Napredne strukture podataka

Fibinačijev hip, B-stabla

Predmet: Uvod u Algoritme 17 - ESI053
Studijski program: Primenjeno softversko inženjerstvo

DEPARTMAN ZA RAČUNARSTVO I AUTOMATIKU
DEPARTMAN ZA ENERGETIKU, ELEKTRONIKU I KOMUNIKACIJE

Fibonačijev hip

- Prednosti:
  - Neke operacije se brže izvršavaju nego upotrebom „običnog“ hipa
  - Omogućava objedinjavanje-uniju hip struktura

- Osnovne operacije (posmatra se min-hip)
  - NAPRAVI-PRAZAN-HIP()
  - DODAJ(*H*, *x*) – dodaj elemenat x u hip H
  - MINIMUM(*H*) – vrati elemenat sa minimanom vrednosti ključa
  - IZDVOJ-MINUMUM(*H*) – vrati i obriši elemenat sa minimanom vrednosti ključa
  - UNIJA(H_1, H_2) – poveži dva hipa u jedan
  - UMANJI-KLJUČ(*H*, *x*, *k*) – za dati elemenat x u hipu H postavi ključ na manju vrednost k
  - OBRIŠI(*H*, *x*) – ukloni elemenat x iz hipa H

Složenosti operacija

Metod	Binarni hip	Fibonačijev hip (amortizovano vreme izvršavanja)
NAPRAVI-PRAZAN-HIP	$\Theta(1)$	$\Theta(1)$
DODAJ	$\Theta(\log_2 n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$\Theta(1)$
IZDVOJ-MINUMUM	$\Theta(\log_2 n)$	$O(\log_2 n)$
UNIJA	$\Theta(n)$	$\Theta(1)$
UMANJI-KLJUČ	$\Theta(\log_2 n)$	$\Theta(1)$
OBRIŠI	$\Theta(\log_2 n)$	$O(\log_2 n)$

- Amortizovano vreme izvršavanja je kod operacija dodavanja, umanjenja ključa i unije konstantno, tj. $\Theta(1)$, dok je kod brisanja i izdvajanja minimuma $O(\log_2 n)$.
  - Ovo znači da je počevši od praznog hipa posle a operacija iz prve grupe i b operacija druge grupe ukupna složenost $O(a + b\log_2 n)$, dok je kod binarnog hipa to $O((a + b)\log_2 n)$
- Fibonačijev hip je pogodan za česte operacije umanjenja-ključa, što rade npr. Dajkstra i Prim algoritmi

Složenosti operacija

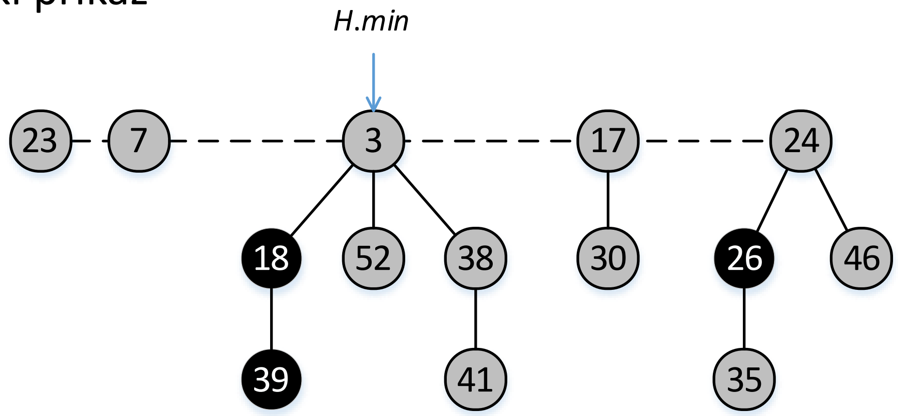
- Uglavnom ima teorijsku osnovu
- Iako je složenost manja od binarnog hipa, konstante su dosta veće, pa su realne brzine manje
- Praktična opravdanost je retka
  - Sem u nekim slučajevima kod velike količine podatakaka

- I binarni i Fibonačijev hip imaju spore operacije pretrage

Struktura Fibonačijevog hipa (1/2)

- Zasnovan je na nekoliko stabala (kolekciji stabala) koja su uređena kao min-hip.
  - U svakom stablu su zadovoljene osobine min hipa: **ključ čvora je veći ili jednak ključu roditelja**
- Ceo hip sadrži
  - pokazivač *min* na korenski čvor sa najmanjim ključem (to je minimalan čvor)
  - Korene stabala
  - n – broj čvorova/elemenata u hipu

Logički prikaz

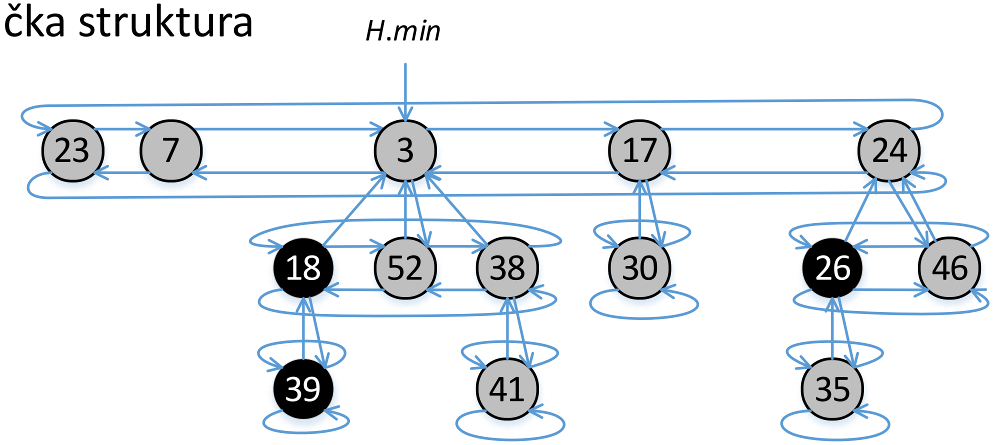


- Svaki elemenat se posmatra kao čvor nekog stabla i sadrži:
  - Ključ
  - Stepen (*degree*) – broj dece
  - Oznaku da je izgubio dete od trenutka kada je postavljen kao dete drugom čvoru (ovde je nazivamo „oznaka čvora“, a na slici je prikazana crnim čvorom)
    - Svaki novododani čvor, kao i čvor koji se postavi kao dete drugom čvoru ima oznaku „laž“

Fizička struktura Fibonačijevog hipa

- Koreni stabala su preko pokazivača *levo* i *desno* povezani u dvostruko spregnutu cirkularnu listu
- Svaki čvor sadrži:
  - Pokazivač na roditeljski čvor
  - Pokazivač na jednog od potomaka (dece)
  - Pokazivače *levo* i *desno*
- Deca su povezana u dvostruko spregnutu cirkularnu listu
  - Deca u listi nisu posebno uređena

Fizička struktura



Potencijal (vidi kasnije):
$\Phi(H) = 5 \cdot 1 + 3 \cdot 2 = 11$

Dodatne osobine

- Posledica strukture (postojanja stabala koja nemaju propisan oblik, a mogu biti i pojedinačni elementi) je mogućnost izvršavanja nekih operacija u „lenjom maniru“ gde se deo posla odlaže za kasnije pozive.
  - Npr. spajanje hipova se svodi na spajanje dve liste stabala; umanjenje ključa nekada odvoji čvor od roditelja i formira novo stablo
- Ipak, kasniji pozivi preslažu čvorove (konsoliduju hip) - da bi se „uveo red“ i postiglo željeno (amortizovano) vreme trajanja operacija
  - Stepeni čvorova ostaju niski, ne veći od $O(\log_2 n)$
- Veličina podstabla čiji koren ima stepen k je najmanje F_{k+2}, gde je F_i i-ti Fibonačijev broj
  - Ovo je postignuto pravilom da se može odvojiti najviše jedno dete od roditelja koji nije koren
  - Kada se odvoji drugo dete čvora tada se taj čvor odvaja od svog roditelja i postaje koren novog stabla
- Broj stabala se smanjuje operacijom izdvoj minimum i data se stabla povezuju (konsoliduje se hip)

Fibonačijevi brojevi su: 1, 1, 2, 3, 5, 8, 13, 21, 34, ... ($F_1 = F_2 = 1, F_k = F_{k-1} + F_{k-2}$)

Dodatne osobine

- Neke operacije su brze, kod za druge treba dosta više vremena.
- Analiza amortizovanog vremena uzima za brze operacije konstantnog trajanja više vremena nego je zaista potrebno i taj „višak vremena“ akumulira za kasnije operacije kada će se oduzeti od stvarnog vremena izvršavanja sporih operacija.
- Definiše se funkcija potencijala koja kvantifikuje akumulirani višak vremena

$$\Phi(H) = t(H) + 2m(H)$$

  - $t(H)$ je broj stabala (svaki koren sačuva 1 jedinicu vremena da se kasnije taj koren poveže sa drugim korenom i tada ta amortizovana operacija traje 0)
  - $m(H)$ je broj označenih čvorova ($2m$ čuva 2 jed.vremena: jedna se može koristi kod odvajanja čvora kada on postaje koren, a druga je sačuvana u njemu kao korenu)

- Maksimalan spepen $D(n)$ je najveći spepen svih čvorova u hipu sa n čvorova. Može se dokazati da je $D(n) = O(\log_2 n)$

$D(n) \leq \lfloor \log_\phi n \rfloor,\ \phi = \frac{1+\sqrt{5}}{2} = 1.61803\ \ldots$ (tj. zlatni presek)

Operacije

- Operacije odlažu konsolidaciju – koliko mogu
- Neke operacije su brze jer su jednostavne
  - Npr. dodavanje, gde se dodaje čvor u listu korena hipa
- Druge operacije preslažu čvorove (vrše konsolidaciju)
  - Npr. Izdvoj-Minimum ukloni čvor, ali onda traži najmanji čvor, što je sporo ako su svi dodati čvorovi u listi pa ih zato preslaže - konsoliduje, da bi naredne operacije bile brže
  - Nakon preslaganja veličina korenske liste je najviše $D(n) + 1$, a svako stablo ima stepen jedinstven u toj listi

...

Operacije: koje su jednostavne i brze

- Napravi-Prazan-Hip()
  - Jednostavna operacija konstantnog trajanja $O(1)$
- Minimum(*H*)
  - *H.min* ukazije na minimalan čvor.
    Jednostavna operacija konstantnog trajanja $O(1)$
- Unija dva hipa
  - Spajanje dve liste se vrši u konstantnom vremenu $O(1)$
  - Nema promene potencijala $\Delta\Phi = \Phi(H) - \big(\Phi(H_1) + \Phi(H_2)\big) = 0$

```
NAPRAVI-PRAZAN-HIP()
  H = new FibonačijevHip
  H.n = 0
  H.min = Nil
  return H
```

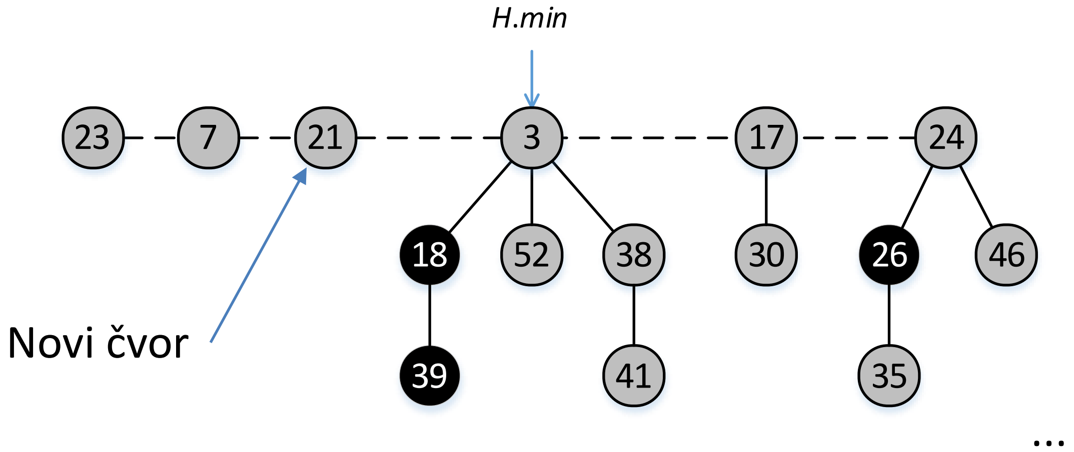
```
MINIMUM(H)
  return H.min
```

```
UNIJA(H1, H2)
  H = NAPRAVI-PRAZAN-HIP()
  H.min = H1.min
  Spoj korenske liste H1 i H2    // trajanje O(1)
  if (H1.min==Nil) or (H2.min ≠ Nil and H2.min.ključ < H1.min.ključ)
      H.min = H2.min
  H.n = H1.n + H2.n
  return H
```

Operacija: Dodavanje čvora

- Doda se novi čvor kao koren stabla.
- Potencijal se poveća za 1.
  Posle dodavanja:
  $\Phi(H_+) = t(H_-) + 1 + 2m(H_-)$
  - H_+ hip posle dodavanja
  - H_- hip pre dodavanja
- Tj. razlika pre i posle dodavanja $\Delta\Phi = 1$
- Stvarno trajanje operacije je $O(1)$, a amortizivano je $O(1) + 1c = O(1)$

```
DODAJ(H, x)
x.stepen = 0
x.r = Nil
x.dete = Nil
x.oznaka = False
if H.min == Nil
   Napraviti listu korena i dodati x
   H.min = x
else dodaj x u listu korena
   if x.ključ < H.min.ključ, H.min = x
H.n = H.n + 1
```

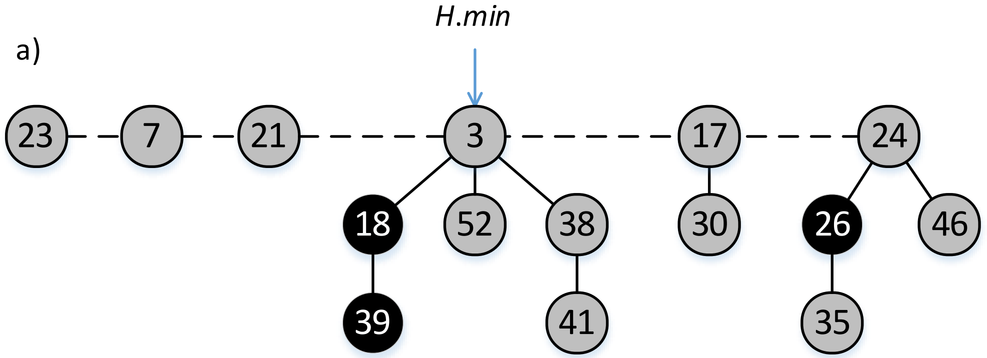


Pravimo se da dodavanje traje duže za c da bi se taj „višak vremena“ akumulirao za kasnije operacije.

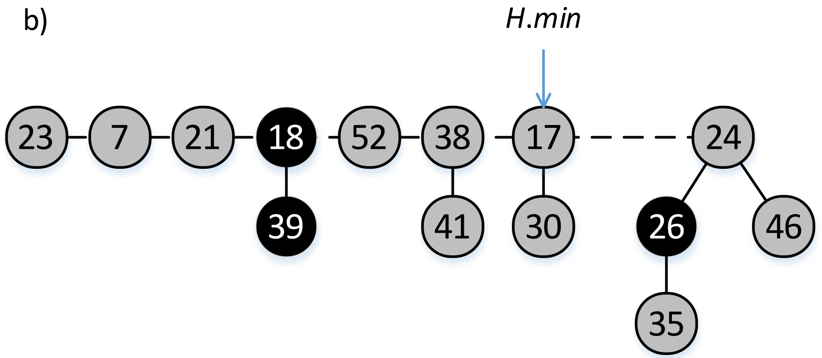
Operacija: Izdvoj minumum

- Sprovodi se u tri faze:
  a) Izbaci se koren sa minimumom
  b) Njegova deca postaju koreni novih stabala
  c) Ovde se pojavljuje konsolidacija ...

a)



b)



Stanje pre konsolidacije

```
IZDVOJ-MINUMUM(H)
z = H.min
if z ≠ Nil
    for each x ∈ dete(z)
        dodaj x u korensku listu
        x.r = Nil
    Ukloni z iz korenske liste
    if z == z.desno
        H.min = Nil
    else H.min = z.desno
        KONSOLIDUJ(H)
    H.n = H.n - 1
return z
```

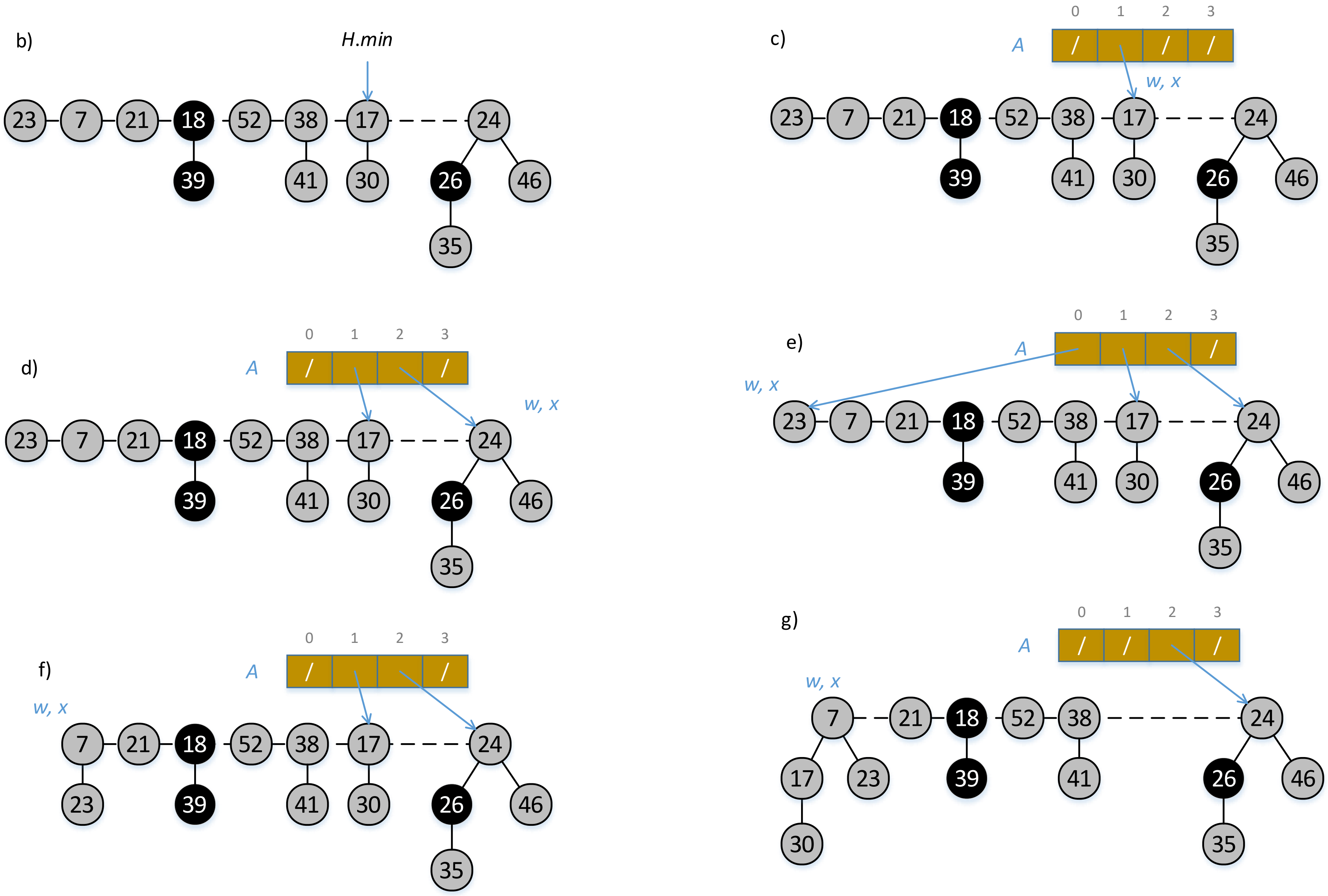
Konsolidacija

- Pomoćna operacija koja prepakuje korenska stabla, tj. svako stablo ili ostavlja ili ga spoji sa drugim stablom koje u korenu ima isti stepen (isti broj dece)
- Posledica: na kraju svi korenski čvorovi imaju drugačiji stepen, tj. broj dece
  Niz A ukazuje na korenske čvorove stepena 0, 1, 2 ..., $D(n)$
  - Podsećanje: $D(n)$ je maks. stepen čvora

```
POVEŽI(H, y, x)
  ukloni y iz korenske liste
  Postavi y kao dete od x, x.stepen += 1
  y.oznaka = False
```

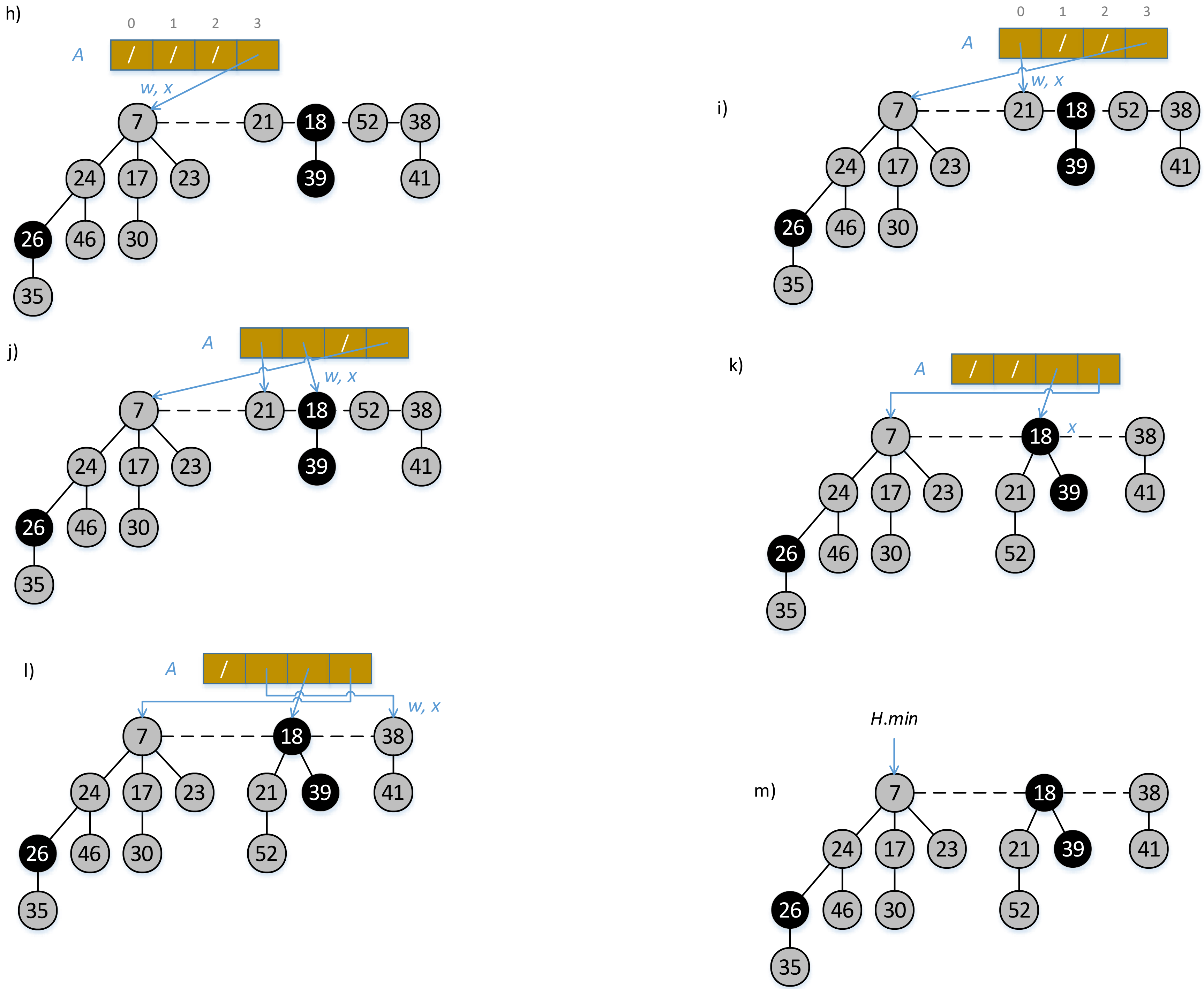
```
KONSOLIDUJ(H)
  Novi niz A[0..D(H.n)]
  for i = 0 to D(H.n)
     A[i] = Nil
  for each čvor w u korenskoj listi
     x = w
     d = x.stepen
     while A[d] ≠ Nil
        y = A[d]
        if x.kluč > y.ključ
           zameni x i y
        POVEŽI(H, y, x)
        A[d] = Nil
        d = d + 1
     A[d] = x
 H.min = Nil
 for i = 0 to D(H.n)
     if A[i] ≠ Nil
        if H.min == Nil
           Napravi korensku listu sa A[i]
           H.min = A[i]
        else dodaj A[i] u korensku listu
           if A[i].ključ < H.min.key
              H.min = A[i]
```

Primer:
Konsolidacija (1/2)



Primer:

Konsolidacija (2/2)



Složenost izdvajanja minimuma

- „Stvarno“ vreme izvršavanja je $O\big(D(n) + t(H)\big)$
  - Brisanje stabla sa minimumom u korenu i postavljanja njegove dece kao nove korene zahteva $O(D(n))$ i poveća potencijal za $D(n) - 1$ (čvor ima max $D(n)$ dece, a -1 zbog brisanja čvora).
  - For petlja 4-14 ima unutrašnju while petlju gde se vrši spajanje korena i broj iteracija ne može biti veći od broja korena, tj. proporcionalan je $D(n) + t(H)$
- Amortizovana složenost je $O\big(D(n)\big) = O(\log_2 n)$
  - Razlika u potencijalima posle $D(n) + 1 + 2m(H)$ i pre konsolidacije $t(H) + 2m(H)$ je $D(n) + 1 - t(H)$
  - Amortizovano vreme izvršavanja je (stvarno vreme + razlika potencijala) $O\big(D(n) + t(H)\big) + D(n) + 1 - t(H) = O\big(D(n)\big) + O\big(t(H)\big) - t(H) = O\big(D(n)\big)$.
- U osnovi stvarno vreme je umanjeno zbog smanjenja broja korena

Pravimo se da izdvajanje minumuma traje kraće jer se „manjak vremena“ nadoknadio iz akumulacije za kasnije operacije.

Operacija: Umanji ključ

- Ako umanjenje ključa x ne narušava strukturu hipa onda se ključ promeni
- Ako umanjenje ključa x narušava strukturu hipa onda se:
  - x odvaja od roditelja y i postavlja u korensku listu
  - roditelj y se postavlja kao korensko stablo ako je x 2. dete koje gubi (ovo se kaskadno primenjuje na njegovog roditelja)
- Amortizovano vreme izvršavanja je $O(1)$

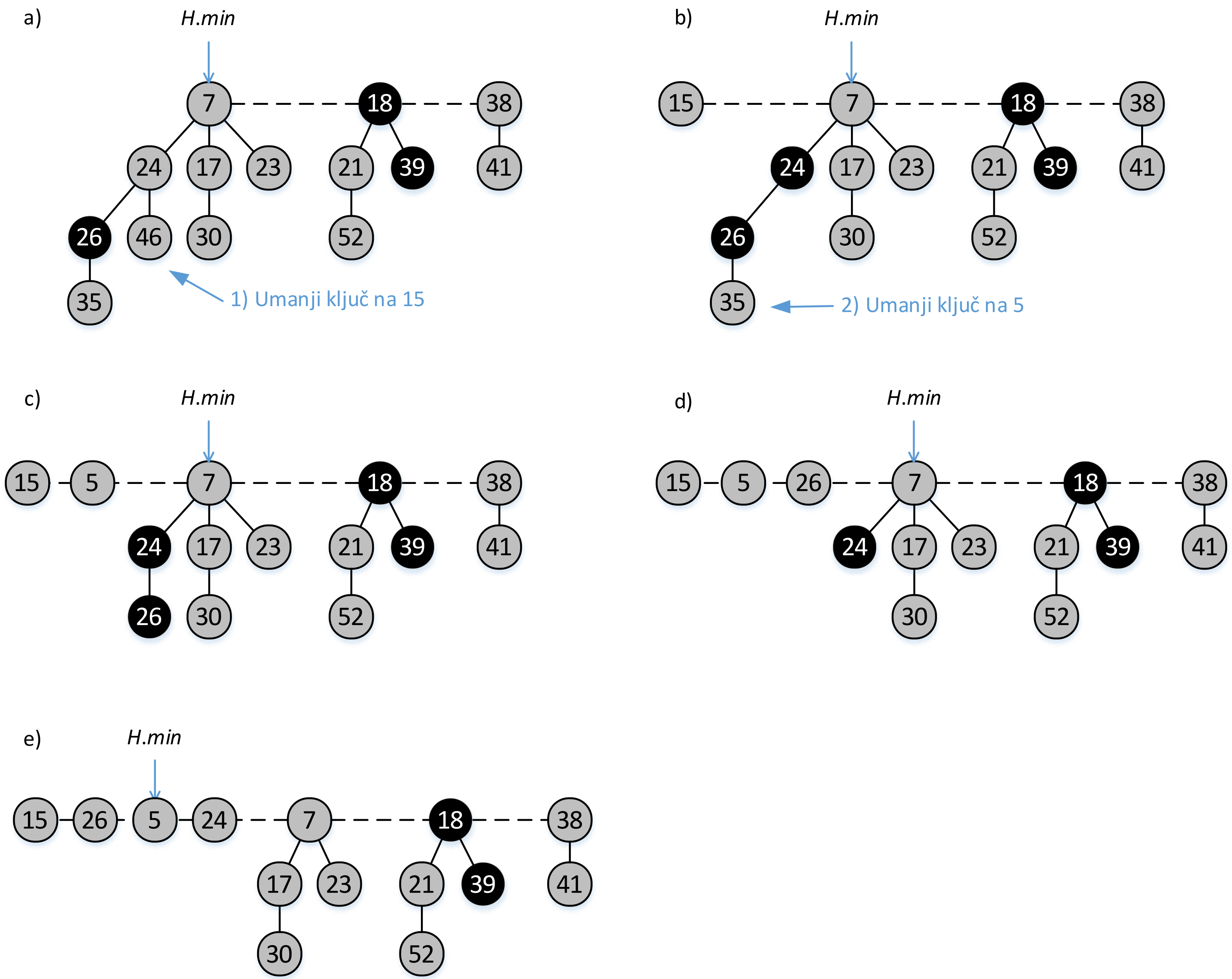
Oznaka se postavi kada se odvoji prvo dete, a obriše se kada se postavi u korensku listu.

```
UMANJI-KLJUČ(H, x, k)
if k < x.ključ
     error „novi ključ je veći od prethodnog“
x.ključ = k
y = x.r
if y ≠ Nil and x.ključ < y.ključ
   ODVOJ(H, x, y)
   KASKADNO-ODVAJAJ(H, y)
if x.ključ < H.min.ključ
   H.min = x

ODVOJ(H, x, y)
Ukloni x iz liste y dece, y.stepen -= 1
Dodaj x u korensku listu
x.r = Nil
x.oznaka = False

KASKADNO-ODVAJAJ(H, y)
z = y.r
if z ≠ Nil
   if y.oznaka == False
      y.oznaka = True
   else ODVOJ(H, y, z)
      KASKADNO-ODVAJAJ(H, z)
```

Primer:
Umanji ključ



Složenost umanjenja ključa

- Odvajanje čvora x traje $O(1)$, a ako dođe do kaskadnog odvajanja $c-1$ puta (c je konstanta) tada nastaje novih c korena i briše se do $c-2$ oznaka ($c-1$ poziva Kaskadno-Odvajaj gde poslednji poziv ne menja oznaku).
- Stvarno vreme je $O\big(1+(c-1)\big)=O(c)$
- Promena potencijala je:
  $\Delta\Phi(H)=\Big(\big(t(H)+c\big)+2\big(m(H)-(c-2)\big)\Big)-\big(t(H)+2m(H)\big)=4-c$
- Amortizovana složenost je najviše $O(c)+4-c=O(1)$ i menja se potencijal.

Operacija: Brisanje čvora

- Svodi se na smanjenje ključa na -∝ kako bi ga uklonilo izdvajanje minimuma
- Amortizovano vreme izvršavanja je $O\big(D(n)\big) = O(\log_2 n)$

```
OBRIŠI(H, x)
  UMANJI-KLJUČ(H, x, -∝)
  IZDVOJ-MINUMUM(H)
```

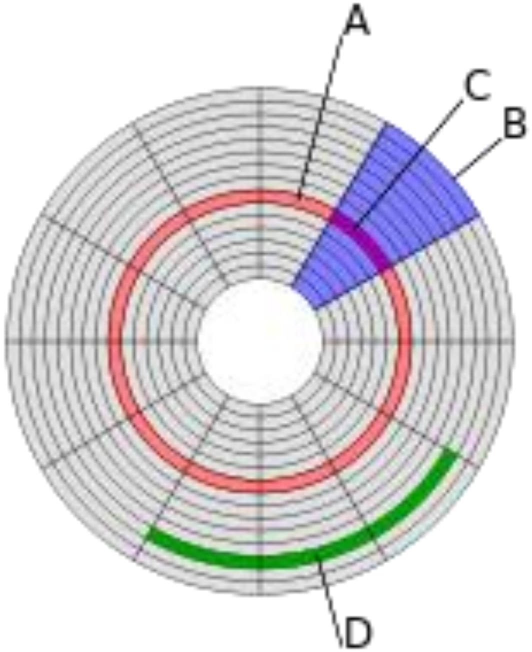
B-stabla

B-stabla

- B-stabla su balansirana stabla pretrage dizajnirana za velika skladišta podataka (na diskovima) – gde svi podaci ne mogu da stanu u radnu memoriju (RAM)
- Slična su crveno-crnim stablima, ali su prilagođena da minimizuju disk I/O operacije
- Generalizuje se koncept binarnog stabla tako da svaki čvor sadrži n ključeva i ima $n+1$ opseg vrednosti

- B-stabla imaju mnogo dece u čvorovima (nekoliko hiljada) – faktor grananja je veliki
- Visina stabla je $O(\log_2 n)$, ali je stvarna visina znatno manja od crveno-crnog stabla zbog velikog faktora grananja
- Većina operacija vezana za dinamičku strukturu podataka se sprovodi u $O(\log_2 n)$

Organizacija podataka na rotirajućim diskovima



Oranizacija podataka na rotirajućem disku:
A – staza
B – sektor
C – blok
D – podatak

- Operacije sa diskom:
  - Čitanje: DiskOčitaj(x)
  - Zapisivanje: DiskZapiši(x)
- Pretpostavka je da se pristup traženim podacima vrši preko ključa
- Iako se ovde posmatra rotirajući disk, blokovska organizacija podataka je prisutna i u drugim tipovima diskova (npr. SSD)

Definicija B-stabla

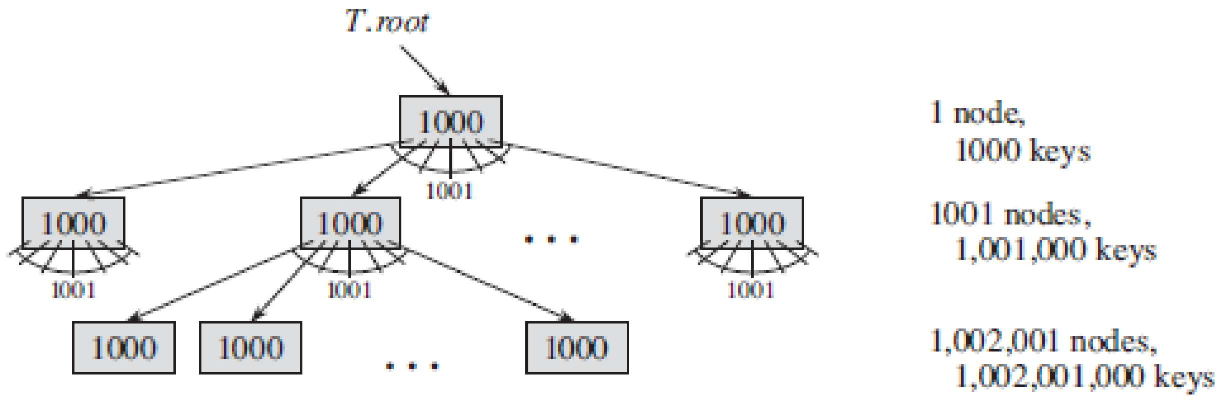
- Smatraćemo da su sa svakom ključu asocirani satelitski podaci, kao i da se satelitski podaci nalaze smešteni u čvorovima
  - Moguća je izmenjena implementacija gde se u čvorovima nalaze pokazivači, a da su satelitski podaci smešteni „sa strane“

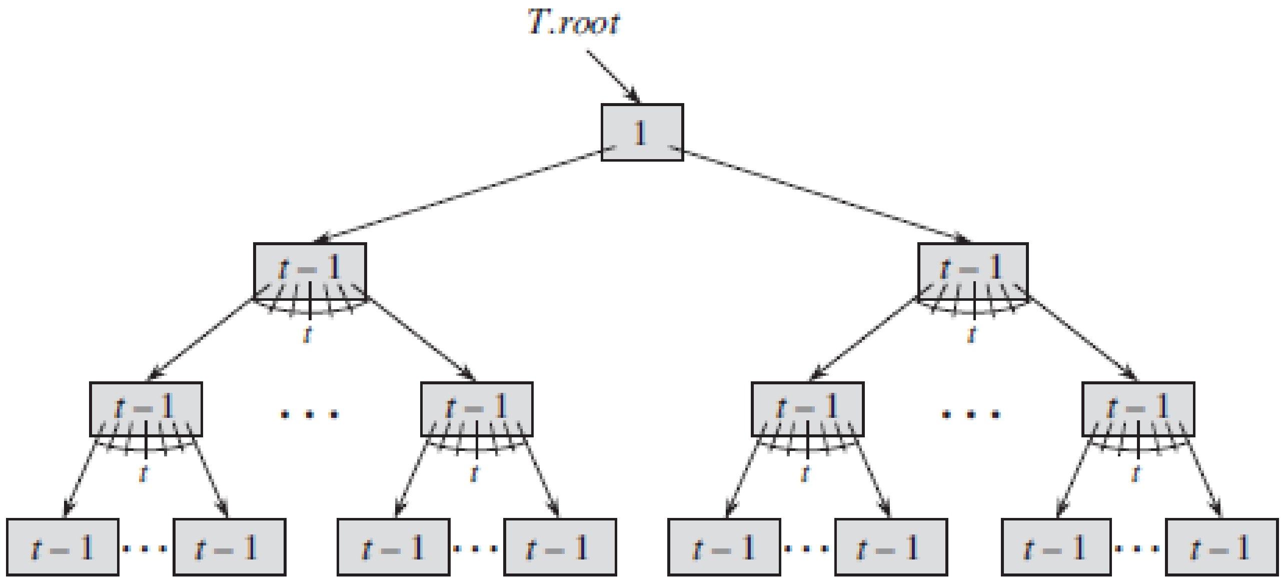
- B^+-stabla su varijanta B-stabala gde su satelitski podaci smešteni samo u listovima, dok se u internim čvorovima nalaze samo ključevi i veze ka čvorovima.

Struktura B-stabla

- Svaki čvor ima:
  - broj ključeva n
  - Same ključeve $k_1, k_2, k_3 \dots k_n$ za koje važi $k_1 \leq k_2 \leq k_3 \dots \leq k_n$
  - Indikator *list* (bulovu vrednost) koji govori da li je čvor list ili je interni čvor
  - $n + 1$ pokazivača na decu
- Svi listovi se nalaze na istoj dubini h
- Čvorovi imaju donju i gornju granicu broja dece definisanu preko parametra $t \geq 2$ (t je minimalan stepen čvora)
  - Svaki čvor sem korena mora imati najmanje $t - 1$ ključeva i t dece
  - Ako stablo nije prazno koren ima barem jedan ključ
  - Svaki čvor može sadržati najviše $2t - 1$ ključeva i $2t$ dece.
  - **Pun čvor** sadrži maksimalan broj ključeva.
- Najjednostavnije B-stablo se dobija za $t = 2$ i tada je to 2-3-4-stablo
- U praksi se koristi veća vrednost za t da bi stablo imalo manju visinu

Struktura B-stabla





Visina B-stabla

Teorijski, stablo sa n ključeva i minimalnim stepenom $t \geq 2$, ima visinu

$$h \leq \log_t \frac{n+1}{2}$$

Najmanje 1 ključ u korenu, svi ostali čvorovi $t-1$. Znači, najmanje 2 čvora na dubini 1, najmanje $2t$ čvorova na dubini 2, najmanje $2t^2$ čvorova na dubini 3, … najmanje $2t^{h-1}$ čvorova na dubini h

Ukupno ključeva: $n \geq 1 + (t-1)\sum_{i=1}^{h} 2t^{i-1} = 1 + 2(t-1)\frac{t^h-1}{t-1} = 2t^h - 1$

Odatle je: $t^h \leq (n+1)/2$

Osnovne operacije

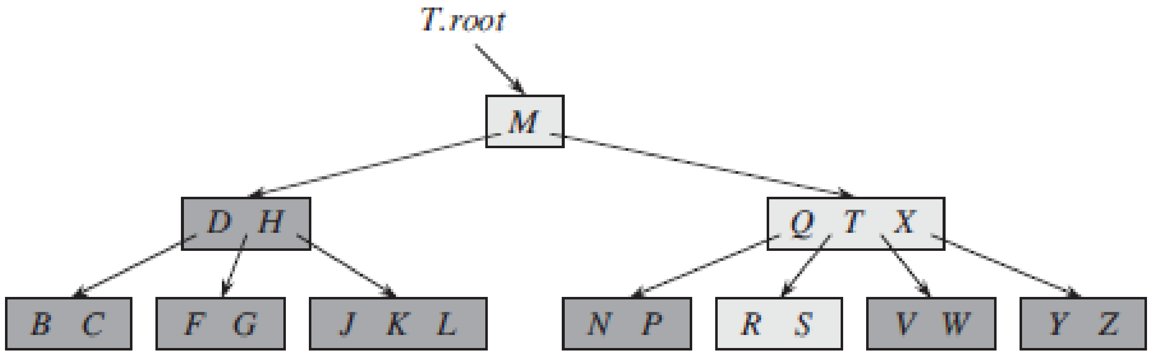
- Osnovne operacije su:
  - Pretraga,
  - Izgradnja praznog stabla,
  - Dodavanje i
  - Brisanje

- Principi
  - Koren stabla je uvek u glavnoj memoriji (RAM-u)
    - Nikada nema potrebe za čitanjem sa diska, ali svaki put kada se promeni koren potrebno je promenu zapisati na disk
  - Svaki čvor koji se prosleđuje kao parametar mora biti prethodno pročitan sa diska
  - Sve procedure imaju jednosmeran sled od korena nadole (nikada nema vraćanja u više nivoe hijerarhije stabla)

Pretraga B-stabla

- Funkcioniše slično pretrazi u binarnom stablu, ali se vrši poređenje sa više ključeva tako da ima $n + 1$ mogućih ishoda
- Složenost: $\mathrm{O}(h) = O(\log_t n)$ je broj rekurzivnih poziva, tj. broj čitanja sa diska. Kako je broj ključeva $< 2t$, onda je broj poređenja $O(t)$. Ukupna složenost je $O(th) = O(t \log_t n)$

```
PRONAĐI(x, k)
  i = 1
  while i < x.n and k > x.k_i
     i = i + 1
  if i < x.n and k == x.k_i
     return (x, i)
  elseif x.list
     return Nil
  else DISK-OČITAJ(x.d_i)
     return PRONAĐI(x.d_i, k)
```



Izgradnja praznog B-stabla

- Izgradnja zahteva i zapisivanje na disk
- Složenost: $O(1)$ za CPU (procesorske) operacije, i $O(1)$ za operacije sa diskom

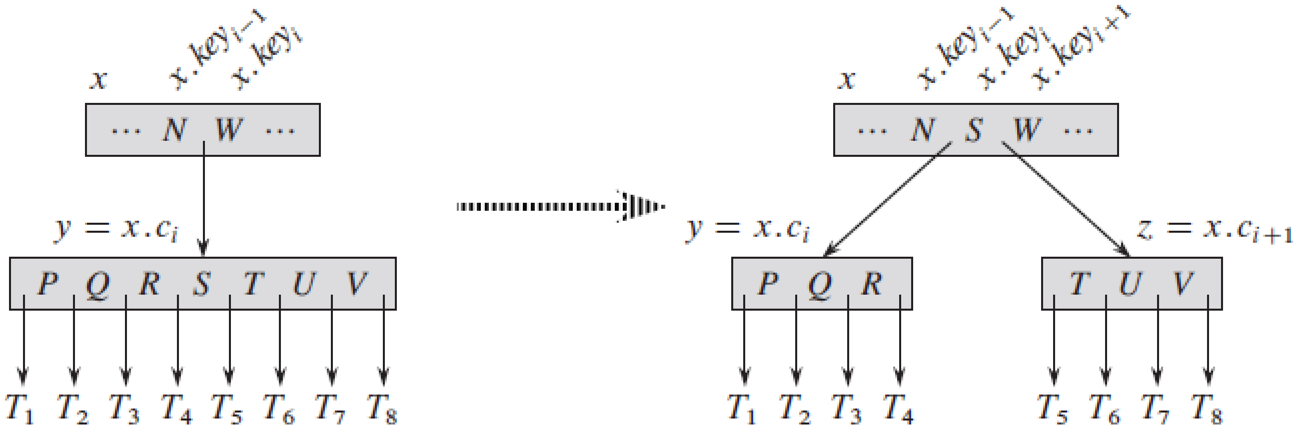
```
IZGRADI(T)
    x = ALOCIRAJ-ČVOR()      // zauzimi jednu stranicu, O(1)
    x.list = True
    x.n = 0
    DISK-ZAPIŠI(x)
    T.koren = x
```

Dodavanje ključa u B-stablo

- Značajno je složenije od dodavanja ključa u binarno stablo
- Ključ se dodaje u postojeći čvor-list.
  - Ako se ključ ne može dodati jer je čvor pun (ima $2t-1$ ključeva), tada se čvor deli na dva u okolini medijane postojećih ključeva tako da oba čvora imaju po $t-1$ ključeva.
  - Ključ medijane se „penje“ u roditeljski čvor da bi identifikovao mesto podele kod novododanih čvorova, a ako ni u njemu nema onda se on deli i tako redom do korena.
  - Međutim, da bi se procedura sprovela „od gore-ka dole“ u jednom prolazu, svaki pun čvor se odmah deli pre nego se „siđe“ na niži nivo u hijerarhiji.
    - Posledica: Kada dođe do podele punog čvora, podrazumeva se da roditelj nije pun čvor.

Podela čvora

- Ulaz je čvor x čije je i-to dete $x.d_i$ pun čvor. Podrazumeva se da su ovi podaci u memoriji.
- Procedura deli čvor $x.d_i$ na dva i ažurira x
- Složenost: procesor $O(t)$, disk $O(1)$

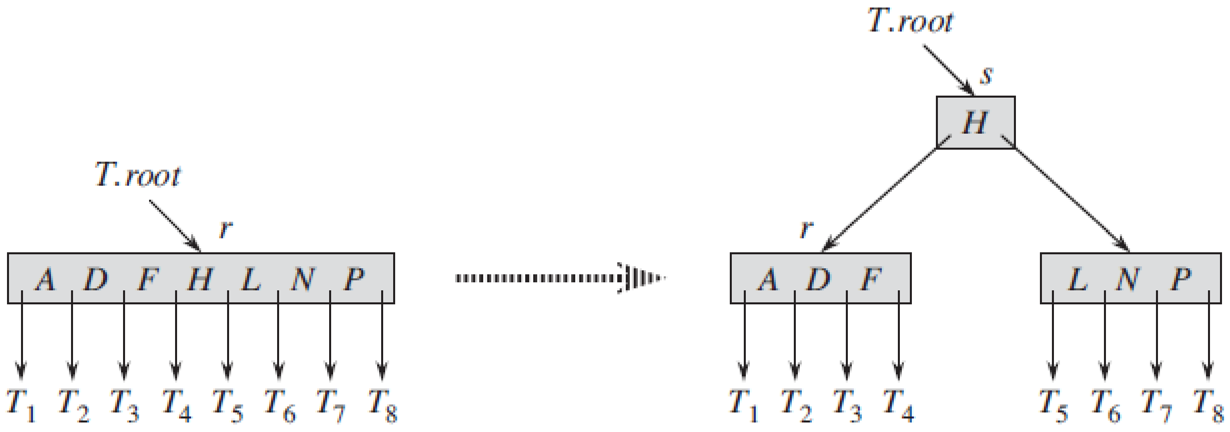


Napomena: Na slici je dete označeno $x.c_i$ a u kodu $x.d_i$

```
PODELI-ČVOR-DETETA(x,i)
    z = ALOCIRAJ-ČVOR()
    y = x.d_i
    z.list = y.list
    z.n = t-1
    for j = 1 to t-1
        z.k_j = y.k_{j+t}
    if not y.list
        for j = 1 to t
            z.d_j = y.d_{j+t}
    y.n = t-1
    for j = x.n+1 downto i+1
        x.d_{j+1} = x.d_j
    x.d_{i+1} = z
    for j = x.n downto i
        x.k_{j+1} = x.k_j
    x.k_i = y.k_t
    x.n = x.n + 1
    DISK-ZAPIŠI(y)
    DISK-ZAPIŠI(z)
    DISK-ZAPIŠI(x)
```

Dodavanje ključa u jednom prolazu

- Posebno se posmatra situacija kada je korenski čvor pun.
- Jedini način da se poveća visina stabla je podela korenskog čvora.
- Izmene se događaju počev od korena na dole
  - DODAJ-KADA-NIJE-PUN se rekurzivno spušta na dole i ako je potrebno pravi podelu čvora.




```
DODAJ(T,k)
    r = T.koren
    if r.n == 2t - 1
        s = ALOCIRAJ-ČVOR()
        T.koren = s
        s.list = False
        s.n = 0
        s.d1 = r
        PODELI-ČVOR-DETETA(s,1)
        DODAJ-KADA-NIJE-PUN(s,k)
    else DODAJ-KADA-NIJE-PUN(r,k)
```

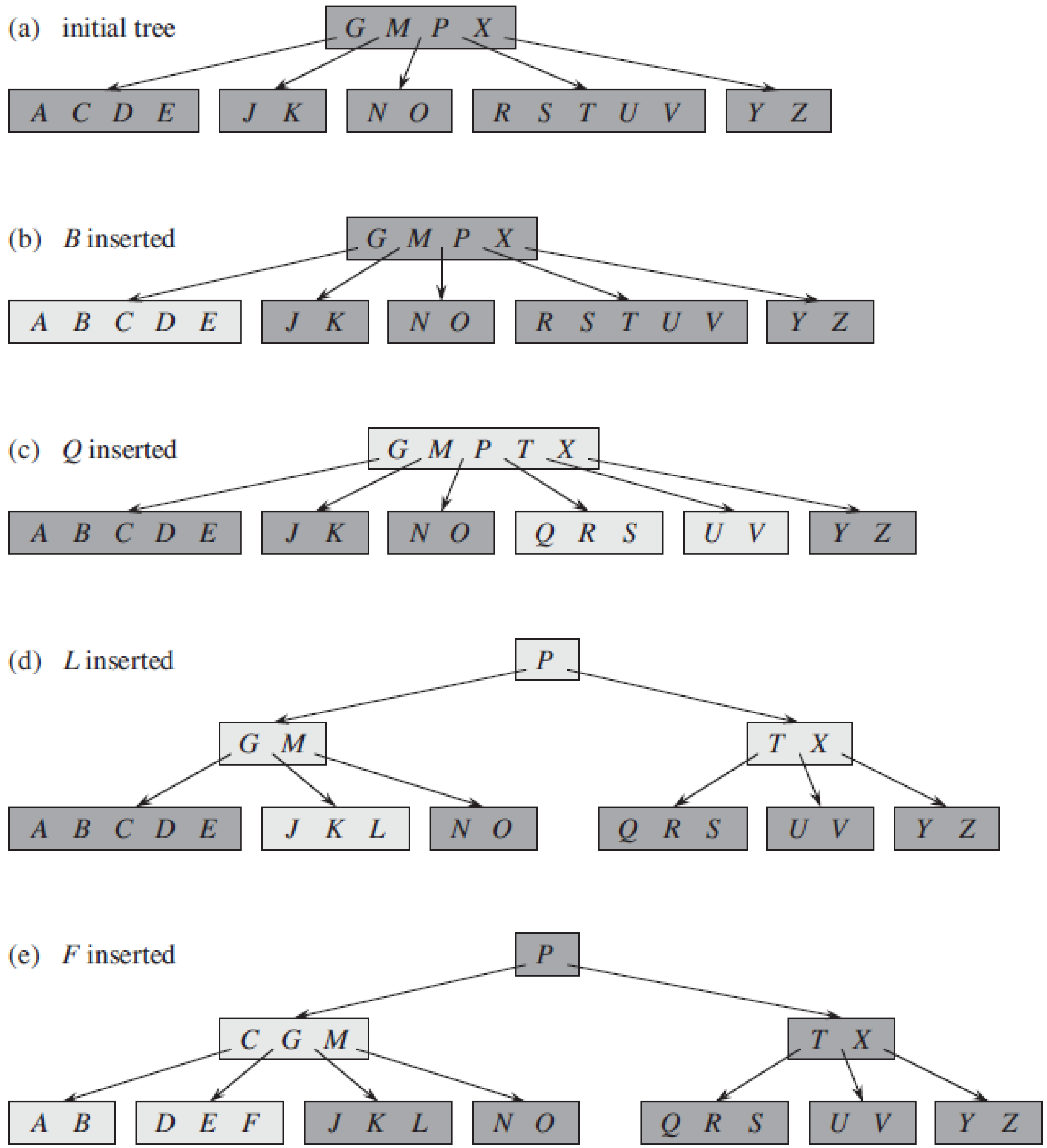
Dodavanje čvora

- za stablo visine h potrebno je $O(h)$ disk pristupa jer je $O(1)$ čitanja i pisanja u svakom pozivu DODAJ-KADA-NIJE-PUN
- Složenost: procesor $O(th) = O(t \log_t n)$
- Broj stranica sa diska koji se drže u RAMu je $O(1)$

```
DODAJ-KADA-NIJE-PUN(x,k)
i = x.n
if x.list
    while i ≥ 1 and k < x.k_i
        x.k_{i+1} = x.k_i
        i = i - 1
    x.k_{i+1} = k
    x.n = x.n + 1
    DISK-ZAPIŠI(x)
else while i ≥ 1 and k < x.k_i
        i = i - 1
    i = i + 1
    DISK-OČITAJ(x.d_i)
    if x.d_i.n == 2t - 1
        PODELI-ČVOR-DETETA(x,i)
        if k > x.k_i
            i = i + 1
    DODAJ-KADA-NIJE-PUN(x.d_i,k)
```

Primer dodavanja

- $t = 3$, tj. svaki čvom može imati najviše 5 ključeva



Brisanje iz B-stabla

- Brisanje je analogno dodavanju, ali je složenije jer se ključ može obrisati iz svakog čvora.
- Pravilo B-stabla nameće da je minimalan broj dece t
- U osnovi brisanje ažurira čvorove od gore ka dole, ali nekim slučajevima je potrebno vratiti se na ažuriranje roditelja nakon ažuriranja deteta kako bi se očuvalo pravilo
- Generalno, većina ključeva je u B-stablu je u lišću, i za njih važi najjednostavnija procedura ...

Procedura brisanja iz B-stabla

Postoji nekoliko slučajeva koji određuju proceduru:

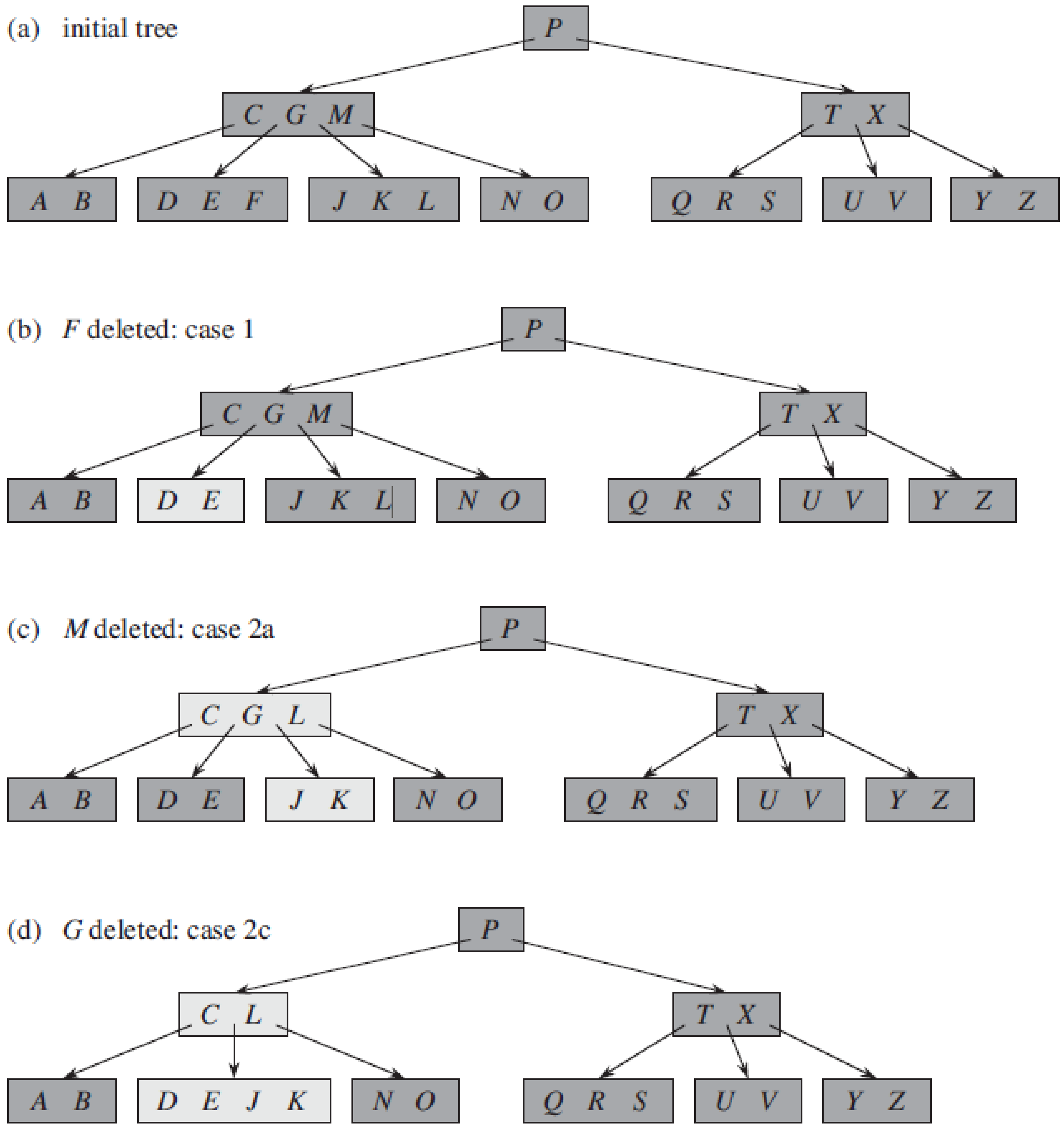
1) Ako je ključ u listu, obrisati ključ
2) Ako je ključ u internom čvoru:
   a) ako levi čvor deteta (ispred brisanog ključa) ima „višak ključeva“ ($\geq t$) onda se njegov najveći ključ prebaci umesto obrisanog. (Brisanje najvećeg ključa se se rekurzivno sprovodi u dubinu).
   b) ako levi čvor nema „višak ključeva“ onda se gleda desni čvor deteta (po ključu iza brisanog) i ako on ima „višak ključeva“ ($\geq t$) onda se njegov najmanji ključ prebaci umesto obrisanog (rekurzivno).
   c) ako neposredno i levo i desno dete „oko“ brisanog ključa imaju po minimalan broj ključeva (po $t - 1$) onda se oni spajaju u jedan čvor i željeni ključ se briše
3) Ako ključ nije u internom čvoru x, onda se proverava čvor ispod y_i u čijem delu stabla treba da postoji ključ. Ako čvor y_i ima $t - 1$ ključeva onda se osigurava da ima barem t ključeva tako što se:
   a) Proveri da li neposredno (levo y_{i-1} ili desno y_{i+1}) susedno dete od y_i ima barem t ključeva i ako ima onda se (najveći ili najmanji) ključ prebaci u x umesto ključa koji se doda u y_i
   b) Ako y_{i-1} i y_{i+1} imaju po $t - 1$ ključeva onda se y_i spoji sa jednim od njih tako što se ključ iz x (koji ih razdvaja) prebaci u spojen čvor

   Cela procedura se rekurzivno ponavlja za čvor y

Primer brisanja (1/2)

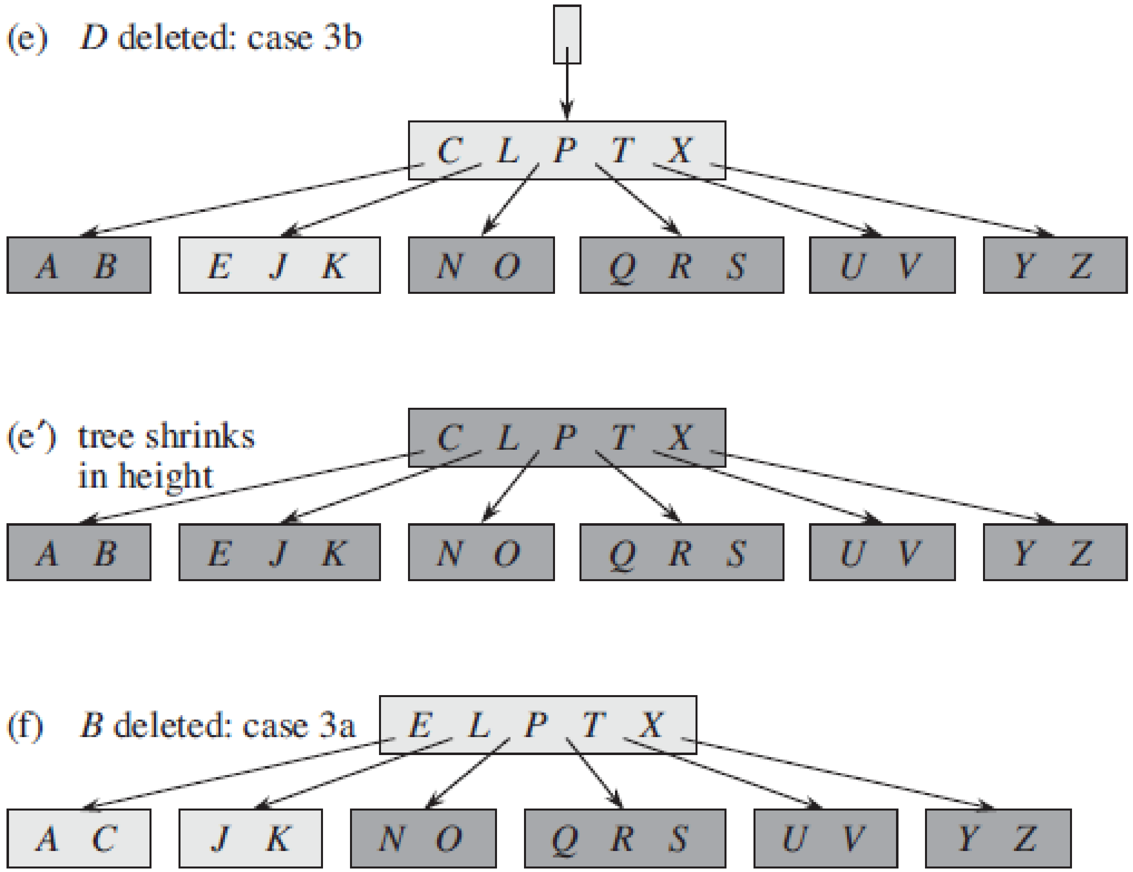
- $t = 3$, tj. minimalan broj ključeva u čvoru je 2

a) Početno stanje
b) Jednostavno brisanje iz lista
c) Brisanje iz čvora koji nije list: prethodni sa nižeg nivoa L zamenju obrisanog M
d) G se spušta na niži nivo da poveže DEGJK i onda se uklanja



Primer brisanja (2/2)

- (nastavak)



e) P se „spušta“ da poveže CLPTX, a zatim se briše D (kao u a)). Prazan koren se briše i broj nivoa se smanjuje za 1.
f) C se spušta ne mesto obrisanog B, a E se podiže na upražnjeno mesto od C.

Brisanja

- Operacija brisanja je jednostavna kada je ključ u listu.
- To je najčešći slučaj jer je broj listova u stablu najveći.

- Složenost: procesor $O(th)$, i $O(h)$ disk operacija.