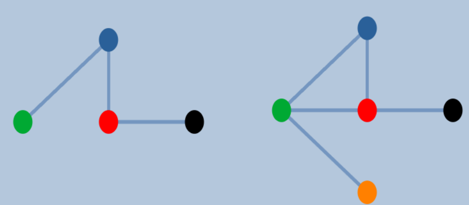
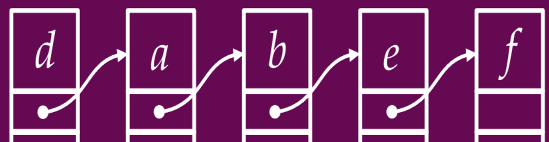
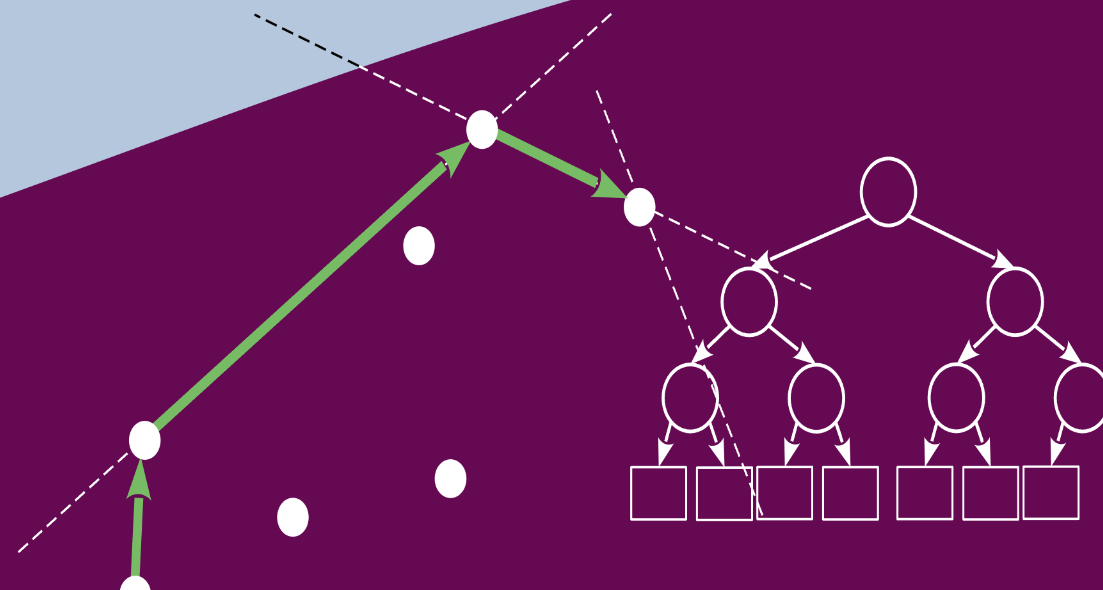
Napredni algoritmi i strukture podataka

Tjedan 2: B-stabla i Crveno-crna stabla (RB)







Creative Commons

slobodno smijete:

dijeliti — umnožavati, distribuirati i javnosti priopćavati djelo

prerađivati djelo

pod sljedećim uvjetima:

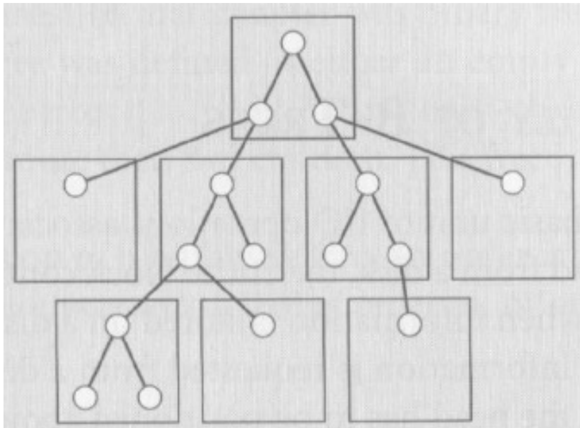
imenovanje: morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).

nekomercijalno: ovo djelo ne smijete koristiti u komercijalne svrhe.

dijeli pod istim uvjetima: ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

Motivacija

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



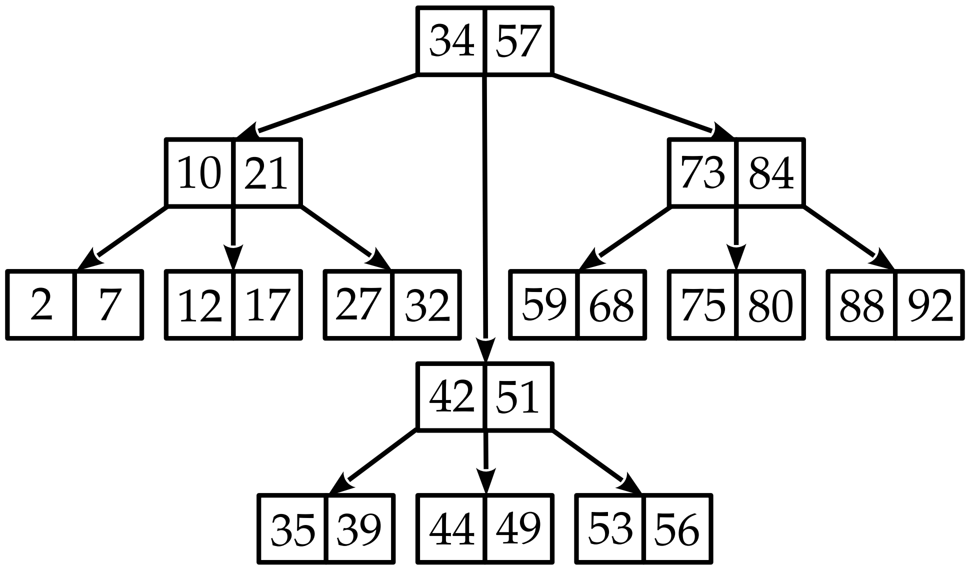
- B-stabla ublažavaju efekte ograničenja sekvencijalnog blokovskog čitanja
  - Veličina čvora se prilagođava veličini bloka

Karakteristike

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

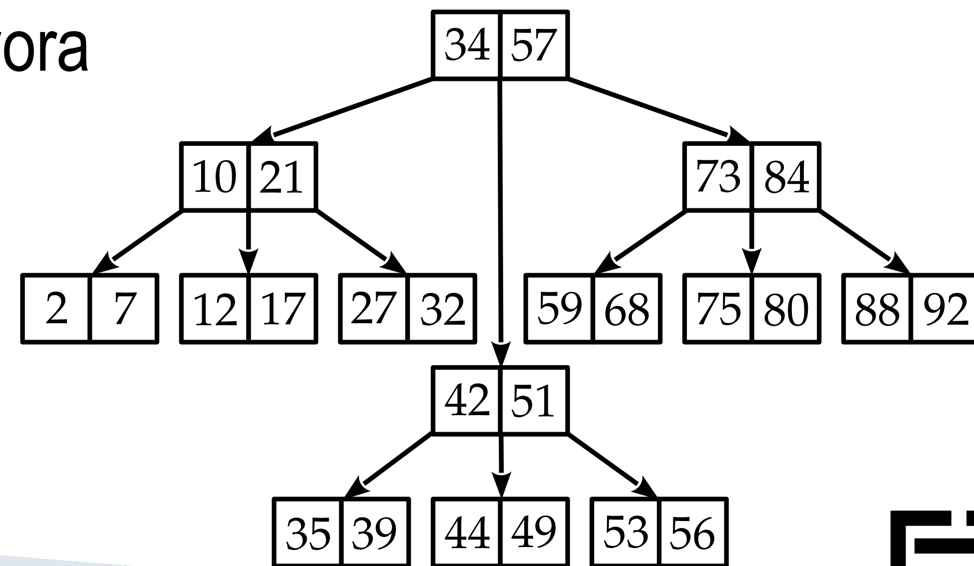
M stabla

- M stabla (*multiway tree*): stabla u kojima čvorovi mogu imati proizvoljan broj djece
- M stablo *m*-tog reda: M stablo u kojem čvorovi mogu imati najviše *m* djece.



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-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

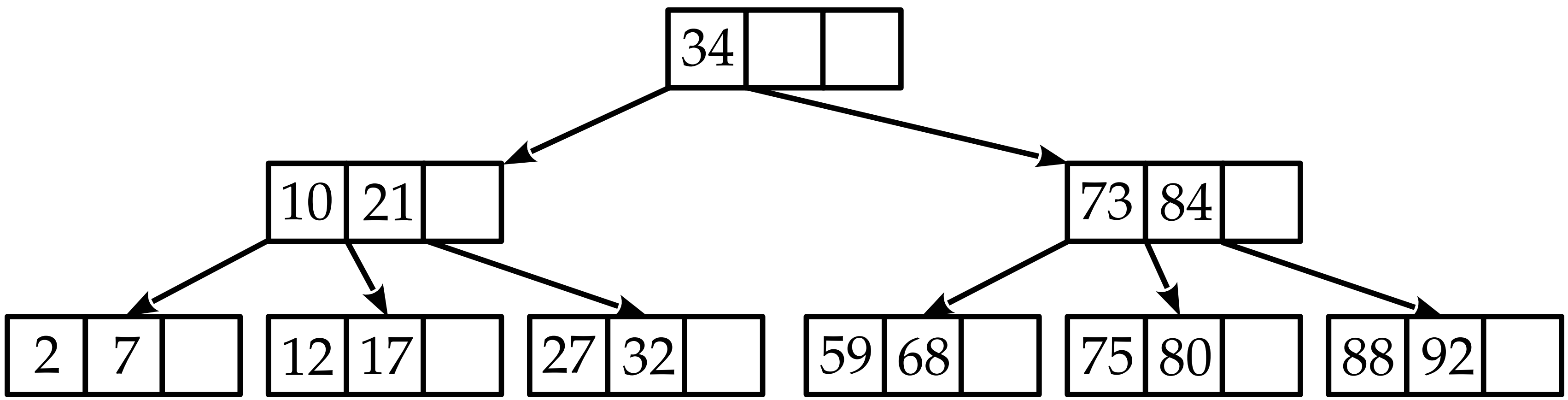
$$\lceil m/2 \rceil \leq k \leq m$$

4. Svi listovi su na istoj razini

Savršena uravnoteženost

B-stablo

- Osobitosti
  - Popunjenost barem 50%
  - Savršeno uravnoteženo



B-stablo

- Implementacija #1
  - Struktura (klasa) s poljem od *m-1* ključeva i poljem od *m* pokazivača – u Pythonu može biti i jedno jedinstveno polje gdje se izmjenjuju pokazivači i ključevi
  - Moguće dodati podatke radi lakšeg održavanja (npr. broj upisanih podataka u čvoru)

- Implementacija #2
  - Svaki čvor je dvostruko povezana lista
  - Svaki ključ ima pokazivače na djecu – samo posljednji ključ koristi oba pokazivača

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 se 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

```
function BTreeSearch(n, v_s)
    n_v ← starting value of the node n
    while value(n_v) < v_s and next(n_v) is not nil do
        n_v ← next(n_v)
    if value(n_v) = v_s then
        return n
    else if next(n_v) is nil and value(n_v) < v_s then
        if rightChild(n_v) is not nil then
            return BTreeSearch(rightChild(n_v), v_s)
        else
            return no searched key
    else
        if leftChild(n_v) is not nil then
            return BTreeSearch(leftChild(n_v), v_s)
        else
            return no searched key
```

- Napomena
  - value(n_v) – vrijednost ključa n_v – npr. cijeli broj
  - next(n_v) – sljedeći ključ nakon n_v u listi ključeva čvora – ovo ovisi o implementaciji čvora
  - leftChild(n_v) i rightChild(n_v) – lijevo i desno dijete ključa n_v

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” (*split*) 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

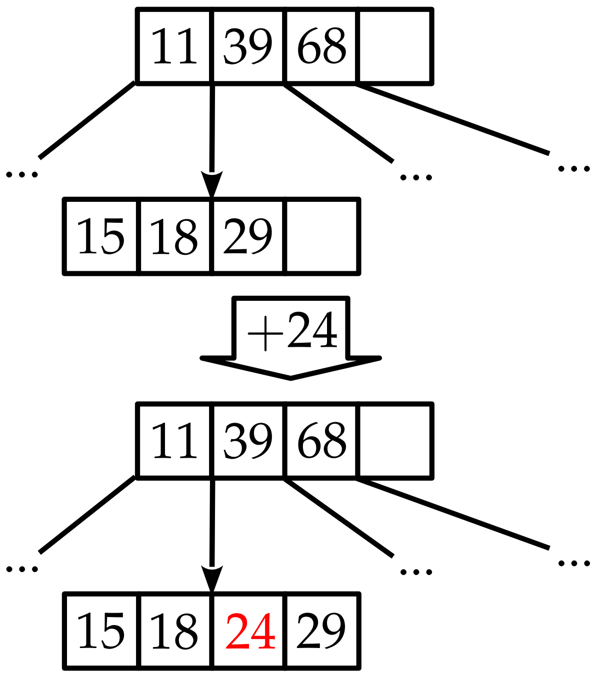
Dodavanje podataka u B-stablo

- 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**

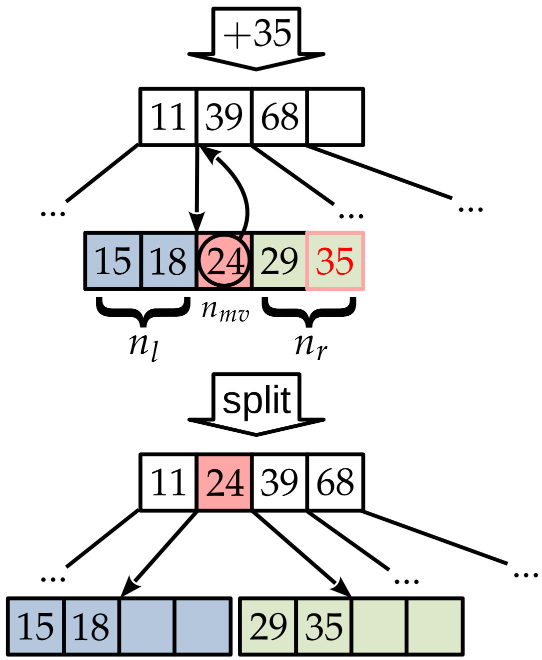
Primjer dodavanja podataka u B-stablo



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

Max. degree = 5

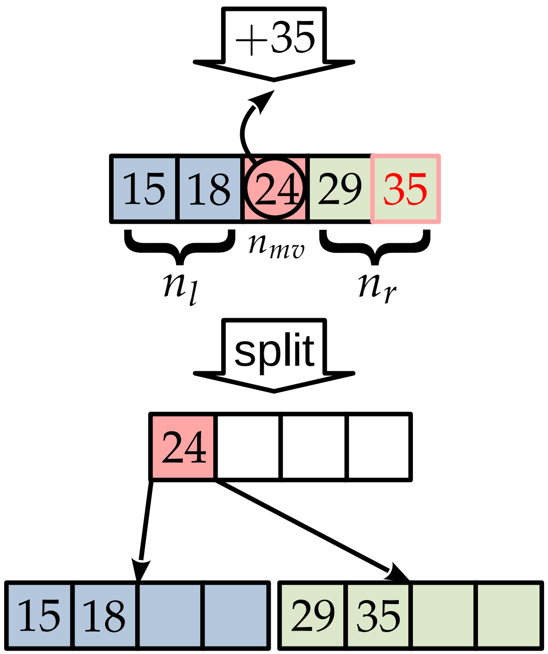
Primjer dodavanja podataka u B-stablo



- **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**

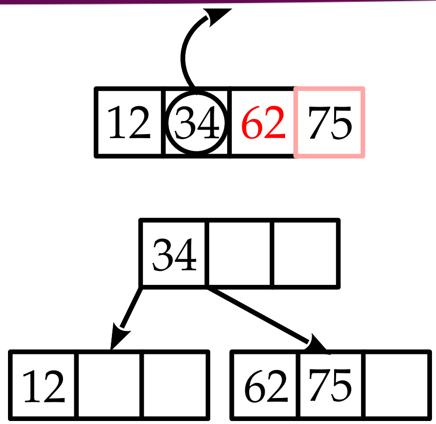
12		

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

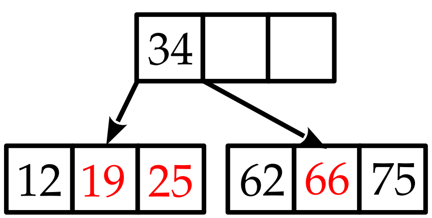
12	34	75

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

Primjer dodavanja podataka u B-stablo



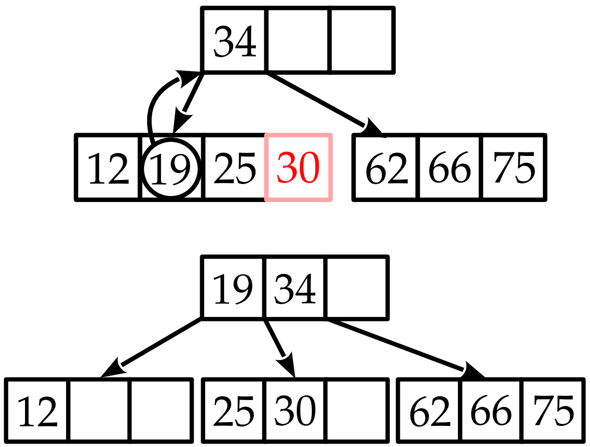
- **Korak 3**: Dodavanjem ključa **62**, dolazi do preljeva u korijenskom čvoru, kojeg razdvajamo uz stvaranje novog korijenskog čvora.



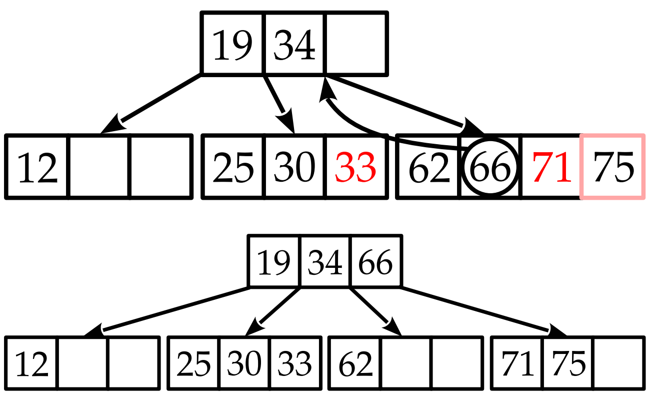
- **Korak 4**: Dodajemo ključeve **19**, **25** i **66** direktno u listove.

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

Primjer dodavanja podataka u B-stablo



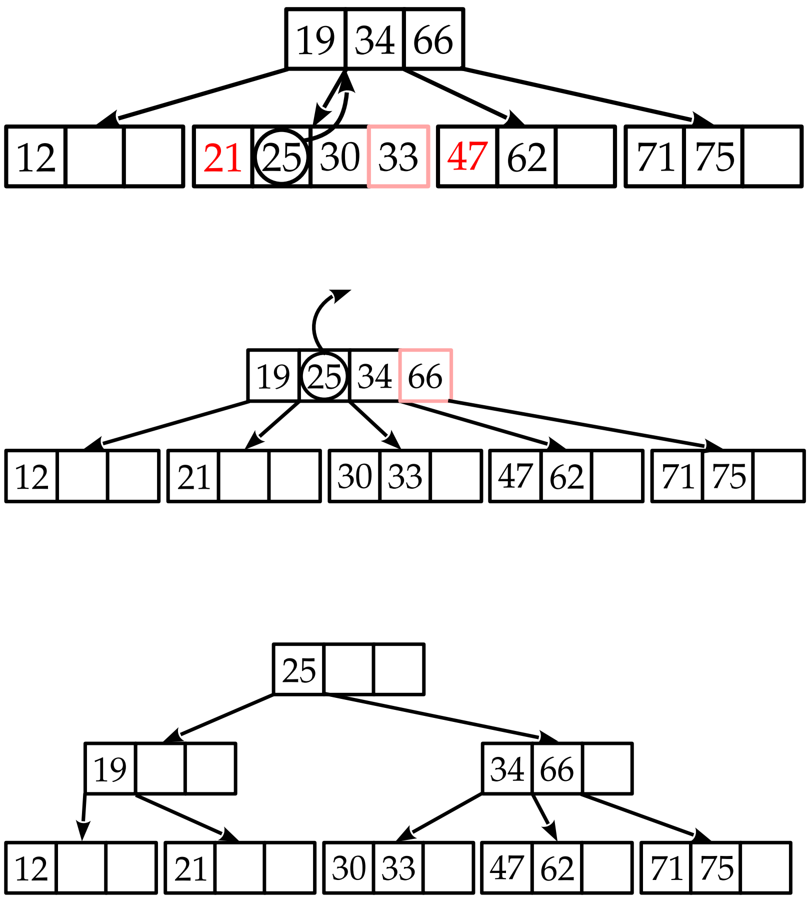
- **Korak 5**: Dodavanjem ključa **30**, dolazi do preljeva u lijevom listu, kojeg razdvajamo uz ubacivanje srednjeg ključa u korijenski čvor.



- **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

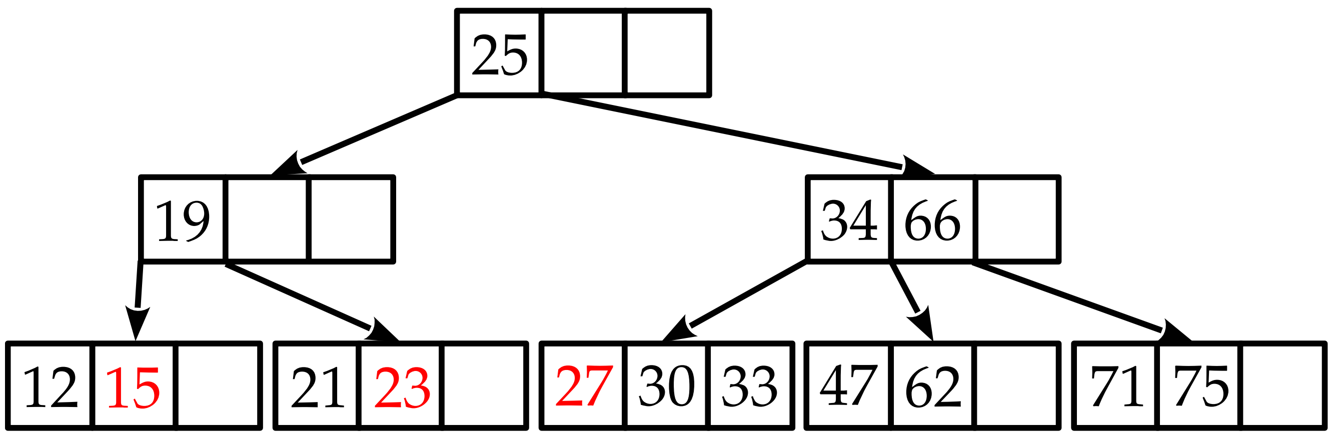


- **Korak 7**: Dodajemo ključ **47**, a zatim ključ **21**.
  - Dodavanjem **21** izazivamo preljev u listu, kojeg razdvajamo uz ubacivanje srednjeg ključa u korijenski čvor.
  - Ubacivanjem 25 u korijenski čvor izazivamo preljev korijenskog čvora, kojeg razdvajamo uz stvaranje novog korijenskog čvora.

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.

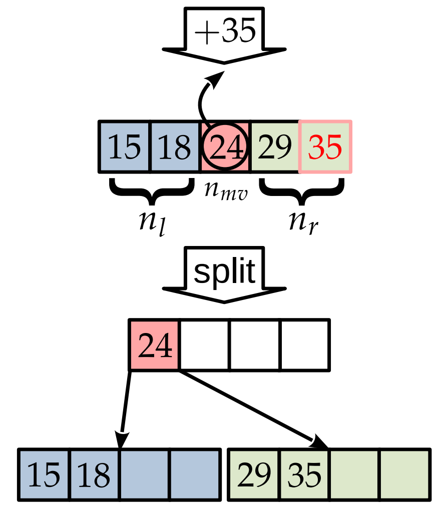


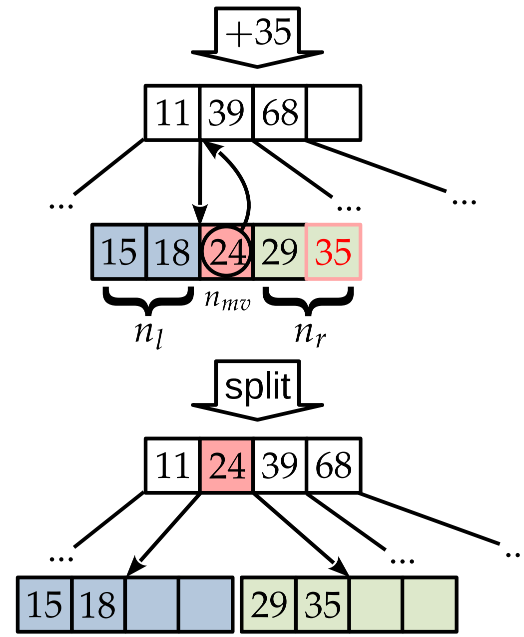
12,75,34,62,19,25,66,30,33,71,47,21,15,23,27

Implementacija dodavanja u B-stablo

procedure BTREESPLIT($btree, n$)
  if $|n| == m$ **then**
    $n_l \leftarrow$ new node having first $\lceil m/2 \rceil - 1$ values in the node n
    $n_{mv} \leftarrow$ the next value in the node n
    $n_r \leftarrow$ new node having the rest of values from the node n
    $n_{last} \leftarrow$ the last value in n_l
    $rightChild(n_{last}) \leftarrow leftChild(n_m)$
    if $parent(n)$ is nil **then**
      $n_{root} \leftarrow$ new node
      $n_{mv}' \leftarrow$ insert $value(n_{mv})$ into the node n_{root}
      $root$ of $btree \leftarrow n_{root}$
      $leftChild(n_{mv}') \leftarrow n_l$
      $rightChild(n_{mv}') \leftarrow n_r$
    else
      $n_{mv}' \leftarrow$ insert $value(n_{mv})$ into the parent node of n
      $leftChild(n_{mv}') \leftarrow n_l$
      $rightChild(n_{mv}') \leftarrow n_r$
      BTREESPLIT($btree$,parent of n)

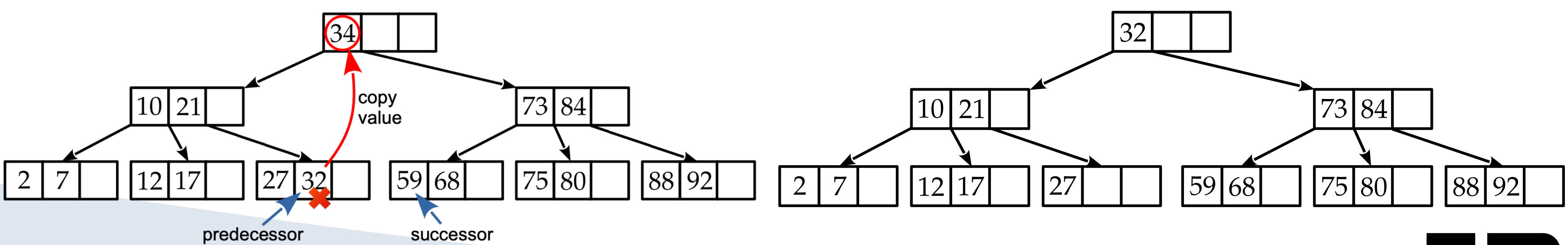
procedure BTREEINSERT($btree, v_n$)
  $(n, n_v) \leftarrow$BTREESEARCH($root$ of $btree, v_n$)
  insert v_n into the node n
  BTREESPLIT($btree, n$)





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

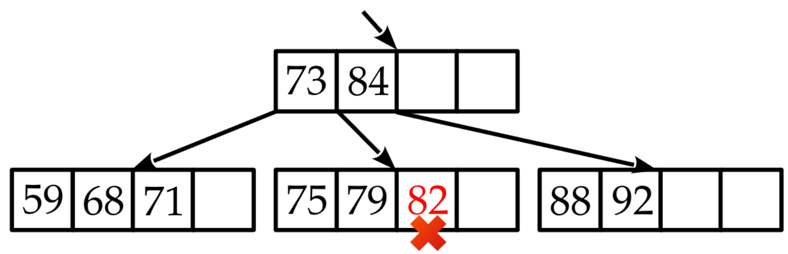


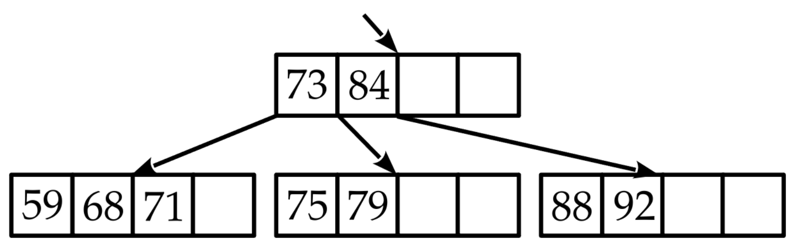
Brisanje podataka u B-stablu

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**

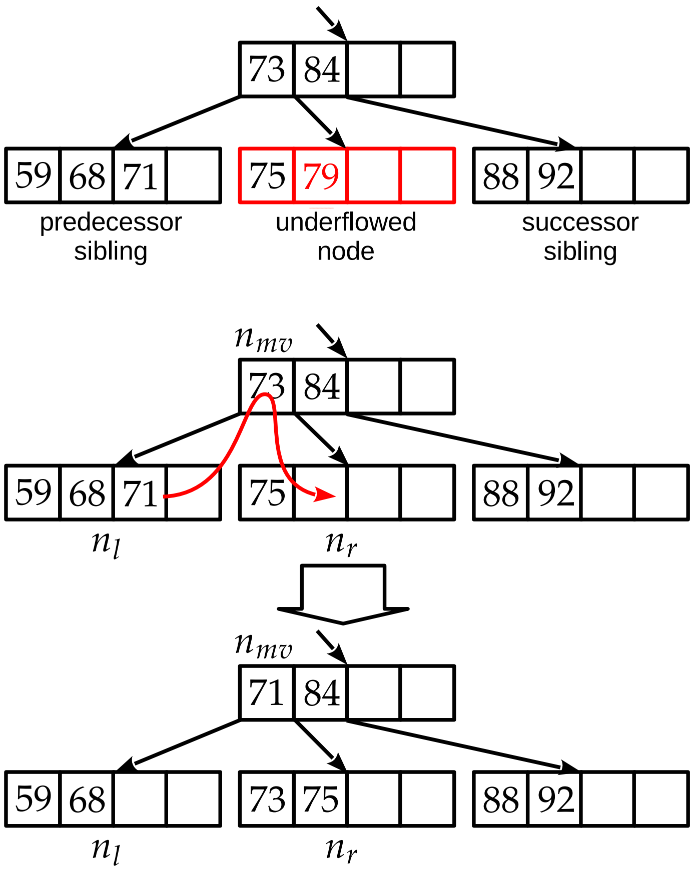
Primjer brisanja podataka iz B-stabla

- **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.



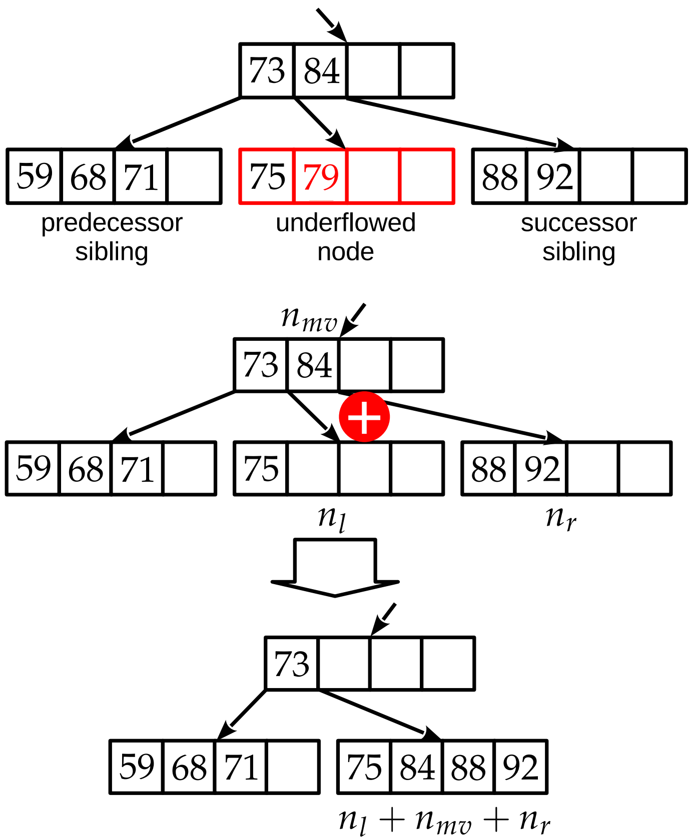


Primjer brisanja podataka iz B-stabla



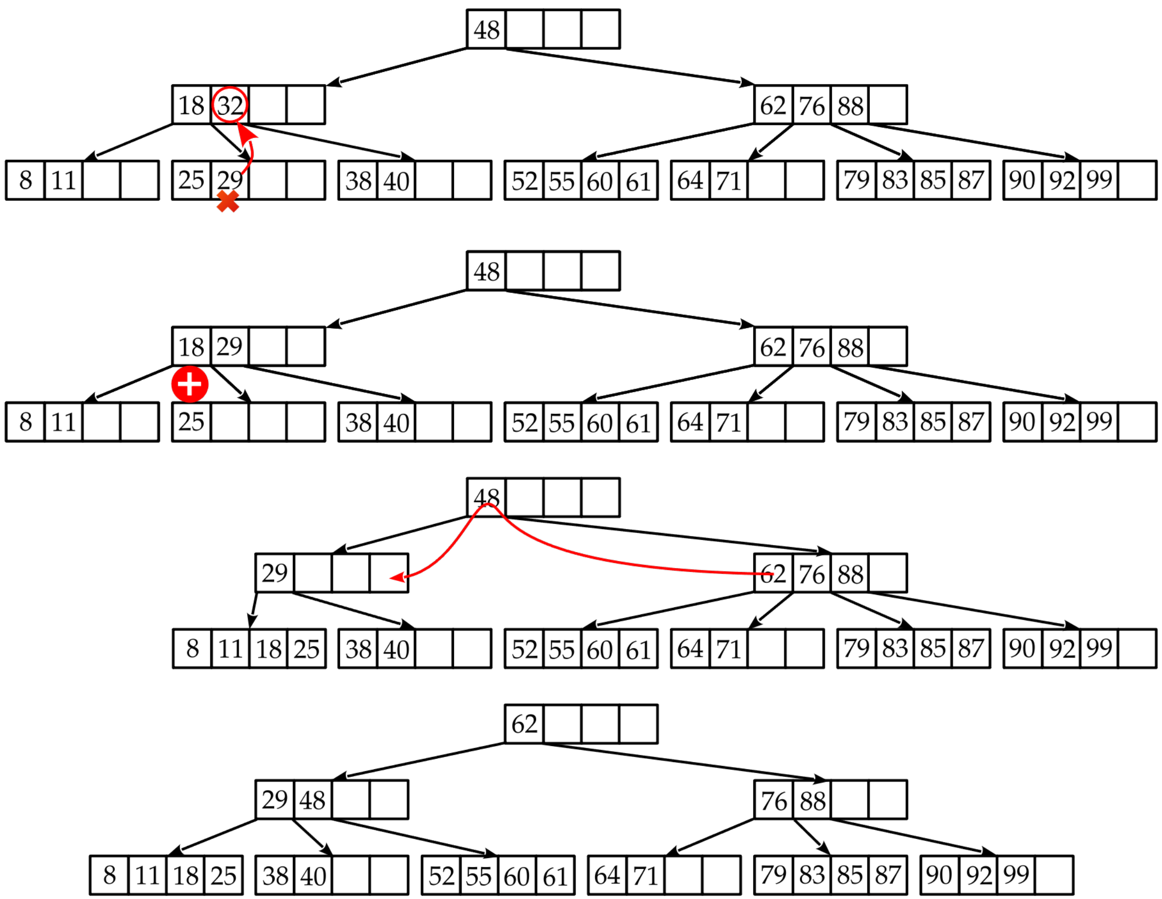
- **Primjer 2**: brišemo ključ **79** iz lista. List nakon brisanja nije više popunjen ≥50% zato jer ima < ⌈m/2⌉-1 ključeva.
- Gledamo li lijevog susjeda (blizanca), vidimo da je on popunjen > ⌈m/2⌉-1
  - Radimo restrukturiranje tako da prebacujemo ključeve iz lijevog susjeda u čvor koji je u podljevu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

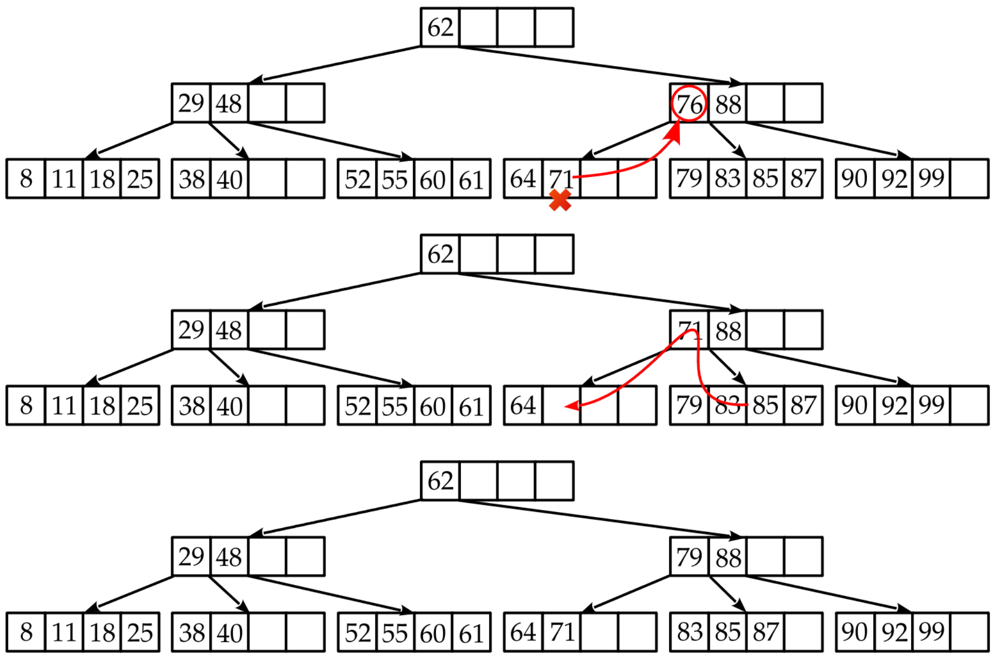
Primjer brisanja podataka iz B-stabla



- 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

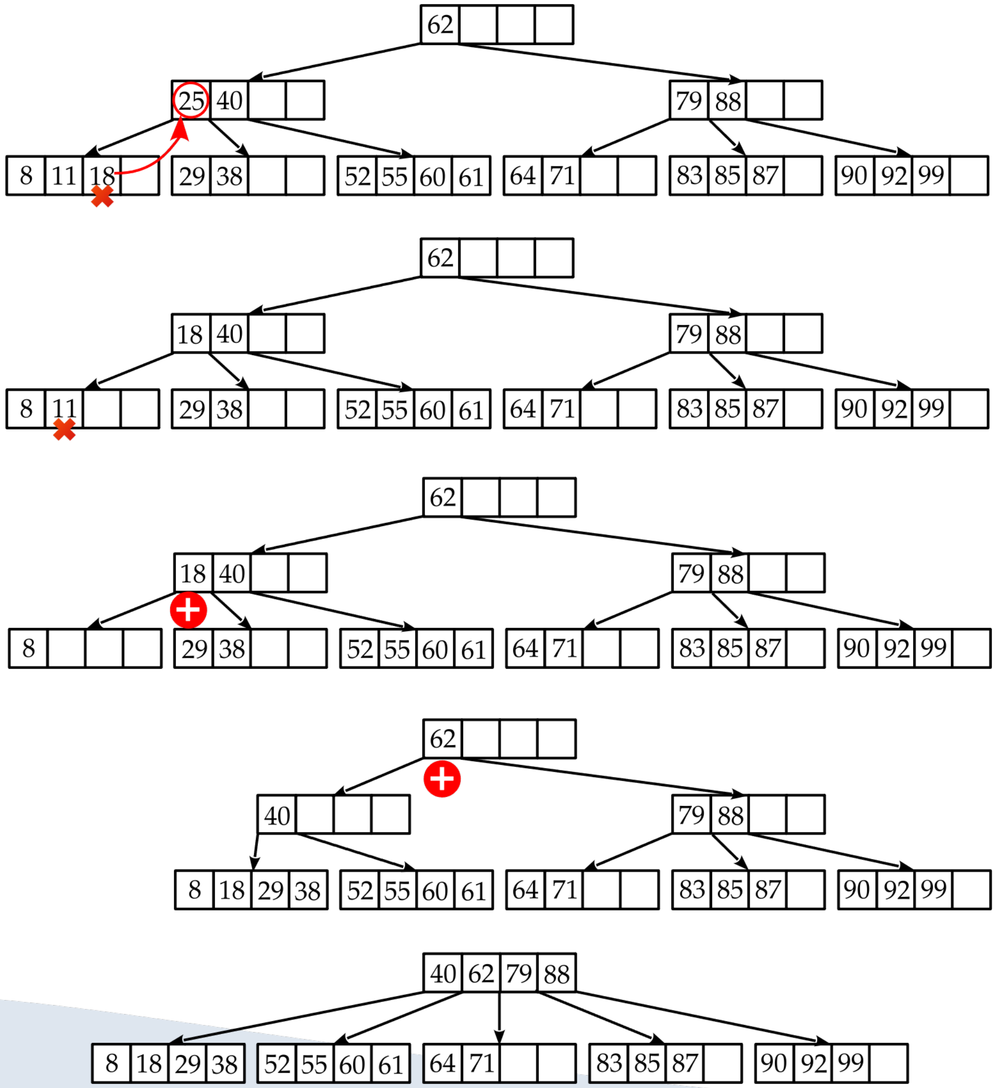
Primjer brisanja podataka iz B-stabla



- **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

32, 76, 48, 25 i 11

Primjer brisanja podataka iz B-stabla



- **Korak 3**: brišemo ključ **25** postupkom kopiranja, a zatim brišemo ključ **11**. List u kojem smo obrisali 18 i 11 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

```
procedure BTreeRemoval(btree, val)
    (n_rem, n_v) ← BTreeSearch(root of btree, val)
    if n_v is not nil then
        remove value val from the node n_rem
        BTreeRemovalConsolidation(btree, n_rem)
```

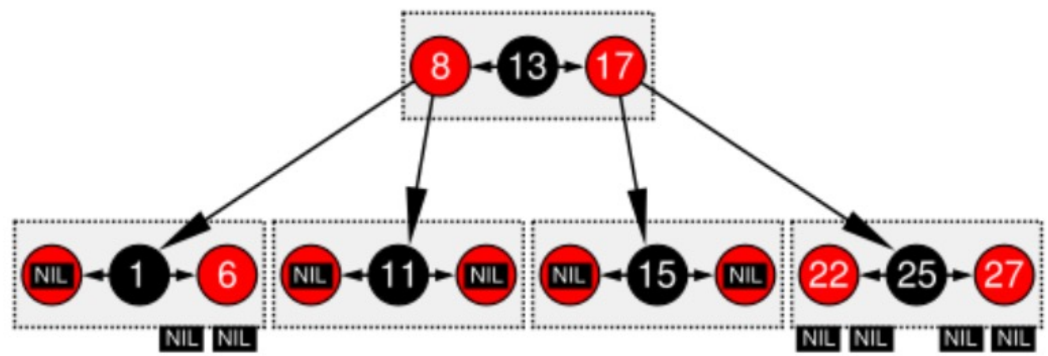
```
procedure BTreeRemovalConsolidation(btree, n)
    if |n| < ⌈degree of btree/2⌉ − 1 then
        ps ← the predecessor sibling
        n_root ← nil
        if ps is not nil then
            n_mv ← the shared parent value between ps and n
            if |ps| > ⌈degree of btree/2⌉ − 1 then
                BTreeRedistribute(btree, ps, n_mv, n)
            else
                n_root ← BTreeMerge(btree, ps, n_mv, n)
        else
            ss ← the successor sibling
            n_mv ← the shared parent value between ss and n
            if |ss| > ⌈degree of btree/2⌉ − 1 then
                BTreeRedistribute(btree, n, n_mv, ss)
            else
                n_root ← BTreeMerge(btree, n, n_mv, ss)
        if n_root is nil and parent of n exists then
            BTreeRemovalConsolidation(btree, parent of n)
```

Crveno-crna stabla

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

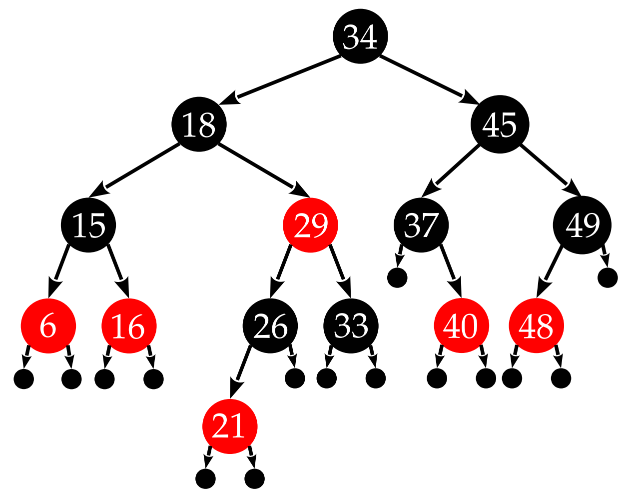
- Binarno stablo koje **idejno** proizlazi iz B-stabla 4. reda ako mu se elementi čvorova smatraju obojanima prema strogim pravilima

- 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
  $h \leq 2log_2(n+1)$
  Dokaz:
  Binarno stablo visine h ima najviše $n = 2^h - 1$ čvorova
  Zbog 4. pravila, barem polovica visine je crna visina pa je $hb \geq h/2$. 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
  $h \leq 2log_2(n+1)$
- 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

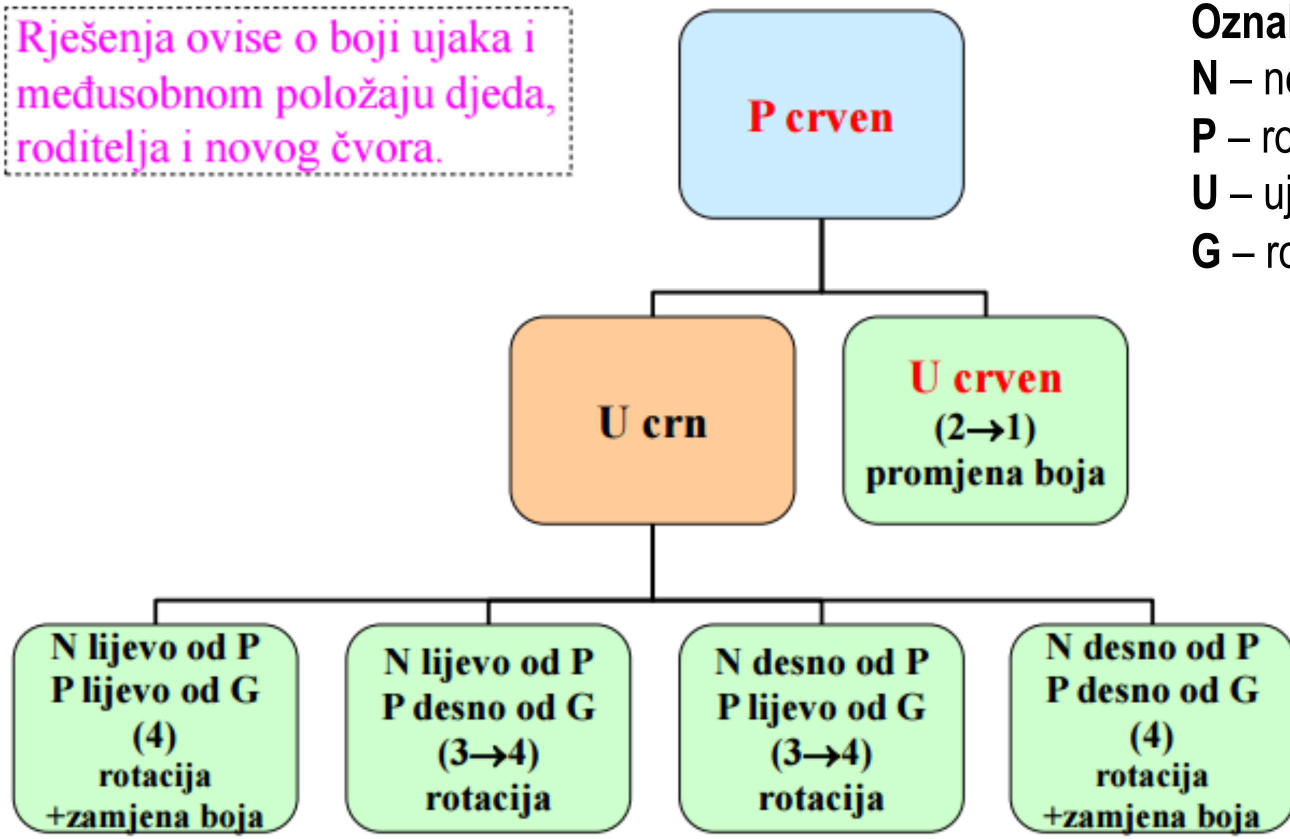
1. ubaciti novi čvor kao u svako drugo binarno search stablo i pridijeliti mu crvenu boju
2. restrukturirati stablo (primjenom rotacija i bojanjem čvorova) da bi zadovoljilo definicijska pravila

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

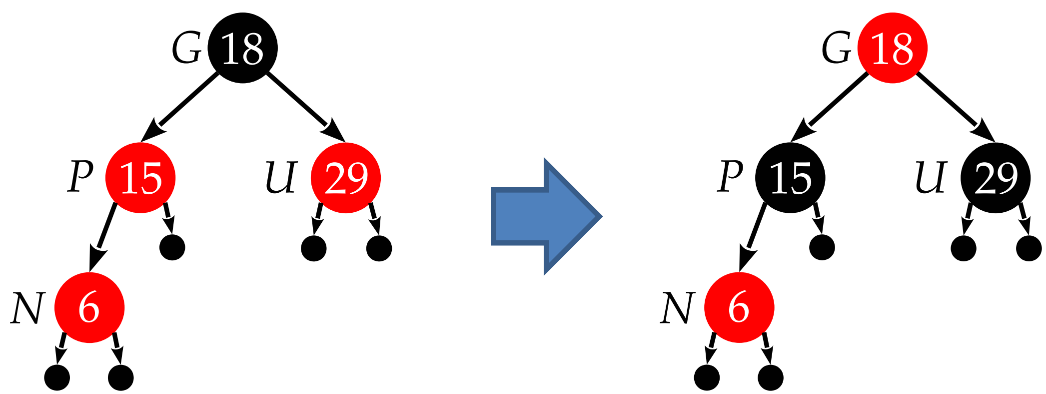
- Restrukturiranje:
  2. roditelj novog čvora je crven (korak 1):



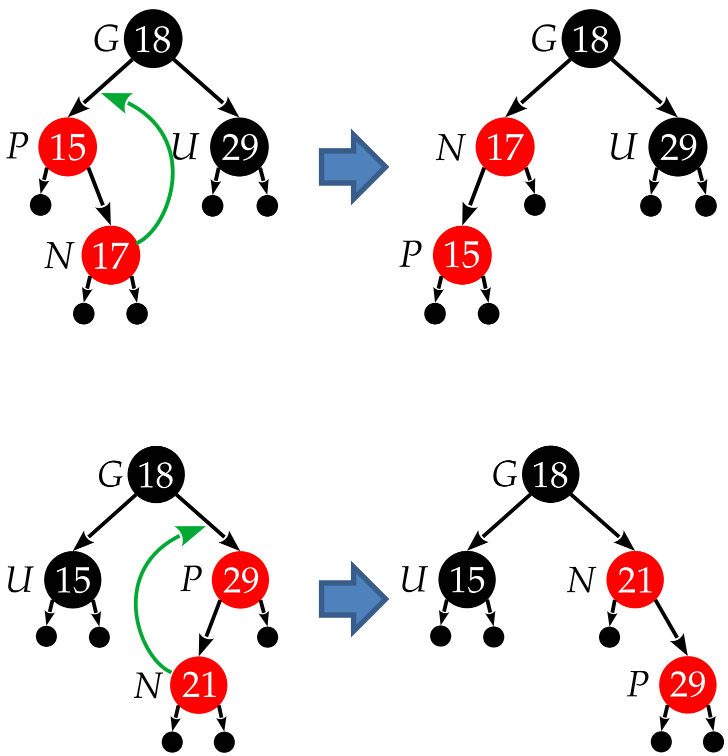
Oznake:
N – novi čvor
P – roditelj od N
U – ujak (blizanac od P)
G – roditelj od P

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)

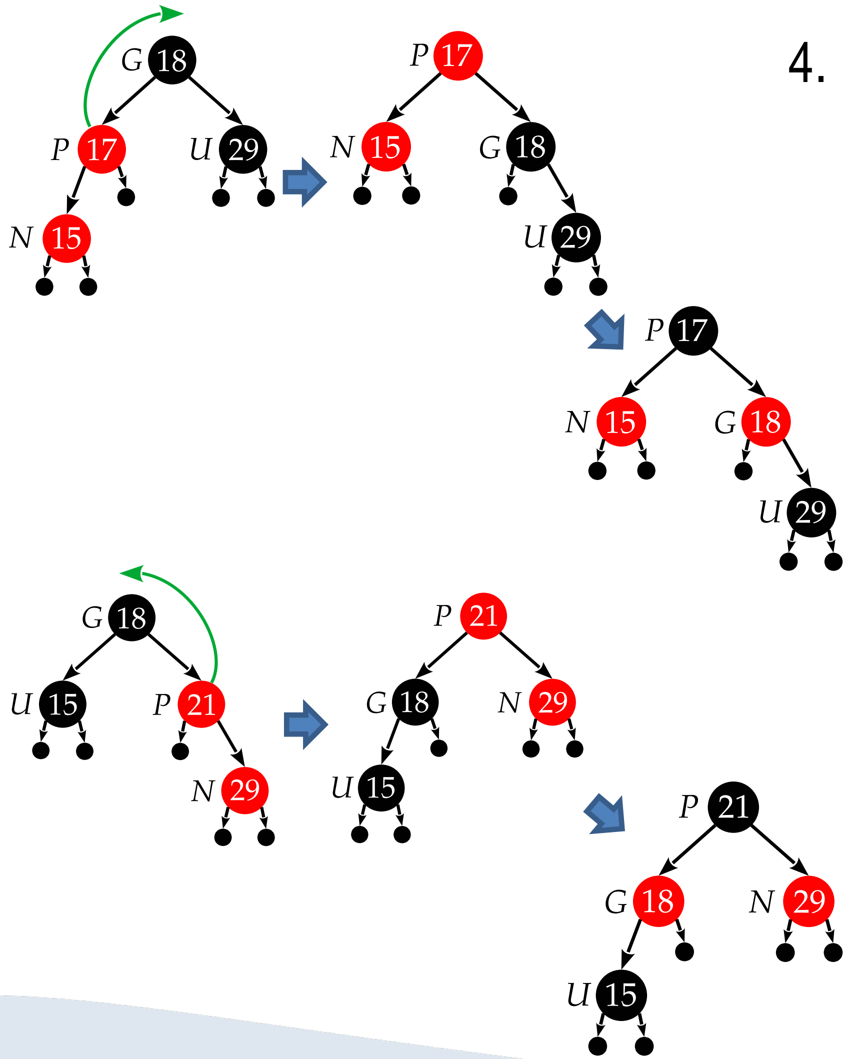


Dodavanje čvora u RB-stablo



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

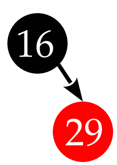


4. Roditelj crven i ujak crn ("linijski" poredak N, P i G)
   - Dva simetrična slučaja:
     - N lijevo dijete od P i P lijevo dijete od G
     - N desno dijete od P i P desno dijete od G
   - Rješenje:
     - *rotacija* P oko G
     - *zamjena boja* P i G (znamo da je G crn jer u protivnom P ne bi mogao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

Primjer dodavanja podataka u RB-stablo

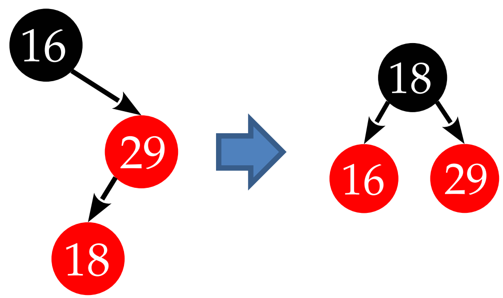
- Dodajemo redom ključeve: **16, 29, 18, 34, 26, 15, 45, 33, 6, 37, 49, 48, 40**

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

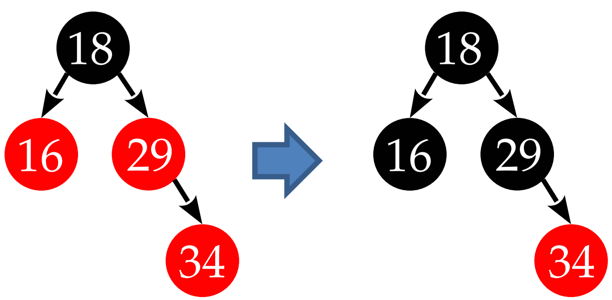


- **Korak 2**: U stablo dodajemo ključ **29**. Nema restrukturiranja RB-stabla.

Primjer dodavanja podataka u RB-stablo



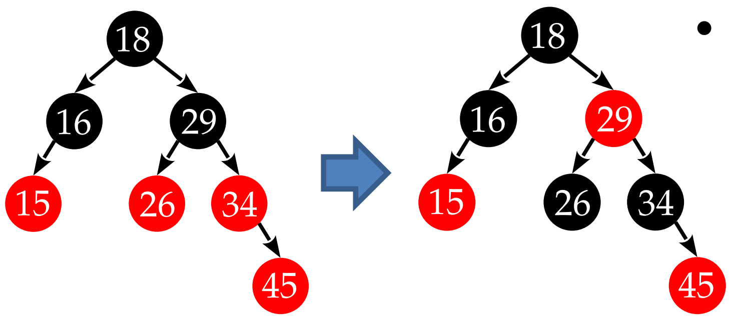
- **Korak 3**: U stablo dodajemo ključ **18**.
  - **Slučaj 3**: Desna rotacija **18** oko **29**
  - **Slučaj 4**: Lijeva rotacija **18** oko **16** + zamjena boja.



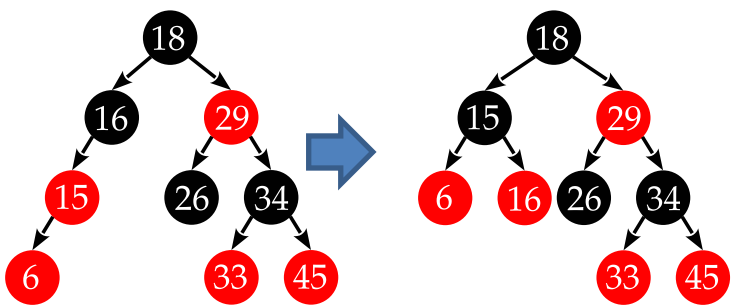
- **Korak 4**: U stablo dodajemo ključ **34**.
  - **Slučaj 2**: Postavi **18** u crveno, a **16** i **29** u crno
  - **Slučaj 1**: Postavi korijen **18** u crno

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

Primjer dodavanja podataka u RB-stablo



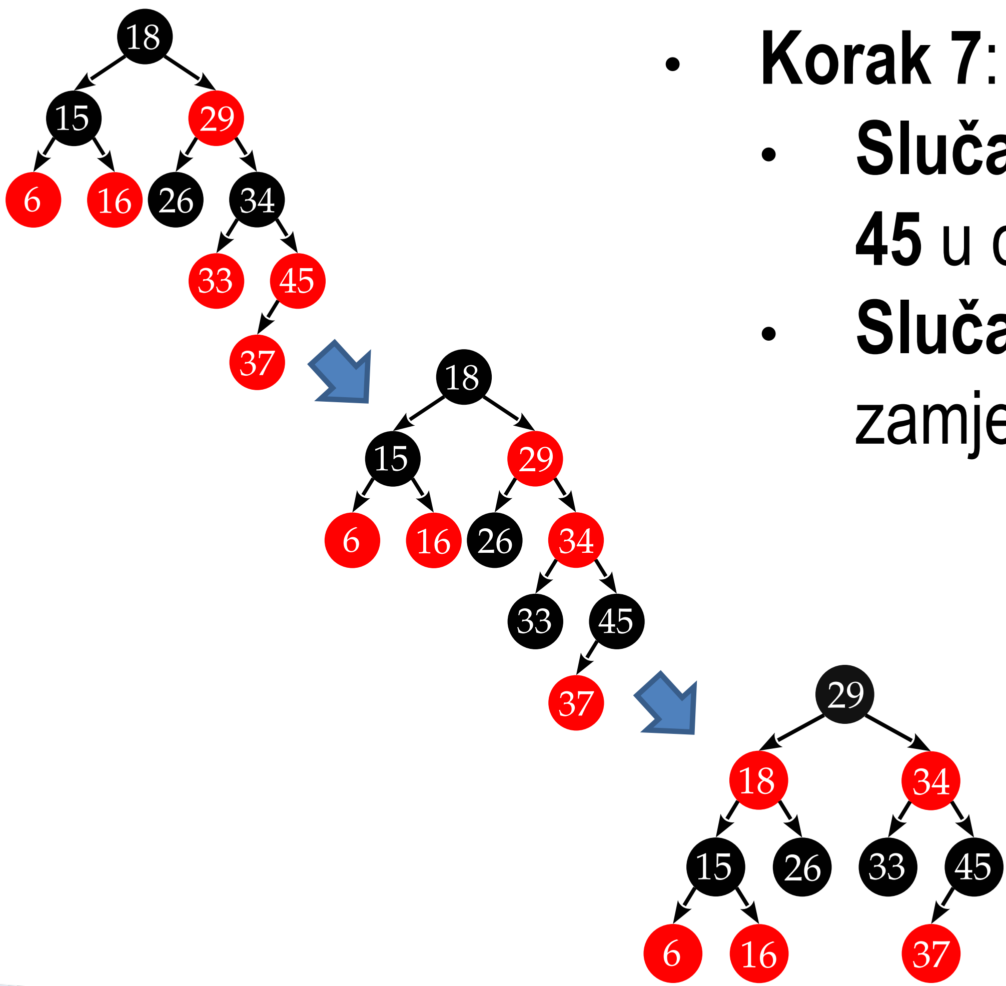
- **Korak 5**: U stablo dodajemo ključeve **26**, **15** i **45**.
  - Nakon dodavanja **45** imamo **slučaj 2**: Postavi **29** u crveno, a **26** i **34** u crno.



- **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

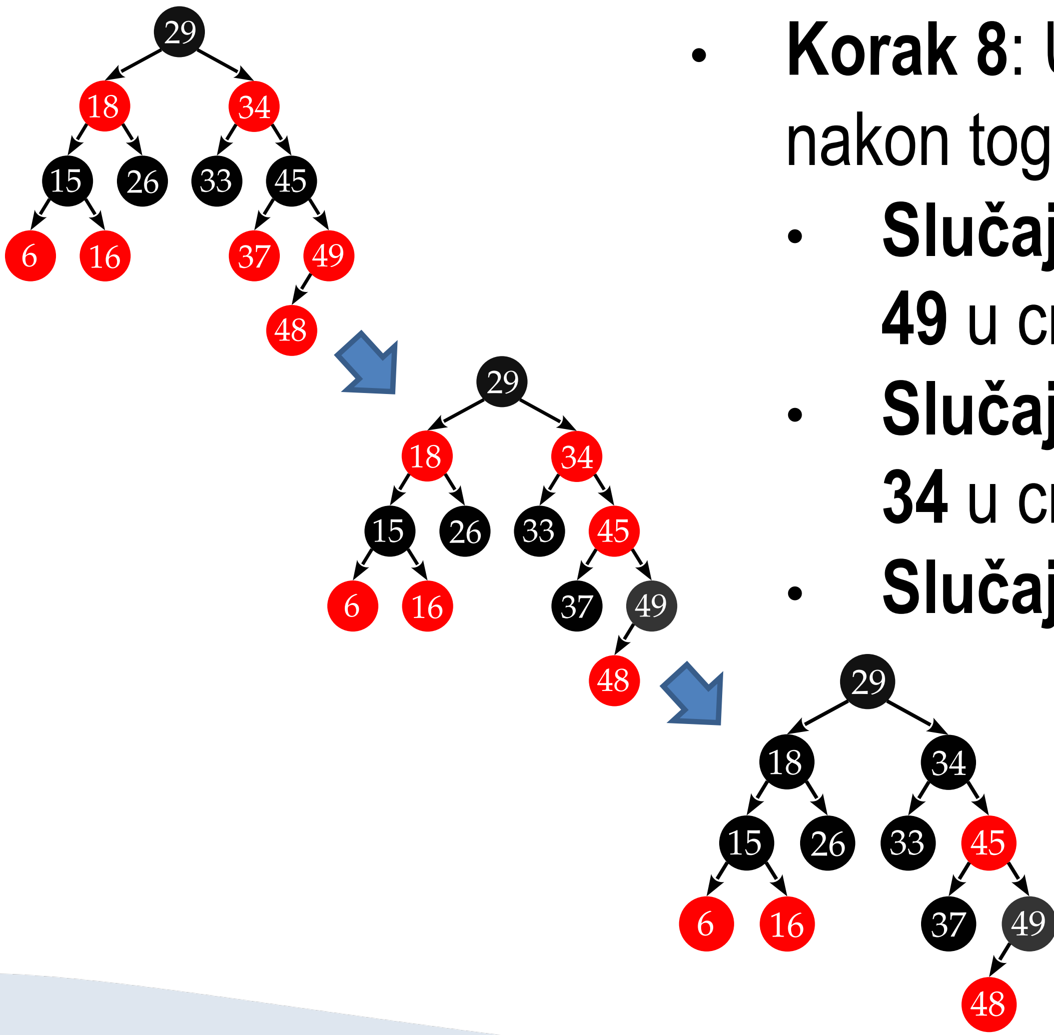
Primjer dodavanja podataka u RB-stablo



- **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

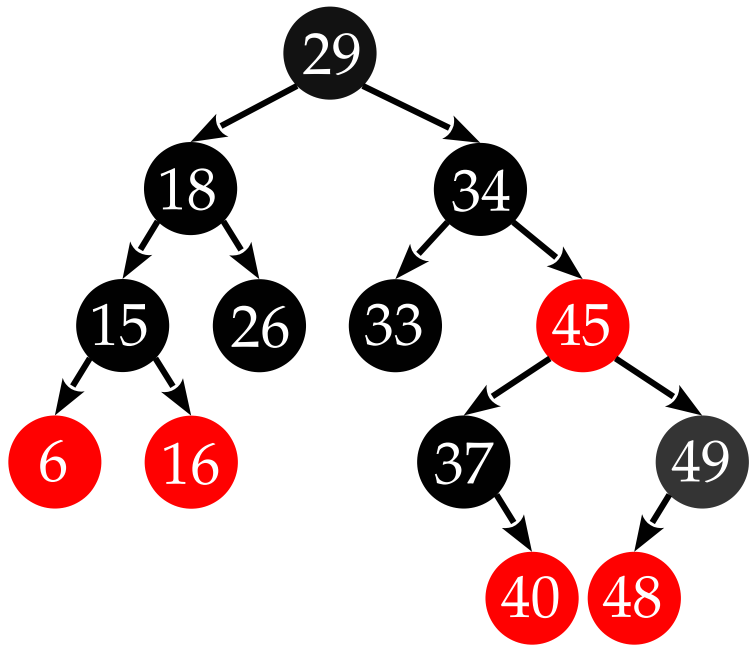


- **Korak 8**: U stablo dodajemo ključ **49**, te nakon toga ključ **48**.
  - **Slučaj 2**: Postavi **45** u crveno, a **37** i **49** u crno.
  - **Slučaj 2**: Postavi **29** u crveno, a **18** i **34** u crno.
  - **Slučaj 1**: Postavi korijen **29** u crno.

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

```
procedure RBTreeInsert(rbtree, N)
    P ← parent(N)
    G ← parent(P)
    while P is 🔴 do
        if P is the left child then
            case ← LL
            if N is the right child then
                case ← LR
            U ← the right child of G
        else
            case ← RR
            if N is the left child then
                case ← RL
            U ← the left child of G
        if U and P are 🔴 then
            P ← U ← ⚫
            G ← 🔴
            N ← G
        else if P is 🔴 and U is ⚫ then
            if case ∈ {LL, RR} then                    ▷ straight cases
                if case is LL then
                    right rotate P around G
                else
                    left rotate P around G
                switch P and G colors
                break
            else                                        ▷ broken cases
                if case is LR then
                    right rotate N around P
                else
                    left rotate N around P
                N ← P
        P ← parent(N)
        G ← parent(P)
    root(rbtree) ← ⚫                                    ▷ Rule 2
```

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.

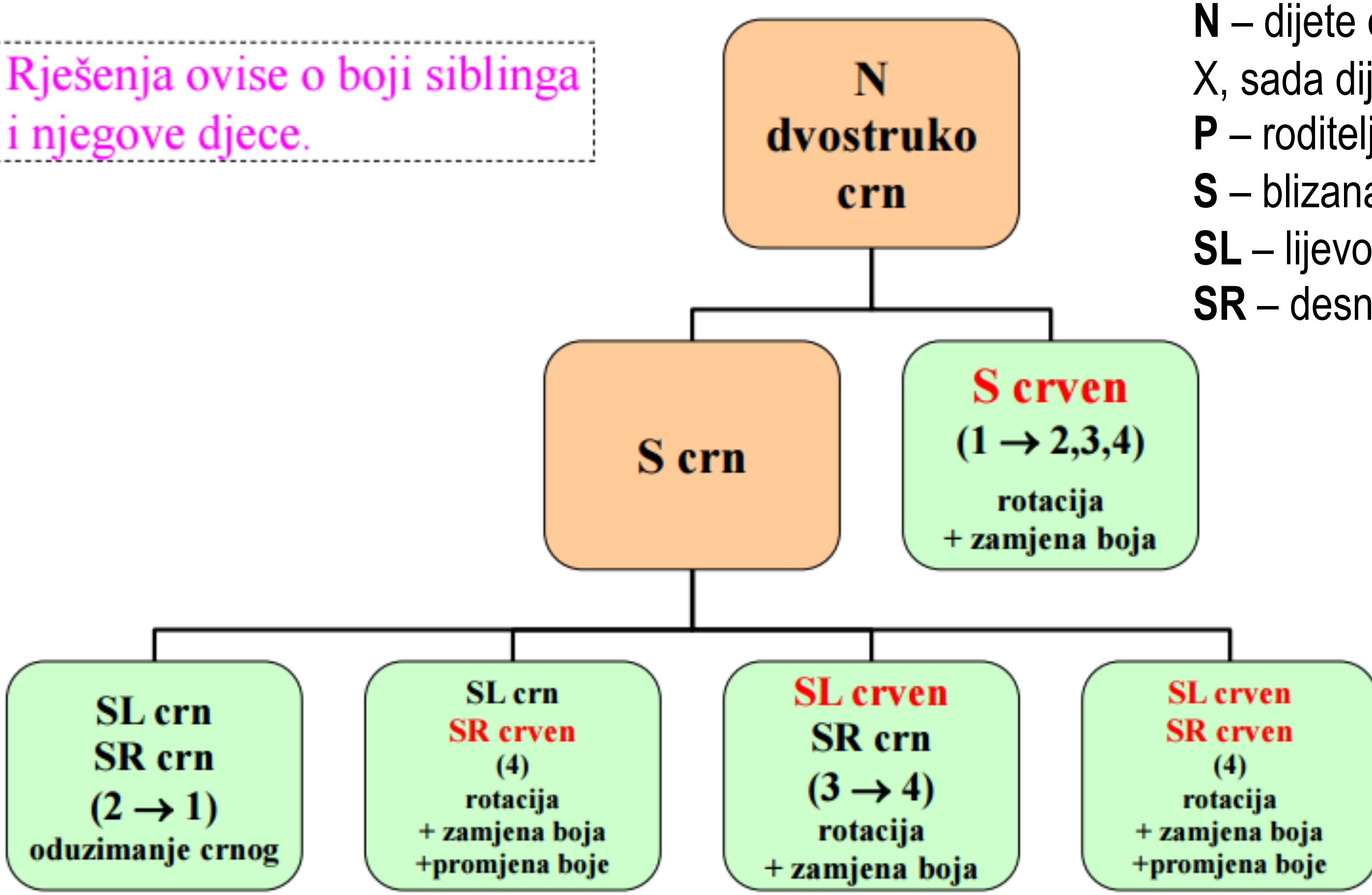
- Rješenje:
  - Ako je crveno-crn, dovoljno je prebojati ga u čisto crno.
  - Ako je dvostruko crn, ideja je proslijediti višak crnog prethodniku i tako taj višak podizati sve dok ne dođe na mjesto gdje ga možemo trajno ugraditi u stablo ili dok ne dođe u korijen gdje ga možemo zanemariti.

Uklanjanje crnog čvora (dvostruko crn)

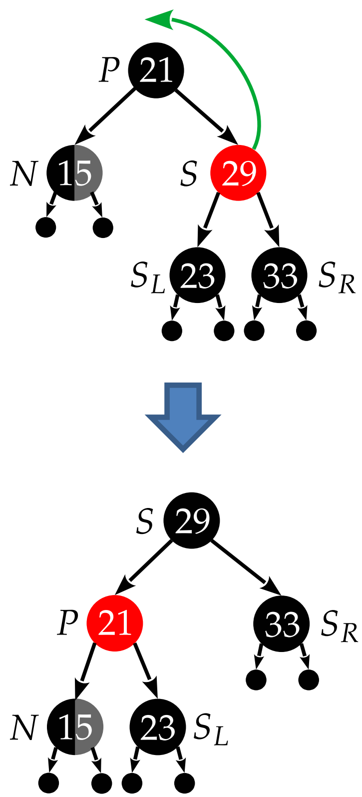
- 4 (+4 simetrična) su moguća slučaja, a ovise o boji čvora blizanca (dijete istog roditelja kao i N) i njegove djece

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

Rješenja ovise o boji siblinga i njegove djece.



Uklanjanje crnog čvora (dvostruko crn)

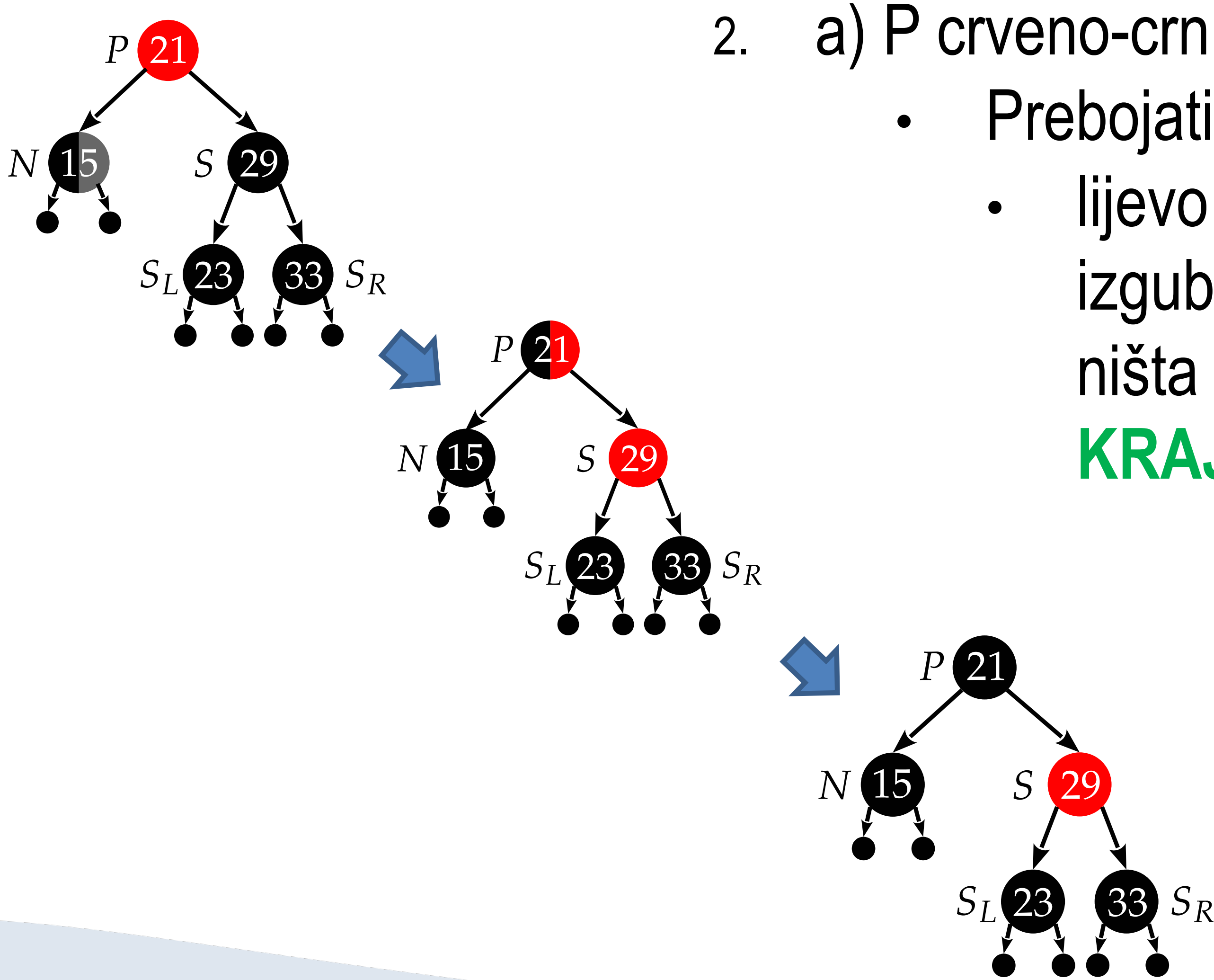


1. Blizanac S je crven
   - P je sigurno crn jer ima crveno dijete
   - Nakon brisanja X crna visina lijevog podstabla od P za jedan je manja od crne visine desnog podstabla (tj. N dvostruko crn)
   - Rješenje: rotirati S oko P (simetrija) pa zamijeniti boje P i S
   - **NASTAVAK** uravnotežavanja iz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

Uklanjanje crnog čvora (dvostruko crn)

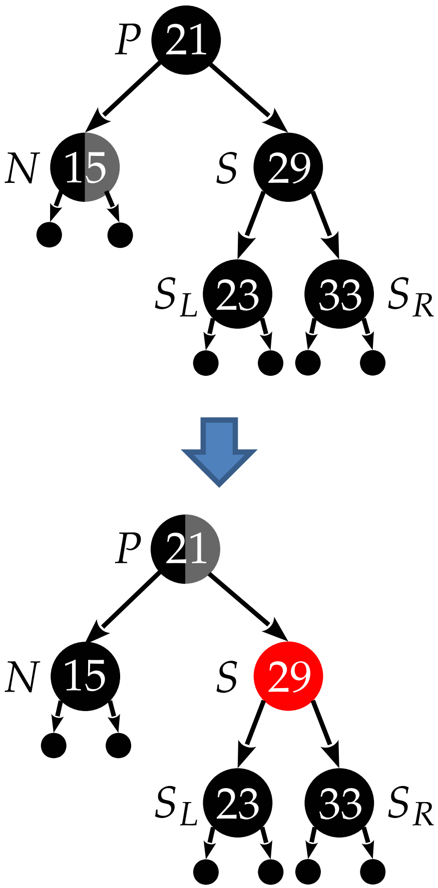
2. S crn, djeca od S crna
   - Oduzeti jedno crno N-u i S-u; N ostaje jednostruko crn, a S postaje crven
   - Taj višak crnoga proslijediti višoj razini (konvergencija!), tj. P-u koji time postaje ili crveno-crn ili dvostruko crn
   - O P-u ovisi postupanje nakon intervencije (slučajevi 2a i 2b):

Uklanjanje crnog čvora (dvostruko crn)



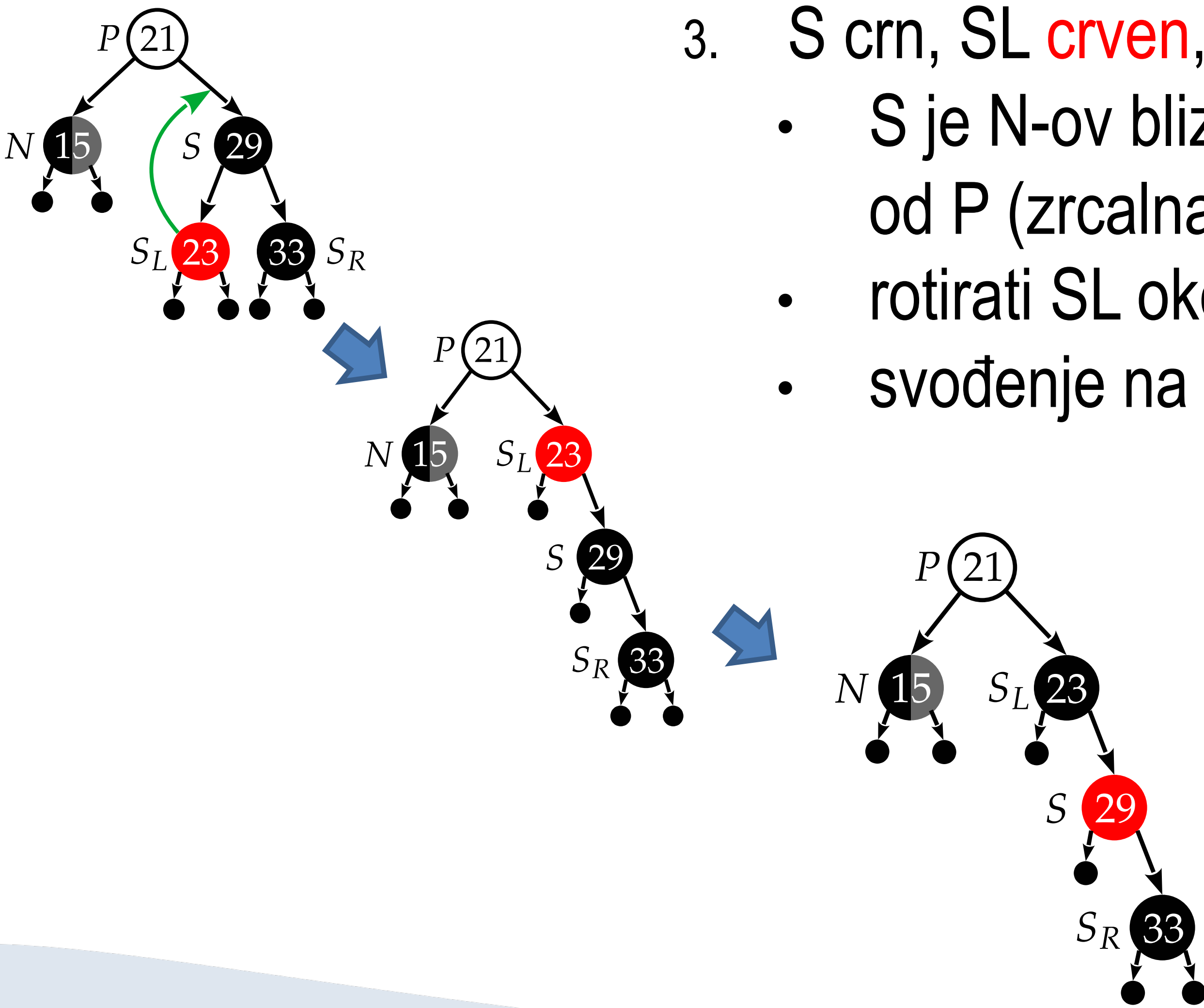
2. a) P crveno-crn
    - Prebojati P u crno
        - lijevo podstablo time dobiva izgubljeno crno, a desnom se ništa ne mijenja jer je S crven; **KRAJ**

Uklanjanje crnog čvora (dvostruko crn)



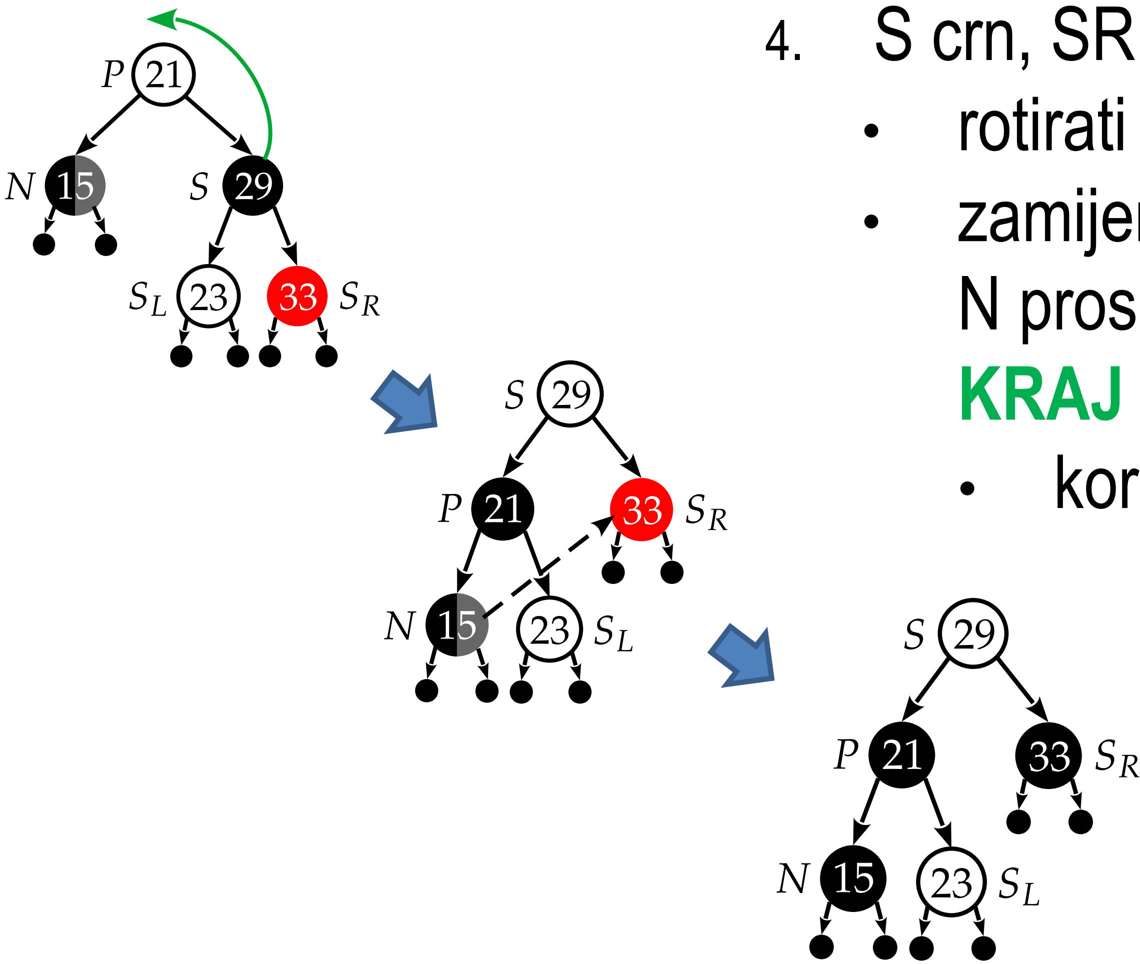
2. b) P dvostruko crn
   - P je korijen
     - višak crnog se odbacuje; **KRAJ**
   - P nije korijen
     - natrag na slučaj 1 promatrajući P kao N; **NASTAVAK**
     - problem je razinu više (konvergencija!)

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

```
procedure RBTreeRemove(rbtree, N)
    while N is not root(rbtree) and N is ● do
        P ← the parent of N
        if N is the left child of P then
            S ← the right child of P
            S_L, S_R ← children of S
            if S is 🔴 then
                S ← ●
                P ← 🔴
                left rotate S around P
            if S_L is ● and S_R is ● then
                S ← 🔴
                N ← the parent of N
            else
                if S_R is ● then
                    S_L ← ●
                    S ← 🔴
                    right rotate S_L around S
                color of S ← color of P
                P ← S_R ← ●
                left rotate S around P
                N ← root(rbtree)
        else                                ▷ implement the symmetrical cases
    N ← ●
```