljedeće upite, koje smo prije zapisali u relacijskoj algebri, sada zapišite u relacijskom računu orijentiranom na n-torke.

Upit 1: Ispišite restorane koji poslužuju neko jelo koje voli gurman Joe.

Upit 2: Ispišite gurmane koji posjećuju bar jedan restoran koji poslužuje neko jelo koje oni vole.

Upit 3: Ispišite sve restorane koji poslužuju sva jela koje voli Joe.

*Rješenje.* Sastavljamo izraze u računu orijentiranom na n-torke koji daju odgovor na svaki od postavljenih upita.

Upit 1:

$$\{s.RESTAURANT \mid SERVES(s) \text{ and } \exists l (LIKES(l) \text{ and } l.DISH = s.DISH \text{ and } l.GOURMET = \text{'Joe'})\} .$$

Upit 2:

$$\{f.GOURMET \mid FREQUENTS(f) \text{ and } \exists s (SERVES(s) \text{ and } s.RESTAURANT = f.RESTAURANT \text{ and } \exists l (LIKES(l) \text{ and } l.DISH = s.DISH \text{ and } l.GOURMET = f.GOURMET))\} .$$

Upit 3:

$$\{s.RESTAURANT \mid SERVES(s) \text{ and } \forall l (LIKES(l) \text{ and } l.GOURMET = \text{'Joe'} \text{ and } l.DISH = s.DISH)\} .$$

### A.2.2 Upiti o gurmanima, račun orijentiran na domene

*Zadatak.* I dalje promatramo bazu podataka o gurmanima iz odjeljka A.1.1. Iste upite sada još zapišite i u relacijskom računu orijentiranom na domene.

Upit 1: Ispišite restorane koji poslužuju neko jelo koje voli gurman Joe.

Upit 2: Ispišite gurmane koji posjećuju bar jedan restoran koji poslužuje neko jelo koje oni vole.

Upit 3: Ispišite sve restorane koji poslužuju sva jela koje voli Joe.

*Rješenje.* Sastavljamo izraze u računu orijentiranom na domene koji daju odgovor na svaki od postavljenih upita.

Upit 1:

$$\{R \mid \exists D (SERVES(RESTAURANT : R, DISH : D) \text{ and } LIKES(GOURMET : \text{'Joe'}, DISH : D))\} .$$

Upit 2:

$$\{G \mid \exists R (FREQUENTS(GOURMET : G, RESTAURANT : R) \text{ and } \exists D (SERVES(RESTAURANT : R, DISH : D) \text{ and } LIKES(GOURMET : G, DISH : D)))\} .$$

Upit 3:

$$\{R \mid \forall D (\text{if } LIKES(GOURMET : \text{'Joe'}, DISH : D) \text{ then } SERVES(RESTAURANT : R, DISH : D))\} .$$



## Dodatak B

# PRIMJERI VEZANI UZ FIZIČKU GRADU BAZE PODATAKA

### B.1 Korištenje raznih organizacija datoteki

Svaka datoteka može se organizirati na razne načine. Da bi mogli procijeniti koja organizacija je najpogodnija, moramo znati kako se datoteka misli koristiti, to jest koje se operacije trebaju efikasno obavljati. U idućem nizu primjera pretpostavljamo da se blok vanjske memorije sastoji od 512 byte, a adresa bloka zauzima 4 byte.

#### B.1.1 Jednostavna organizacija

*Zadatak.* Datoteka s najboljim rezultatima neke računalne igre sastoji se od zapisa sljedećeg oblika:

BODOVI	PREZIME I IME	DATUM
--------	---------------	-------

Duljina zapisa je 40 byte. Datoteka nikad ne sadrži više od 100 zapisa. Najčešća operacija je ispis rang liste najboljih rezultata (prvih  $n$  mjesta). Predložite pogodnu organizaciju.

*Rješenje.* Budući da je datoteka mala i čita se sekvencijalno, najpogodnija je *jednostavna organizacija*. Zapisi su silazno sortirani po podatku *BODOVI*. U jedan blok stane najviše 12 zapisa plus pointer na idući blok plus byte-ovi za puno/prazno. Novi zapis se mora ubaciti na “pravo mjesto” u sortiranom redoslijedu - zato je dobro da u blokovima ostavimo slobodnih mjesta. Izgled datoteke vidi se na slici B.1.

#### B.1.2 Datoteka u obliku hash tablice

*Zadatak.* Datoteka automobila u nekom gradu sastoji se od zapisa oblika:

<u>REGISTARSKA</u> <u>OZNAKA</u>	<i>TIP</i> <i>AUTOMOBILA</i>	<i>GODINA PROIZVODNJE</i> <i>AUTOMOBILA</i>	<i>PREZIME I</i> <i>IME VLASNIKA</i>	...
-------------------------------------	---------------------------------	--	---	-----

Duljina zapisa je 150 byte. Očekujemo da će datoteka sadržavati nekoliko tisuća zapisa. Najčešća operacija je: traženje podataka o automobilu sa zadanim registarskim brojem. Predložite pogodnu organizaciju.

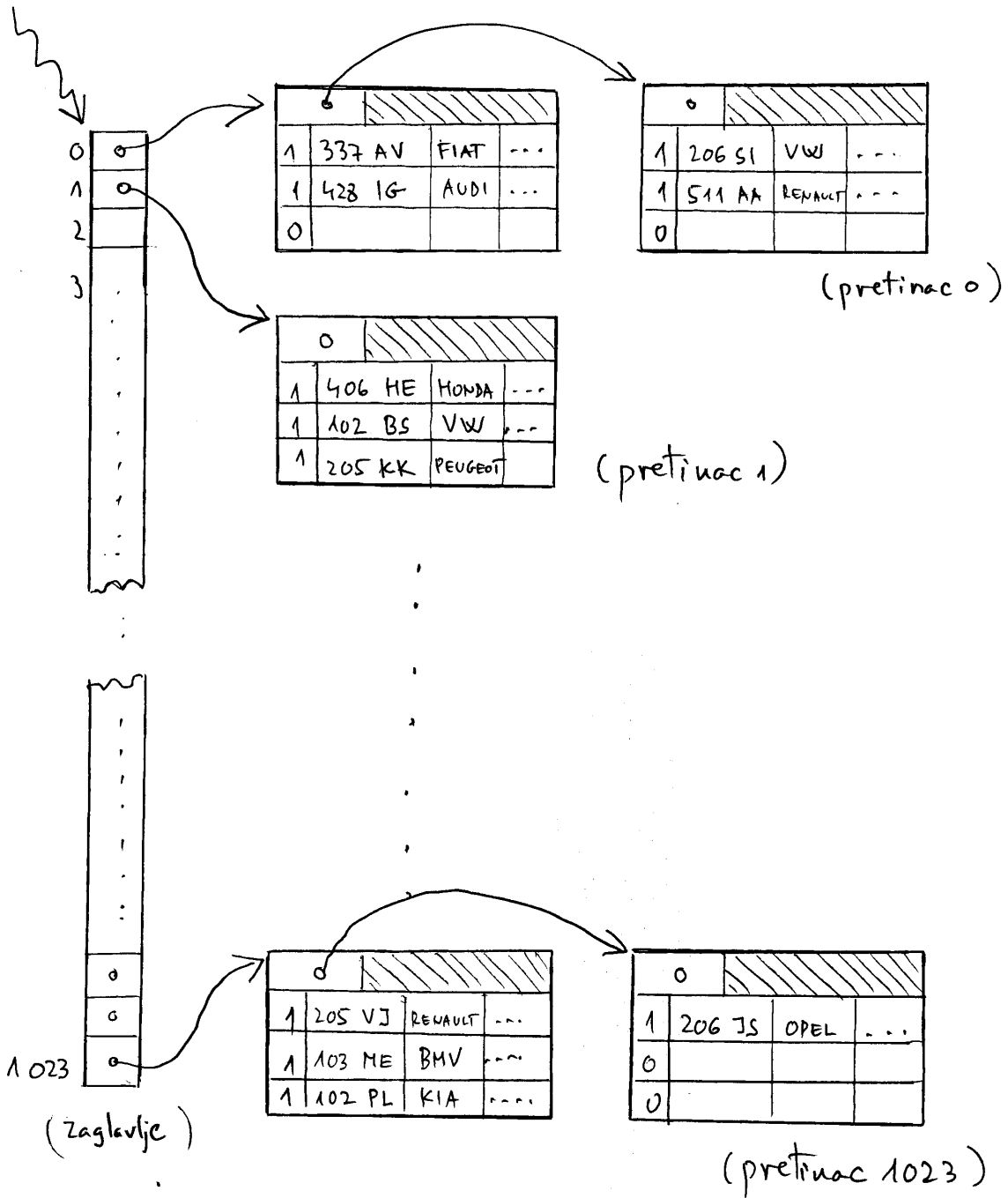
*Rješenje.* Budući da je potreban brzi pristup na osnovu primarnog ključa, a nije potreban sortirani redoslijed u odnosu na ključ, najbolja je *organizacija u obliku hash tablice*. U jedan blok stane 3 zapisa (plus pointer na idući blok plus byte-ovi za puno/prazno). Pretinac u prosjeku smije imati svega nekoliko blokova. Stoga broj pretinaca  $P$  mora biti oko 1000, na primjer  $P = 1024$ . Da bi smjestilo 1024 adresa blokova, samo zaglavlje mora zauzimati 8 (po mogućnosti uzastopnih) blokova. Hash funkciju treba odabrati onako kako je bilo opisano na predavanju (dijeljenje ključa na segmente, zbrajanje tih segmenata modulo  $P$ ). Izgled datoteke otprilike se vidi na slici B.2. Zbog lakšeg crtanja, podaci na slici B.2 nasumično su raspoređeni po pretincima, što se može tumačiti kao primjena neke nepoznate (ali sigurno postojeće) hash funkcije.

1	3912	MATIĆ IVO	12-12-07
1	7346	PERIĆ FRANJO	01-03-08
1	7220	SAVIĆ JOSIP	03-04-07
1	7220	MATIĆ IVO	05-07-07
1	6165	MATIĆ IVO	01-02-08
1	6064	KATIĆ VESNA	27-05-07
1	6060	PERIĆ FRANJO	08-08-07
0			
0			
0			
0			
0			

1	5392	PERIĆ FRANJO	03-05-06
1	5287	KATIĆ VESNA	05-07-07
1	5110	MATIĆ IVO	05-07-07
1	5028	KIKIĆ SAŠA	12-12-06
1	4993	KATIĆ VESNA	13-05-07
1	4876	HORVAT JASNA	25-02-08
1	4560	BABIĆ MIRA	26-03-08
1	4350	KIKIĆ SAŠA	11-10-07
0			
0			
0			
0			

Slika B.1: Jednostavno organizirana datoteka s rang listom rezultata neke računalne igre.



Slika B.2: Datoteka u obliku hash tablice s podacima o automobilima.

### B.1.3 Indeks sekvencijalna datoteka

*Zadatak.* Datoteka sadrži rječnik stranih riječi i sastoji se od zapisa oblika:

<u>STRANA RIJEČ</u>	<u>PRIJEVOD</u>
---------------------	-----------------

Strana riječ zauzima 15 znakova, a prijevod 35 znakova; znači duljina zapisa je 50 byte. Rječnik sadrži oko 10000 riječi. Najčešća operacija je ispis prijevoda za zadanu stranu riječ. Pritom se dozvoljava zadavanje početka strane riječi (prvih nekoliko slova) - tada treba ispisati sve riječi koje počinju na zadani način i njihove prijevode. Predložite pogodnu organizaciju.

*Rješenje.* Treži se direktan pristup na osnovu primarnog ključa, te (nakon toga) sekvencijalno čitanje idućih zapisa u smislu sortiranog redoslijeda po ključu. U ovakvim slučajevima najbolja je *indeks sekvencijalna organizacija*. Zapisi su uzlasno sortirani po ključu u smislu leksikografskog uređaja. U jedan blok stane najviše 10 zapisa plus byte-ovi za puno/prazno. Znači, osnovna datoteka će zauzimati oko 1000 blokova. Razrijeđeni indeks sadrži prvu stranu riječ i adresu za svaki blok osnovne datoteke. Jedan par (riječ, adresa) zauzima 19 byte. Možemo računati da u jedan blok stane oko 25 parova (riječ, adresa), naime potreban je i dodatni prostor za organizaciju samog indeksa. Znači, indeks će zauzeti oko 40 blokova. Na slici B.3 nacrtano je da je indeks sam za sebe jednostavno organiziran. No, budući da je on dosta glomazan (oko 40 blokova), isplati ga se pretvoriti u B-stablo.

### B.1.4 Vezane liste zapisa

*Zadatak.* Datoteka sadrži meteorološke podatke koji se odnose na jedan grad. Zapis je oblika:

<u>DATUM</u>	<u>PADALINE</u> ( $l/m^2$ )	<u>TEMPERATURA</u> U 13 h ( $^{\circ}C$ )	<u>TLAK U</u> 13 h (mbar)	...
--------------	--------------------------------	--	------------------------------	-----

Duljina zapisa je 90 byte. Očekujemo da će datoteka sadržavati nekoliko tisuća zapisa. Osim traženja podataka za zadani dan ili zadani mjesec, potrebno je brzo naći dane za koje je količina padalina ( $l/m^2$ ) bila u zadanom intervalu. Predložite pogodnu organizaciju.

*Rješenje.* Da bi omogućili traženje po primarnom ključu (zadani dan, zadani mjesec) služimo se *indeks-sekvencijalnom organizacijom* (kao u prethodnom primjeru). Da bi olakšali traženje po padalinama, sve zapise osnovne datoteke s istom vrijednošću za padaline povežemo u *vezanu listu*. Uvodimo i *indeks listi*. Pointer na zapis stane u 5 byte (adresa bloka plus redni broj unutar bloka). Zato se duljina zapisa povećava na 95 byte. U jedan blok stane 5 zapisa. Nije potrebno ostavljati prazna mjesta jer se zapisi ubacuju redom sa sve većim datumima. Razrijeđeni indeks bi kao i u prošlom primjeru mogao biti glomazan, te bi ga trebalo prikazati kao B-stablo. Indeks listi sadrži parove oblika (količina padalina, pointer na početak odgovarajuće vezane liste). Jedan par zahtijeva na primjer 8 byte (3 byte za troznamenkasti broj, 5 byte za adresu). Možemo računati da u jedan blok stane 60 takvih parova. Budući da količina padalina neće biti veća od  $200 l/m^2$ , cijeli indeks zauzima  $200/60 \approx 4$  bloka, pa može biti jednostavno organiziran. Izgled organizacije datoteke vidi se na slici B.4.

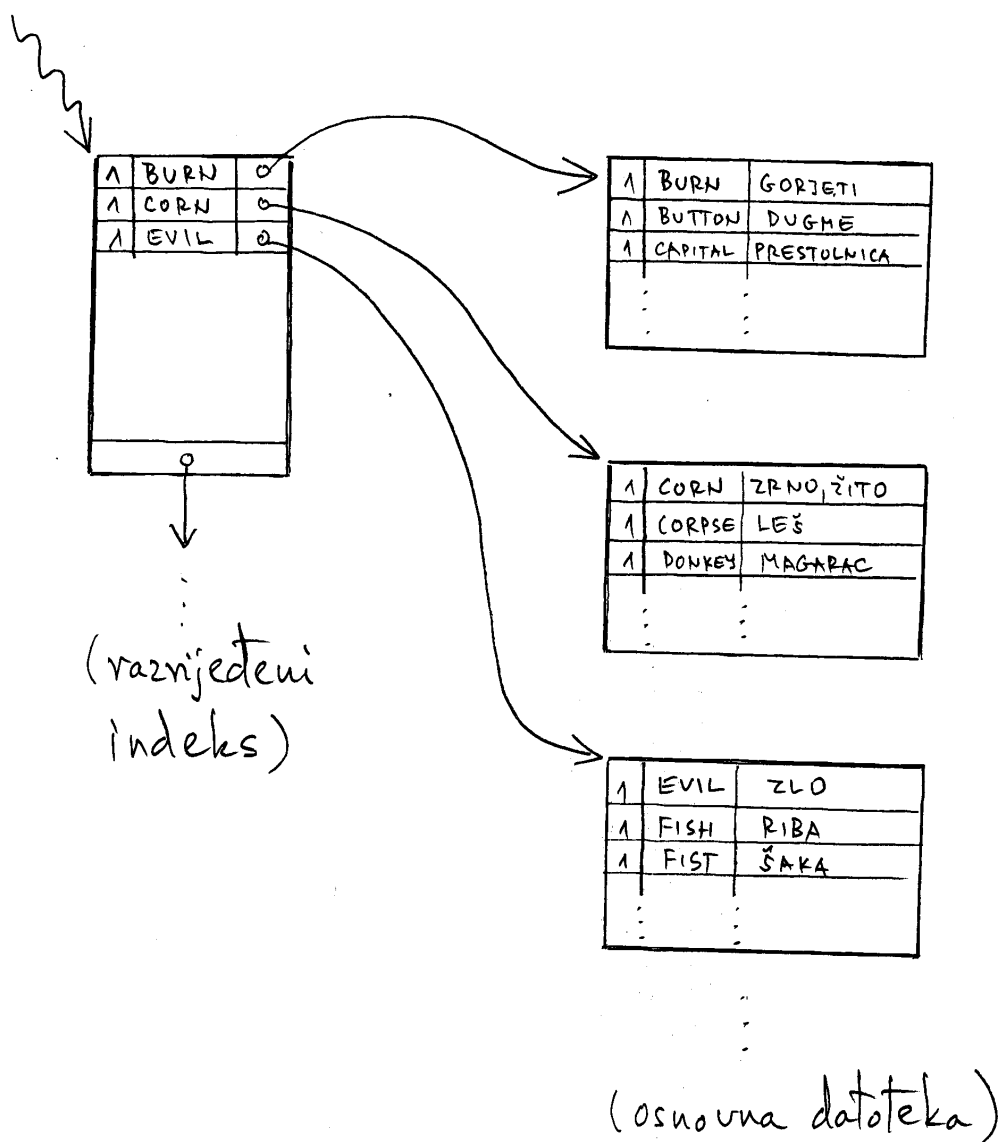
### B.1.5 Invertirana datoteka

*Zadatak.* Datoteka sadrži tehničke karakteristike vijaka koji se koriste u nekoj tvornici. Zapis je oblika:

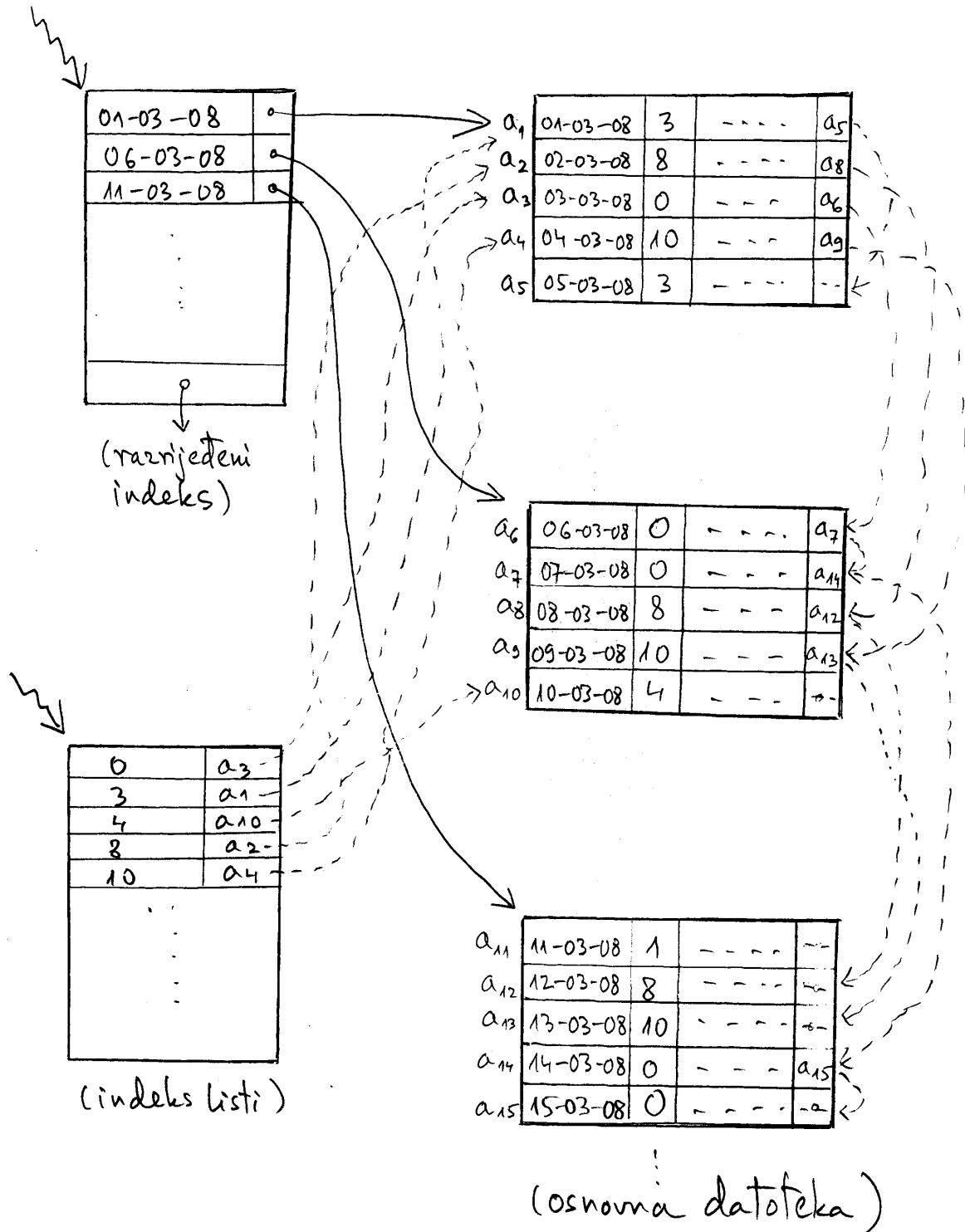
<u>IDENT</u>	<u>PROMJER</u>	<u>DULJINA</u>	<u>PROMJER</u>	...
<u>BROJ</u>	<u>VIJKA</u> (mm)	<u>NAVOJA</u> (mm)	<u>GLAVE</u> (mm)	

Duljina zapisa je oko 120 byte. Očekujemo da će datoteka sadržavati više od tisuću zapisa. Osim traženja po identifikacijskom broju, javlja se potreba za odgovaranjem na sljedeće upite.

- Koji vijci imaju promjer u zadanom intervalu?
- Koji su vijci s duljinom navoja u zadanom intervalu?
- Koji vijci imaju promjer glave u zadanom intervalu?
- Razne Boole-ovske kombinacije prethodnih upita.



Slika B.3: Rječnik stranih riječi organiziran kao indeks sekvencijalna datoteka.



Slika B.4: Meteorološki podaci - indeks sekvencijalna datoteka s dodatnim vezanim listama.

Predložite pogodnu organizaciju datoteke.

*Rješenje.* Odgovaranje na pojedine upite mogli bi ubrzati uvođenjem vezanih listi, kao u prethodnom primjeru. To bi onda bile višestruke vezane liste. No budući da se traže i Boole-ovske kombinacije jednostavnijih upita, najpogodnija je *invertirana organizacija*. Uvodimo gusti primarni indeks, te sekundarne indekse za podatke *PROMJER VIJKA*, *DULJINA NAVOJA* i *PROMJER GLAVE*. Zapis osnovne datoteke složeni su u blokove u proizvoljnom redosljedju. U jedan blok stane 4 zapisa (plus byte-ovi za puno/prazno). Svi indeksi su glomazni pa se prikazuju kao B-stabla. Izgled cijele organizacije vidi se na slici B.5.

### B.1.6 Organizacija s podijeljenom hash funkcijom

*Zadatak.* Datoteka sadrži podatke o zaposlenima u nekom poduzeću. Zapis je oblika:

<u>MATIČNI</u> <u>BROJ</u>	<u>PREZIME</u> <u>I IME</u>	<u>ZANIMANJE</u>	<u>ODJEL</u>	<u>...</u>
-------------------------------	--------------------------------	------------------	--------------	------------

Zbog fluktuacije kadra česta su ubacivanja i izbacivanja zapisa. Duljina zapisa je 200 byte. Ukupan broj zaposlenih je nekoliko stotina. Osim traženja na osnovu matičnog broja, traže se i zaposleni sa zadanim zanimanjem ili iz zadanog odjela. Često se traže i zaposleni koji istovremeno zadovoljavaju dva ovakva uvjeta, dakle oni sa zadanim zanimanjem i iz zadanog odjela. Predložite pogodnu organizaciju datoteke.

*Rješenje.* Predložimo *hash datoteku s podijeljenom hash funkcijom*. U jedan blok stanu samo dva zapisa. S obzirom da pretinac ne smije imati suviše mnogo blokova, potrebno je da broj pretinaca  $P$  bude oko 100, na primjer  $128 = 2^7$ . Cijelo zaglavlje tada zauzima  $128 \cdot 4 = 512$  byte, pa stane u jedan blok. Hash adresa se sastoji od 7 bitova. Hash funkciju zadajemo na primjer ovako.

- Gornja tri bita u hash adresi ovise o *MATIČNOM BROJU*, na primjer:  
 $h_1(\text{MATIČNI BROJ}) = (\text{zbroj znamenki u MATIČNOM BROJU}) \% 8$ .
- Iduća dva bita u hash adresi ovise o *ZANIMANJU*, na primjer:  
 $h_2(\text{ZANIMANJE}) = (\text{broj znakova u ZANIMANJU koji su različiti od bjeline}) \% 4$ .
- Donja dva bita u hash adresi ovise o *ODJELU*, na primjer:  
 $h_3(\text{ODJEL}) = (\text{redni broj ODJELA}) \% 4$ .

Da bi pronašli zapis sa zadanim matičnim brojem, treba pretražiti 16 pretinaca. Da bi pronašli sve zaposlene sa zadanim zanimanjem, treba pretražiti 32 pretinca. Slično je kad tražimo zaposlene iz zadanog odjela. No da bi pronašli zaposlene koji su zadanog zanimanja i rade u zadanom odjelu, dovoljno je pretražiti 8 pretinaca. Datoteka je prikazana na slici B.6.

### B.1.7 Računanje veličine datoteke

U dosadašnjim primjerima uzimali smo da se blok vanjske memorije sastoji od 512 byte, te da adresa bloka zauzima 4 byte. Na raznim računalima te vrijednosti se mogu razlikovati, što može bitno utjecati na svojstva pojedinih organizacija datoteki. To ćemo ilustrirati sljedećim primjerom.

*Zadatak.* Neka je  $l_a$  duljina adrese bloka,  $l_b$  duljina bloka (sve u byte-ovima). Promatramo varijantu jednostavne organizacije datoteke prikazanu na slici B.7. Dakle, datoteka se sastoji od niza blokova čije adrese su popisane u posebnoj (jednom) bloku - zaglavlju. Odredite maksimalnu veličinu ovakve datoteke ako je:

- $l_a = 4, l_b = 512$ ,
- $l_a = 4, l_b = 1024$ ,
- $l_a = 3, l_b = 2048$ ,
- $l_a = 2, l_b = 4096$ .

110238	$a_3$
172634	$a_9$
202474	$a_7$
227164	$a_{13}$
258320	$a_{14}$
281376	$a_1$
331202	$a_{12}$
337274	$a_4$
376551	$a_6$
462108	$a_5$
472164	$a_2$
515043	$a_{11}$
528117	$a_8$
700105	$a_{10}$

(gusti primarni indeks)

$a_1$	1	281376	10	50	12	...
$a_2$	1	472164	12	40	16	...
$a_3$	1	110238	16	60	20	...
$a_4$	1	337274	10	50	14	...

$a_5$	1	462108	8	40	12	...
$a_6$	1	376551	10	50	16	...
$a_7$	1	202474	11	60	14	...
$a_8$	1	528117	15	70	20	...

$a_9$	1	172634	16	70	20	...
$a_{10}$	1	700105	10	50	16	...
$a_{11}$	1	515043	8	40	14	...
$a_{12}$	1	331202	8	40	12	...

$a_{13}$	1	227164	11	50	14	...
$a_{14}$	1	258320	12	60	20	...
0					...	
0					...	

(sekund. indeks za promjer vijka)



8	$a_5, a_{11}, a_{12}$
10	$a_1, a_4, a_6, a_{10}$
11	$a_7, a_{13}$
12	$a_2, a_{14}$
15	$a_8$
16	$a_3, a_9$

(sekund. indeks za duljinu navoja)



40	$a_2, a_5, a_{11}, a_{12}$
50	$a_1, a_4, a_6, a_{10}, a_{13}$
60	$a_3, a_7, a_{14}$
70	$a_8, a_9$

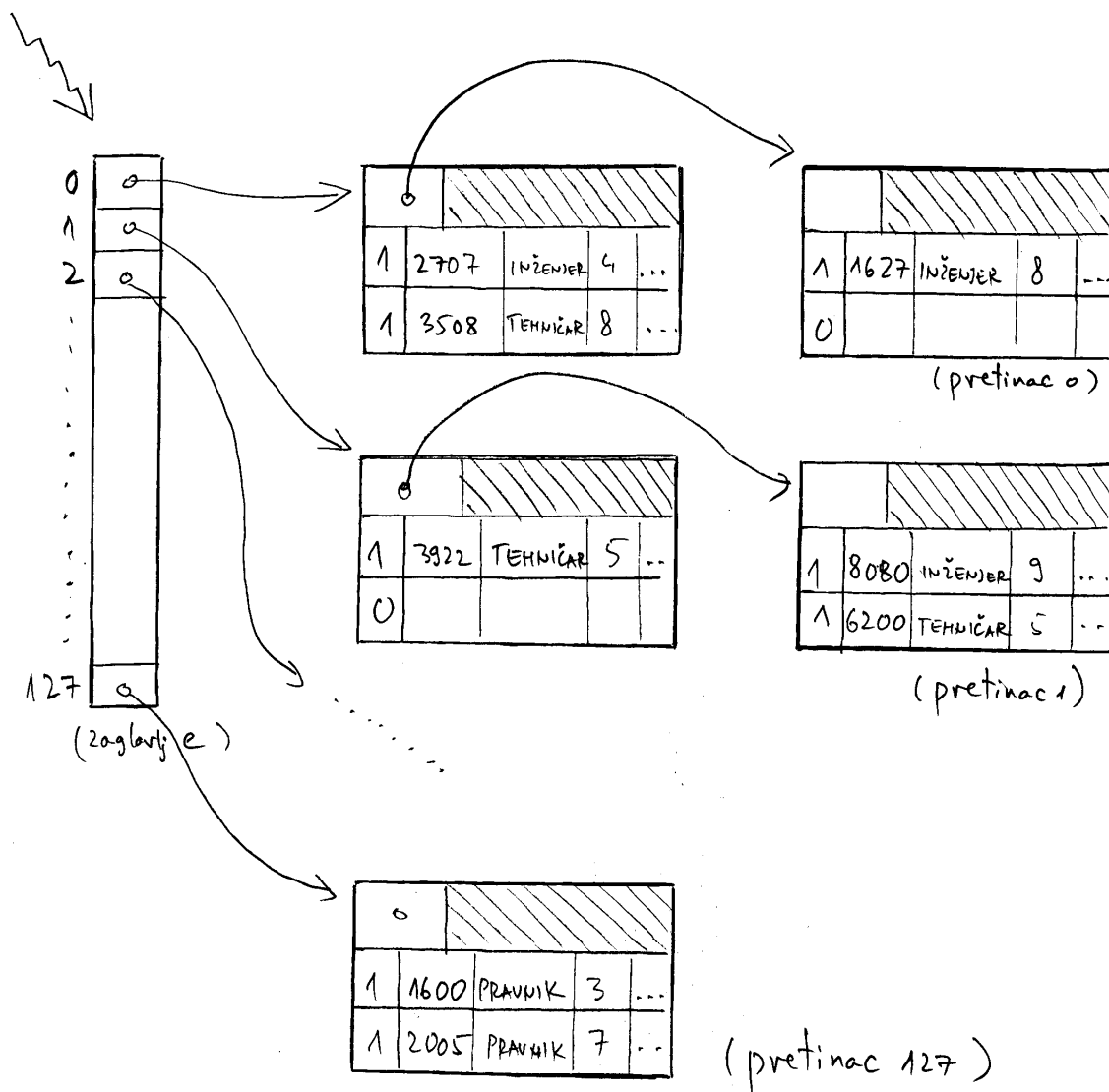
(sekund. indeks za promjer glave)



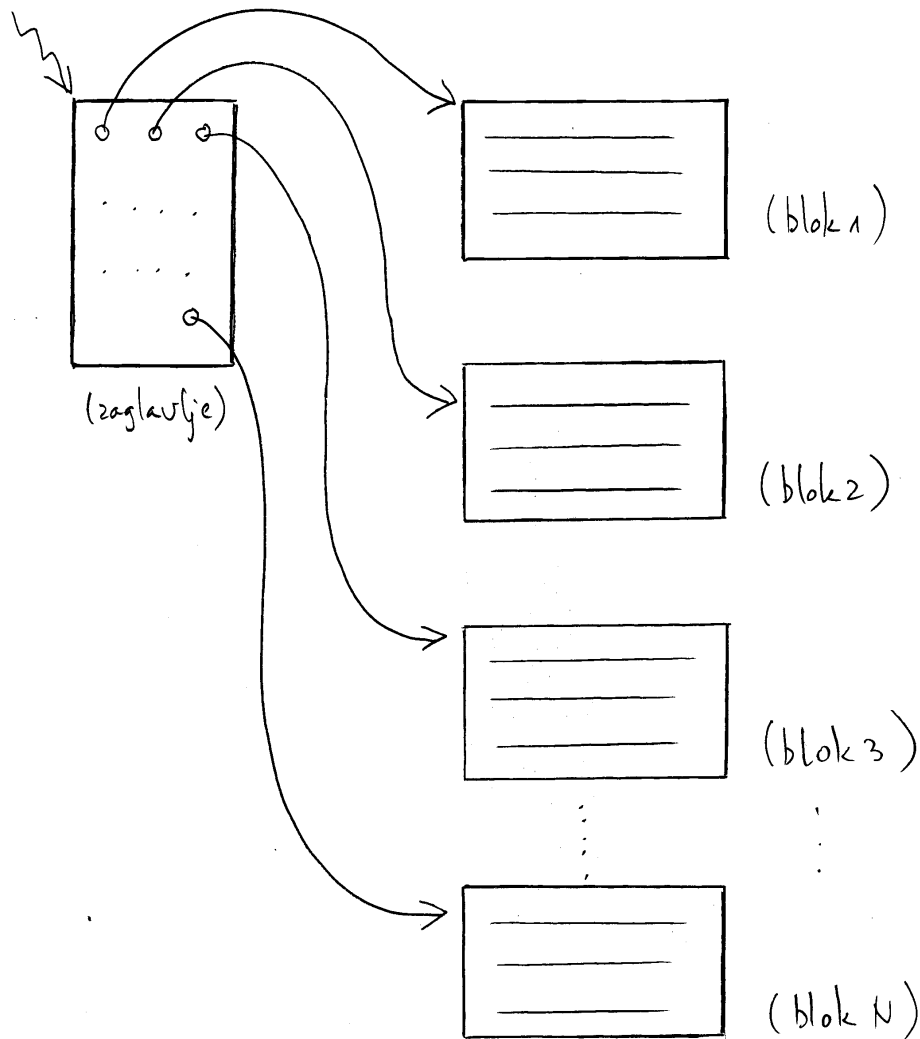
12	$a_1, a_5, a_{12}$
14	$a_4, a_7, a_{11}, a_{13}$
16	$a_2, a_6, a_{10}$
20	$a_3, a_8, a_9, a_{14}$

(osnova datoteka)

Slika B.5: Invertirana organizacija za datoteku s tehničkim karakteristikama vijaka.



Slika B.6: Podaci o zaposlenima u poduzeću - hash datoteka s podijeljenom hash funkcijom.



Slika B.7: Jednostavna organizacija datoteke - varijanta sa zaglavljem.

*Rješenje.* Računamo maksimalnu veličinu posebno za svaki slučaj.

- i) U zaglavlje stane  $512/4 = 128$  adresa, pa je maksimalna veličina datoteke  $128 \text{ blokova} = 128 \cdot 512 \text{ byte} = 64 \text{ Kbyte}$ .
- ii) U zaglavlje stane  $1024/4 = 256$  adresa, pa je maksimalna veličina datoteke  $256 \text{ blokova} = 256 \cdot 1024 \text{ byte} = 256 \text{ Kbyte}$ .
- iii) U zaglavlje stane  $2048/3 = 682$  adresa, pa je maksimalna veličina datoteke  $682 \text{ bloka} = 682 \cdot 2048 \text{ byte} = 1364 \text{ Kbyte} \approx 1.33 \text{ Mbyte}$ .
- iv) U zaglavlje stane  $4096/2 = 2048$  adresa, pa je maksimalna veličina datoteke  $2048 \text{ blokova} = 2048 \cdot 4096 \text{ byte} = 8 \text{ Mbyte}$ .

Ustvari, vrijedi ovakva općenita formula: (maksimalna veličina datoteke) =  $\lfloor l_b/l_a \rfloor \cdot l_b \approx l_b^2/l_a$ .

## B.2 Rad s B-stablima

Indeksi bilo koje vrste obično se fizički prikazuju kao B-stabla. Znamo da je B-stablo reda  $m$   $m$ -narno stablo sa sljedećim svojstvima:

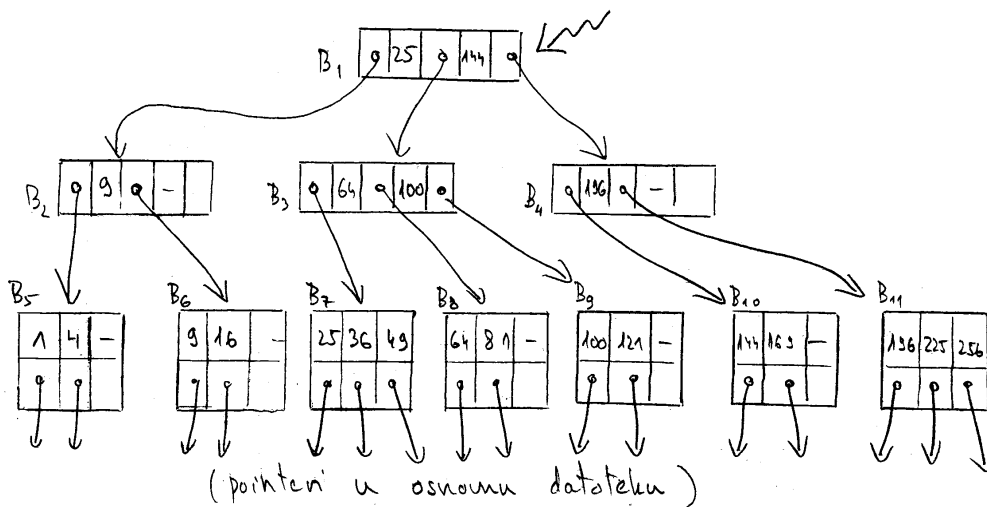
- korijen je ili list ili ima bar dvoje djece;
- svaki čvor izuzev korijena i listova ima između  $\lceil m/2 \rceil$  i  $m$  djece;
- svi putovi od korijena do lista imaju istu duljinu.

Čvorovi B-stabla zapravo su blokovi vanjske memorije. Cijelo B-stablo može se promatrati kao hijerarhija (jednostavno organiziranih) indeksa. Listovi čine “najniži” indeks s pointerima u osnovnu datoteku. Unutrašnji čvorovi neposredno iznad listova čine razrijeđeni indeks najnižeg indeksa. Unutrašnji čvorovi na idućem nivou čine indeks od indeksa od indeksa, ..., i tako dalje, ..., korijen predstavlja “najviši” indeks. Cijela hijerarhija može se smatrati i jednim (hijerarhijski građenim) indeksom.

### B.2.1 Prikaz gustog indeksa pomoću B-stabla

*Zadatak.* Zapisi osnovne datoteke sadrže sljedeće vrijednosti primarnog ključa: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256. Prikažite gusti indeks za tu datoteku pomoću B-stabla reda 3.

*Rješenje.* Jedno od mogućih B-stabala vidi se na slici B.8. Naglasimo da to rješenje nipošto nije jedinstveno. Pretpostavili smo da u jedan blok-list stanu 3 vrijednosti ključa s pripadnim pointerima.



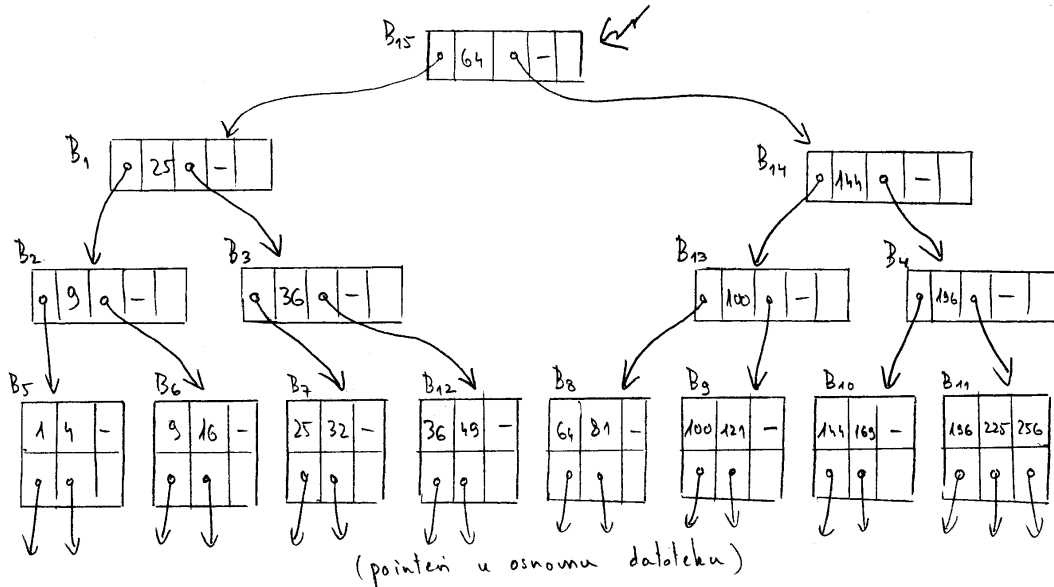
Slika B.8: Jedan od mogućih prikaza zadanog indeksa pomoću B-stabla.

### B.2.2 Ubacivanje u B-stablo

*Zadatak.* U osnovnu datoteku iz prošlog primjera ubačen je novi zapis s vrijednošću primarnog ključa 32. Uskladite indeks B-stablo sa slike B.8 s tom promjenom.

*Rješenje.* Slijedimo put od korijena B-stabla prema dolje. Stižemo u list koji bi morao sadržavati vrijednost ključa 32 - to je blok  $B_7$ . Vrijednost 32 ne možemo ubaciti u  $B_7$  jer je taj blok već pun. Zato uključujemo novi blok  $B_{12}$ . Vrijednosti ključa i pointerne rasporedimo između  $B_7$  i  $B_{12}$ , i to tako da 25 i 32 budu u  $B_7$ , a 36 i 42 u  $B_{12}$ .

Dalje moramo umetnuti 36 i pointer prema  $B_{12}$  u blok  $B_3$ . To uzrokuje da se  $B_3$  prepuni, pa zato uključujemo još jedan novi blok:  $B_{13}$ . Pointeri prema  $B_7$  i  $B_{12}$  (zajedno s graničnom vrijednošću ključa) idu u  $B_3$ , dok pointeri prema  $B_8$  i  $B_9$  (s pripadnim vrijednostima ključa) idu u  $B_{13}$ . Dalje pokušavamo ubaciti 64 i pointer na  $B_{13}$  u  $B_1$ . Sada se  $B_1$  prepunio, pa uključujemo novi blok  $B_{14}$ . Pointere (i vrijednosti ključa) iz prepunjenog  $B_1$  raspodijelimo između  $B_1$  i  $B_{14}$ . Također, uvodimo novi korijen  $B_{15}$  s pointerima na  $B_1$  i  $B_{14}$  (i vrijednošću ključa 64). Rezultirajuće B-stablo vidi se na slici B.9.



Slika B.9: Ubacivanje nove vrijednosti 32 u B-stablo s prethodne slike.

### B.2.3 Izbacivanje iz B-stabla

*Zadatak.* Iz osnovne datoteke iz prošlog primjera dalje se izbacuje zapis s vrijednošću ključa 64. U skladu s time ažurirajte indeks B-stablo sa slike B.9.

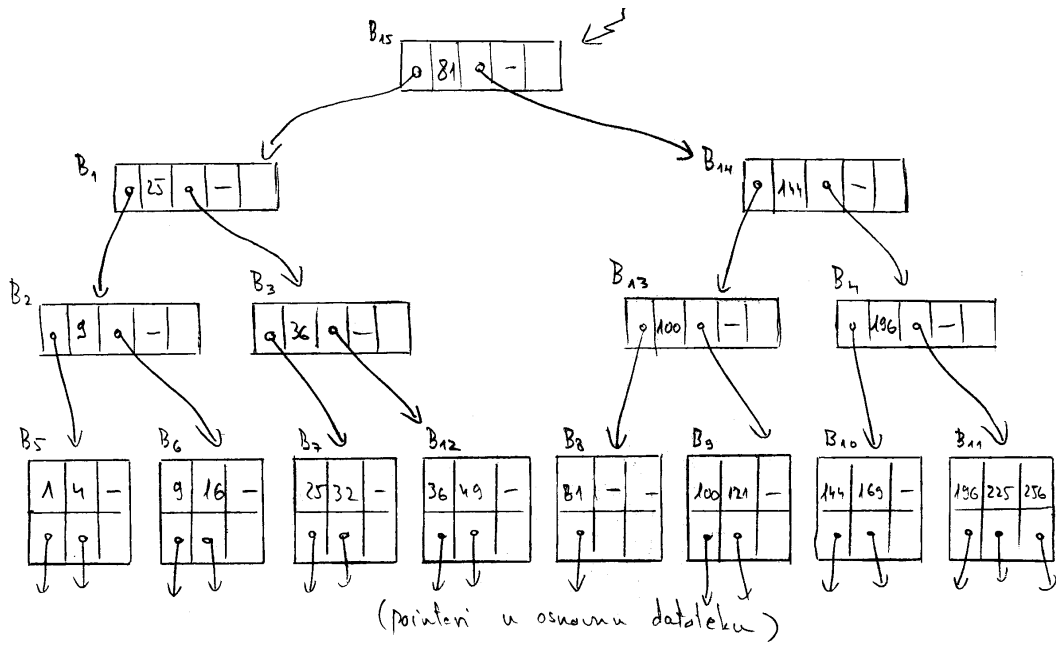
*Rješenje.* Slijedimo put od korijena stabla prema dolje i pronalazimo vrijednost ključa 64 u bloku  $B_8$ . Izbacujemo 64 i pripadni pointer iz  $B_8$ . Time se promijenila najmanja vrijednost ključa u  $B_8$  - ona više nije 64 nego 81. To zahtijeva da se korigira jedan od predaka bloka  $B_8$ . Slijedimo put od  $B_8$  prema korijenu i tražimo vrijednost 64. Ona se pojavljuje tek u korijenu  $B_{15}$  - mijenjamo je u 81. Dobiveno B-stablo vidi se na slici B.10.

### B.2.4 Još jedno izbacivanje iz B-stabla

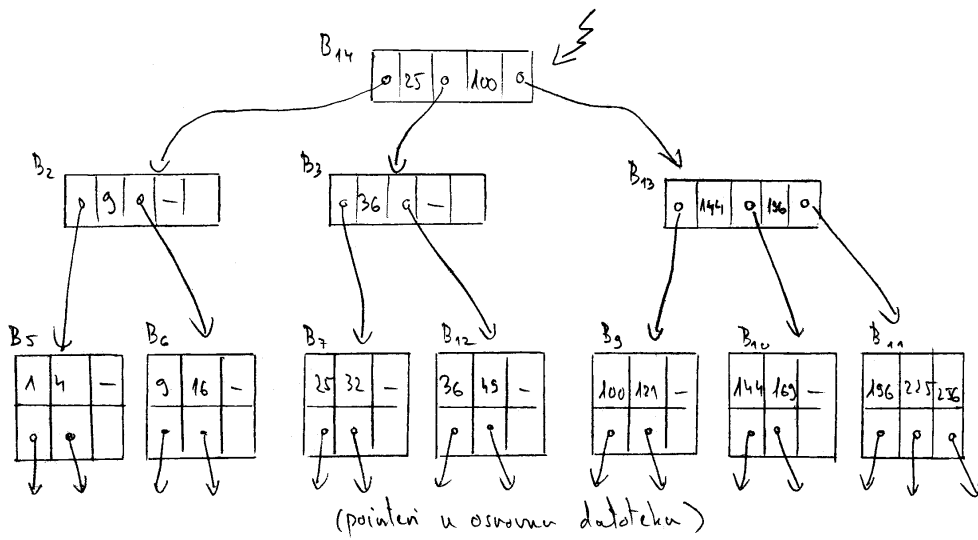
*Zadatak.* Iz osnovne datoteke iz prošlog primjera u nastavku se izbacuje i zapis s vrijednošću ključa 81. Uskladite indeks B-stablo sa slike B.10 s tom promjenom.

*Rješenje.* Opet slijedimo put od korijena stabla prema dolje i pronalazimo vrijednost ključa 81 u bloku  $B_8$ . Izbacujemo 81 i pripadni pointer iz  $B_8$  - time  $B_8$  postaje prazan te se isključuje iz stabla.

U skladu s nestankom  $B_8$  korigiramo njegovog roditelja  $B_{13}$ , tako da izbacimo pointer na  $B_8$ . No sad  $B_{13}$  ima samo jedno dijete, što je nedopustivo za stabla reda 3. Potrebno je ujediniti  $B_{13}$  i njegovog brata  $B_4$ . Točnije rečeno, premještamo pointere iz  $B_4$  u  $B_{13}$ , a  $B_4$  isključimo iz stabla. U skladu s nestankom  $B_4$  korigiramo zajedničkog roditelja od  $B_{13}$  i  $B_4$ , dakle  $B_{14}$ . Sad bi  $B_{14}$  ostao samo s jednim djetetom, što nije dozvoljeno. Zato  $B_{14}$  ujedinjavamo s njegovim bratom  $B_1$ , to jest pointere iz  $B_1$  premještamo u  $B_{14}$ , a  $B_1$  isključujemo iz stabla. Blok  $B_{15}$  tada više nije potreban pa i njega isključujemo, tako da  $B_{14}$  postaje novi korijen. U svim promijenjenim blokovima po potrebi ažuriramo granične vrijednosti ključa. Rezultirajuće B-stablo vidi se na slici B.11.



Slika B.10: Izbacivanje vrijednosti 64 iz B-stabla s prethodne slike.



Slika B.11: Daljnje izbacivanje vrijednosti 81 iz B-stabla s prethodne slike.

### B.2.5 Određivanje optimalnog reda i visine B-stabla

Rad s B-stablom je to brži što je red  $m$  stabla veći. Zato za prikazivanje indeksa treba odabrati B-stablo što većeg reda. s druge strane,  $m$  ne može biti po volji velik jer je ograničen odnosom između veličine bloka i broja byte-ova potrebnih za prikaz adrese ili vrijednosti ključa. To nam ilustrira sljedeći primjer.

*Zadatak.* Blok vanjske memorije sastoji se od 512 byte, a adresa bloka zauzima 4 byte. Osnovna datoteka sastoji se od milijun zapisa duljine 100 byte. Vrijednost primarnog ključa zauzima 8 byte. Želimo sagraditi B-stablo koje će služiti kao gusti indeks za opisanu datoteku. Odaberite optimalni red stabla  $m$ . Također, procijenite visinu stabla.

*Rješenje.* Unutrašnji čvor B-stabla je jedan blok u kojem piše najviše  $m$  adresa blokova i najviše  $(m - 1)$  vrijednosti ključa. Zbog zadanih veličina vrijedi:

$$4m + 8(m - 1) \leq 512, \text{ dakle } 12m \leq 520, \text{ odnosno } m \leq 520/12$$

Znači, optimalni  $m$  je  $m = 43$ .

U jedan blok osnovne datoteke stane najviše 5 zapisa. Zato možemo uzeti da adresa zapisa (adresa bloka plus redni broj zapisa u bloku) zauzima 5 byte. Jedan par (vrijednost ključa, pripadna adresa zapisa u osnovnoj datoteci) zauzima dakle  $8 + 5 = 13$  byte. Zato u jedan list B-stabla stane najviše  $512/13 = 39$  takvih parova.

Kod procjene visine B-stabla uzet ćemo da su listovi polu-puni, to jest da u prosljeku sadrže 20 parova (vrijednost ključa, adresa zapisa). Budući da osnovna datoteka ima 1000000 zapisa, slijedi da stablo ima oko 50000 listova. Visina stabla kreće se između  $\approx \log_{43} 50000$  i  $\approx \log_{22} 50000$ , ovisno o tome da li je prosječni broj djece unutrašnjeg čvora bliži  $m$  ili  $\lceil m/2 \rceil$ . Dakle, procjenjujemo da je visina stabla između 2.876683 i 3.500366, dakle između 3 i 4. Znači, da bi pronašli zapis sa zadanom vrijednošću ključa, moramo pročitati 4 do 5 blokova u indeksu, plus još jedan blok u osnovnoj datoteci.