SERIJSKA DATOTEKA

Vežbe

Sadržaj

Karakteristike serijske datoteke

Karakteristike serijske datoteke

- Tip organizacije datoteka u kojoj su slogovi smešteni u sukcesivne memorijske lokacije, najčešće prema hronološkom redosledu njihovog nastanka
- Fizička struktura datoteke ne poseduje informaciju o logičkim vezama između slogova
- Ne postoji veza između vrednosti ključa sloga i adrese lokacije u koju je smešten

Slog

- Osnovna jedinica obrade podataka koja predstavlja jedan zapis u datoteci
- Slog se sastoji od polja koja imaju svoj naziv i tip
- Svaki slog ima polje ključ koje služi kao jedinstveni identifikator
- Ključ sloga se koristi za pretragu, ažuriranje i organizaciju datoteke

Blok

- Osnovna jedinica prenosa podataka
- Grupa slogova koji se kombinuju zajedno radi povećanja efikasnosti
- Rezultat grupisanja slogova je smanjen broj operacija čitanja/pisanja
- Faktor blokiranja datoteke predstavlja broj slogova u bloku

Primer fizičke strukture serijske datoteke

- Fizička strukture jedne serijske datoteke od 8 slogova sa faktorom blokiranja f = 3
- Slogovi su prikazani kao dvojke (k(Si), p(Si)), gde je k(Si) vrednost ključa, a p(Si) predstavlja konkretizacije ostalih obeležja sloga
- Slogovi su smešteni u fizičke blokove sa adresama A1, A2 i A3
  - Blok sa adresom A3 je samo delimično popunjen
  - Oznaka kraja datoteke je ***,** te je slog sa tom vrednošću ključa poslednji popunjen slog

A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$
	6		**11**		**4**	
A_2		$p(S_4)$		$p(S_5)$		$p(S_6)$
	55		**35**		**2**	
A_3		$p(S_7)$		$p(S_8)$		
	16		*****			

Pojašnjenje sintakse pseudokoda

Struktura procesa

- Struktura procesa

```
POČETAK PROCESA  ime_procesa
        Naredbe pseudokoda
KRAJ PROCESA  ime_procesa
```

- Poziv procesa

```
POZOVI ime_procesa(spisak_stvarnih parametara)
```

Pojašnjenje sintakse pseudokoda

```
PROCES ime_procesa(U(ulazni_parametri), I(izlazni_parametri), UI(ulazno-izlazni_parametri))
```

- Parametri procesa
  - **U** - ulazni parametar
    - Nadređeni proces prosleđuje vrednost podređenom procesu
    - Za podređeni proces se stvaraju privatne kopije ulaznih parametara
  - **I** - Izlazni parametar
    - Parametri koje pozvani proces vraća nadređenom procesu
    - Njihova vrednost u nadređenom procesu nema uticaja na vrednost u podređenom procesu
    - Na početku izvršavanja podređenog procesa vrednost im je nedefinisana.
  - **UI** - Ulazno-izlazni parametri
    - Ulazni parametri koje podređeni proces modifikuje i vraća nadređenom procesu
    - Na početku izvršavanja podređenog procesa vrednost im je predefinisana.

Pojašnjenje sintakse pseudokoda

- Parametar *status* predstavlja status ulazno-izlaznih (U/I) operacija
  - 0 - poslednja U/I operacija se uspešno završila,
  - 1 - došlo se do kraja datoteke i
  - > 1 - došlo je do neke greške u fizičkoj realizaciji razmene podataka (neuspešno čitanje ili pisanje, neuspešno otvaranje ili zatvaranje datoteke i slično)

- Veličina bloka i oznaka kraja datoteke su u nastavku pseudokoda označene su kao: **VELIČINA_BLOKA** i **OZNAKA_KRAJA_DATOTEKE**

Zadaci

Zadatak 1 - sistem za praćenje događaja

- Napisati C program koji će omogućiti rad sa podacima o događajima koji su se dogodili u okviru informacionog sistema. Događaji se evidentiraju u okviru blokirane serijske datoteke sa faktorom blokiranja f=3. Za svaki novi događaj beleži se:
  - Evidencioni broj događaja (jedinstven broj do 12 cifara),
  - Vreme događaja (datum i vreme u formatu DD/MM/GGGG_HH:mm:SS),
  - Tip događaja (informativni, upozorenje, greška – označeno kao INFO, WARNING, ERROR),
  - Korisnički ID (ako se događaj odnosi na određenog korisnika; do 10 karaktera ili SYSTEM za sistemske događaje) i
  - Kratak naziv događaja (tekstualni naziv događaja, do 20 karaktera; razmaci zamenjeni donjom crtom).

Zadatak 1 - sistem za praćenje događaja

- Omogućiti:
  a. odabir datoteke,
  b. formiranje datoteke,
  c. unos novog sloga,
  d. ispis svih slogova,
  e. traženje u okviru datoteke,
  f. modifikacija sloga (promena tipa događaja i/ili naziva događaja),
  g. fizičko brisanje sloga i
  h. logičko brisanje.

Zadatak 2

- Napisati program koji će omogućiti rad sa podacima o evidentiranim parkiranjima na privatnom parkingu. Svi podaci moraju biti smeštani u binarnu *serijski organizovanu datoteku* sa faktorom blokiranja ***f*=4**. Za svako parkiranje beleži se:
  - id oznaka (do 5 cifara),
  - registarska oznaka vozila (najviše 10 karaktera),
  - datum i vreme parkiranja (niz od 16 karaktera u formatu YYYY-MM-DD HH:MM),
  - oznaka upotrebljenog parking mesta (tačno 3 karaktera) i
  - dužina boravka u minutima (do 1 000 000 minuta).

Zadatak 2

- Omogućiti:
  a. formiranje prazne datoteke,
  b. upis novog sloga u datoteku, uz proveru postojanja vrednosti identifikatora u datoteci, direktnim unosom podataka u realnom vremenu,
  c. traženje sloga u datoteci i njegov prikaz zajedno sa adresom bloka i rednim brojem sloga u bloku,
  d. prikaz svih slogova datoteke zajedno sa adresom bloka i rednim brojem sloga u bloku,
  e. unos novih slogova iz (samostalno pripremljene) tekstualne datoteke *parkiraliste.csv*, u okviru koje je semantika svake od kolona istaknuta u zaglavlju,
  f. fizičko brisanje svih slogova kod kojih je dužina boravka jednaka nuli,
  g. prikaz izveštaja u kom će za svako parking mesto biti prikazano koliko puta je upotrebljeno i koliko je ukupno minuta bilo u upotrebi.

Formiranje blokirane serijske datoteke

Formiranje blokirane serijske datoteke

Koraci formiranja serijske datoteke:

1. Definisanje strukture
2. Kreiranje serijske datoteke
3. Otvaranje serijske datoteke
4. Pokretanje procesa preuzimanja podataka
5. Dok postoje podaci za unos:
    a. Preuzimanje podataka
    b. Validacija podataka
    c. Formiranje sloga
    d. Upis sloga
6. Zatvaranje serijske datoteke

Napomena: dodavanje slogova je objašnjeno u delu Upis novog sloga

Formiranje blokirane serijske datoteke - Definisanje strukture

Definisanje strukture serijske datoteke obuhvata definisanje:

- strukture sloga
  - naziv i tip svih polja sloga
  - ključ sloga
  - polje za logičko brisanje sloga
- faktora blokiranja
- oznake kraja datoteke

Formiranje serijske datoteke - Kreiranje serijske datoteke

```
PROCES formiranje_serijske_datoteke( U(naziv_serijske_datoteke), I(status), UI() )
POČETAK PROCESA formiranje_serijske_datoteke

    OTVORI_DATOTEKU( U(naziv_serijske_datoteke), I(status_otvaranja_datoteke), UI() )

    AKO JE status_otvaranja_datoteke = 0
        INICIJALIZUJ_BLOK(I(prviBlok))
        POSTAVI prviBlok.slogovi[0].ključ <- OZNAKA_KRAJA_DATOTEKE

        UPISI_BLOK( U(naziv_serijske_datoteke, prviBlok, 0 ), I(status_upisa_u_datoteku), UI( ) ),
        ZATVORI_DATOTEKU( U(naziv_serijske_datoteke ), I(status_zatvaranja_datoteke), UI( ))
        POSTAVI status <- 0

    INAČE

        POSTAVI status <- 2

    KRAJ AKO

KRAJ PROCESA formiranje_serijske_datoteke
```

Ispis sadržaja

Ispis sadržaja - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
3. Učitavanje bloka
4. Provera uspešnosti čitanja i provera statusa čitanja bloka:
   a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
   b. Ako je uspešno, nastavljanje procesa.
5. Prolazak kroz sve slogove u trenutnom bloku
   a. Ako je slog validan (nije logički obrisan), prikazati njegov sadržaj.
6. Prelazak na sledeći blok i ponavljanje koraka 3-6
   a. Završetak kada se dostigne kraj datoteke

Ispis sadržaja

fseek(fajl, 0, SEEK_SET);

```
PROCES ispis_sadrzaja(U(serijska_datoteka), I(status))
POČETAK PROCESA ispis_sadrzaja
    AKO JE serijska_datoteka == NULL
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(serijska_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))  (* pripremi prostor za blok podataka *)
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI ispis_blokova DOK JE status_čitanja == 0
            POSTAVI i <- 0
            RADI ispis_slogova DOK JE i < VELIČINA_BLOKA
               AKO JE blok.slogovi[i].ključ != OZNAKA_KRAJA_DATOTEKE I blok.slogovi[i].obrisan != 1
                     ISPISI blok.slogovi[i]
                KRAJ AKO
                POSTAVI i <- i + 1
            KRAJ RADI ispis_slogova
            POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
            ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        KRAJ RADI ispis_blokova
        POSTAVI status <- 0
    KRAJ AKO
KRAJ PROCESA ispis_sadrzaja
```

Traženje sloga

Traženje sloga

- Traženje se u serijskoj datoteci vrši primenom metode linearnog traženja
- Traženje počinje od početka datoteke pristupanjem sukcesivno memorijskim blokovima
- Dva ishoda traženja:
  - *Uspešno* - pronađen slog sa vrednošću ključa jednakom argumentu traženja
  - *Neuspešno* - ako se traženi slog ne pronađe ni u poslednjem bloku datoteke

Traženje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Inicijalizacija brojača blokova kako bi traženje počelo od prvog bloka
3. Učitavanje bloka
4. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
5. Prolazak kroz sve slogove u trenutnom bloku
    a. Ako je slog validan (nije logički obrisan), proveriti da li se njegov ključ poklapa sa traženim ključem.
        i. Ako se ključ poklapa, ispisati podatke o slogu i završiti proces.
        ii. Ako se ključ ne poklapa, nastaviti sa sledećim slogom.
6. Prelazak na sledeći blok i ponavljanje koraka 3-6
    a. Ako se dostigne kraj datoteke (i traženi slog nije pronađen), obavestiti korisnika da slog nije pronađen.

Traženje sloga

```
PROCES nadji_slog(U(serijska_datoteka, ključ_sloga), I(pronadjeni_slog), UI())
POČETAK PROCESA nadji_slog
    AKO JE serijska_datoteka == NULL
        POSTAVI pronadjeni_slog <- NULL
    INAČE
        POMERI_POKAZIVAČ(U(serijska_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok)) (* pripremi prostor za blok podataka *)
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
            POSTAVI i <- 0
            RADI provera_slogova
                NAREDNI SLAJD
            KRAJ RADI provera_slogova
            POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
            ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        KRAJ RADI provera_trenutnog_bloka
        POSTAVI pronadjeni_slog<- NULL
    KRAJ AKO
KRAJ PROCESA nadji_slog
```

Traženje sloga

- Provera slogova:

```
RADI provera_slogova DOK JE i < VELIČINA_BLOKA
    AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE
        POSTAVI pronadjeni_slog <- NULL
        IZLAZ IZ PROCESA
    KRAJ AKO
    AKO JE blok.slogovi[i].ključ == ključ_sloga I blok.slogovi[i].obrisan == 0
        POSTAVI pronadjen_slog <- blok.slogovi[i]
        IZLAZ IZ PROCESA
    KRAJ AKO
    POSTAVI i <- i + 1
KRAJ RADI provera_slogova
```

Upis novog sloga

Upis novog sloga

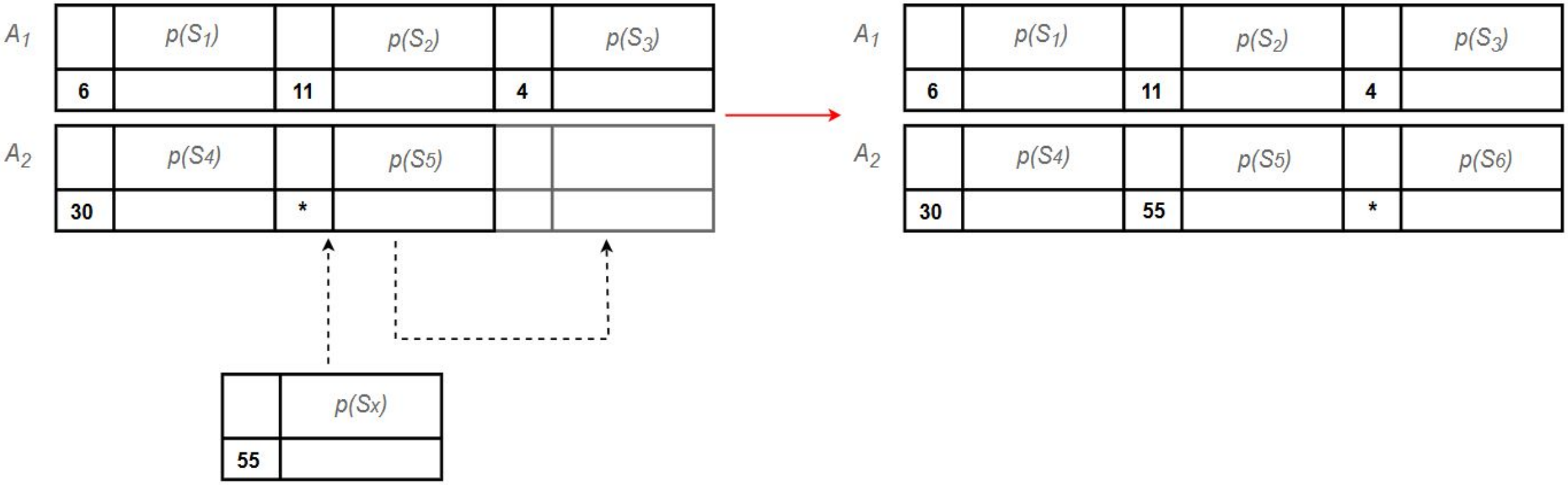
- Novi slogovi se upisuju na kraj datoteke
  - Hronološki prvi slog smešta se u prvu lokaciju memorijskog prostora dodeljenog datoteci
  - Slogovi nastali hronološki kasnije, smeštaju se u sukcesivne lokacije sa većom adresom
- Upisu novog sloga prethodi neuspešno traženje
  - Da bi se upisao novi slog, potrebno je prvo proveriti da li se slog sa datim ključem već nalazi u datoteci
- Novi slog se upisuje u prvu slobodnu lokaciju na kraju datoteke

Upis novog sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Provera da li slog za zadatim ključem već postoji
   a. Prekinuti unos ukoliko postoji
3. Inicijalizacija novog sloga koji će biti upisan
4. Inicijalizacija brojača blokova na poslednji blok i učitavanje poslednjeg bloka
5. Provera da li je poslednji blok pun
   a. Ako je poslednji blok pun (oznaka kraja datoteke u poslednjem slogu), treba dodati novi blok
6. Upis novog sloga na mesto sloga sa oznakom kraja datoteke i pomeranje oznake kraja datoteke:
      i. U sukcesivnu lokaciju ukoliko ima mesta u trenutnom bloku
      ii. Dodati novi blok i oznaku kraja datoteke postaviti u prvi slog novog bloka

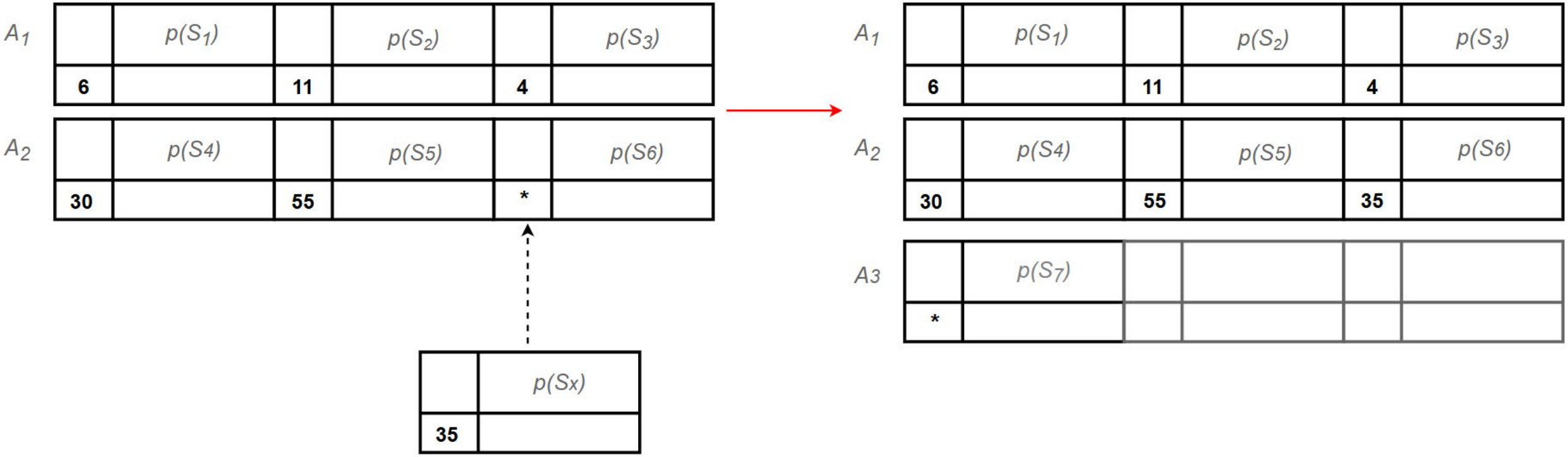
Upis novog sloga - upis u poslednji blok

```
ključ_novog_sloga = 55
```



Upis novog sloga - dodavanje novog bloka

```
ključ_novog_sloga = 35
```



Upis novog sloga

```
PROCES dodaj_slog(U(serijska_datoteka, novi_slog), I(status))
POČETAK PROCESA dodaj_slog
    AKO JE serijska_datoteka == NULL
        POSTAVI status <- 2
    INAČE
        POSTAVI slogStari <- pronadji_slog(U(serijska_datoteka, ključ_novog_sloga))
        AKO JE slogStari != NULL
            POSTAVI status <- 2  (* Greška: slog već postoji *)
        INAČE
            INICIJALIZUJ_BLOK(I(blok)) (* pripremi prostor za blok podataka *)
            POMERI_POKAZIVAČ(U(serijska_datoteka, kraj), -VELIČINA_BLOKA)
            ČITAJ_BLOK(U(serijska_datoteka), UI(blok))
            POSTAVI i <- 0
            RADI pronalazak_pozicije DOK JE i < VELIČINA_BLOKA
                AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE
                    POSTAVI blok.slogovi[i] <- novi_slog
                    POSTAVI i <- i + 1  (* pomeri poziciju za kraj datoteke *)
                    KRAJ pronalazak_pozicije
                KRAJ AKO
                POSTAVI i <- i + 1
            KRAJ pronalazak_pozicije
            NAREDNI SLAJD
        KRAJ AKO
    KRAJ AKO
KRAJ PROCESA dodaj_slog
```

Upis novog sloga

```
AKO JE i < VELIČINA_BLOKA
    POSTAVI blok.slogovi[i].ključ <- OZNAKA_KRAJA_DATOTEKE
    POMERI_POKAZIVAČ(U(serijska_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
    UPISI_BLOK(U(serijska_datoteka), UI(blok))
INAČE
    POMERI_POKAZIVAČ(U(serijska_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
    UPIŠI_BLOK(U(serijska_datoteka), UI(blok))
    INICIJALIZUJ_BLOK(I(novi_blok))
    POSTAVI novi_blok.slogovi[0].ključ <- OZNAKA_KRAJA_DATOTEKE
    UPIŠI_BLOK(U(serijska_datoteka), UI(novi_blok))
KRAJ AKO
```

Logičko brisanje sloga

Logičko brisanje sloga

- Logičko brisanje sloga datoteke podrazumeva izmenu statusnog polja
- Potrebno je proći kroz čitavu datoteku i proveriti da li se ključ nekog sloga poklapa sa ključem sloga za brisanje
- Ako je slog pronađen, potrebno je izmeniti statusno polje i izmenjeni slog upisati nazad u datoteku

Logičko brisanje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Provera da li slog postoji
3. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
4. Učitavanje bloka
5. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
6. Prolazak kroz sve slogove u trenutnom bloku
    a. Proveriti da li se njegov ključ poklapa sa ključem sloga za logičko brisanje
        i. Ako se ključ poklapa:
            1. izmeniti polje za logičko brisanje sloga
            2. i upisati izmenjeni slog u datoteku
            3. prekid procesa
        ii. Ako se ključ ne poklapa, nastaviti sa sledećim slogom.

Logičko brisanje sloga - koraci

7. Prelazak na sledeći blok i ponavljanje koraka 3-6
    a. Ako se dostigne kraj datoteke (i traženi slog nije pronađen), obavestiti korisnika da slog nije pronađen.

Logičko brisanje sloga

```
PROCES logicko_brisanje_sloga(U(serijska_datoteka, ključ_sloga), I(status), UI())
POČETAK PROCESA logicko_brisanje_sloga
    AKO JE serijska_datoteka == NULL
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(serijska_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka ), I(status_čitanja), UI(blok))
        (* učitavanje po blokovima *)
        RADI učitavanje_sadržaja_datoteke DOK JE status_čitanja == 0
                    POSTAVI i <- 0
                    RADI provera_slogova DOK JE i < VELIČINA_BLOKA
                        NAREDNI SLAJD
                    KRAJ RADI provera_slogova
                POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
                ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka ), I(status_čitanja), UI(blok))
        KRAJ RADI učitavanje_sadržaja_datoteke
    KRAJ AKO
KRAJ PROCESA logicko_brisanje_sloga
```

Logičko brisanje sloga - *provera_slogova*

```
RADI provera_slogova DOK JE i < VELIČINA_BLOKA

    (* dosli smo do kraja datoteke *)
    AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE TADA
        POSTAVI status <- 1
    KRAJ AKO

    (* dosli smo do sloga koji treba logicki obrisati *)
    AKO JE blok.slogovi[i].ključ == ključ_sloga TADA
        POSTAVI blok.slogovi[i].obrisan <- 1;
        POMERI_POKAZIVAČ(U(serijska_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
        UPIŠI_BLOK(U(serijska_datoteka, blok, redni_broj_bloka), I(status_upisa_u_datoteku), UI( ))
        POSTAVI status <- 0
    KRAJ AKO

    POSTAVI i <- i + 1
KRAJ RADI provera_slogova
```

Fizičko brisanje sloga

Fizičko brisanje sloga

- Fizičko brisanje sloga u datoteci zahteva njegovo prethodno nalaženje
- Nakon uspešnog nalaženja sloga, sve sledeće slogove u datoteci je potrebno pomeriti za jedno mesto unazad
- Počevši od bloka u kome je nađen slog za brisanje, svi sledeći blokovi se ažuriraju
- Zbog popunjavanja poslednjeg sloga u bloku, za izmenu svakog bloka potrebno je učitavanje i narednog bloka
- Da bi se pomeranje slogova datoteke nastavilo, vrednost sloga koji se briše se ažurira ključem prvog sloga u narednom bloku
- Ako je poslednji slog datoteke obrisan, potrebno je uraditi skraćivanje datoteke

Fizičko brisanje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Provera da li slog postoji
3. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
4. Učitavanje bloka
5. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
6. Prolazak kroz sve slogove u trenutnom bloku
    a. Proveriti da li se njegov ključ poklapa sa ključem sloga za fizičko brisanje
        i. Ako se ključ poklapa potrebno je pomeriti sve slogove do kraja datoteke za jedno mesto
        ii. Po potrebi obrisati poslednji blok ukoliko je ostao prazan

Fizičko brisanje sloga

```
PROCES fizičko_brisanje_sloga(U(serijska_datoteka, ključ_sloga), I(status), UI())
POČETAK PROCESA nadji_slog
    nadji_slog(U(serijska_datoteka, ključ_sloga), I(status_traženja_sloga)) TADA
    AKO JE nadjeni_slog == NULL
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(serijska_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))
        INICIJALIZUJ_BLOK(I(naredni_blok))
        POSTAVI trenutni_ključ_sloga <- ključ_sloga
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
            NAREDNI SLAJD
        KRAJ RADI provera_trenutnog_bloka
        POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
        ČITAJ_BLOK(U(serijska_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
    KRAJ AKO
KRAJ PROCESA fizičko_brisanje_sloga
```

Fizičko brisanje sloga

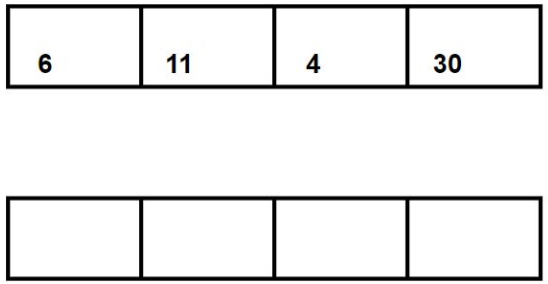
```
fseek(fajl, -sizeof(BLOK), SEEK_END);
long bytesToKeep = ftell(fajl);
ftruncate(fileno(fajl), bytesToKeep);
fflush(fajl); (* da bi se odmah obradio truncate *)
```

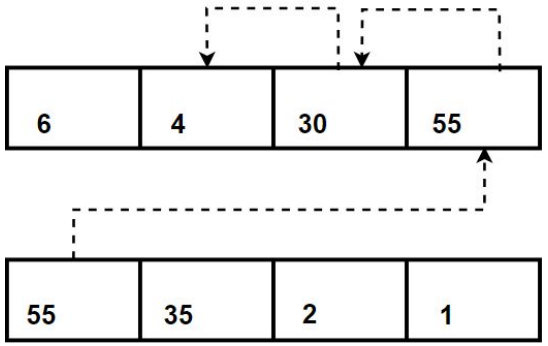
```
RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
    POSTAVI i <- 0
    RADI provera_slogova DOK JE i < VELIČINA_BLOKA
        (* došli smo do poslednjeg sloga datoteke, svi slogovi su pomereni * )
        AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE TADA
            AKO JE i == 0
            (* nakon pomeranja, oznaka kraja datoteke je prvi slog u poslednjem bloku * )
                SKRATI_DATOTEKU(I(serijska_datoteka, redni_broj_bloka - 1))
            KRAJ AKO
            POSTAVI status <- 0
        KRAJ AKO
        AKO JE blok.slogovi[i].ključ == trenutni_ključ_sloga_za_brisanje TADA
            reorganizuj_blokove(I(serijska_datoteka, blok, naredni_blok),
                                IU(trenutni_ključ_sloga, ključ_sloga, redni_broj_bloka))
        KRAJ AKO
        POSTAVI i <- i + 1
    KRAJ RADI provera_slogova
KRAJ RADI provera_trenutnog_bloka
```

Fizičko brisanje sloga

```
PROCES reorganizuj_blokove(I(datoteka, blok, naredni_blok), IU(trenutni_ključ, ključ, rbr_bloka))
POČETAK PROCESA reorganizuj_blokove
    AKO JE blok.slogovi[i].kljuc == ključ I blok.slogovi[i].obrisan == 1 TADA
        KRAJ reorganizuj_blokove (* slog koji korisnik želi da obriše je već logički obrisan*)
    KRAJ AKO
    POSTAVI j <- i + 1
    RADI pomeranje_slogova DOK JE j < VELIČINA_BLOKA
        POSTAVI blok.slogovi[j-1] <- blok.slogovi[j]
        POSTAVI j <- j + 1
    KRAJ pomeranje_slogova
    ČITAJ_BLOK(U(datoteka, rbr_bloka + 1), I(status_čitanja_narednog_bloka), UI(naredni_blok))
    AKO JE status_čitanja_narednog_bloka == 0  TADA (* postoje blokovi posle trenutnog *)
        POSTAVI rbr_bloka <- rbr_bloka - 1
        POSTAVI blok.slogovi[ VELIČINA_BLOKA -1] <- naredniBlok.slogovi[0]
        POSTAVI trenutni_ključ <- naredniBlok.slogovi[0].ključ
    KRAJ AKO
    POSTAVI rbr_bloka <- rbr_bloka - 1
    UPIŠI_BLOK(U(datoteka, blok, rbr_bloka), I(status_upisa_u_datoteku), UI( ))
    AKO JE status_čitanja_narednog_bloka == 0 TADA
        POSTAVI status <-0
    KRAJ AKO
KRAJ reorganizuj_blokove
```

trenutni_ključ_sloga = 11

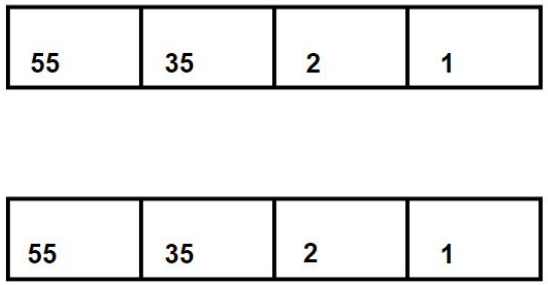


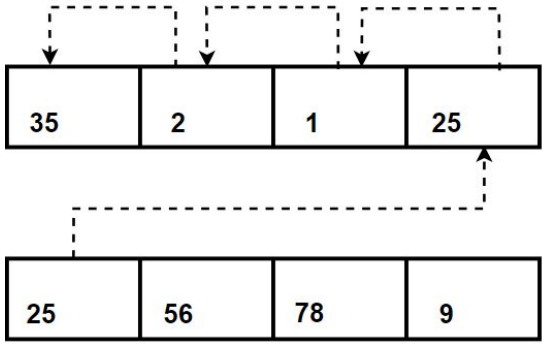


A_1	$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		11		4		30	
A_2	$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	55		35		2		1	
A_3	$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4	$p(S_{13})$		$p(S_{14})$				
	16		*				

A_1	$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		11		4		30	
A_2	$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	55		35		2		1	
A_3	$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4	$p(S_{13})$		$p(S_{14})$				
	16		*				

trenutni_ključ_sloga = 55

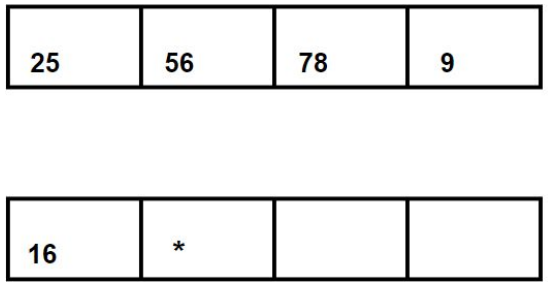




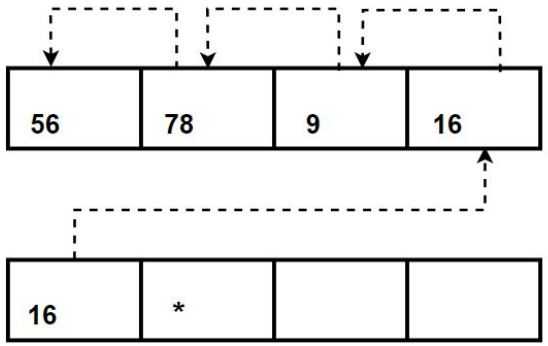
A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2		$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	55		35		2		1	
A_3		$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4		$p(S_{13})$		$p(S_{14})$				
	16		*					

A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2		$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	55		35		2		1	
A_3		$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4		$p(S_{13})$		$p(S_{14})$				
	16		*					

trenutni_ključ_sloga = 25

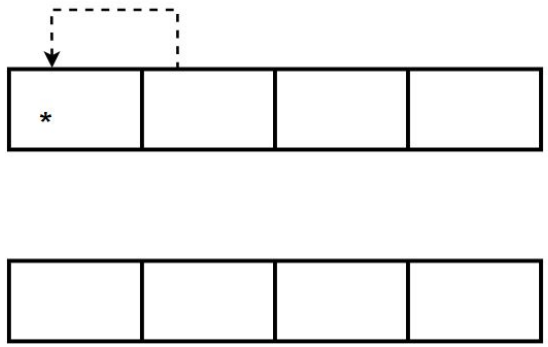


A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2		$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	35		2		1		25	
A_3		$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4		$p(S_{13})$		$p(S_{14})$				
	16		*					



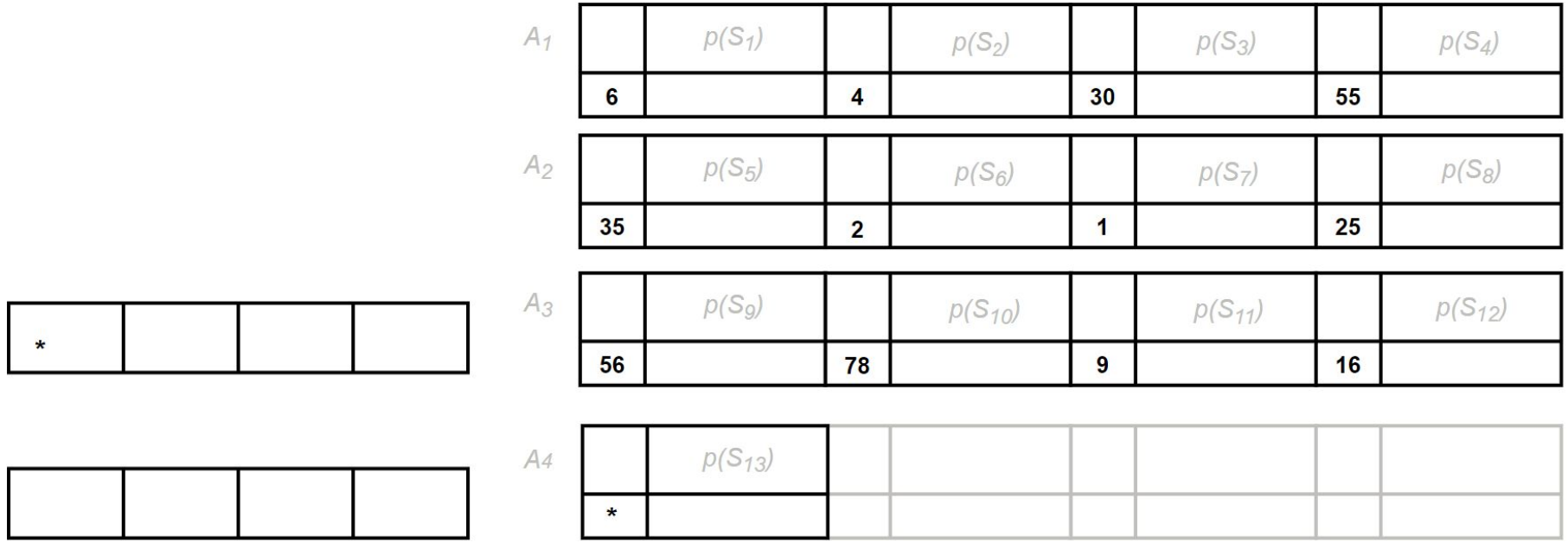
A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2		$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	35	Text	2		1		25	
A_3		$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	25		56		78		9	
A_4		$p(S_{13})$		$p(S_{14})$				
	16		*					

trenutni_ključ_sloga = 16



A_1	$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2	$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	35	Text	2		1		25	
A_3	$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	56		78		9		16	
A_4	$p(S_{13})$		$p(S_{14})$					
	16		*					

A_1	$p(S_1)$		$p(S_2)$		$p(S_3)$		$p(S_4)$
	6		4		30		55	
A_2	$p(S_5)$		$p(S_6)$		$p(S_7)$		$p(S_8)$
	35	Text	2		1		25	
A_3	$p(S_9)$		$p(S_{10})$		$p(S_{11})$		$p(S_{12})$
	56		78		9		16	
A_4	$p(S_{13})$		$p(S_{14})$					
	16		*					


A1
p(S1)
p(S2)
p(S3)
p(S4)
6
4
30
55
A2
p(S5)
p(S6)
p(S7)
p(S8)
35
2
1
25
A3
p(S9)
p(S10)
p(S11)
p(S12)
56
78
9
16
A4
p(S13)
*
*

Ažuriranje sloga

Ažuriranje sloga

- Isti postupak kao kod logičkog brisanja
- Koraci ažuriranja sloga:
  1. Uspešno traženje sloga
  2. Ažuriranje vrednosti sloga
  3. Upis bloka u kom se nalazi slog sa ažuriranim vrednostima

Primena serijske datoteke

Primena serijske datoteke

- Prenos podataka na memorijski medijum u procesu obuhvata podataka nakon čega se može izvršiti njihovo sortiranje ili druga raspodela
  - Često predstavlja polaznu osnovu za izgradnju datoteka sa drugim vrstama organizacije podataka
- Slučajevi gde je jednostavnost implementacije i unosa podataka važnija od brzog pristupa
  - Kada se podaci uglavnom evidentiraju ili dodaju bez potrebe za čestim preuzimanjem ili ažuriranjem (evidentiranje transakcija, logging sistema)
- Za male skupove podataka gde su prednosti složenijih metoda organizacije zanemarljive

Kraj!

Hvala na pažnji!