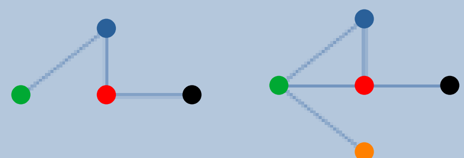
Napredni algoritmi i strukture podataka

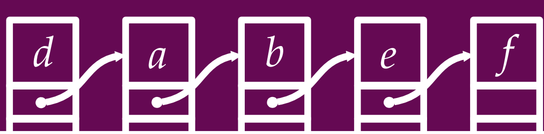
predavanja

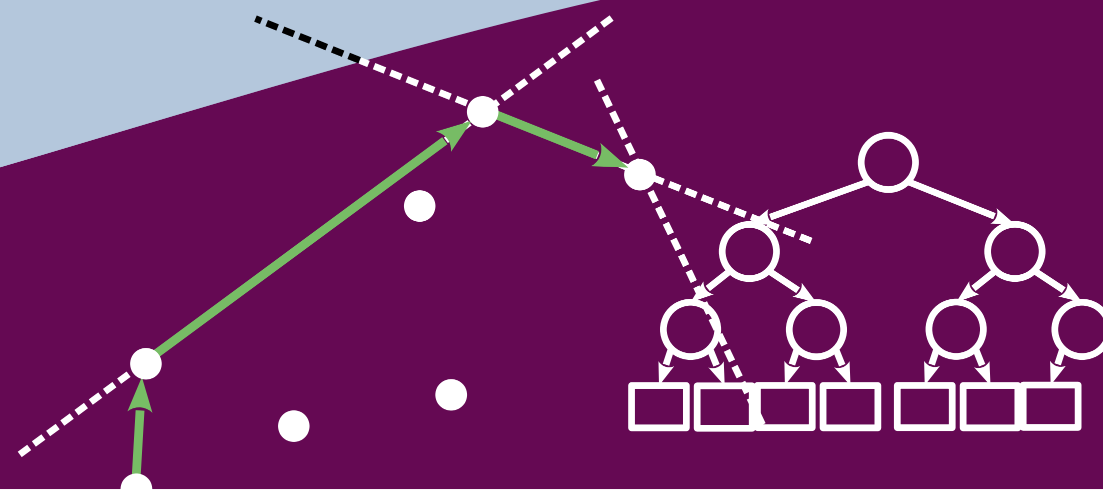
2021./2022.

B-stabla i RB-stabla

(B-trees, Red-Black trees)



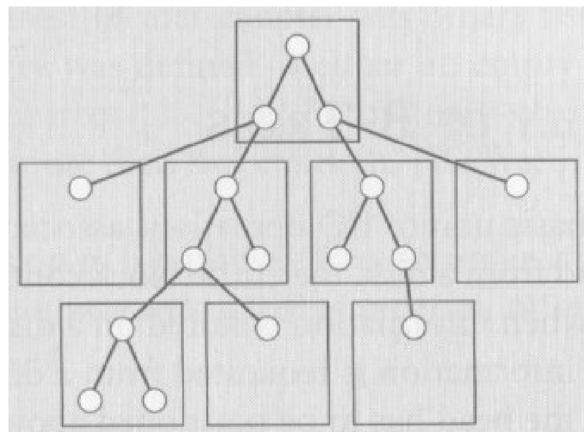




Creative Commons

Motivacija

- Vanjska memorija
  - Sekvencijalno čitanje po blokovima
  - Susjedni čvorovi u stablu mogu biti razasuti u udaljenim blokovima



- B-stabla ublažavaju efekte ograničenja sekvencijalnog blokovskog čitanja
  - Veličina čvora se prilagođava veličini bloka

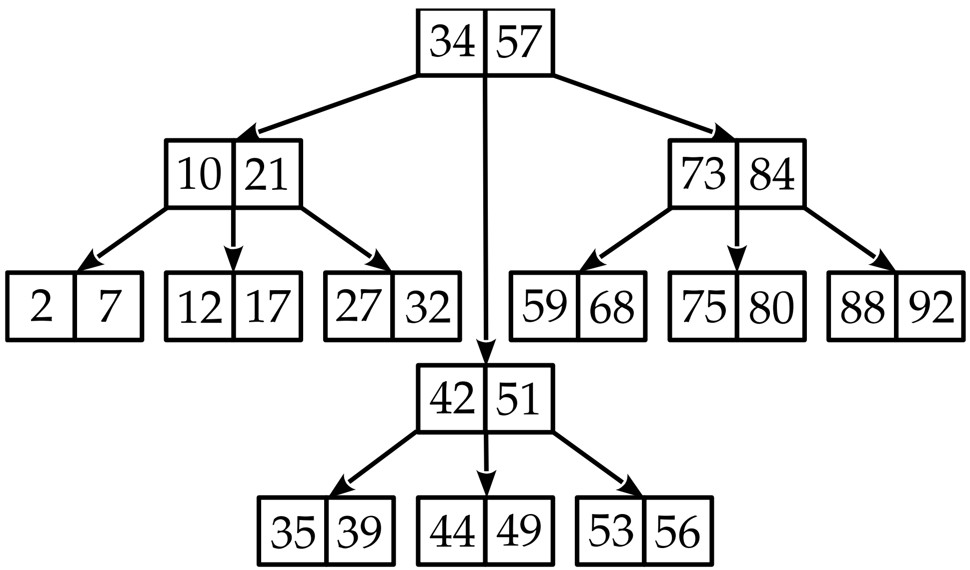
Karakteristike

- Potpuna balansiranost
- Sortiranje podataka po vrijednosti ključa
- Čuvanje određenog broja elemenata u jednom čvoru

M stabla

- M stabla: stabla u kojima čvorovi mogu imati proizvoljan broj djece

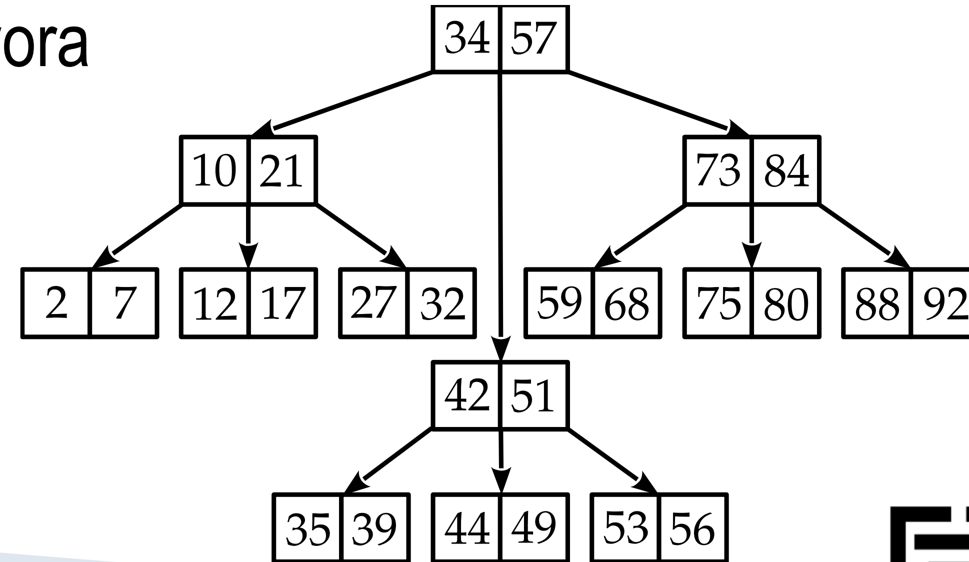
- M stablo *m*-tog reda: M stablo u kojem čvorovi mogu imati najviše *m* djece.



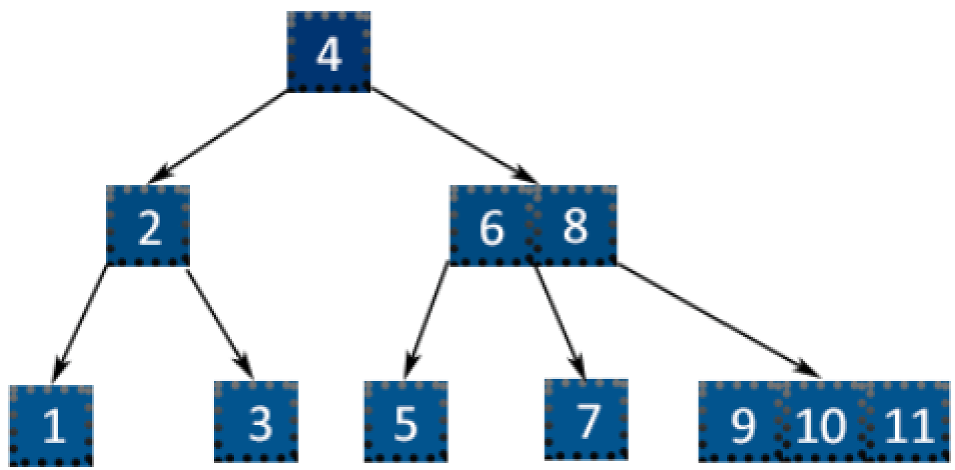
Dozvoljeno je pisati „m stablo” i „m-stablo”

M stabla

- Svojstva M stabla *m*-tog reda:
  1. svaki čvor ima najviše *m* djece i *m–1* podataka (ključeva)
  2. ključevi u čvorovima su sortirani
  3. ključevi u prvih *i* djece nekog čvora su manji od *i*-tog ključa promatranog čvora
  4. ključevi u zadnjih *m–i* djece nekog čvora su veći od *i*-tog ključa promatranog čvora



B-stabla



B-stablo

- B-stablo (*Balanced Tree*)

- B stablo je topološka struktura za pohranu i dohvaćanje informacija u obliku stabla u kojem su svi terminalni čvorovi na istoj udaljenosti od korijena, a svi neterminalni čvorovi imaju između n i 2n podstabla.
  - Ključevi za pretraživanje i podaci pohranjeni u unutarnjim i krajnjim čvorovima (listovima).

- Prostorna složenost O(n)
- Složenost pretraživanja O(log n)
- Složenost dodavanja O(log n)
- Složenost brisanja O(log n)

B-stablo

- B stablo *m*-tog reda je M-stablo sa dodatnim svojstvima:
  1. korijen ima najmanje dvoje djece, osim ako je ujedno i list (jedini čvor u stablu)

Iskorištenje ≥50% - unut.čv.

  2. svaki čvor, osim korijena i listova, sadrži **barem** $k-1$ ključeva i k pokazivača na podstabla (ima k djece), pri čemu je
$$\lceil m/2 \rceil \leq k \leq m$$

Iskorištenje ≥50% - listovi

  3. svi listovi sadrže **barem** $k-1$ ključeva, pri čemu je
$$\lceil m/2 \rceil \leq k \leq m$$

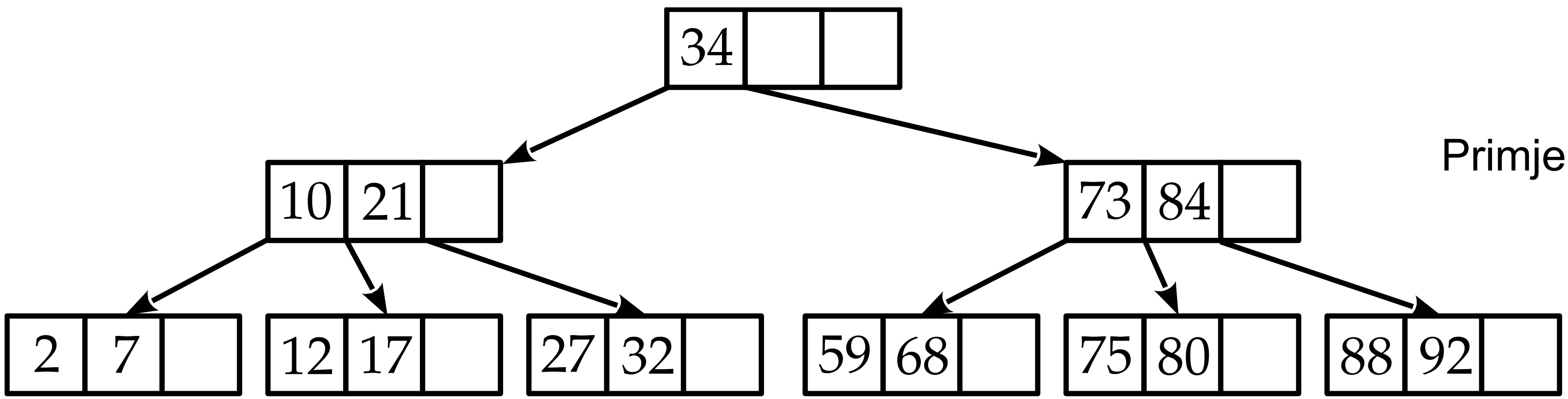
  4. svi listovi su na istoj razini

Savršena uravnoteženost

B-stablo

- Osobitosti
  - Popunjenost barem 50%
  - Savršeno uravnotežena

Uravnoteženo stablo (*Balanced Tree*) – stablo u kojem je razlika visina pod stabala svakog čvora najviše jedan. **Savršeno uravnoteženo stablo** (*Perfectly Balanced Tree*) – uravnoteženo stablo kojemu su svi listovi u najviše dvije razine.



Primjer→

https://www.cs.usfca.edu/~galles/visualization/BTree.html

B-stablo

- Implementacija
  - Struktura (klasa) s poljem od *m-1* ključeva i poljem od *m* pokazivača
  - Moguće dodati podatke radi lakšeg održavanja (npr. broj upisanih podataka u čvoru)

Algoritam pretraživanja B-stabla

1. Ući u čvor i redom pregledavati ključeve sve dok je trenutni manji od traženog, a još ima neprovjerenih
   - Prvi čvor u koji ulazi je korijen

2. Ako je 1. korak završio zbog nailaska na ključ veći od traženog ili zbog dolaska do kraja čvora, spusti se na razinu niže i ponovi prvi korak
   - Ako nema niže razine, nema traženog ključa

Pretraživanje B-stabla - implementacija

```
SearchBTree (key, node):
if (node != NULL):
   for (i=1; i<=node->keyNum && node->keys[i]<key;++i)
       if (i>node->keyNum || node->keys[i]>key):
         SearchBTree (key, node->pointers[i]);
       else:
         return node;
else:
   no searched key;
```

- Napomena
  - *keyNum* je članska varijabla čvora koja sadrži broj ključeva upisanih u čvor
  - polje *keys* je popis ključeva upisanih u čvor
  - polje *pointers* je polje pokazivača na djecu

Dodavanje podataka u B-stablo

- Jednostavnije je graditi B-stablo **odozdo prema gore**

- Algoritam:
  1. pronaći list u koji bi trebalo smjestiti novi podatak
  2. ako ima mjesta, upisati novi podatak
  3. ako je taj list pun, “rascijepiti” ga (napraviti novi list, ravnomjerno raspodijeliti elemente između dva čvora, a središnji element upisati u roditelja)
  4. ako je i roditelj pun, “rascijepiti” i roditelja (ponavljati proceduru iz koraka 3)
  5. ako je i korijen pun, “rascijepiti” ga i napraviti novi korijen

Dodavanje podataka u B-stablo

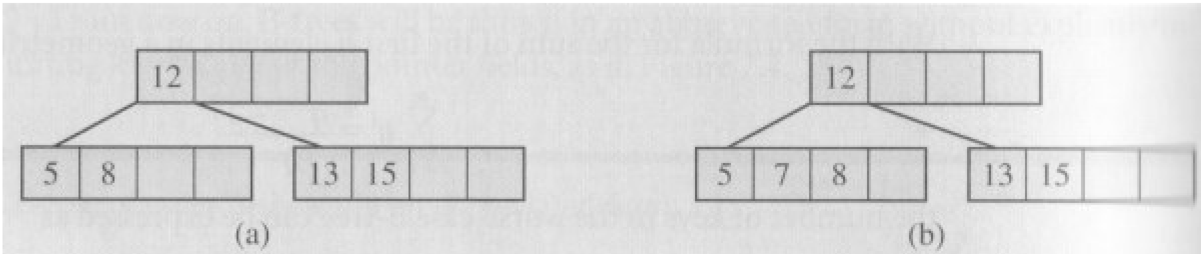
- Prilikom ubacivanja novog podatka moguće su 3 situacije:
  1. list u koji treba ići novi element nije pun
     - ubaciti novi element u taj list na odgovarajuće mjesto, pomičući po potrebi prethodni sadržaj
  2. list u koji treba ići novi element je pun, ali korijen stabla nije
     - list se dijeli (stvara se novi čvor) i svi elementi se ravnomjerno raspoređuju, s tim da se središnji element upisuje u roditelja
  3. list u koji treba ići novi element je pun, a isto tako i korijen stabla
     - kad se razdijeli korijen nastaju dva B-stabla koja treba sjediniti

Dodavanje podataka u B-stablo

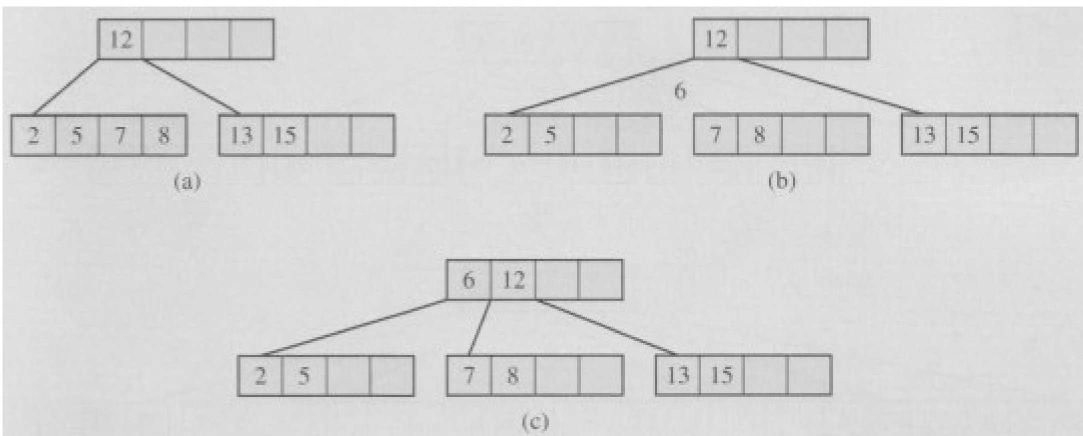
- Sjedinjenje u trećem slučaju se postiže stvaranjem još jednog čvora koji će biti novi korijen i upisivanjem središnjeg elementa u njega
  - To je jedini slučaj koji završava povisivanjem stabla
  - **B-stablo je uvijek savršeno uravnoteženo**

Primjer dodavanja podataka u B-stablo

- Primjer 1: dodavanje **7**



- Primjer 2: dodavanje **6**

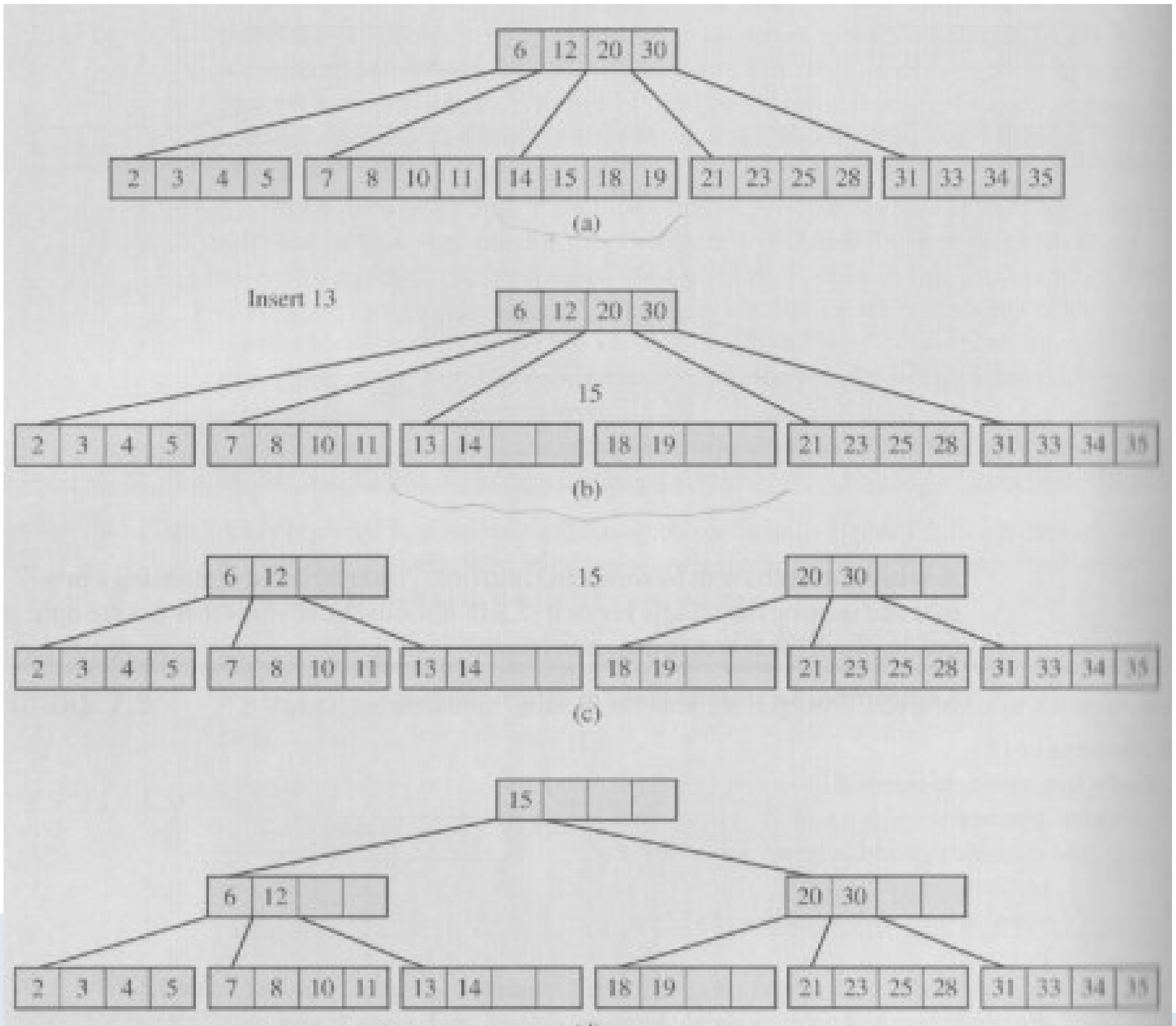


Max. degree = 5

https://www.cs.usfca.edu/~galles/visualization/BTree.html

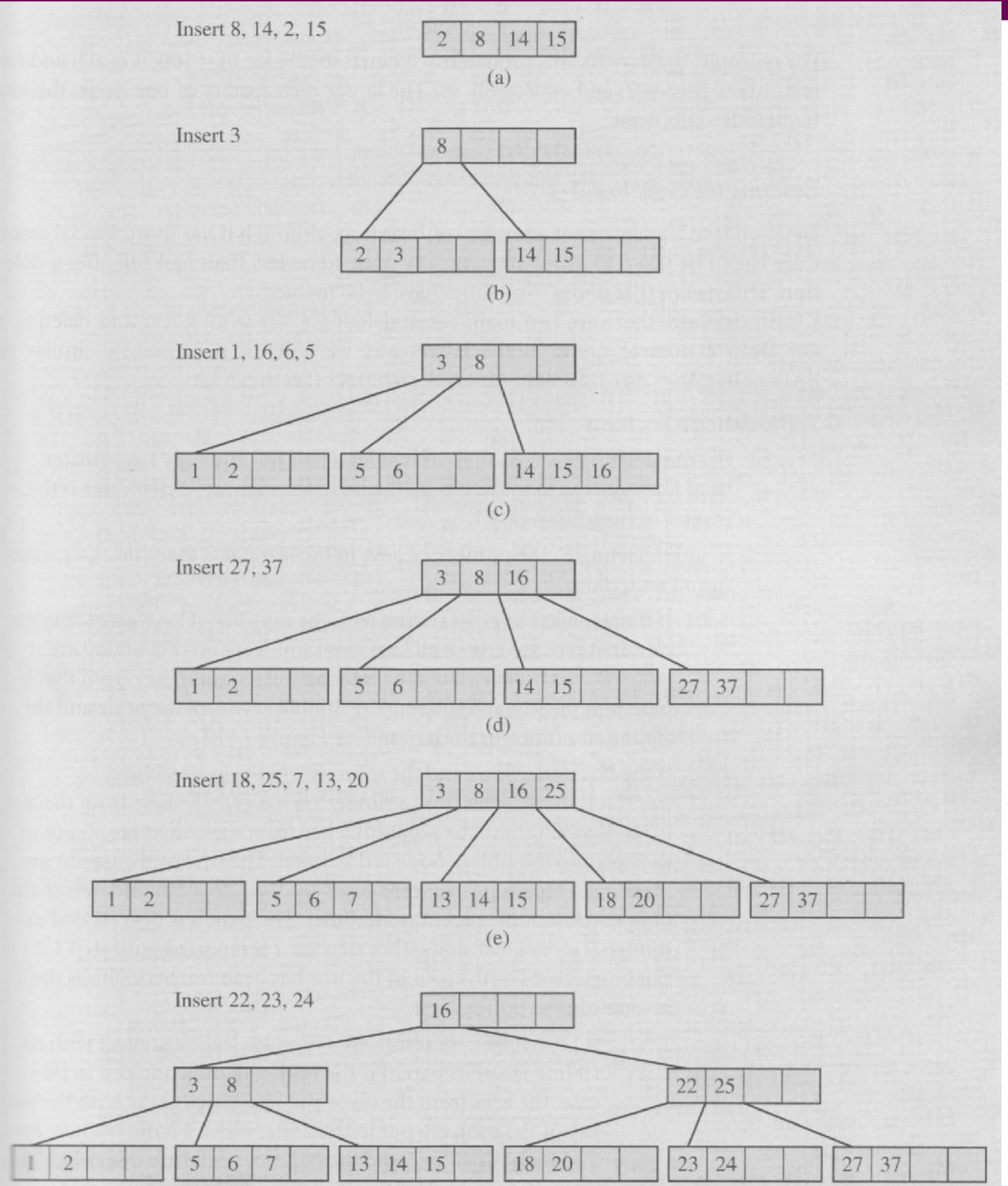
Primjer dodavanja podataka u B-stablo

- Primjer 3: dodavanje **13**

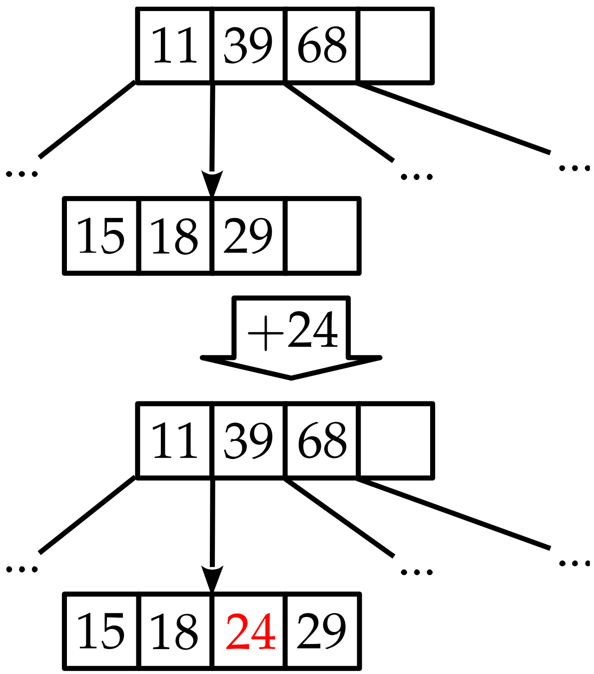


Primjer dodavanja podataka u B-stablo

- Redom se dodaju:
- **8,14, 2, 15, 3, 1, 16, 6, 5, 27, 37, 18, 25, 7, 13, 20, 22, 23 i 24**

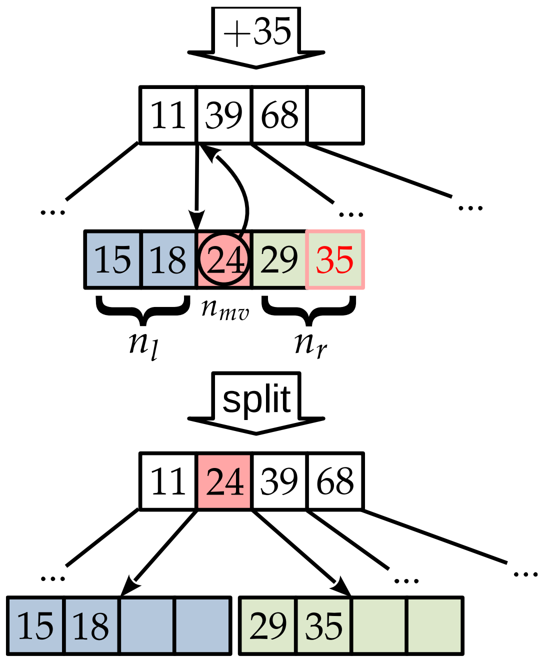


Primjer dodavanja podataka u B-stablo



- **Primjer 1**: dodajemo 24 u list u kojem ima manje od $m - 1$ ključeva. Nema potrebe za restrukturiranjem B-stabla.

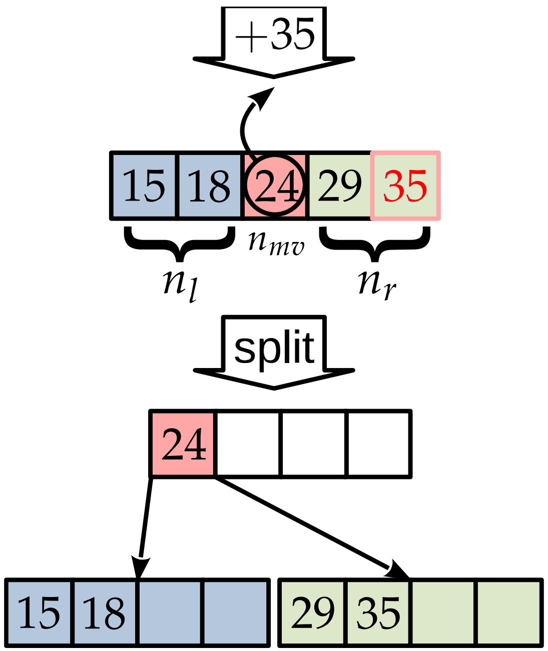
Primjer dodavanja podataka u B-stablo



- **Primjer 2**: dodajemo 35 u list u kojem ima točno $m - 1$ ključeva i koji ima roditeljski čvor.
  - Dešava se preljev u listu.
  - List se dijeli na središnji ključ i dva dijela.
  - Središnji ključ ubacujemo u roditelja, a list razdvajamo na dva čvora.

Max. degree = 5

Primjer dodavanja podataka u B-stablo

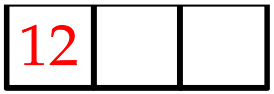


- **Primjer 3**: dodajemo 35 u čvor u kojem ima točno $m - 1$ ključeva i koji **nema** roditeljski čvor (očito je korijenski čvor).
  - Dešava se preljev u čvoru.
  - List se dijeli na središnji ključ i dva dijela.
  - Središnji ključ koristimo za stvaranje novog korijenskog čvora, a list razdvajamo na dva čvora.

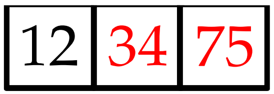
Max. degree = 5

Primjer dodavanja podataka u B-stablo

- B-stablo reda 4 – 4 kazaljke, 3 ključa
- Dodajemo redom ključeve: **12,75,34,62,19,25,66,30,33,71,47,21,15,23,27**



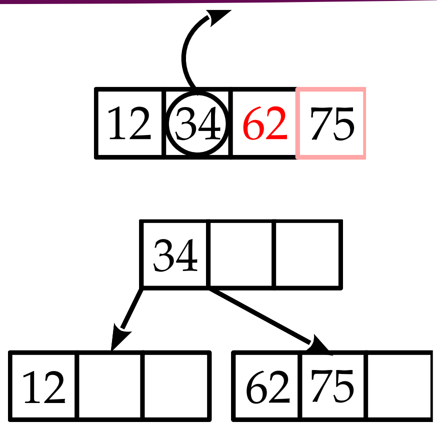
- **Korak 1**: Formiramo korijenski čvor s prvim ključem **12**



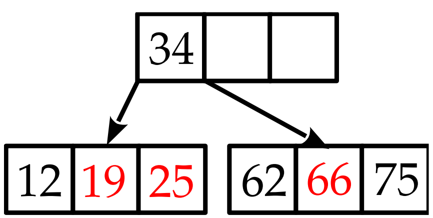
- **Korak 2**: U čvor dodajemo ključeve do dok možemo – dodajemo **75** i **34**

12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

Primjer dodavanja podataka u B-stablo



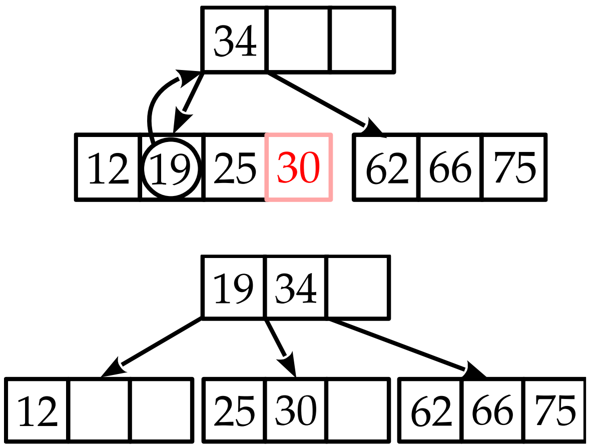
- **Korak 3**: Dodavanjem ključa **62**, dolazi do preljeva u korijenskom čvoru, kojeg razdvajamo uz stvaranje novog korijenskog čvora.



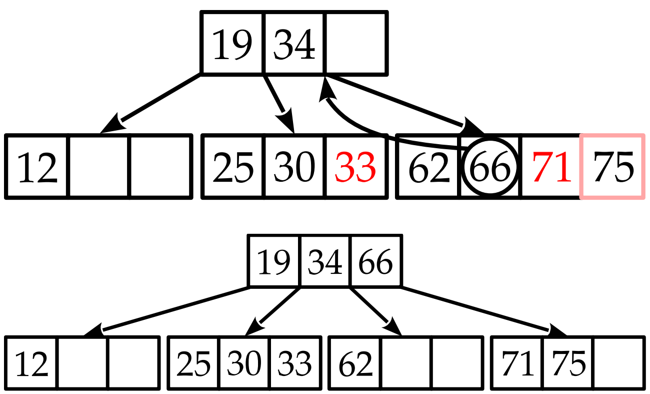
- **Korak 4**: Dodajemo ključeve **19**, **25** i **66** direktno u listove.

12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

Primjer dodavanja podataka u B-stablo



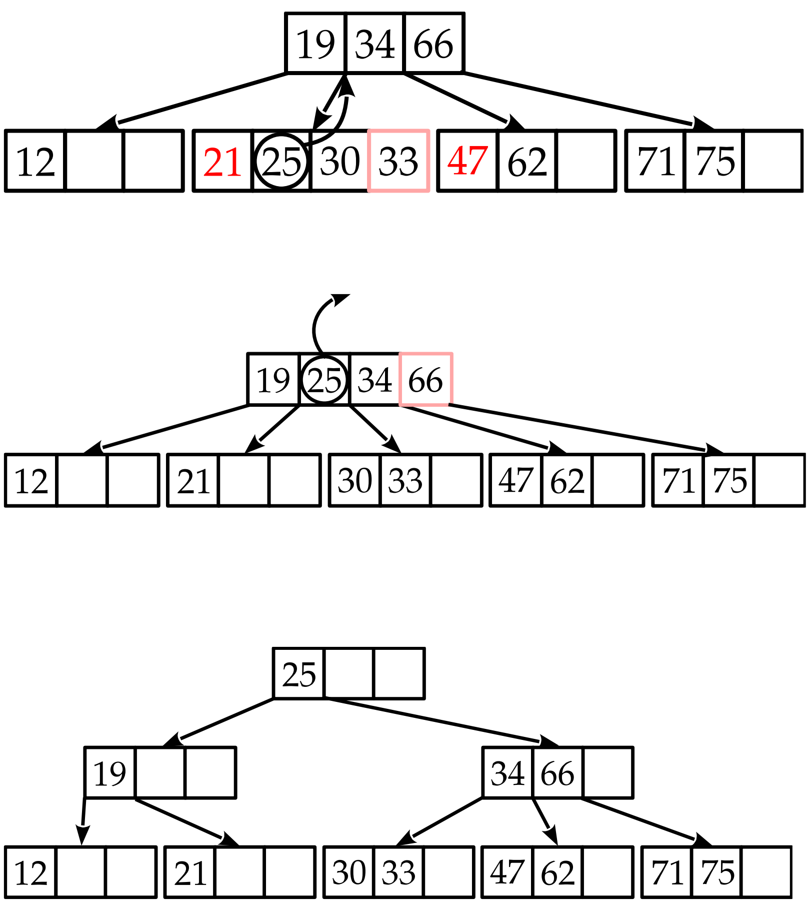
- **Korak 5**: Dodavanjem ključa **30**, dolazi do preljeva u lijevom listu, kojeg razdvajamo uz ubacivanje srednjeg ključa u korijenski čvor.



- **Korak 6**: Dodajemo ključ **33**, a zatim **71**. Dodavanjem **71** izazivamo preljev u desnom listu, kojeg razdvajamo uz ubacivanje srednjeg ključa u korijenski čvor.

12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

Primjer dodavanja podataka u B-stablo

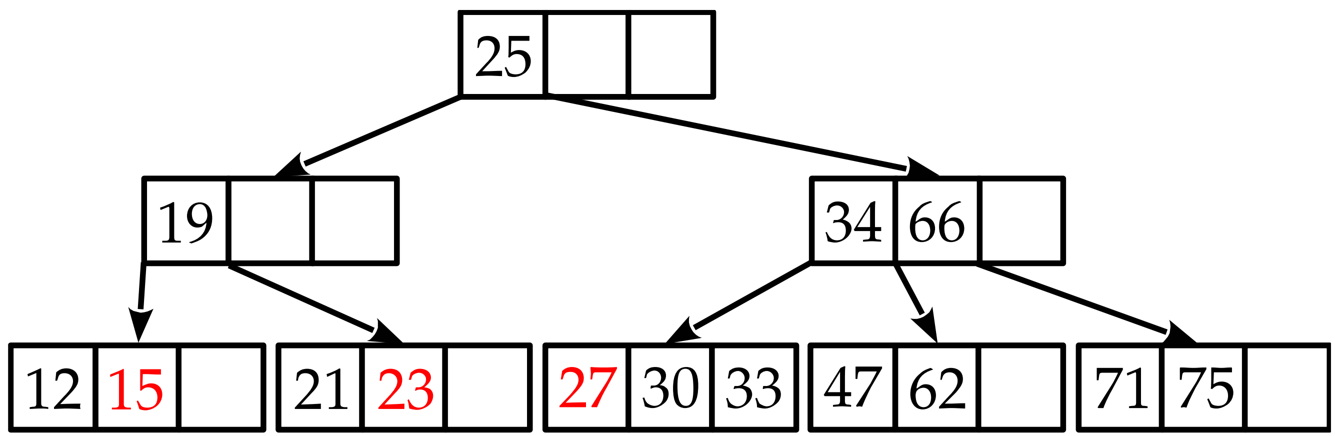


- **Korak 7**: Dodajemo ključ **47**, a zatim ključ **21**.
  - Dodavanjem **21** izazivamo preljev u listu, kojeg razdvajamo uz ubacivanje srednjeg ključa u korijenski čvor.
  - Ubacivanjem 25 u korijenski čvor izazivamo preljev korijenskog čvora, kojeg razdvajamo uz stvaranje novog korijenskog čvora.

12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

Primjer dodavanja podataka u B-stablo

- **Korak 8**: Dodajemo ključeve **15**, **23** i **27** direktno u listove B-stabla bez restrukturiranja.



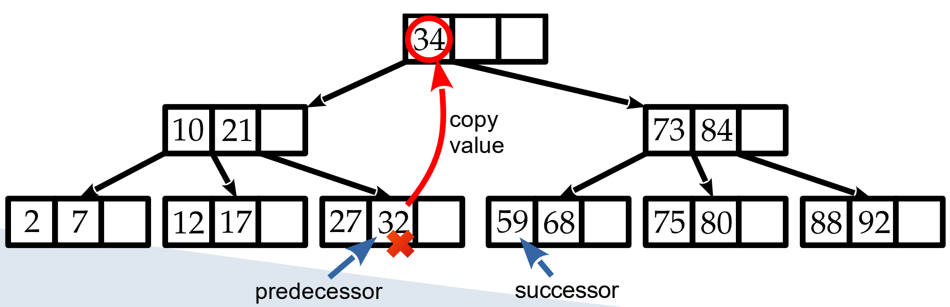
12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

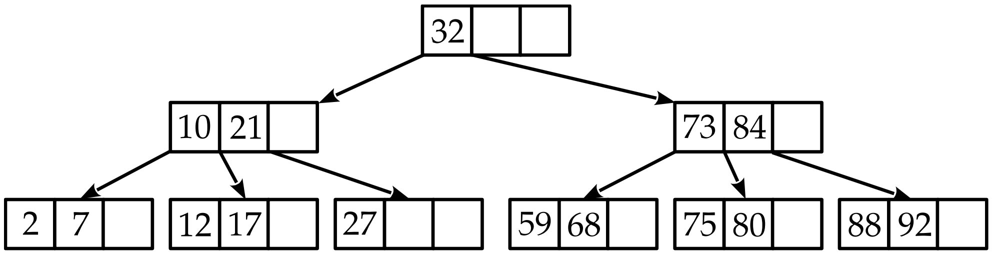
Implementacija dodavanja u B-stablo

```
BTreeInsert (K)
find a leaf node to insert K;
while (true):
find a proper position in array keys for K;
if node is not full:    //case 1
    insert K and increment keyNum;
    return;
else:                   //case 2, case 3
    split node into node1 and node2; // node1 = node, node2 is new
    distribute keys and pointers evenly between node1 and node2 and
initialize properly their keyNum's;
    K = middle key; // ideally, K is median
    if node was the root: // case 3
        create a new root as parent of node1 and node2;
        put K and pointers to node1 and node2 into the root, and set its
keyNum to 1;
        return;
    else:
        node = its parent // case 2; now process the node's parent
```

Brisanje podataka u B-stablu

- 2 slučaja:
  1. brisanje elementa u listu stabla
  2. brisanje elementa u čvoru
     - svodi se na brisanje elementa iz lista
       - na mjesto elementa koji treba izbrisati upisuje se njegov neposredni prethodnik (koji može biti samo u listu), potom se u listu briše prepisani element standardnim postupkom za brisanje lista

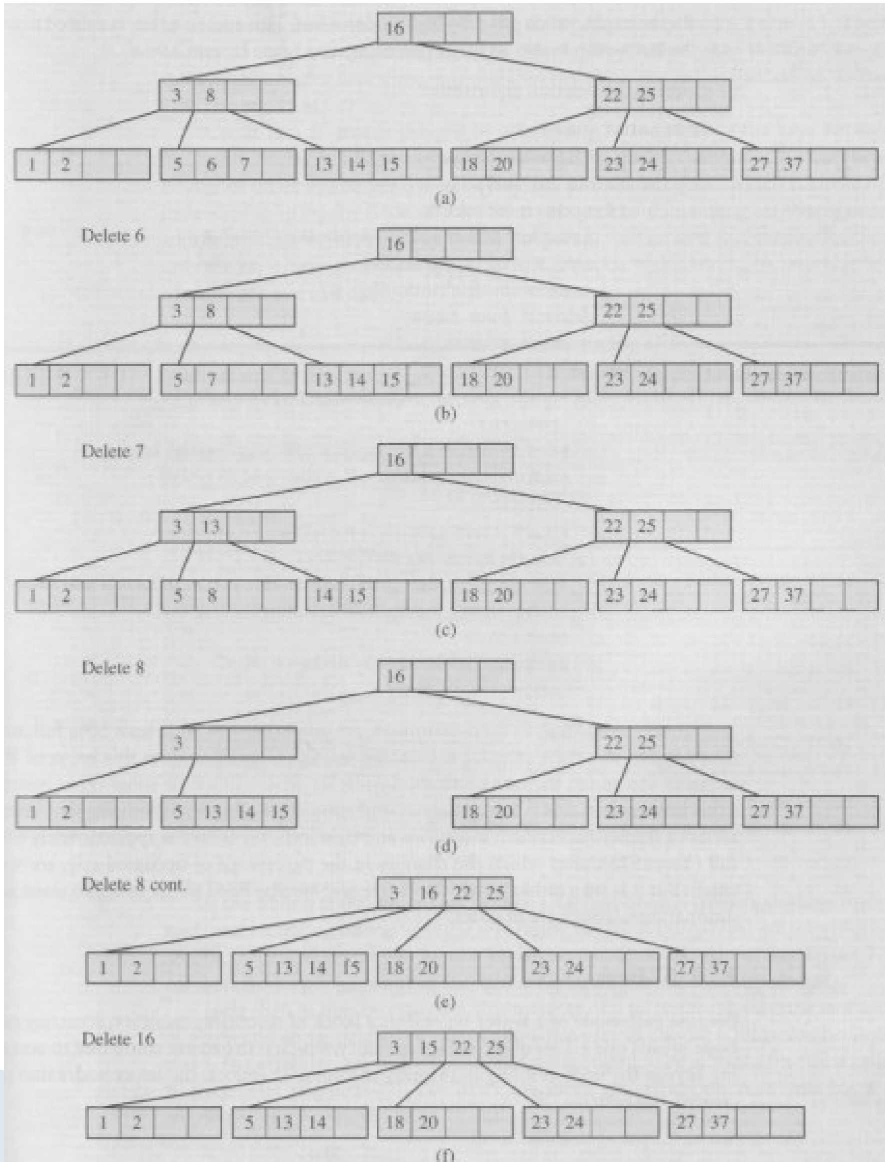




Brisanje podataka u B-stablu

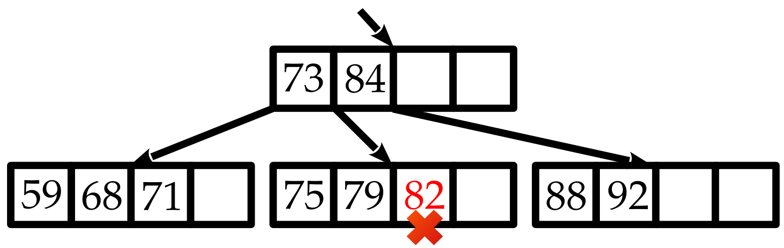
1. list i nakon brisanja elementa ima $\geq \lceil m/2 \rceil - 1$ ključeva; **KRAJ**
2. broj preostalih elemenata(ključeva) < $\lceil m/2 \rceil - 1$
    1. ako lijevo ili desno postoji susjed s > $\lceil m/2 \rceil - 1$ ključeva
        - elemente lista, elemente susjeda i središnji element iz roditelja ravnomjerno rasporediti u list i susjeda, a kao novi središnji element u roditelja upisati središnji element ujedinjenog skupa (unije) elemenata; **KRAJ**
    2. list i susjed se sjedinjuju (svi elementi lista i susjeda + središnji element iz roditelja se upisuju u list, a susjed se briše); **NASTAVITI s roditeljem**
    3. postupkom 2.2 dolazimo do korijena:
        - ako korijen ima više od 1 elementa: sjediniti trenutni čvor i susjeda kao u 2.2; **KRAJ**
        - inače: sve elemente lista, susjeda i korijena upisati u 1 čvor koji postaje novi korijen, a 2 čvora se brišu iz stabla; **KRAJ**

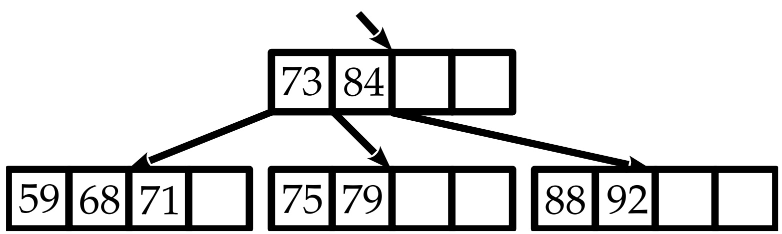
Primjer brisanja podatka u B-stablu



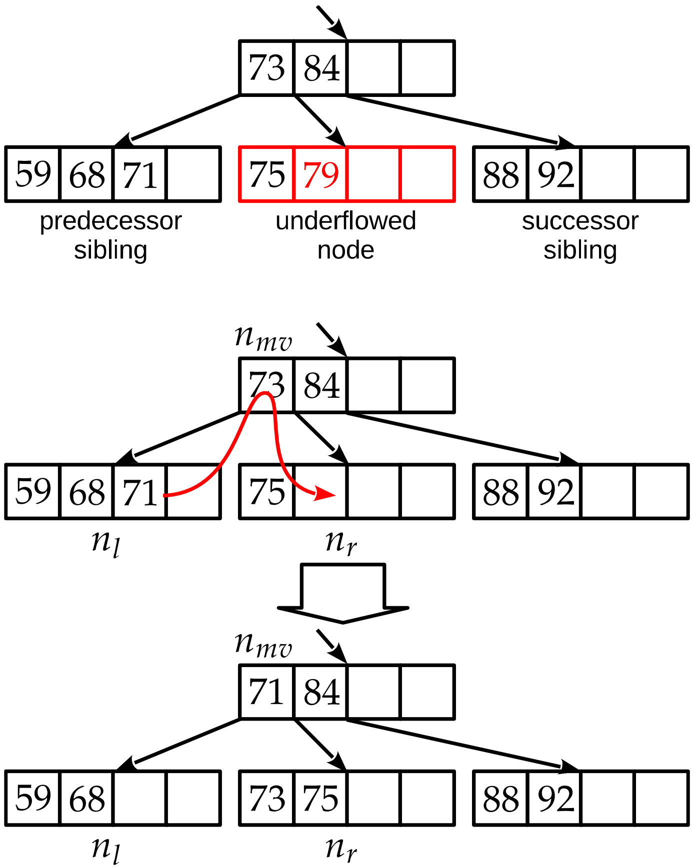
Primjer brisanja podataka iz B-stabla

- **Primjer 1**: brišemo ključ **82** iz lista. List je nakon brisanja još uvijek popunjen ≥50% zato jer ima ≥ ⌈m/2⌉-1 ključeva. Nema potrebe za restrukturiranjem B-stabla.



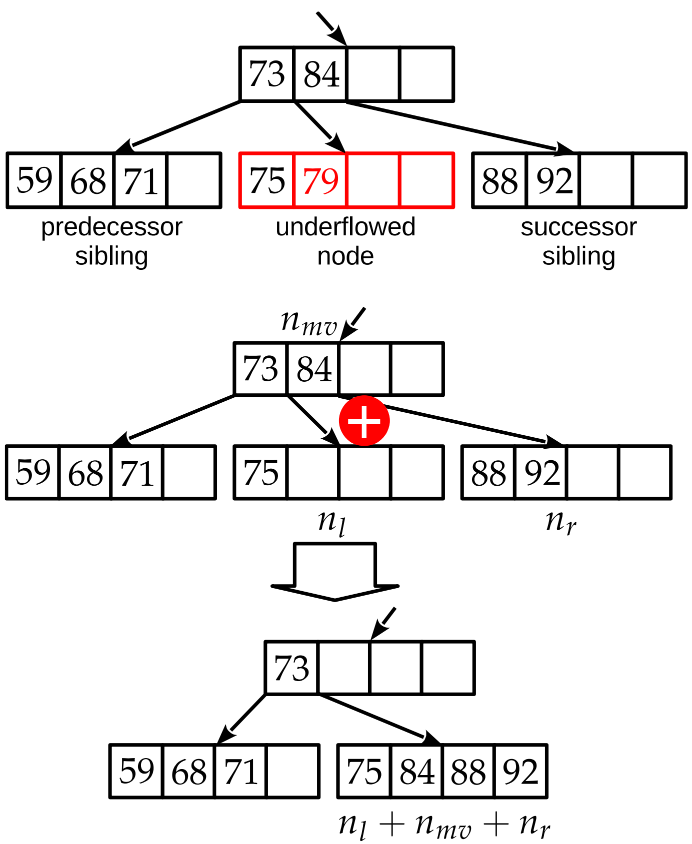


Primjer brisanja podataka iz B-stabla



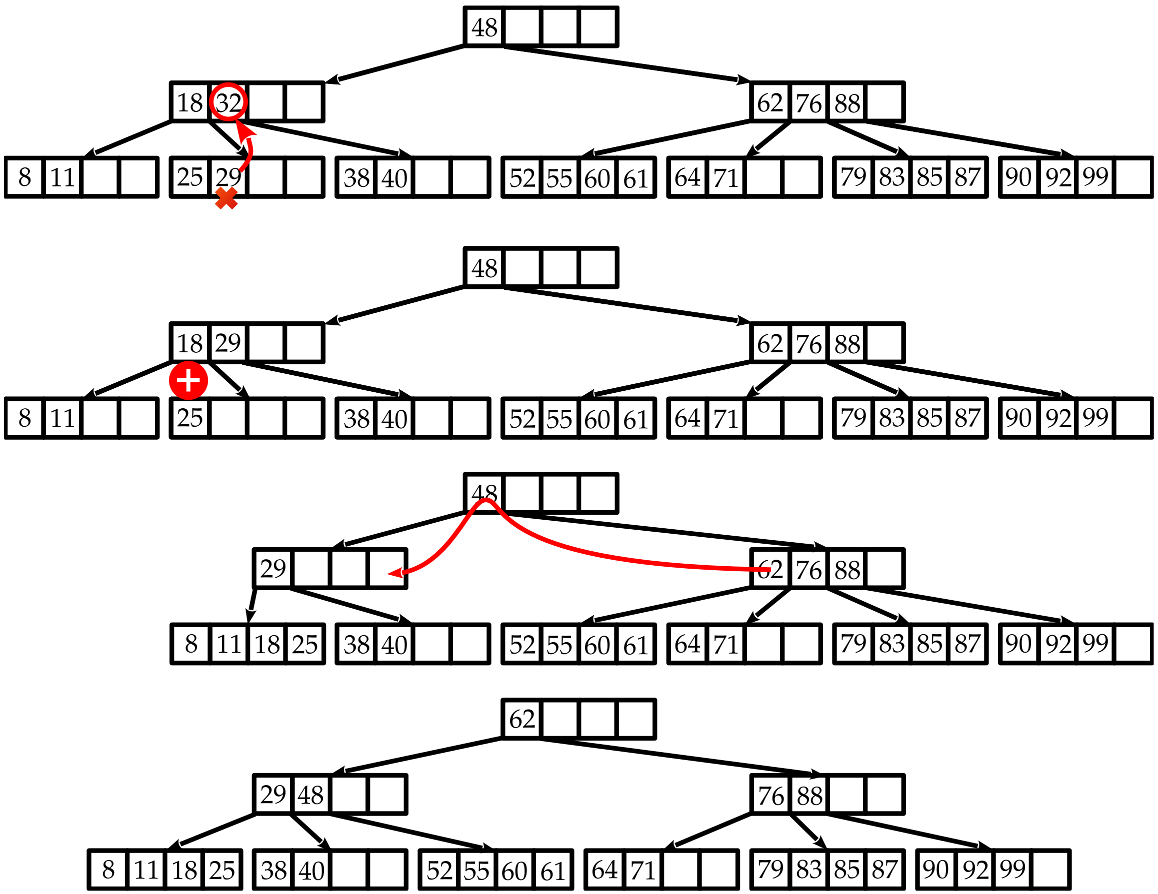
- **Primjer 2**: brišemo ključ **79** iz lista. List nakon brisanja nije više popunjen ≥50% zato jer ima < ⌈m/2⌉-1 ključeva.
- Gledamo li lijevog susjeda (blizanca), vidimo da je on popunjen > ⌈m/2⌉-1
  - Radimo restrukturiranje tako da prebacujemo ključeve iz lijevog susjeda u čvor koji je u podljevu
  - Pri tome pazimo na zajednički ključ u čvoru roditelja

Primjer brisanja podataka iz B-stabla



- **Primjer 3**: brišemo ključ **79** iz lista. List nakon brisanja nije više popunjen ≥50% zato jer ima < ⌈m/2⌉-1 ključeva.
- Gledamo li desnog susjeda (blizanca), vidimo da je on popunjen = ⌈m/2⌉-1
  - Radimo spajanje desnog susjeda i čvora koji je u podljevu (*underflow*)
- Primijetimo da je sada roditeljski čvor u podljevu, što moramo riješiti rekurzivno

Primjer brisanja podataka iz B-stabla

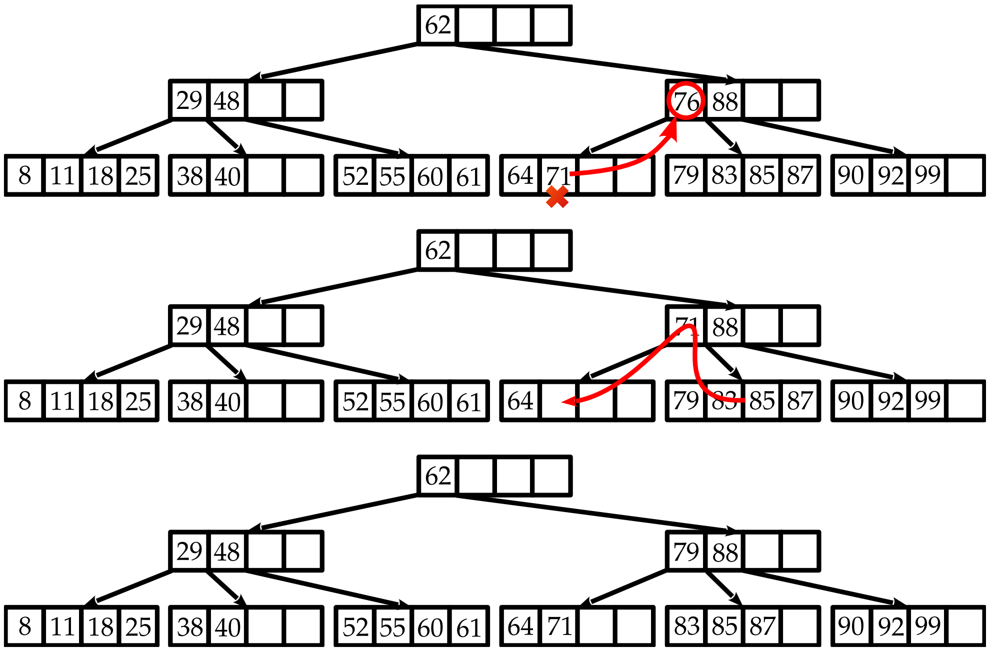


- Iz početnog B-stabla brišemo ključeve

  32, 76, 48, 25 i 11

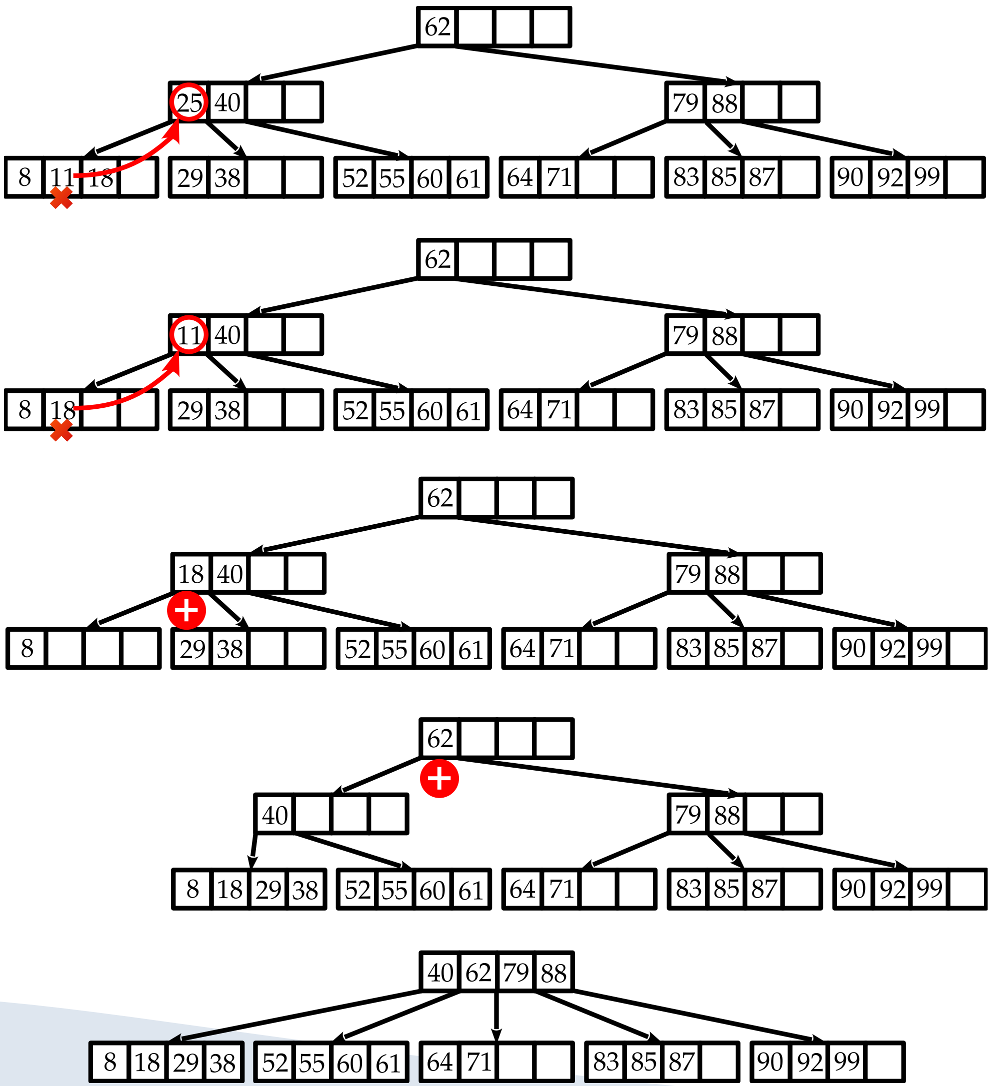
- **Korak 1**: brišemo ključ **32** postupkom kopiranja
  - List u podljevu spajamo s lijevim susjedom
  - To uzrokuje podljev unutarnjeg čvora
  - Restrukturiramo desnog susjeda i unutarnji čvor u podljevu

Primjer brisanja podataka iz B-stabla



- **Korak 2**: brišemo ključ **76** postupkom kopiranja. List u kojem smo obrisali zamjenski ključ je u podljevu.
    - Desni susjed do čvora u podljevu ima 100% popunjenost
    - Restrukturiramo desnog susjeda i čvor u podljevu

Primjer brisanja podataka iz B-stabla



- **Korak 3**: brišemo ključ **25** postupkom kopiranja, a zatim brišemo ključ **11** postupkom kopiranja. List u kojem smo obrisali zamjenske ključeve je sada u podljevu.
  - Desni susjed do čvora u podljevu ima točno 50% popunjenost, te čvor u podljevu spajamo s desnim susjedom
  - Sada je unutarnji čvor u podljevu, a njegov desni susjed ima točno 50% popunjenost, te čvor u podljevu spajamo s desnim susjedom
  - Prethodnim spajanjem nestaje stari korijenski čvor, a novi spojeni čvor postaje korijenski. Dubina stabla smanjuje se za 1.

32, 76, 48, 25 i 11

Implementacija brisanja elementa u B-stablu

```
BTreeDelete (K)
node = SearchBTree (K, root);          // naći čvor s podatkom za brisanje (K)
if (node != NULL):
    if node is not a leaf:
        find a leaf with the closest predecessor S of K;
        copy S over K in node;         // i.e. replace K with S in node
        node = the leaf containing S;  // redirect node
        delete S from node;            // delete S in leaf
    else:
        delete K from node
    while (node underflows):           // u protivnom imamo slučaj 1
        if the first left or right sibling of node has > ⌈m/2⌉-1 keys
            redistribute keys between node and its sibling;    // slučaj 2.1
            return;
    // u nastavku znamo da susjedi imaju točno granično elemenata (m/2 elemenata)
    else if node's parent is the root  // slučaj 2.2.1
        if the parent has only one key:
            merge node, its sibling, and the parent to form a new root;
            return;
        else:
            merge node and its sibling; // kao slučaj 2.2
            return;                    // samo bez nastavka
    else:
        merge node and its sibling;    // slučaj 2.2
        node = its parent;
```

Razlike B stabla i binarnog stabla

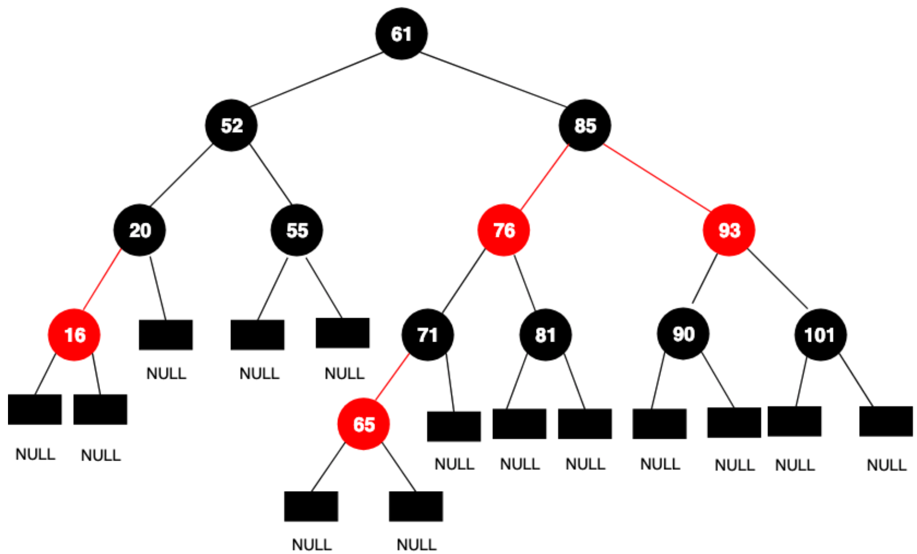
- B-stablo je unapređenje binarnog stabla.

- Za razliku od binarnog stabla, u B-stablu čvor može imati više od dvoje djece.
- Pretraživanje B-stabla slično je pretraživanju binarnog stabla, ali umjesto donošenja binarne ili "dvosmjerne" odluke o grananju na svakom čvoru, donosi se odluku o višestrukom grananju prema broju djece čvora.
- Umetanje ključa u B-stablo složenije je od umetanja ključa u binarno stablo pretraživanja. Temeljna operacija koja se koristi tijekom umetanja je razdvajanje punog čvora oko njegovog medijalnog ključa (srednji broj na popisu brojeva poredanih od najmanjeg do najvećeg) na dva čvora. Podjela ili cijepanje je način na koji B-stablo raste.

Razlike B stabla i binarnog stabla

- Za razliku od binarnog stabla, B-stablo povećava visinu na vrhu umjesto na dnu.
- Budući da je svaki čvor roditelj može imati više podređenih čvorova djece, B-stabla ne trebaju ponovno balansiranje tako često kao druga samobalansirajuća stabla, ali mogu neefikasno koristiti memoriju ako čvorovi nisu potpuno puni.
- Zahtijevajući da svi čvorovi listova budu na istoj dubini, B-stablo se održava uravnoteženim. Dubina će se povećavati kako se stablu dodaje više elemenata, ali povećanje ukupne dubine nije često i rezultira time da su svi čvorovi lista još jedan čvor udaljeniji od korijena. Visina stabla smanjuje se povećanjem broja ključeva unutar svakog unutarnjeg čvora. Također, smanjuje se broj skupih pristupa čvorovima, a rebalans stabla događa se rjeđe.

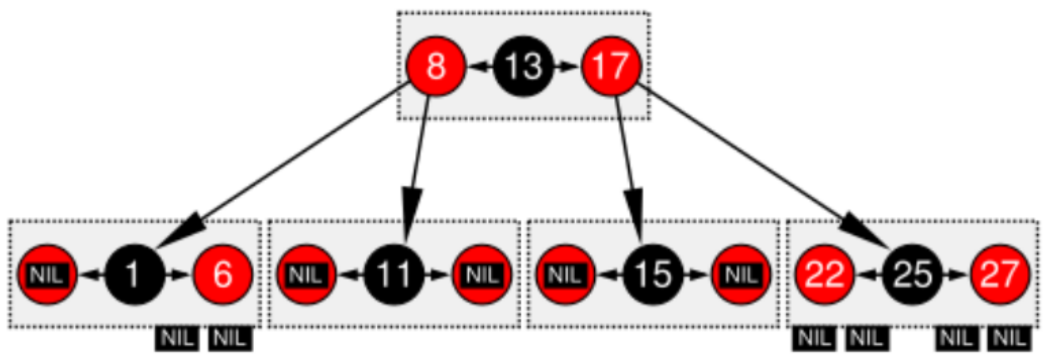
Crveno-crna stabla



Crveno-crno stablo (*Red-black tree*)

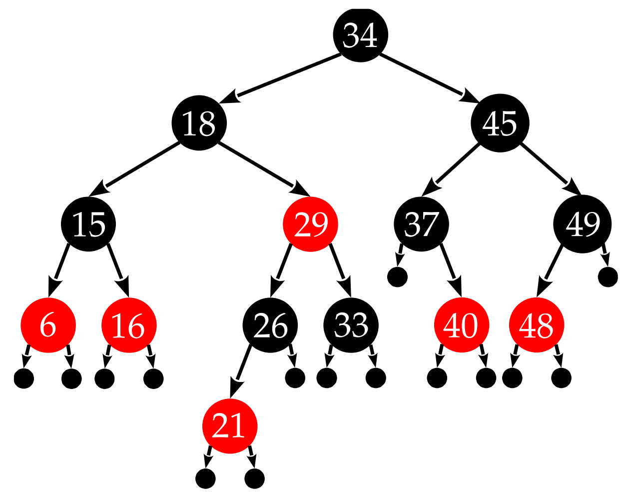
- Binarno stablo koje **idejno** proizlazi iz B-stabla 4. reda ako mu se elementi čvorova smatraju obojanima prema strogim pravilima

- Usporedba s B-stablom:
  - manji utrošak memorije
  - zadržava uravnoteženost
  - složenost ista



Definicijska pravila

1. svaki čvor je **crven** ili **crn**
2. korijen je **crn** (neobavezno, ali uobičajeno)
3. svaki list* je **crn**
4. oba potomka crvenog čvora su **crna**
5. svaka staza od nekog čvora do (bilo kojeg) lista koji je njegov potomak prolazi istim brojem crnih čvorova



- *Listovi u crveno-crnom (RB) stablu ne sadrže informacije pa ne moraju ni postojati, nego roditelji mogu imati NULL pokazivače ili svi pokazivati isti poseban čvor, sentinel

Crvena i crna visina stabla

- Razlikujemo crvenu i **crnu** visinu stabla:
  - rh(x), bh(x)
    - broj čvorova određene boje na putu od čvora x do lista koji mu je potomak (x se ne broji).

- Ključno svojstvo za uravnoteženost RB stabla:
  - najduži put od korijena do nekog lista najviše je dvostruko duži od najkraćeg puta od korijena do nekog (drugog) lista
  - tj. najduži put je najviše dvostruko duži od najkraćeg.

Teorem

- Visina RB-stabla s n unutarnjih čvorova je
  - $h \leq 2\log_2(n+1)$
  - Dokaz:
    - Binarno stablo visine *h* ima najviše $n = 2^h - 1$ čvorova
    - Zbog 4. pravila, barem polovica visine je crna visina pa je *hb* ≥ *h*/2. Budući da je n veći ili jednak broju crnih čvorova na putu od korijena do najnižeg lista, slijedi:
      - $n \geq 2^{hb} - 1 \geq 2^{\frac{h}{2}} - 1$, a iz toga izravno $h \leq 2\log_2(n+1)$
- Pretraživanje binarnog stabla je složenosti O(h) pa je složenost pretraživanja RB-stabla $O(\log_2 n)$

Dodavanje čvora u RB-stablo

- Radi lakše analize, uvode se pojmovi
  - čvor-ujak (uncle) koji se označava s **U**, a znači blizanac roditelja promatranog čvora (roditeljev brat/sestra)
  - čvor-djed koji se označava s **G** (grandparent), a znači roditelj roditelja

1. ubaciti novi čvor kao u svako drugo binarno search stablo i pridijeliti mu crvenu boju
2. restrukturirati stablo (primjenom rotacija i bojanjem čvorova) da bi zadovoljilo definicijska pravila

Dodavanje čvora u RB-stablo

- Definicijska pravila 1, 3 i 5 su uvijek zadovoljena kod dodavanja novog čvora, a 2 i 4 mogu biti ugrožena (ne istodobno) na sljedeće načine:
  - pravilo 2 ako je novi čvor korijen
  - pravilo 4 ako je roditelj novog čvora crven

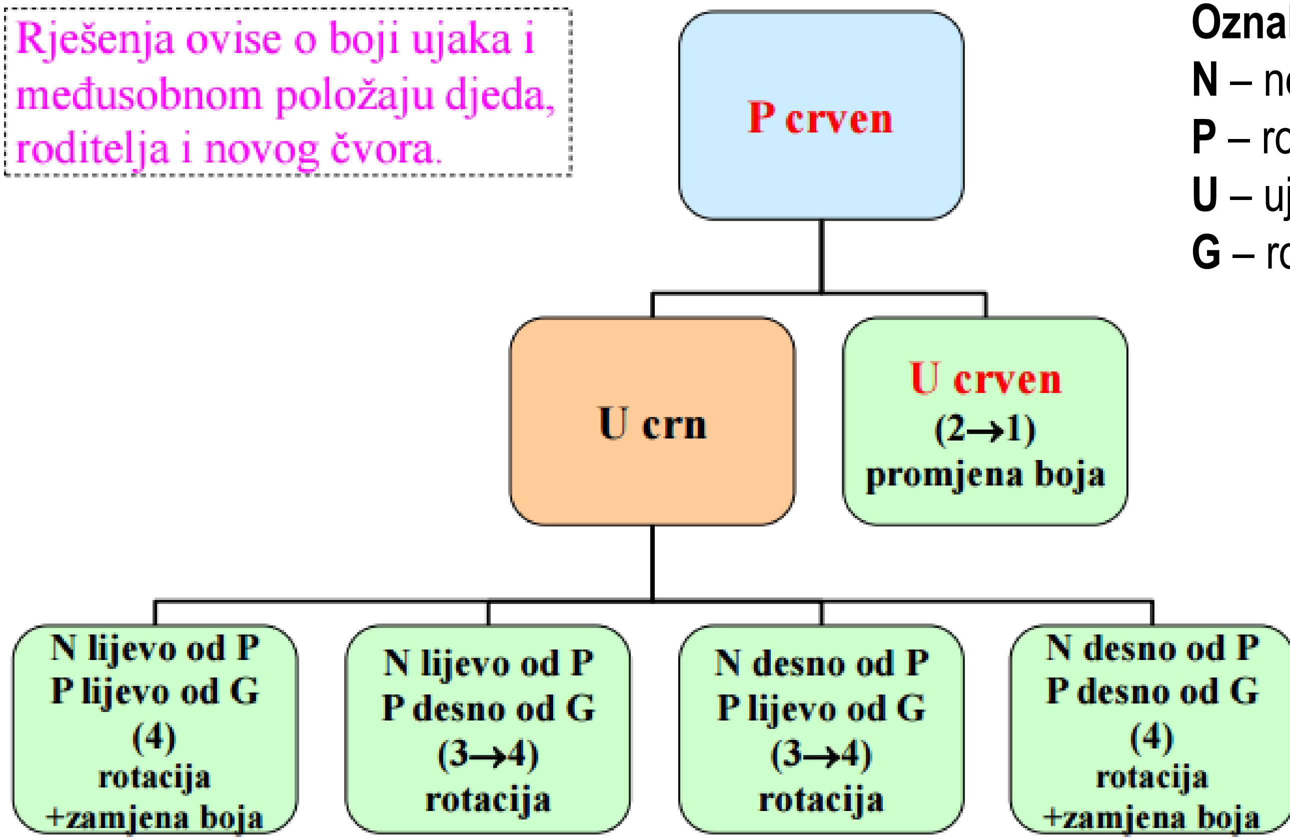
    - U oba slučaja potrebno je restrukturiranje

- Restrukturiranje:
  1. novi čvor je korijen:
     - prebojati ga u **crno** (5. pravilo ostaje zadovoljeno jer je to dodatni crni čvor u svim putevima u stablu)

Dodavanje čvora u RB-stablo

- Restrukturiranje:
  2. roditelj novog čvora je crven (korak 1):

Rješenja ovise o boji ujaka i međusobnom položaju djeda, roditelja i novog čvora.



Oznake:

N – novi čvor

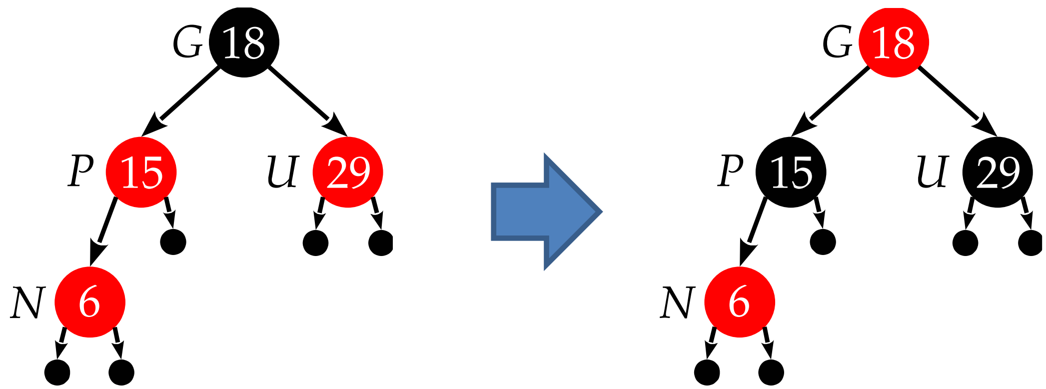
P – roditelj od N

U – ujak (blizanac od P)

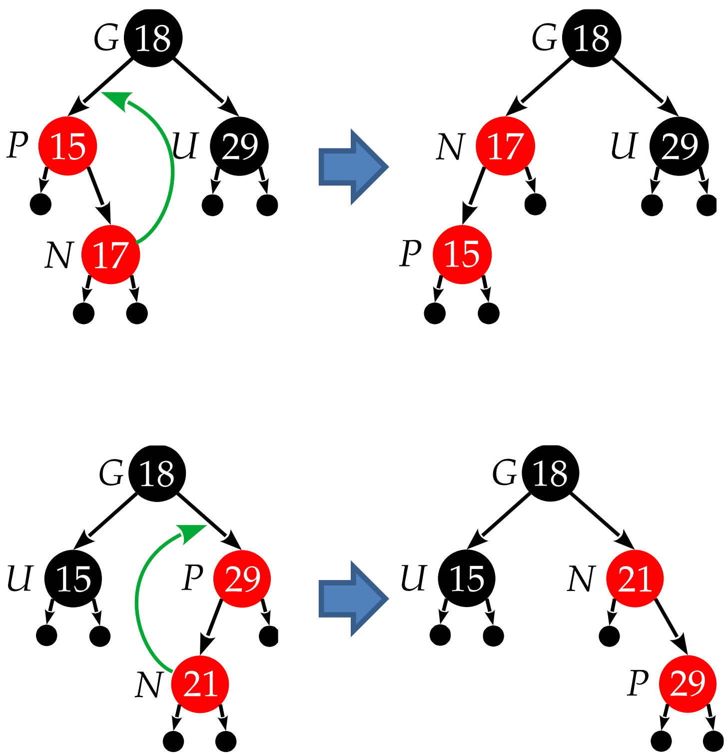
G – roditelj od P

Dodavanje čvora u RB-stablo

2. Roditelj i ujak su crveni
   - Narušeno 4. pravilo (nanizana dva crvena; P i N)
     - prebojati P i U u crno (rješava 4. pravilo), a G u crveno (očuvanje 5. pravila) - sada G može narušavati 4. pravilo ako ima crvenog roditelja ili 2. pravilo ako je korijen
     - **Nastaviti** s provjerom promatrajući G kao novi čvor (N)

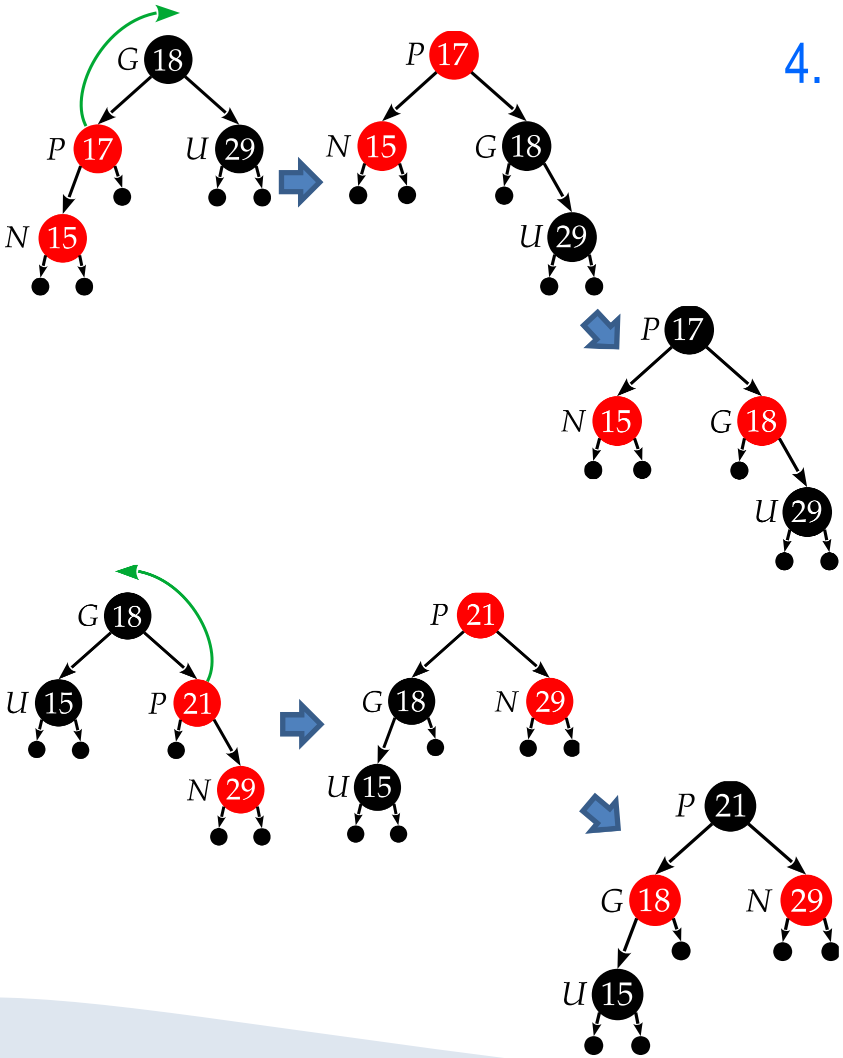


Dodavanje čvora u RB-stablo



3. Roditelj crven i ujak crn ("izlomljeni" poredak N, P i G)
   - Dva simetrična slučaja:
     - N desno dijete od P i P lijevo dijete od G
     - N lijevo dijete od P i P desno dijete od G
   - Rješenje:
     - *rotacija* N oko P, čime se stanje prevodi u "izravnati poredak" N, P i G koji se rješava u 4. provjeri
     - **Nastaviti** s provjerom (4), pridajući P-u ulogu N-a

Dodavanje čvora u RB-stablo

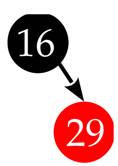


4. Roditelj crven i ujak crn (“linijski” poredak N, P i G)
   - Dva simetrična slučaja:
     - N lijevo dijete od P i P lijevo dijete od G
     - N desno dijete od P i P desno dijete od G
   - Rješenje:
     - *rotacija* P oko G
     - *zamjena boja* P i G (znamo da je G crn jer u protivnom P ne bi mogao biti crven); **KRAJ**

Primjer dodavanja podataka u RB-stablo

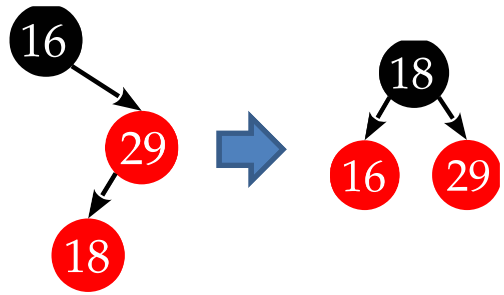
- Dodajemo redom ključeve: **16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40**

- **Korak 1**: Formiramo korijenski čvor s prvim ključem **16**. Nakon dodavanja, čvor je crveni, pa ga pretvorimo u crni (pravilo 2).

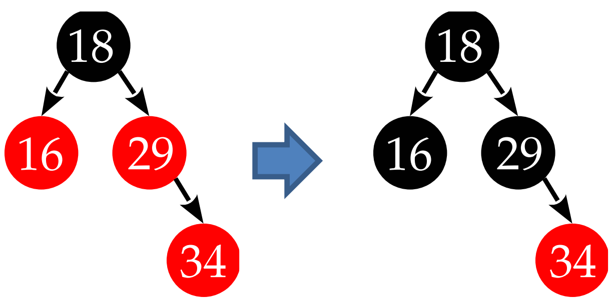


- **Korak 2**: U stablo dodajemo ključ **29**. Nema restrukturiranja RB-stabla.

Primjer dodavanja podataka u RB-stablo



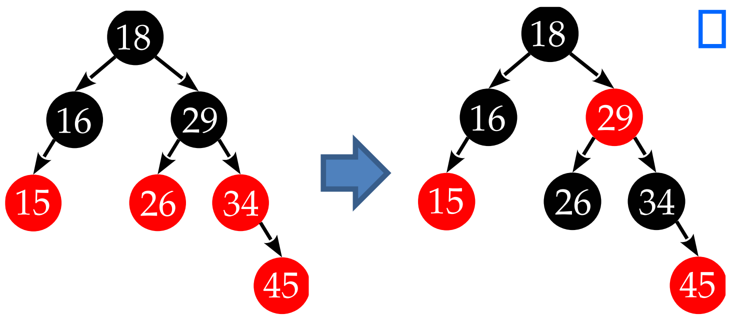
- **Korak 3**: U stablo dodajemo ključ **18**.
  - **Slučaj 3**: Lijeva rotacija **18** oko **29**
  - **Slučaj 4**: Desna rotacija **18** oko **16** + zamjena boja.



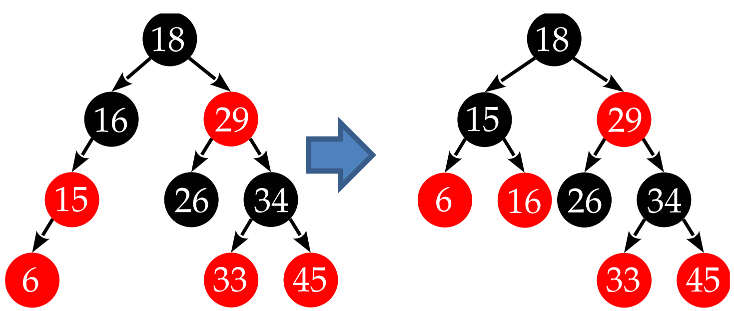
- **Korak 4**: U stablo dodajemo ključ **34**.
  - **Slučaj 2**: Postavi **18** u crveno, a **16** i **29** u crno
  - **Slučaj 1**: Postavi korijen **18** u crno

16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40

Primjer dodavanja podataka u RB-stablo



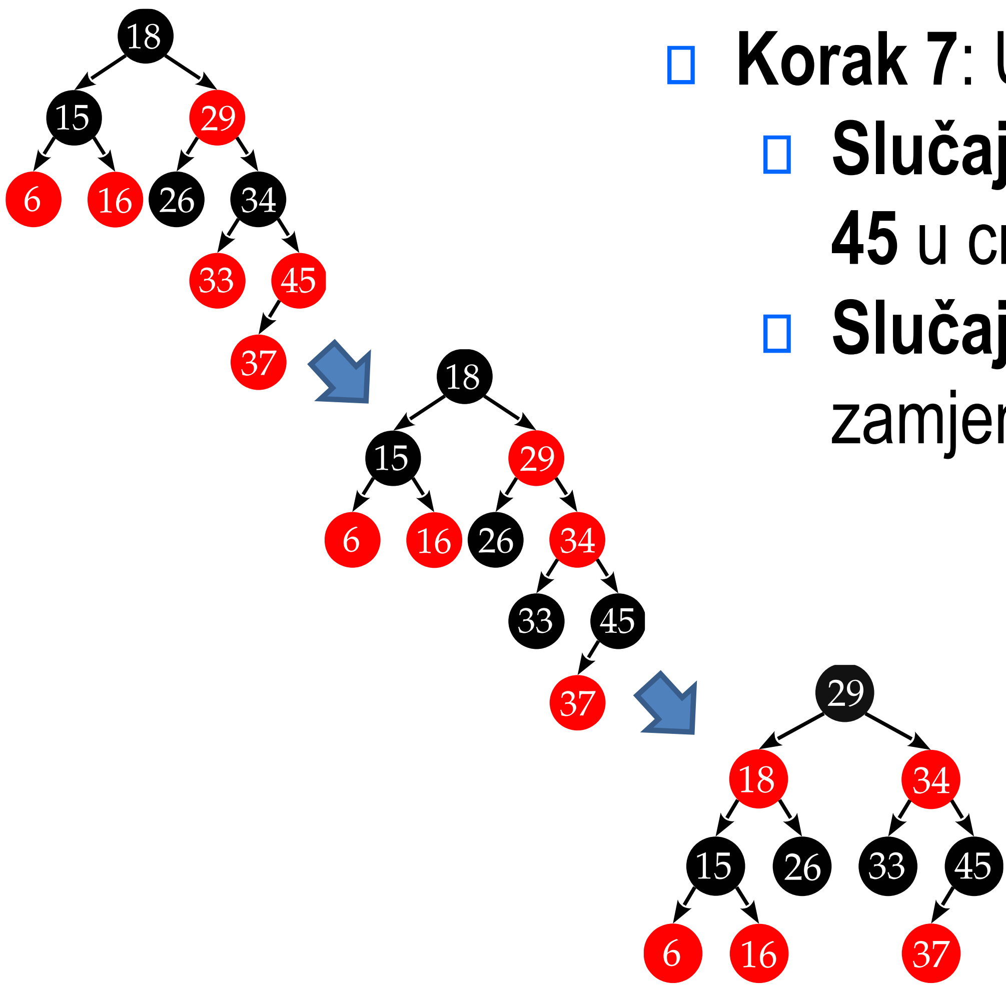
- **Korak 5**: U stablo dodajemo ključeve **26**, **15** i **45**.
  - Nakon dodavanja **45** imamo **slučaj 2**: Postavi **29** u crveno, a **26** i **34** u crno.



- **Korak 6**: U stablo dodajemo ključeve **33** i **6**.
  - Nakon dodavanja **6** imamo **slučaj 4**: desna rotacija **15** oko **16** i zamjena boja između **15** i **16**

16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40

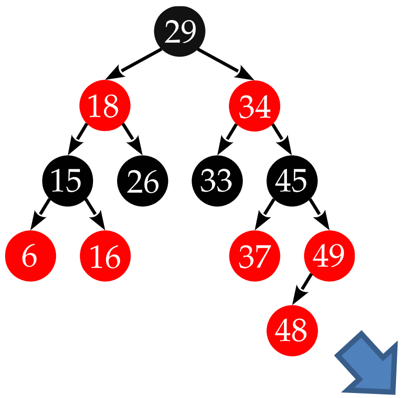
Primjer dodavanja podataka u RB-stablo

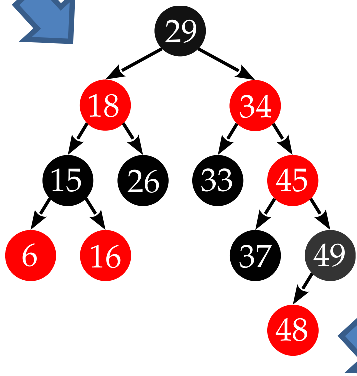


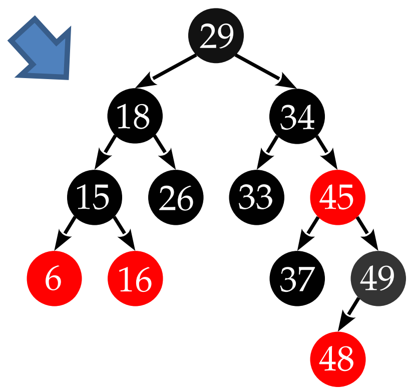
- **Korak 7**: U stablo dodajemo ključ **37**.
  - **Slučaj 2**: Postavi **34** u crveno, a **33** i **45** u crno.
  - **Slučaj 4**: Lijeva rotacija **29** oko **18** i zamjena boja **18** i **29**.

16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40

Primjer dodavanja podataka u RB-stablo





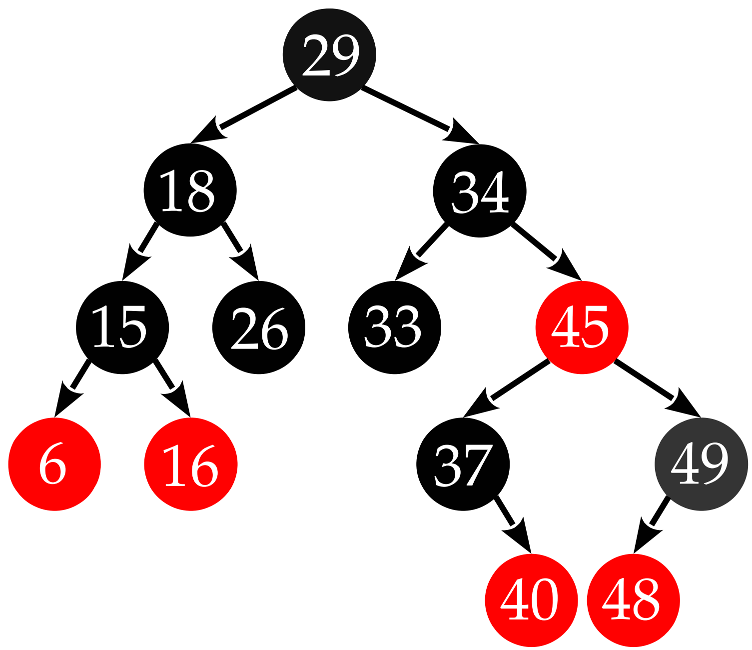


- **Korak 8**: U stablo dodajemo ključ **49**, te nakon toga ključ **48**.
  - **Slučaj 2**: Postavi **45** u crveno, a **37** i **49** u crno.
  - **Slučaj 2**: Postavi **29** u crveno, a **18** i **34** u crno.
  - **Slučaj 1**: Postavi korijen **29** u crno.

16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40

Primjer dodavanja podataka u RB-stablo

- **Korak 9**: U stablo dodajemo ključ **40**



16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40

Implementacija dodavanja čvora u RB-stablo

u proceduri

na slajdovima

```
//it is assumed that an ordinary inserting routine already inserted node N as a red leaf
while colour(p(N)) is red                           //p(N) = parent of N
    if p(N) is left child of g(N)
        U = right child of g(N);
        if colour(U) = red
        {   colour(p(N)) = black;                    //case 1 (2)
            colour(U) = black;                       //case 1 (2)
            colour(g(N)) = red;                      //case 1 (2)
            N = g(N);     }                          //case 1 (2)
        else
        {   if N is right child of P                 //P=p(N), G=g(N)
            {     left-rotate N about P;             //case 2 (3)
                  N = P;   }                         //case 2 (3)
            right-rotate p(N) about g(N);            //case 3 (4)
            colour[P] = black;                       //case 3 (4)
            colour[G] = red;     }                   //case 3 (4)
    else    //the same as above, with left and right exchanged
colour(root) = black;             //rješava i novi==korijen
```

Brisanje čvora u RB-stablu

- Algoritam:
  1. Brisanje kopiranjem (zamjenski čvor; u nastavku oznaka X)
  2. Ukloniti zamjenski čvor; on može imati najviše jedno dijete pa je problem pojednostavnjen

- Ako je zamjenski čvor:
  - crven: svojstva RB-stabla nisu narušena, postupak je gotov
  - **crn**: složeniji postupak

Uklanjanje crnog čvora

- 3 su moguća problema nakon uklanjanja crnog čvora:
  1. ako je uklonjen korijen, mogao je imati samo jedno dijete (N) koje postaje novi korijen, a ono može biti i crveno
     - povreda 2. pravila (korijen je crn)
  2. nakon uklanjanja X, njegovo dijete N i roditelj P su u odnosu dijete-roditelj i ako su oboje crveni
     - povreda 4. pravila (djeca crvenog su crna)
  3. uklanjanje crnog X znači smanjenje crne visine svih njegovih prethodnika (predaka)
     - povreda 5. pravila
- Za prvi slučaj dovoljno je prebojati N u crno i sve je riješeno jer se mijenjanjem boje korijena jednako mijenja crna visina svim čvorovima stabla. Ostala dva slučaja ovise o boji čvora N.

Uklanjanje crnog čvora

- Zamislimo da možemo nekako prenijeti crninu X-a na N. Tada ju uklanjanjem X ne bismo izgubili i RB pravila ne bi bila prekršena:
  - Ako je N prethodno bio crven, postat će crveno-crn i crnoj visini doprinositi 1.
  - Ako je N prethodno bio crn, postat će dvostruko crn i crnoj visini doprinositi 2.

- Rješenje:
  - Ako je crveno-crn, dovoljno je prebojati ga u čisto crno.
  - Ako je dvostruko crn, ideja je proslijediti višak crnog prethodniku i tako taj višak podizati sve dok ne dođe na mjesto gdje ga možemo trajno ugraditi u stablo ili dok ne dođe u korijen gdje ga možemo zanemariti.

Uklanjanje crnog čvora (dvostruko crn)

- 4 (+4 simetrična) su moguća slučaja, a ovise o boji čvora blizanca (dijete istog roditelja kao i N) i njegove djece

Rješenja ovise o boji siblinga i njegove djece.

Oznake:

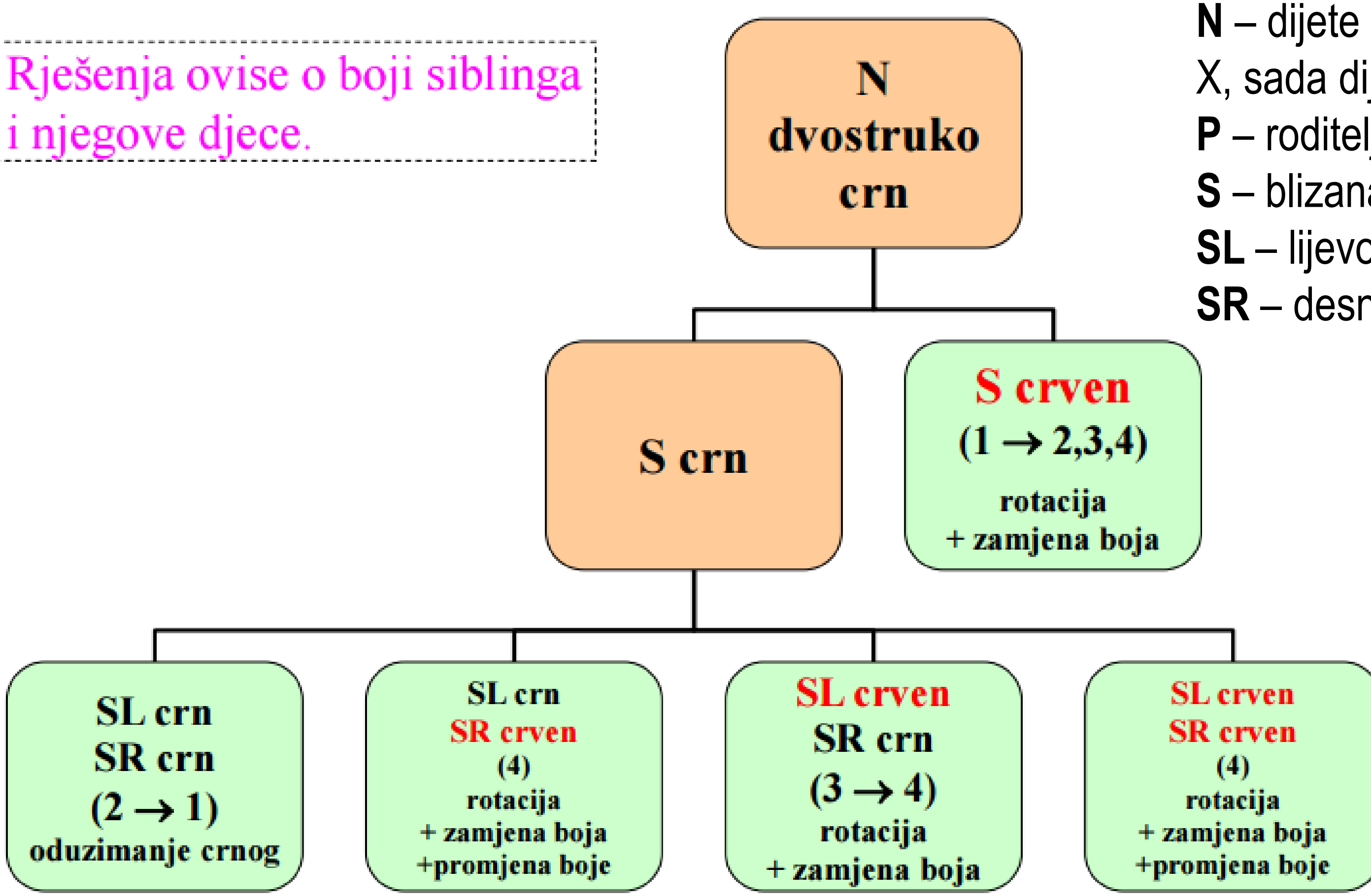
N – dijete od uklonjenog X, sada dijete od P

P – roditelj od N

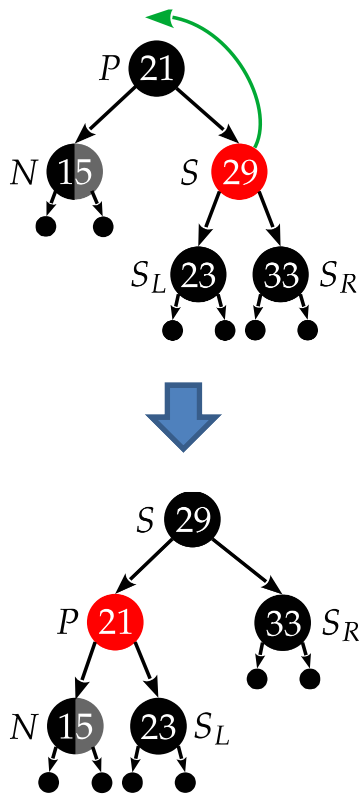
S – blizanac od N

SL – lijevo dijete od S

SR – desno dijete od S



Uklanjanje crnog čvora (dvostruko crn)

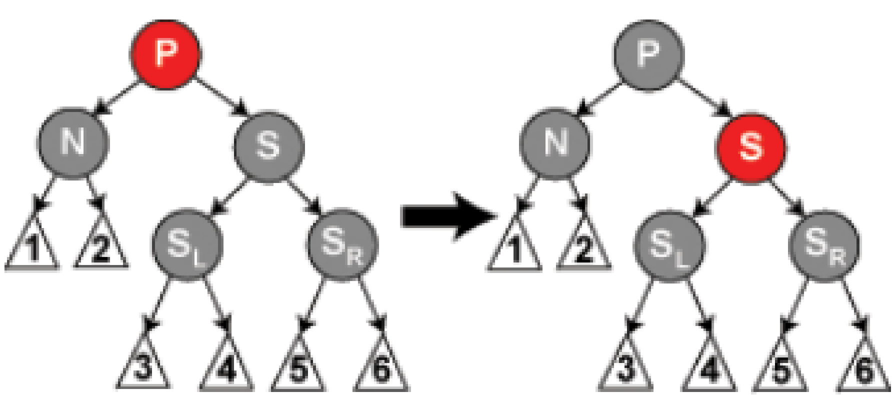


1. Blizanac S je crven
   - P je sigurno crn jer ima crveno dijete
   - Nakon brisanja X crna visina lijevog podstabla od P za jedan je manja od crne visine desnog podstabla (tj. N dvostruko crn)
   - Rješenje: rotirati S oko P (simetrija) pa zamijeniti boje P i S
   - **NASTAVAK** uravnotežavanja iz N

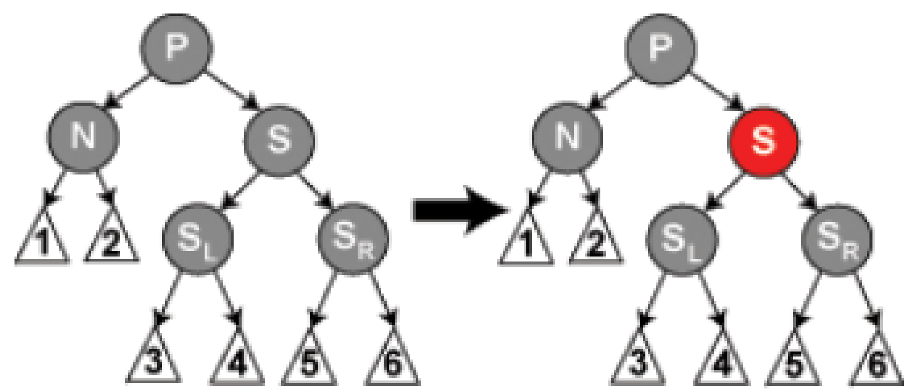
Uklanjanje crnog čvora (dvostruko crn)

2. S crn, djeca od S crna
   - Oduzeti jedno crno N-u i S-u; N ostaje jednostruko crn, a S postaje crven
   - Taj višak crnoga proslijediti višoj razini (konvergencija!), tj. P-u koji time postaje ili crveno-crn ili dvostruko crn
   - O P-u ovisi postupanje nakon intervencije (slučajevi 2a i 2b):

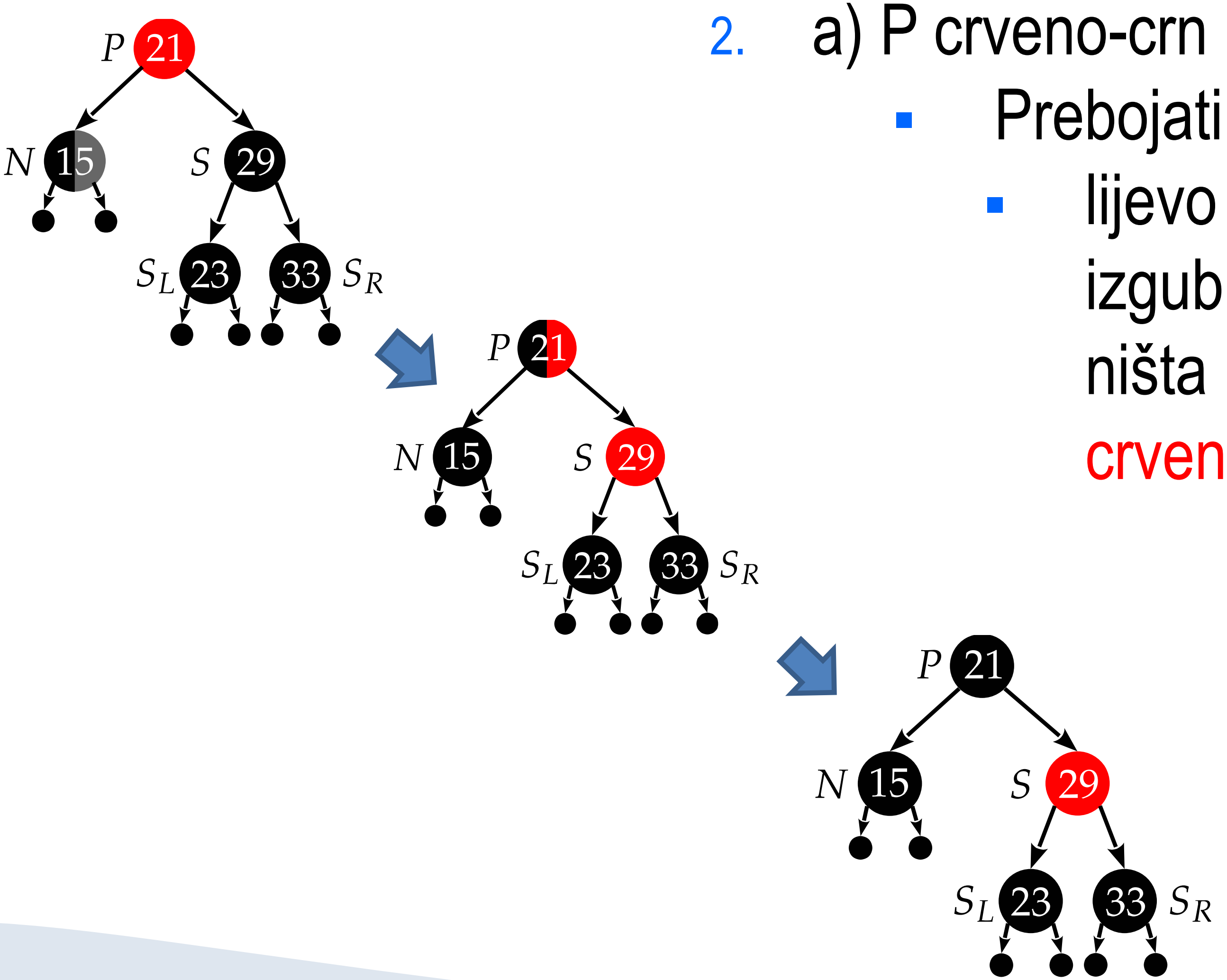
P crveno-crn



P dvostruko crn

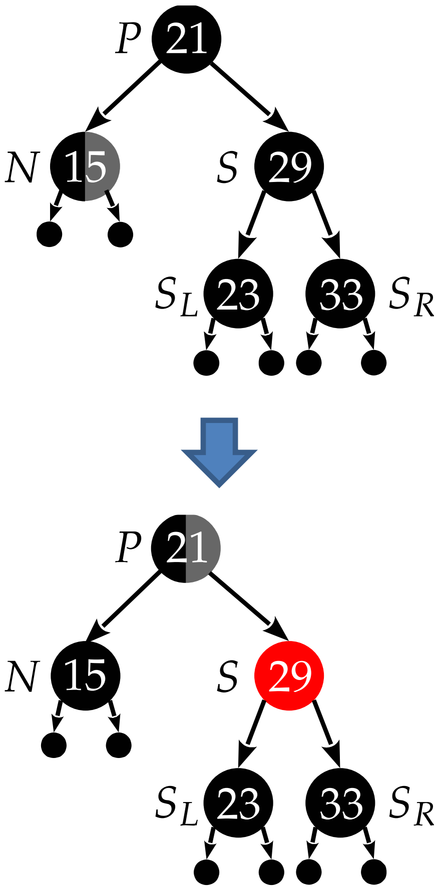


Uklanjanje crnog čvora (dvostruko crn)



2. a) P crveno-crn
   - Prebojati P u crno
     - lijevo podstablo time dobiva izgubljeno crno, a desnom se ništa ne mijenja jer je S crven; **KRAJ**

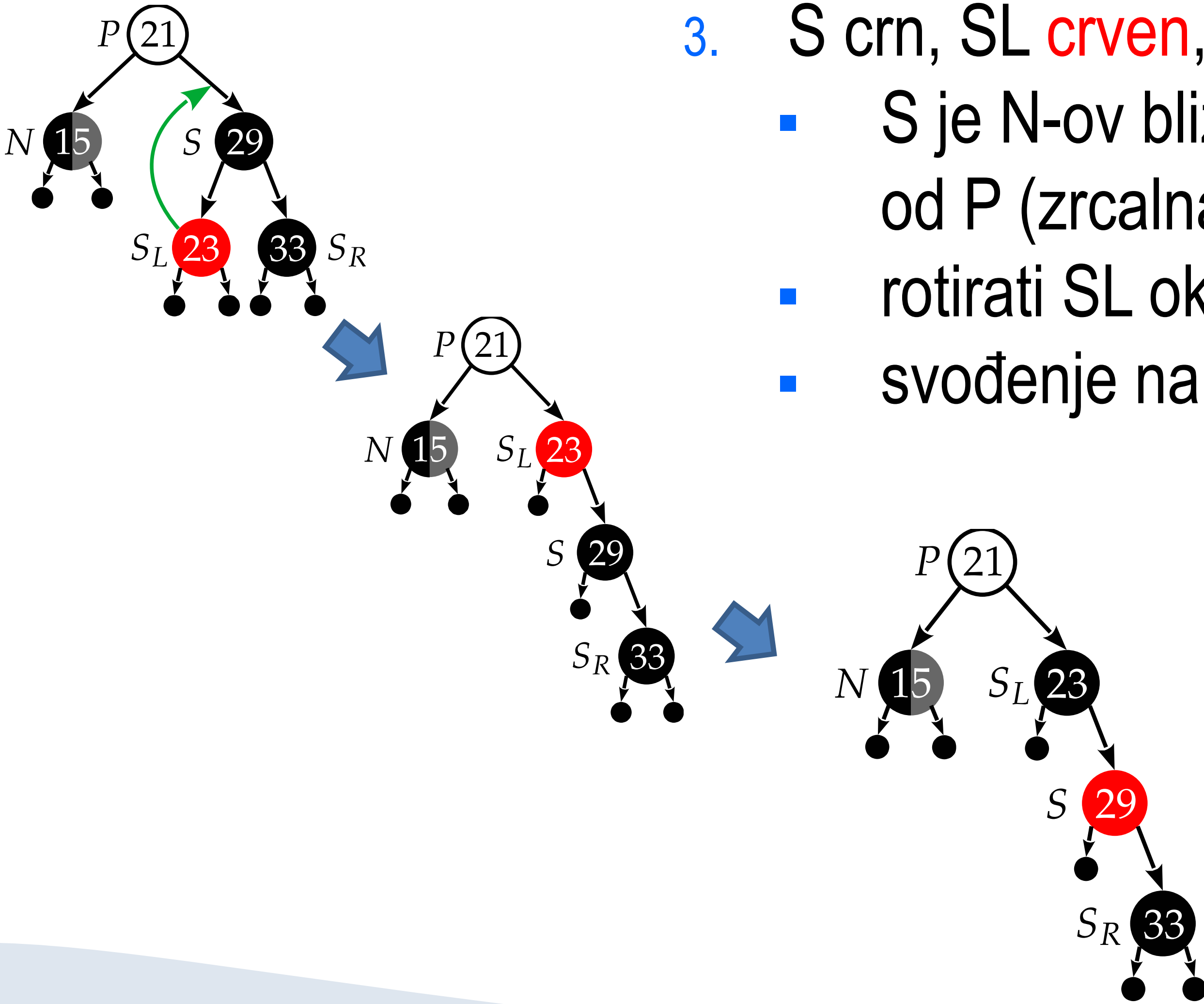
Uklanjanje crnog čvora (dvostruko crn)



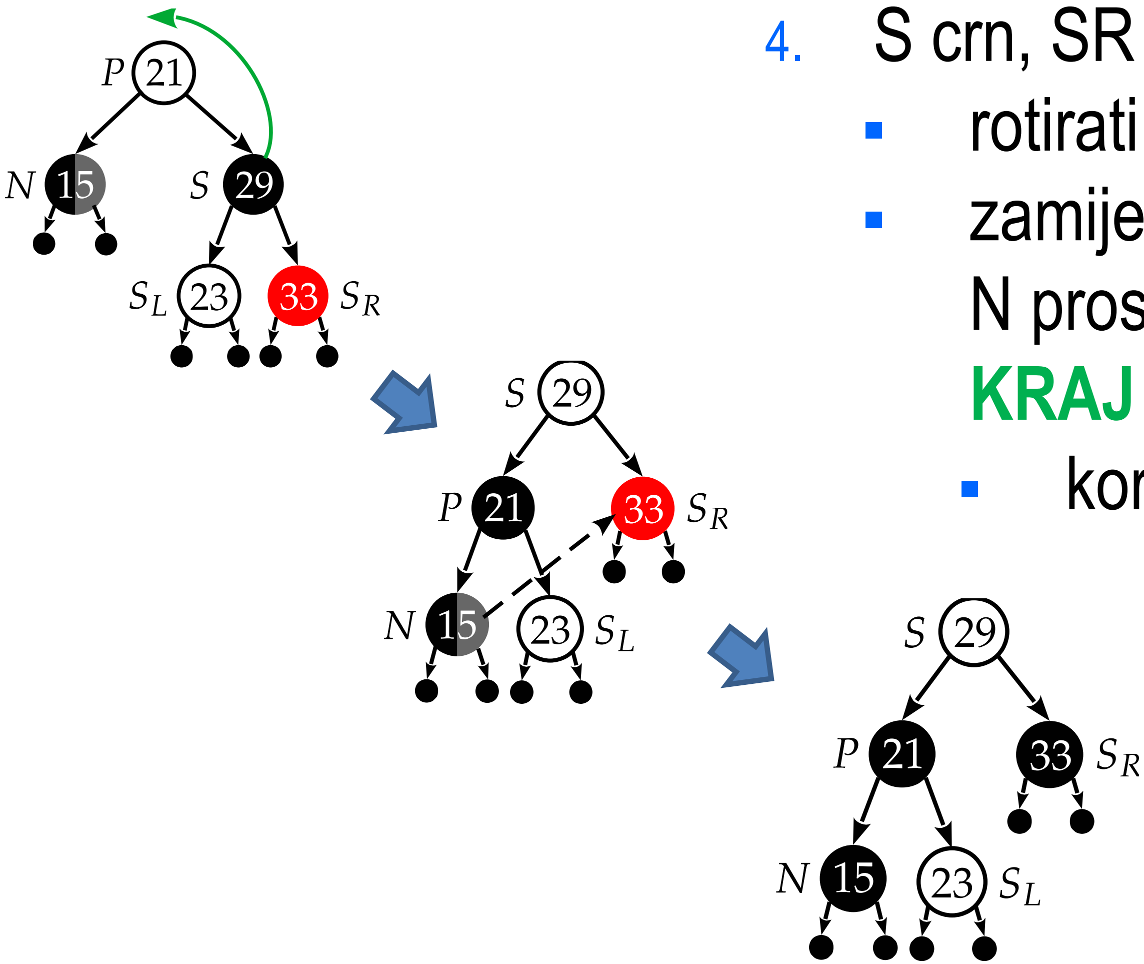
2. b) P dvostruko crn
   - P je korijen
     - višak crnog se odbacuje; **KRAJ**
   - P nije korijen
     - natrag na slučaj 1 promatrajući P kao N; **NASTAVAK**
     - problem je razinu više (konvergencija!)

Uklanjanje crnog čvora (dvostruko crn)

3. S crn, SL crven, SR crn, P nevažan
   - S je N-ov blizanac, a N je <u>lijevo</u> dijete od P (zrcalna simetrija!)
   - rotirati SL oko S i zamijeniti im boje
   - svođenje na slučaj 4; **NASTAVAK**



Uklanjanje crnog čvora (dvostruko crn)



4. S crn, SR crven, P i SL nevažni
   - rotirati S oko P (zrcalna simetrija!)
   - zamijeniti boje S i P, a višak crnog iz N proslijediti u SR (prebojati u crno); **KRAJ**
     - korijen podstabla ostaje iste boje

Implementacija brisanja čvora u RB-stablu

```
RBDeleteFixup(N)
//it is assumed that an ordinary deleting routine has already replaced black X by N
while  N is not root    and    colour(N) is black
    if N  is left child of P
        S = right child of P;                              //sibling
        if colour(S) = red
            colour(S) = black;                             //case 1
            colour(p(N)) = red;                            //case 1
            left-rotate S about P;                         //case 1
        if colour(SL)=black and colour(SR)=black
            colour(S)=red;                                 //case 2
            N = p(N);                                      //case 2
        else
            if colour(SR)=black
            {    colour(SL)=black;                         //case 3
                 colour(S)=red;                            //case 3
                 right-rotate SL about S;  }               //case 3
            colour(S)=colour(P);                           //case 4
            colour(P)=black;                               //case 4
            colour(SR)=black;                              //case 4
            left-rotate S about P;                         //case 4
            N = root;          //or simply break;
    else    //same as if-clause with left and right exchanged
colour(N) = black;
```

Primjeri za vježbanje

[9 bodova] Prikažite polazno prazno RB stablo uslijed sljedećih promjena (redom kojim su navedene):

a) **[5 boda]** upisivanja redom sljedećih brojeva:
25, 7, 24, 15, 27, 37, 47, 36, 34, 5, 45

b) **[4 boda]** brisanja redom:
27, 24, 34