# Strukture podataka

## Stabla: Crveno-crno, AVL, Proširena Crveno-Crna

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


DEPARTMAN ZA RAČUNARSTVO I AUTOMATIKU

DEPARTMAN ZA ENERGETIKU, ELEKTRONIKU I KOMUNIKACIJE

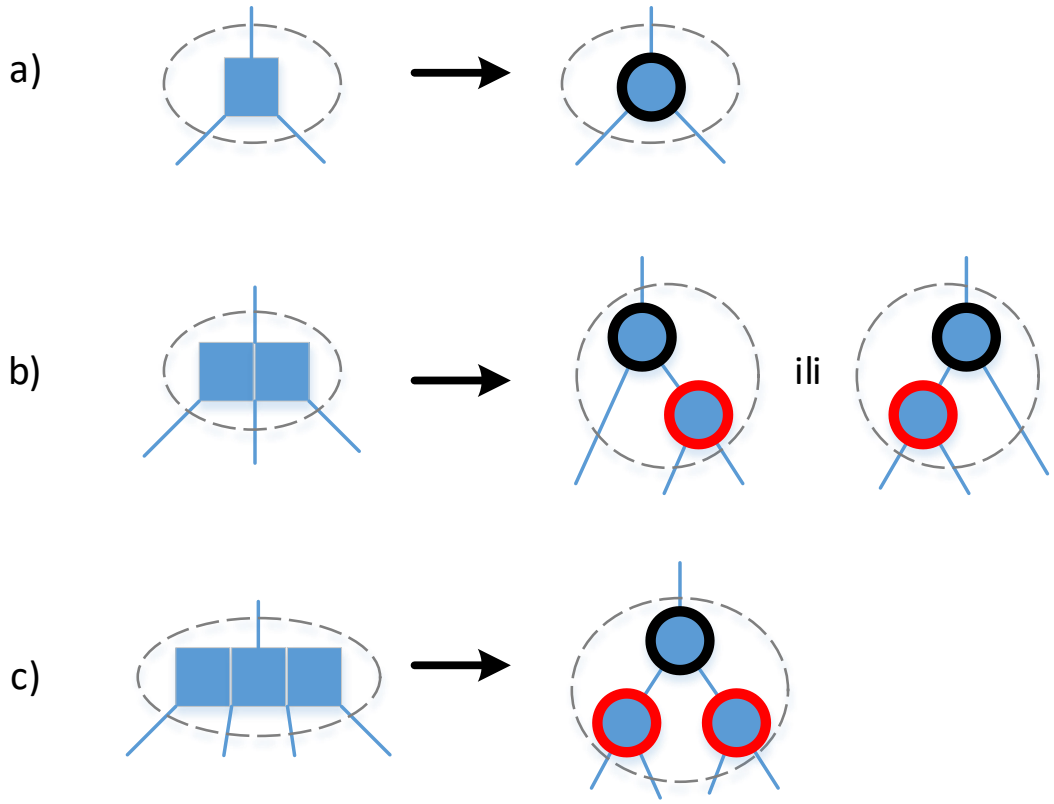
# Crveno-crno (CC) stablo

- ... je u osnovi sa osobinama i strukturom BSP
  - Dodatno, svaki čvor ima boju: crnu ili crvenu
- ... je stablo praktičnije od 2-3-4 stabla
  - 2-3-4 stablo nije binarno – ima više ključeva u čvoru, više poređenja, ... $\rightarrow$ složenija implementacija
- ... nije idealno balansirano kao što 2-3-4 jeste (da je jednako dugačka putanja od korena do svakog lista), ali ima osobinu da su sve dužine putanja do listova unutar opsega sa faktorom 2, tj. da je najduža putanja „samo“ do 2 puta duža od najkraće putanje.
  - To je dovoljno dobro i kažemo da je crveno-crno stablo **aproksimativno balansirano**.
- Kako je operacije (dodavanja i brisanja čvora, pretrage, minimuma, maksimuma, prethodnog i narednog) izvršavaju u vremenu $O(h)$, za balansirano stablo to je $O(\log_2 n)$

# Konverzija 2-3-4 čvora u binarne

- Posmatramo sve moguće slučajeve veličine čvorova u 2-3-4 stablu:
  a) 1 ključ
  b) 2 ključa
  c) 3 ključa

  i pravi se analogija klasteru čvorova.
- Klaster ima 1 crni čvor u vrhu i ostale crvene
  - Svakom nivou 2-3-4 stabla odgovara jedan crni čvor
- Broj grana koje dodaje svaki klaster (nivo) u putanju od korena do lista je najviše jednak broju nivoa u 2-3-4 stablu
  - Potvrđuje da je max dužina putanje $\leq$ 2 puta putanja u 2-3-4 stablu

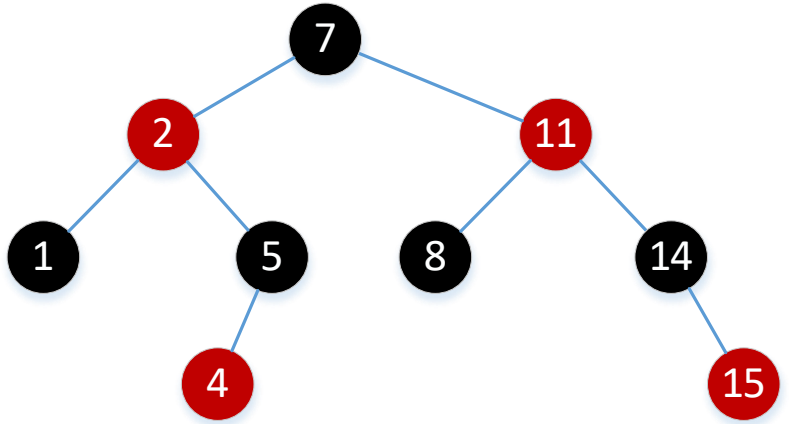


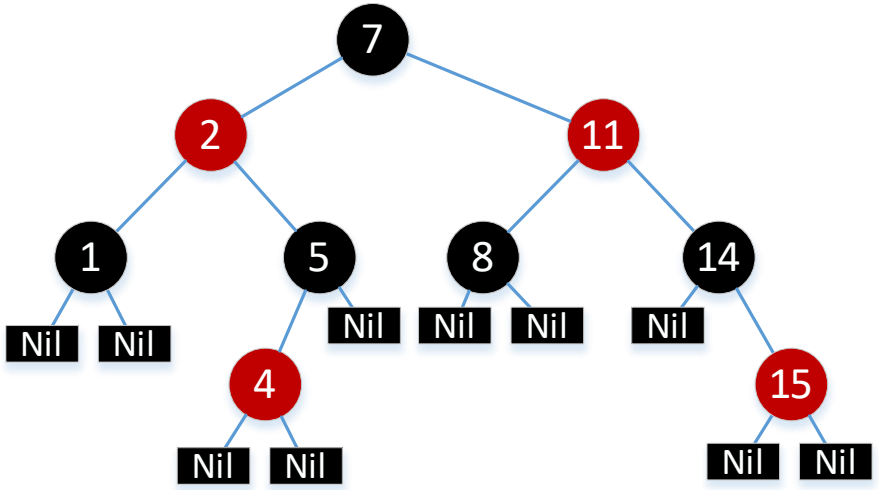
# Pravila CC stabla

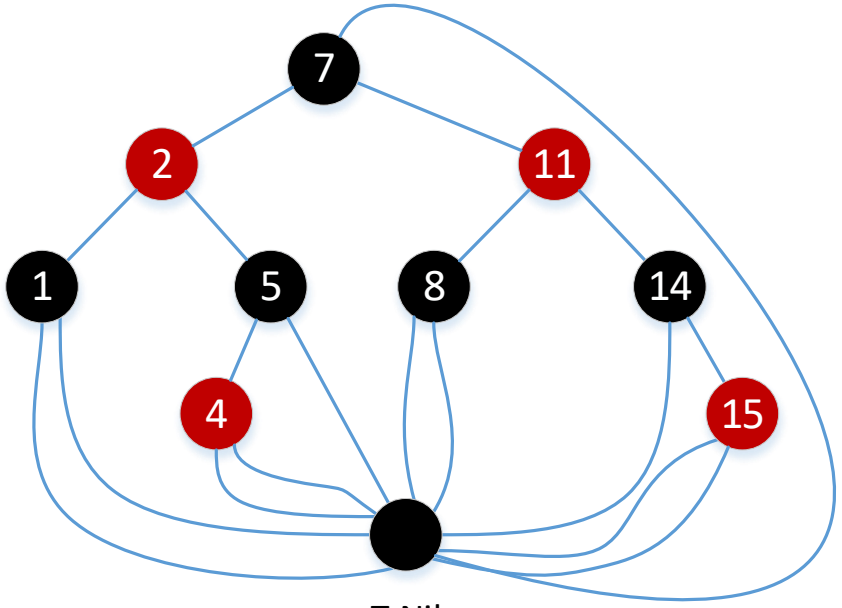
1. Svaki čvor je crn ili crven
2. Čvor u korenu je crn
3. Svi listovi su crni i ne sadrže ključeve/vrednosti
   - To je kao dodatni sloj – kao „dekoracija“
4. Ako je čvor crven oba deteta su crna
   - putanja nikada nema dva susedna crvena čvora
5. Za svaki čvor, sve putanje od njega do listova (u njegovom podstablu) su sa jednakim brojem crnih čvorova

- Ova pravila moraju biti ispunjena i nakon dodavanja/brisanja čvorova
  - npr. ako se ukloni crni čvor na putanji ...crveni-crni-crveni... pravilo 4. se naručava!
- Da bi se očuvale gornje osobine potrebno je transformacijama preurediti strukturu stabla:
  - rotacijama i/ili
  - promenama boje nekih čvorova

# Primeri predstave stabla



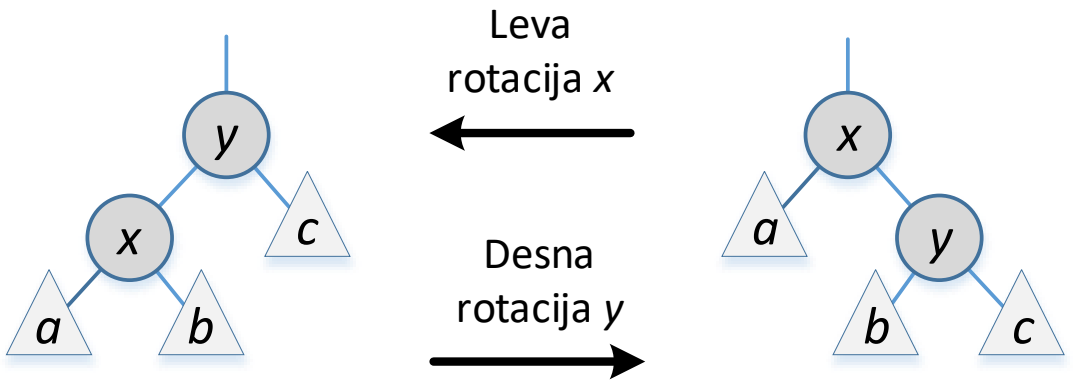




T.Nil

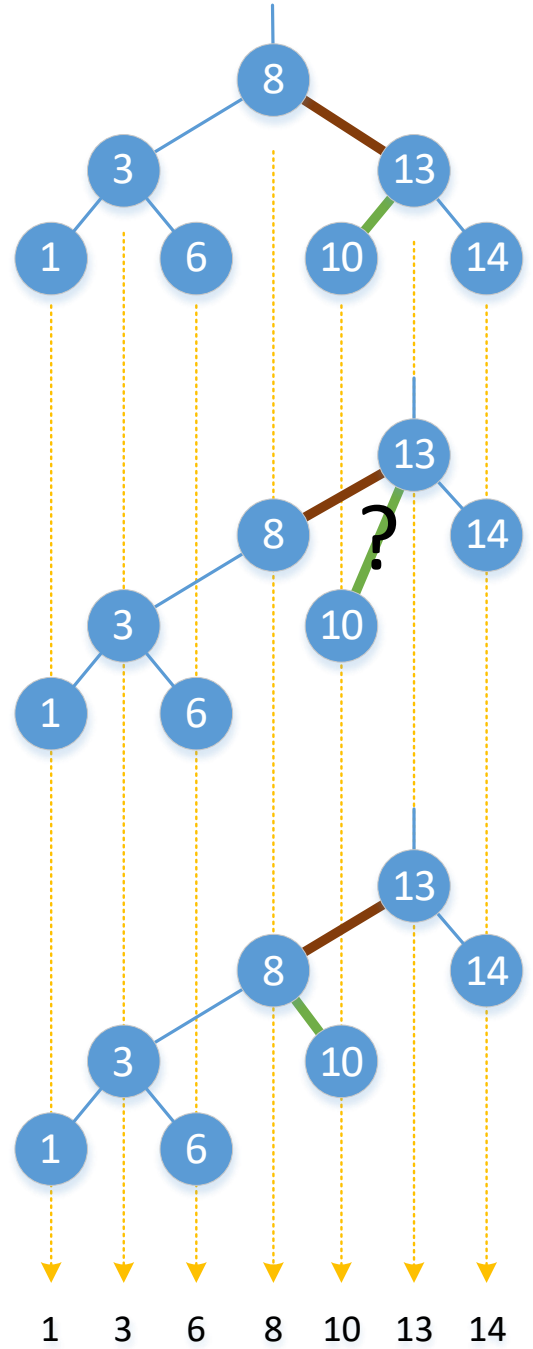
# Rotacije

- Rotira se grana (npr. *x*-*y*), tj. roditelj (*y*) i dete (*x*) zamenjuju mesta
  - Posledica je da se desno podstablo *b* (čvora *x* promovisanog u roditelja) prevezuje kao levo podstablo prethodnog roditelja (*y*)
- Analogno se može izvršiti suprotna rotacija




Posmatra se čvor koji se „spušta“ sa leve/desne strane

```
ROTIRAJ-LEVO(x)
1    y = x.desno
2    x.desno = y.levo
3    if y.levo ≠ T.Nil
4        y.levo.r = x
5    y.r = x.r
6    if x.r == T.Nil
7        T.koren = y
8    elseif x == x.r.levo
9        x.r.levo = y
10   else
11       x.r.desno = y
12   y.levo = x
13   x.r = y
```



# Dodavanje

- Dodavanje čvora se vrši na dno – uvek.
- Kada posmatramo neku putanju broj crnih čvorova bi trebalo održavati malim
  - Ako pokušamo da dodamo crni čvor, to ne remeti pravila, ali time generišemo dodatni nivo, što je loše
  - Zato dodamo crveni čvor, ali možemo imati „problem“ kada se nadovezuje na postojeći crveni čvor
- Mogu se pojaviti 3 nedozvoljene situacije koje se rešavaju korekcijama (transformacijama).
  - Poziv metode KORIGUJ-DODAJ
- Radi pojednostavljenja koda ubačen je jedan čuvar T.Nil u celo stablo i sve Nil vrednosti ukazuju na njega.

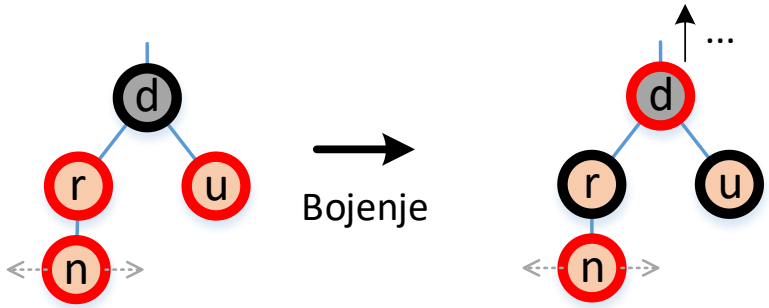
```
DODAJ(T, z)
1   y = T.Nil
2   x = T.koren
3   while x ≠ T.Nil
4       y = x
5       if z.ključ < x.ključ
6               x = x.levo
7       else    x = x.desno
8   z.r = y
9   if y == T.Nil
10      T.koren = z
11  elseif z.ključ < y.ključ
12          y.levo = z
13  else    y.desno = z
14  z.levo = T.Nil
15  z.desno = T.Nil
16  z.boja = CRVENA
17  KORIGUJ-DODAJ(T, z)
```

# Pre dodavanja čvora

- Stablo je pravilno – ispunjava svih 5 osobina
- Nakon dodavanja (crvenog čvora) je možda narušena neka od osobina
  - Ako je prazno stablo narušena je osobina #2
  - Ako nije prazno stablo jedino osobina #4 može biti narušena
- Narušena osobina se ispravlja
  - Rotacija – složenost operacije $O(1)$
  - promena boje nekih čvorova – složenost operacije $O(1)$

# Korekcije nakon dodavanja – slučajevi

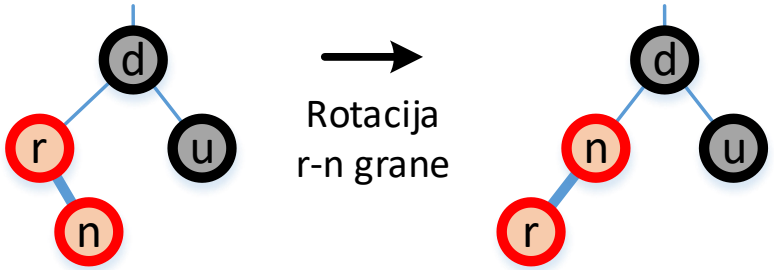
a)




Jedan nivo crnih, kao na početku, ali potencijalno se vezuje na crvenog pradedu (nastavlja se korekcijama)

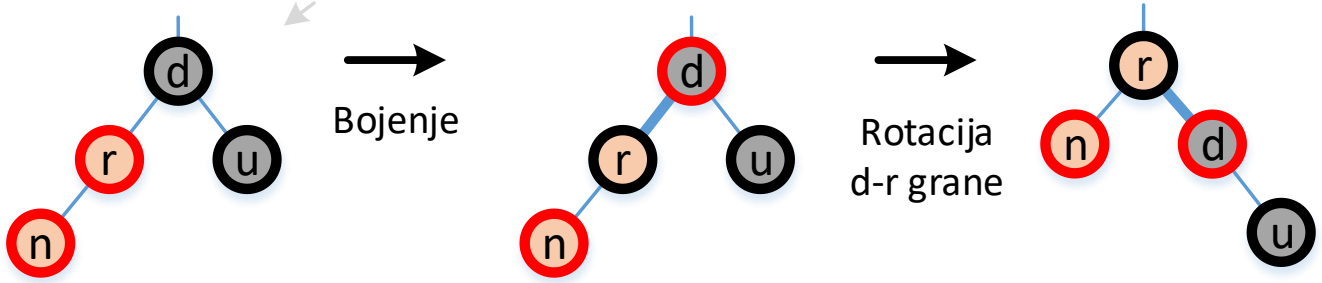
Prikazane transformacije imaju ujaka sa desne strane.
Analogne transformacije postoje kada je ujak sa level strane.

b)




Ovo je slučaj c)

c)




Dva nivoa crnih, kao na početku

**Legenda:**
n – novi čvor
r – roditelj
d – deda
u – ujak

# Korekcije nakon dodavanja

- Napomena: čvorovi koji se posmatraju imaju svoje potomke (podstabla)
- Incijalni slučaj: dodati čvor nema dece, ali u naknadnim korekcijama posmatrani čvorovi mogu imati decu (u svim razmatranim slučajevima)
- Ima 6 slučajeva: razmatramo 3 slučaja kada je „ujak“ desno, i 3 slučaja kada je on levo
  - b) se nastavlja u c)
  - c) završava korekcije
  - a) potencijalno se nastavlja od „dede“ (ako je dedin roditelj crven) ili se prekida (dedin roditelj je crn)
    - To znači „penjanje“ za po dva nivoa, možda sve do korena
      - Ako se boja korena promeni u crvenu (denin-dedin-..deda je postao crven) red #17 je postavlja na crnu
    - Može se nastaviti nekim od slučajeva a), b) ili c)

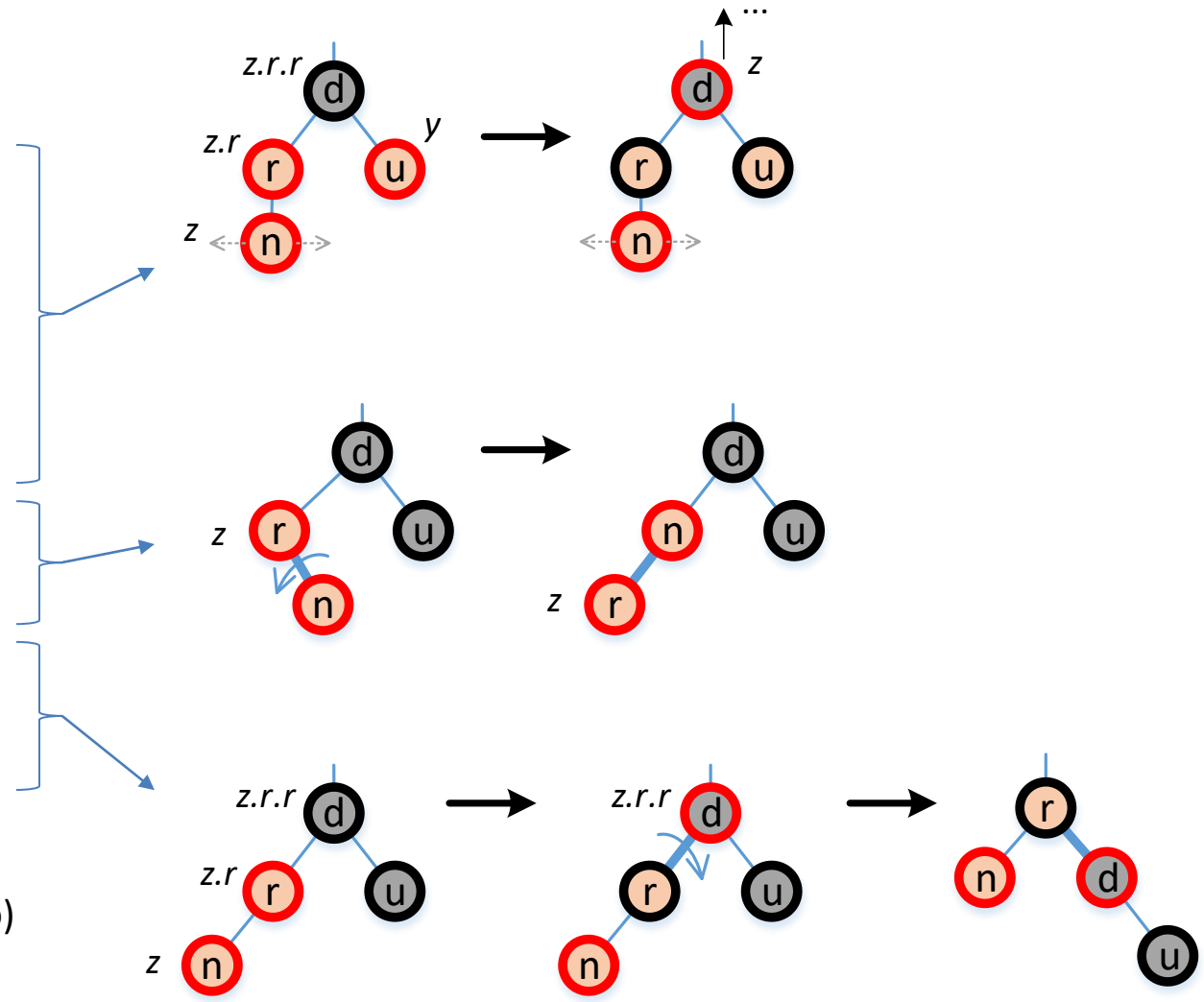
# Korekcije nakon dodavanja

- Dodaje se crveni čvor.
- Ako je roditelj crn ne treba korigovanje
- Mađutim, narušena se pravila ako je to:
  - Prvi čvor (koren) u stablu
  - Ako je roditelj dodatog čvora crven

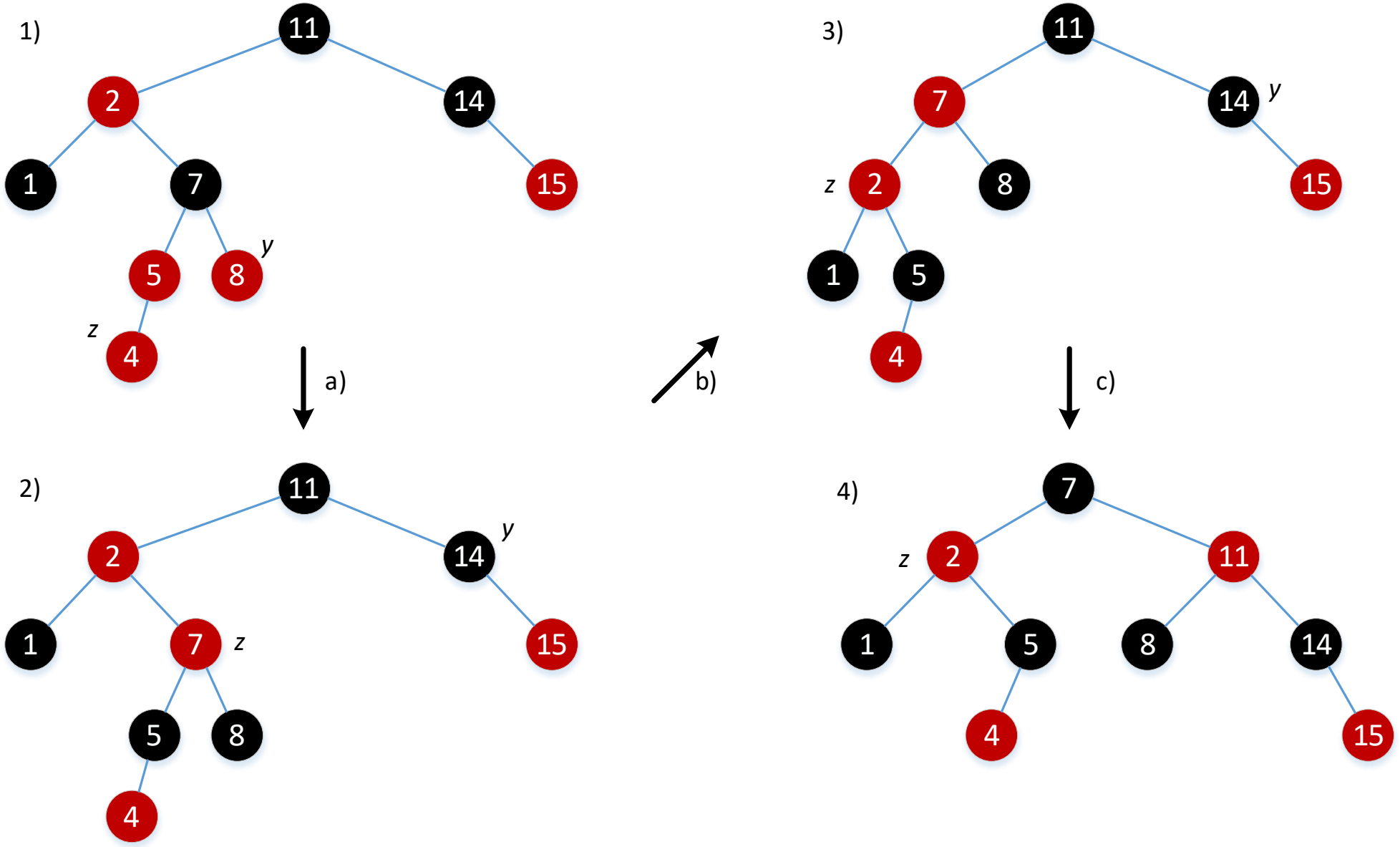
```
KORIGUJ-DODAJ(T, z)
1   while z.r.boja == CRVENA
2       if z.r == z.r.r.levo
3           y = z.r.r.desno          // ujak
4           if y.boja == CRVENA
5               z.r.boja = CRNA
6               y.boja = CRNA
7               z.r.r.boja = CRVENA
8               z = z.r.r   // nastavi sa dedom
9           else if z == z.r.desno
10                  z = z.r
11                  ROTIRAJ-LEVO(T, z)
12              z.r.boja = CRNA
13              z.r.r.boja = CRVENA
14              ROTIRAJ-DESNO(T, z.r.r)
15      else
16          … roditelj je desno dete
            (isto kao gore, ali se zamene atributi levo/desno)
17    T.koren.boja = CRNA
```

# Korekcije nakon dodavanja (2)

```
KORIGUJ-DODAJ(T, z)
1   while z.r.boja == CRVENA
2       if z.r == z.r.r.levo
3           y = z.r.r.desno          // ujak
4           if y.boja == CRVENA
5               z.r.boja = CRNA
6               y.boja = CRNA
7               z.r.r.boja = CRVENA
8               z = z.r.r   // nastavi sa dedom
9           else if z == z.r.desno
10                  z = z.r
11                  ROTIRAJ-LEVO(T, z)
12              z.r.boja = CRNA
13              z.r.r.boja = CRVENA
14              ROTIRAJ-DESNO(T, z.r.r)
15      else
16          … roditelj je desno dete
            (isto kao gore, ali se zamene atributi levo/desno)
17  T.koren.boja = CRNA
```



# Primer


1)
11
2
14
1
7
15
5
8
y
z
4
a)
2)
11
2
14
y
1
7
z
15
5
8
4
b)
3)
11
7
14
y
z
2
8
15
1
5
4
c)
4)
7
z
2
11
1
5
8
14
4
15

# Složenost operacije dodavanja

- Ubacivanje novog čvora $O(\log n)$
- Potencijalna korekcija
  - Slučaj a): promena boja najviše u $(\log n)\ /2$ nivoa + slučaj b) ili c)
  - Slučaj b): 1 rotacija i bojenje + slučaj c)
  - Slučaj c): 1 rotacija i bojenje

- Ukupna složenost je $O(\log n)$

# Zamena čvora drugim čvorom

- I ovde se zamena čvora koristi kod brisanja čvora
- Zamena iz BSP je ovde prilagođena ...
  - Nil je zamenjen sa T.Nil
  - uslov **if** *y* ≠ Nil je sklonjen iz 6. reda jer postoji *y.r* i kada *y* pokazuje na „čuvara“.

```
ZAMENA(T, x, y)
1  if x.r == T.Nil
2     T.koren = y
3  elseif x == x.r.levo
4        x.r.levo = y
5  else x.r.desno = y
6  if y ≠ Nil
7  y.r = x.r
```

# Brisanje čvora

- Slično je BSP brisanju čvora, ali se mora voditi briga o pravilima CC stabla
- Da se ne menja broj crnih čvorova na putanjama vrše se korekcije:
  - kada se briše crni čvor koji ima manje od dva deteta: crna boja se „prenosi na dete“ $x$ ili
  - kada se briše čvor koji ima oba deteta i zamenjuje se narednim čvorom koji je crn.
    Naredni čvor dolazi na mesto obrisanog i dobija boju obrisanog (red 20), ali ako mu je prethodna boja bila crna prenosi se na njegovo dete $x$

- U oba slučaja $x$ čvor se dodatno opterećuje prenetom crnom bojom:
  - ako je crven onda treba da postane crn,
  - ako je crn onda je „duplo crn“ i korekcijama se to dodatna crna prenosi na druge čvorove...

```
OBRIŠI(T, z)
1   y = z
2   bojay = y.boja
3   if z.levo == T.Nil
4      x = z.desno
5      ZAMENA(T, z, z.desno)
6   elseif z.desno == T.Nil
7      x = z.levo
8      ZAMENA(T, z, z.levo)
9   else y = MINIMUM(z.desno)
10     bojay = y.boja
11     x = y.desno
12     if y.r == z
13        x.r = y
14     else ZAMENA(T, y, y.desno)
15        y.desno = z.desno
16        y.desno.r = y
17     ZAMENA(T, z, y)
18     y.levo = z.levo
19     y.levo.r = y
20     y.boja = z.boja
21  if bojay == CRNA
22     KORIGUJ-OBRIŠI(T, x)
```

# Korekcija brisanja

$x$ je levo dete i ako je crven onda pocrni (red 23), inače: postoje 4 slučaja, gde je njegov brat (sestra) $w$:

a) crven $w$ (deca su mu uvek crna!) – transformiše se slučaj b), c) ili d)

b) crn $w$ i njegova deca su crna - $w$ postaje crven, a dodatno crno od $x$ se prebacuje na roditelja (petlja 1)

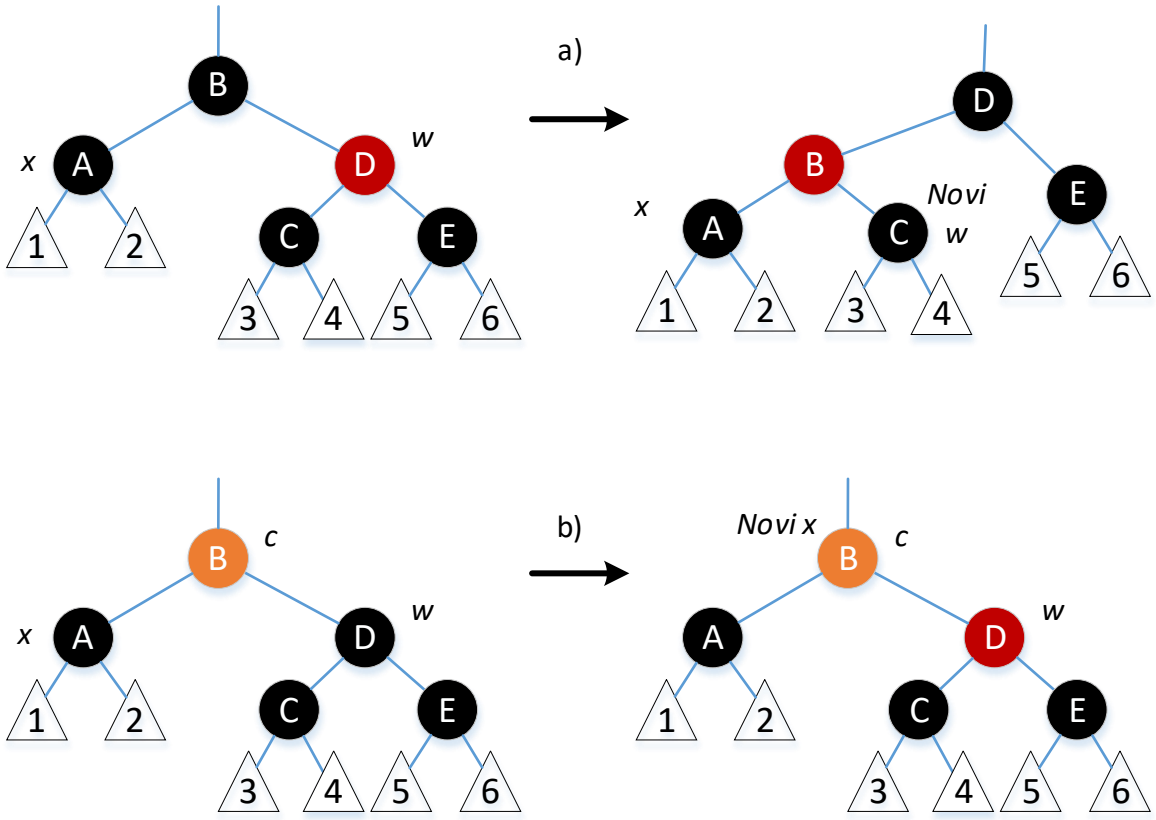
c) crn $w$ i njegovo levo dete je crveno, a desno crno - transformiše se u slučaj d)

d) crn $w$ i njegovo desno dete je crveno (a levo crno ili crveno) – rotiraju se $w$ i roditelj i zamene boje, a desno dete pocrni.
Ovde se prekidaju dalje korekcije jer je $x$ preneo dodatno crno na roditelja (na putanji do njega je dodatni crni čvor).

Kada je $x$ desno dete ista 4 slučaja zahtevaju simetrične korekcije.

```
KORIGUJ-OBRIŠI(T, x)
1   while x ≠ T.koren & x.boja == CRNA
2      if x == x.r.levo
3         w = x.r.desno
4         if w.boja == CRVENA
5            w.boja = CRNA
6            x.r.boja = CRVENA
a)       ROTIRAJ-LEVO(T, x.r)
8            w = x.r.desno
9         if w.levo.boja == CRNA & w.desno.boja == CRNA
b)          w.boja = CRVENA
11          x = x.r
12        else if w.desno.boja == CRNA
13              w.levo.boja == CRNA
c)            w.boja = CRVENA
15            ROTIRAJ-DESNO(T, w)
16            w = x.r.desno
17          w.boja = x.r.boja
18          x.r.boja = CRNA
d)          w.desno.boja = CRNA
20          ROTIRAJ-LEVO(T, x.r)
21          x = T.koren
22     else (…isto kao gore, ali levo <-> desno)
23  x.boja = CRNA
```

# Slučajevi korekcije brisanja




Narandžasti čvorovi imaju oznaku *c* i boja im je crna ili crvena.

# Korekcija brisanja – još jednom

- Kada je obrisan crveni čvor koji ima jedno dete – nema korekcije
- Kada je obrisan crni čvor koji ima jedno dete – dete pocrni (red 23)
- Kada je obrisan čvor sa dva deteta naredni čvor $y$ dolazi na njegovo mesto i poprima njegovu boju ($y$ preuzima boju od $z$ i nema narušavanja pravila kod pozicije $z$), ali potencijalno pravi problem na mestu na kome se nalazio.
  - Ako je $y$ bio crven – nema korekcija
  - Ako je $y$ bio crn onda ima potencijalna 3 problema:
    1. čvor $y$ je bio koren sa crvenim detetom $x$ koje nakon premeštanja $y$ postaje crveni koren – $x$ pocrni (red 23)
    2. čvor $x$ (koji je dete $y$) se postavlja kao dete od $y.r$, i oba čvora $x$ i $x.r$ mogu biti crvena – $x$ pocrni (red 23)
    3. čvor $x$ putanja ima jedan crni čvor manje
       - Ako je $x$ crven rešenje je promeniti mu boju u crnu (red 23)
       - Ako je $x$ crn onda je „duplo crn“ i dodatnu crnu (od brisanja $y$) treba ukloniti – rešava se slučajevima korekcije brisanja:
         a) konvertuje se u neki od slučajeva: b, c) ili d)
         b) čvor $w$ postaje crven, a crna se „seli“ sa $x$ i $w$ na (B) čvor
            - Ako je (B) bio crven postaje crn i korekcije se zaustavljaju
            - Ako je (B) bio crn postaje duplo crn i konverzije se nastavljaju od roditeljskog nivoa ($x.r$)
         c) Konvertuje u d)
         d) Rotacija + (B) čvor preuzima extra crnu od $x$, a (D) preuzima boju od (B). Korekcije se ovde zaustavljaju.

# Složenost operacije brisanja

- Brisanje čvora sadrži zamenu $O(1)$ i potencijalno traženje narednog čvora $O(\log n)$
- Potencijalna korekcija sadrži operacije rotacija $O(1)$ i zamena boja $O(1)$.
  Od mogućih slučajeva samo b) se prenosi na roditeljski nivo i u najgorem slučaju se korekcije mogu proteći do korena stabla, tj. $O(\log n)$ nivoa.

- Ukupna složenost je $O(\log n)$

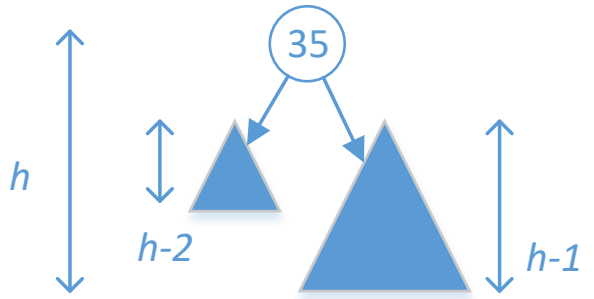
# AVL stablo

- Adelson-Velsky & Landis-ovo stablo = samobalansirajuće BSP
- Balansira se visina stabla
  Glavna osobina: **za svaki čvor razlika visina levog i desnog podstabla je najviše $\pm 1$**

- Najgori slučaj: desno podstablo je za 1 više od levog podstabla za svaki čvor (ili obratno).
  - Posmatramo minimalan broj čvorova na visini $h$

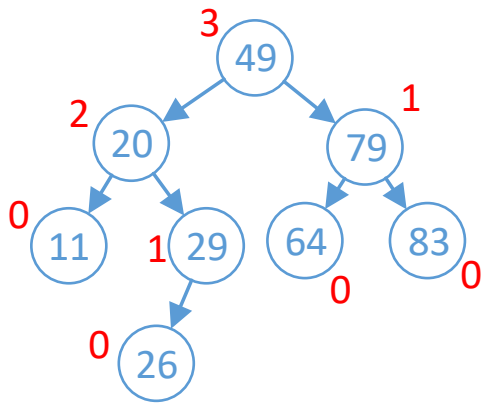
$$N_h = N_{h-2} + 1 + N_{h-1} > 1 + 2N_{h-2} > 2N_{h-2}$$

  - Rekurzija daje $N_h > 2^{h/2}$, odakle je $h < 2 \log_2 N_h$
  - Tačan rezultat: $h \leq 1.440 \log_2 n$
- Visina AVL stabla je $h = O(\log_2 n)$
  gde je $n$ broj čvorova u stablu



# Visina čvora

- Svaki čvor pamti svoju visinu kao dodatni atribut
- Visina čvora je dužina (broj grana) najduže putanje do lista




- Visina lista $= 0$ (visina nepostojećeg potomka, označenog sa Nil, je $-1$)
- Visina svakog drugog čvora

$$= \max(visina\ levog\ podstabla, visina\ desnog\ podstabla) + 1$$

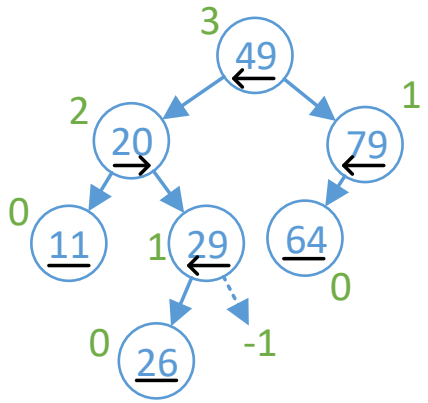
# Dodavanje čvora u AVL stablo

Ima dva koraka:

1. Ubaciti čvor kao da je BSP uz ažuriranje visina čvorova
2. Ako je narušena balansiranost – primeniti rotacije (vidi CC stabla)
   AVL balansiranost je narušena u čvoru gde je razlika visina njegovih podstabla veća od 1

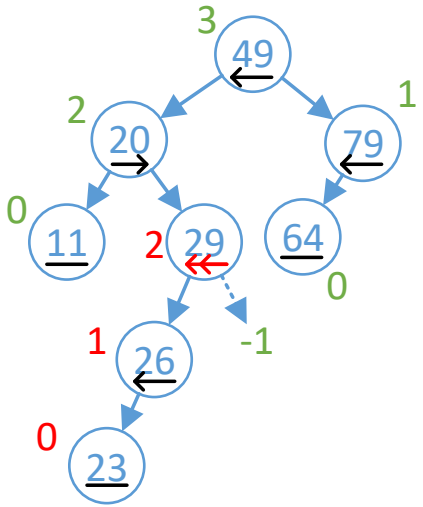
Primeri:

- Polazi se od „korektnog“ AVL stabla (primer desno)
- Dodaju se čvorovi koji narušavaju osnovnu osobinu balansiranosti
- Postoje dva slučaja, koje ilustruju:
  - `DODAJ(23)`
  - `DODAJ(70)`

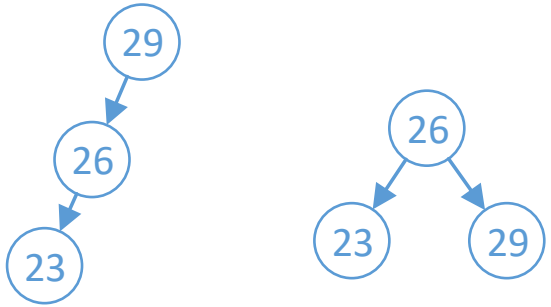


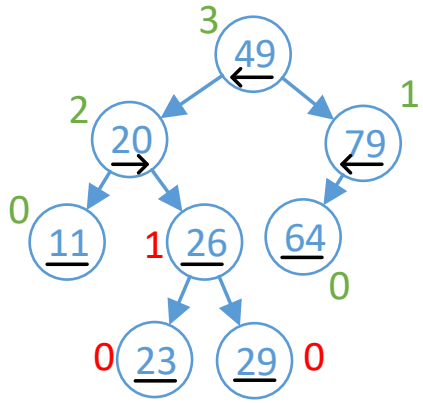
# Slučaj 1. dodavanja čvora u AVL stablo

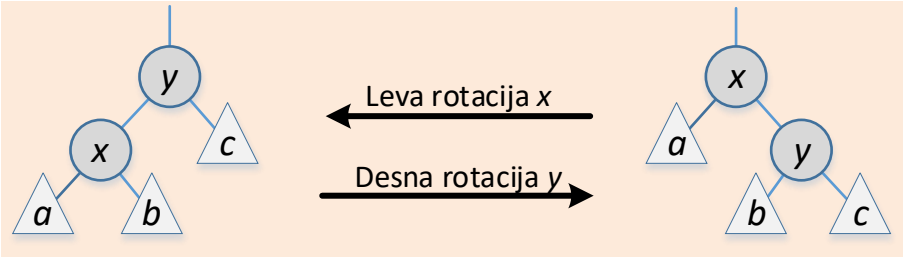
- Primer: DODAJ(23)
  Ovaj slučaj rešava jedna rotacija.




ROTIRAJ-DESNO(29)

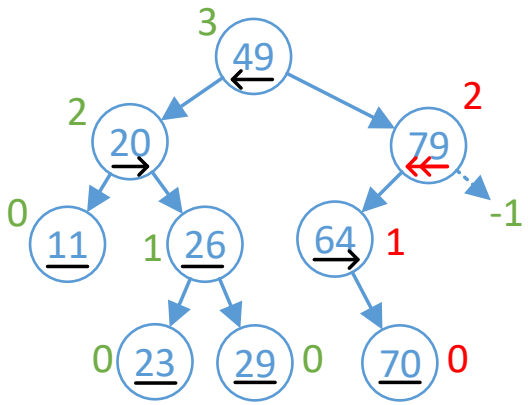


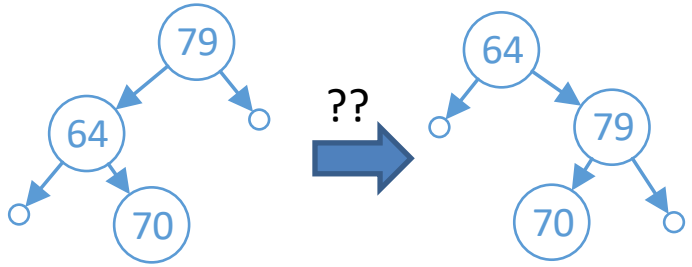



# Slučaj 2. dodavanja čvora u AVL stablo

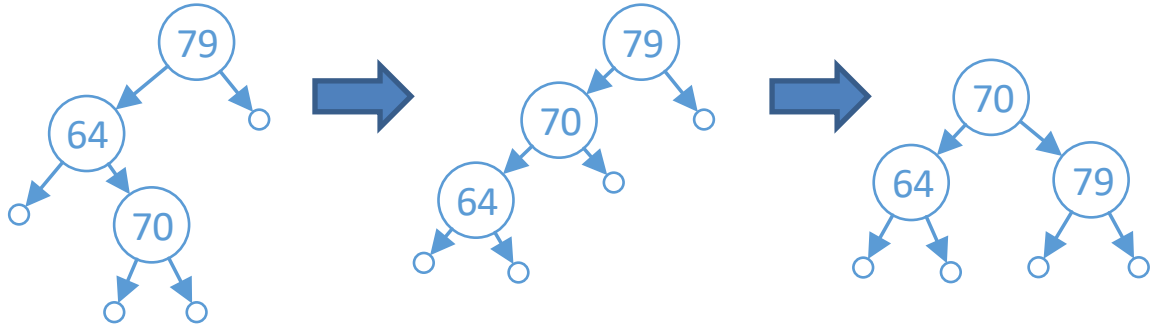
- Primer: DODAJ(70)
  Ovaj slučaj ne rešava jedna rotacija.
  Potrebne su dve rotacije.




ROTIRAJ-DESNO(79)




Pogrešno




Ispravno

ROTIRAJ-LEVO(64)  ROTIRAJ-DESNO(79)

# Brisanje čvora iz AVL stabla

Ima dva koraka (slično dodavanju):

1. Obrisati čvor kao da je BSP uz ažuriranje visina čvorova
   Obratiti pažnju da zamena čvorova menja visinu (metoda ZAMENI)
2. Ako je narušena balansiranost – primeniti rotacije

# Algoritam za popravljanje balansiranosti AVL stabla

- Neka je $x$ najniži čvor gde je narušena AVL balansiranost
- Neka je $y$ desna strana od $x$ čija je visina veća
- Kada je desna strana od $y$ viša ili je $y$ balansiran čvor uraditi ROTIRAJ-LEVO($x$)
  u suprotnom (leva strana od $y$ je $z$ i ona je viša) uraditi dve rotacije:
  ROTIRAJ-DESNO($y$) i ROTIRAJ-LEVO($x$)
- Ukoliko rotacije narušavaju balansiranost roditeljskog čvora (pre rotacije(a) je to bio roditelj od $x$) primeniti algoritam na njega i tako sve do korena stabla – dokle god je narušena balansiranost čvorova.

- Opisani postupak treba prilagoditi kada je $y$ leva strana od $x$ čija je visina veća

# Primena rotacija

Ponovo:

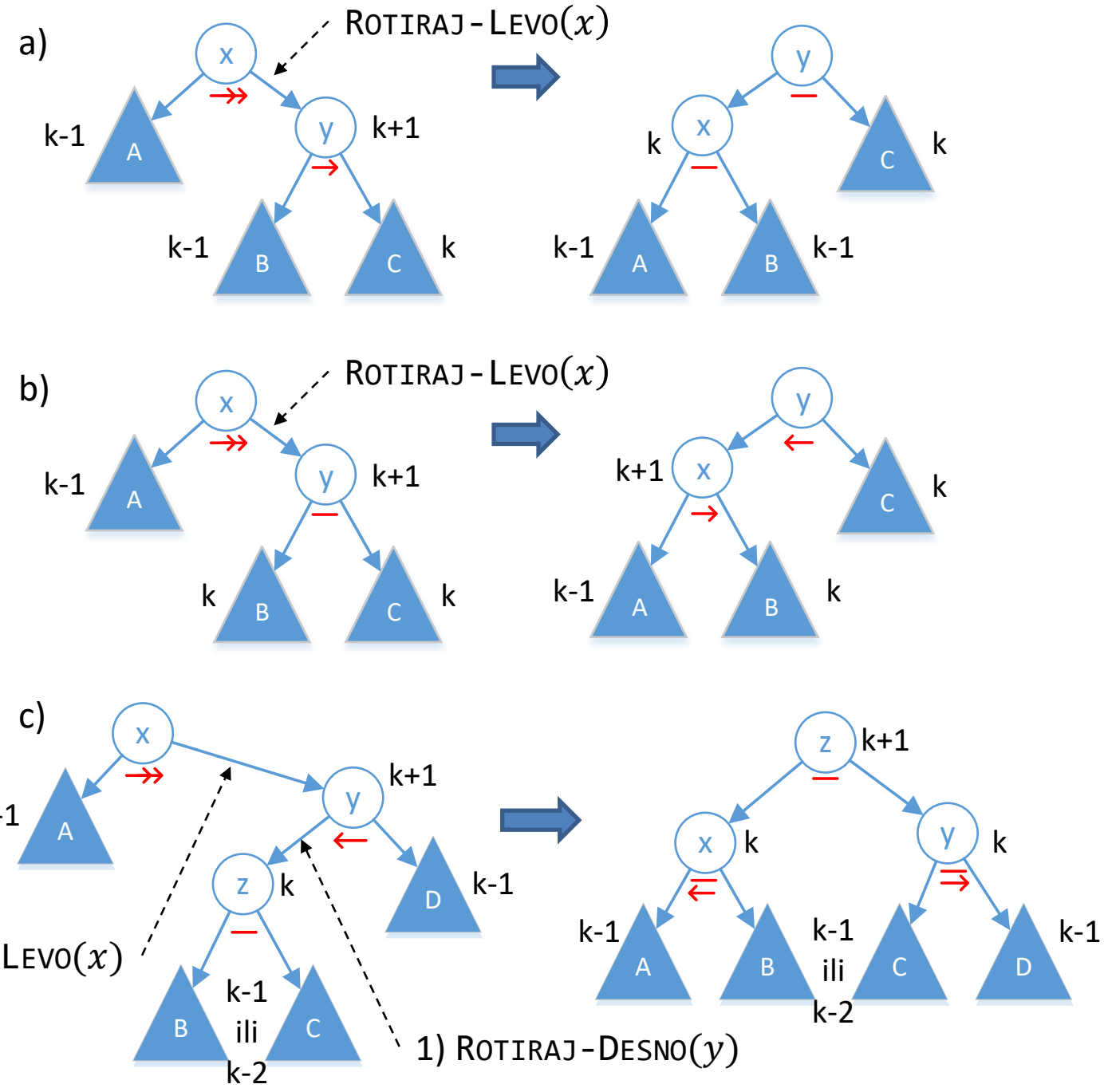
Tri moguća slučaja kada je narušena balansiranost čvora $x$ sa desne strane:

a) desno dete od $y$ je više

b) deca od $y$ su iste visine (rešenje je isto kao za 1)

c) levo dete od $y$ je više

Slično kada je narušena balansiranost $x$ sa leve strane.



# Proširenje strukture čvora CC stabla

- Čvor CC stabla ima: ključ, roditelja, levo i desno dete, boju + satelitski podaci
- Kod AVL stabla je dodat nivo – visina čvora
- Dodatne operacije (obično) zahtevaju dodavanje informacije u čvor:
  - Broj elemenata u podstablu
  - Interval vrednosti
  - Itd

Uopšteno
Koraci kod proširivanja neke poznate strukture podataka:

1. Izbor osnovne strukture podataka (npr. CC stablo)
2. Dodavanje informacije u osnovnu strukture
3. Proveriti da se dodata informacija može održavati osnovnim operacijama (npr. dodavanja, brisanja...) i prilagoditi ih
4. Razviti nove operacije

# Prošireno CC stablo i redosledna statistika

- Traži se $i$-ti po redosledu element u skupu elemenata, tj. $i$-ti „najmanji ključ“
  - Ranije smo videli da se može pronaći u nesortiranom nizu u vremenu $O(n)$
- Sada se posmatra dinamička struktura podataka i zato koristimo CC stablo
- Ideja: svaki čvor CC stabla $x$ se proširi brojem elemenata u podstablu

$$x.brel = x.levo.brel + x.desno.brel + 1$$

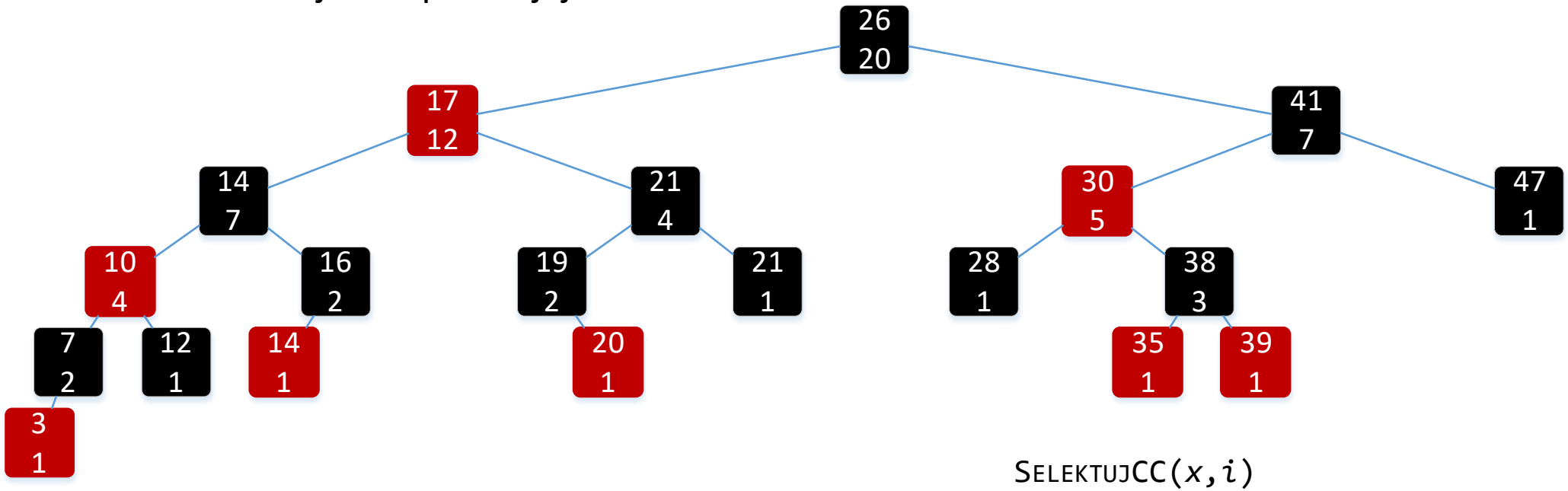
- Sada se svaka selekcija može uraditi u vremenu $O(\log_2 n)$

```
SELEKTUJCC(x,i)
1  r = x.levo.brel + 1
2  if i==r
3     return x
4  elseif i < r
5        return SELEKTUJCC(x.levo, i)
6  else return SELEKTUJCC(x.desno, i-r)
```

Poziv: SELEKTUJCC($t.koren, i$)

# Stablo redosledne statistike

- Stablo redosledne statistike je CC stablo sa modifikacijom čvora – dodatim $brel$
  (u primeru gornji broj je $ključ$, a donji $brel$)
  - Primetiti da se ključevi ponavljaju



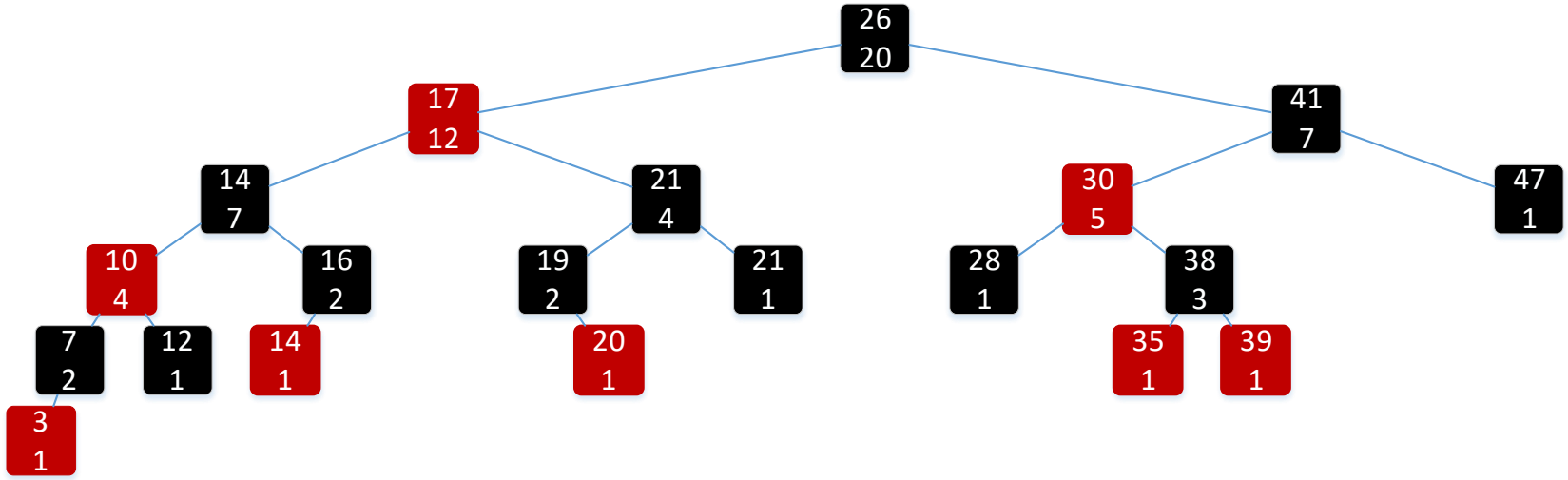

Npr. Koji je 17. po redu?

```
SELEKTUJCC(x,i)
1   r = x.levo.brel + 1
2   if i==r
3      return x
4   elseif i < r
5         return SELEKTUJCC(x.levo, i)
6   else   return SELEKTUJCC(x.desno, i-r)
```

# Određivanje redne pozicije čvora

- Za čvor $x$ treba odrediti koji je po redu u redoslednoj statistici
- Kolika je složenost ovog algoritma?

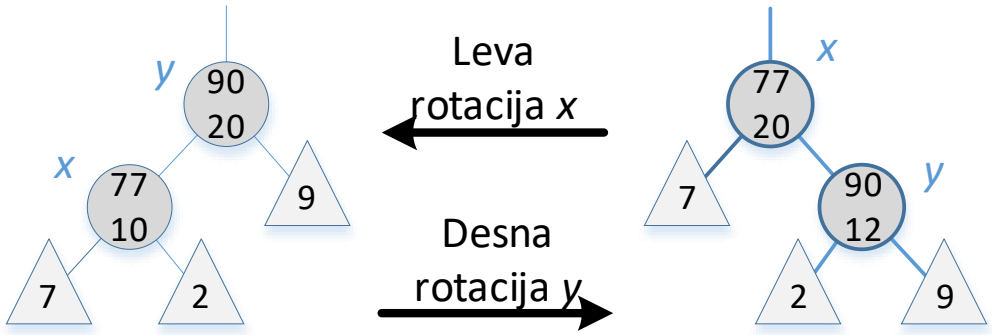
```
REDNAPOZICIJA(T, x)
1  r = x.levo.brel + 1
2  y = x
3  while y ≠ T.koren
4     if y == y.r.desno
5        r = r + y.r.levo.brel + 1
6     y = y.r
7  return r
```



# Ažuriranje polja dodatog u strukturu čvora

Bitne operacije su dodavanje i brisanje

- Dodavanje u CC stablo ima dve faze:
  1. pronalaženje mesta novom čvoru polazeći od korena – za svaki posećeni čvor se poveća $brel$ za 1.
     Za dodati čvor je $brel = 1$
  2. eventualne strukturne promene – rotacije (utiče samo na 2 čvora)
- Brisanje takođe ima 2 faze
  1. briše se čvor (ili se pomeri) nagore u stablu – za svaki čvor od obrisanog do korena se umanji $brel$ za 1.
  2. eventualne strukturne promene – rotacije (utiče samo na 2 čvora)
- Obe operacije imaju složenost $O(\log_2 n)$

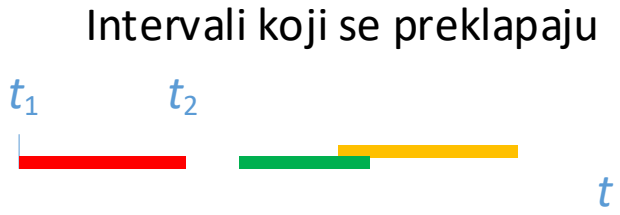



Proširenje za ROTIRAJ-LEVO(T, $x$)

```
...
y.brel = x.brel
x.brel = x.levo.brel
       + x.desno.brel + 1
```

Slično za desnu rotaciju.

# Intervalno stablo




- Intervalno stablo je prošireno CC stablo koje podržava operacija na dinamičkim intervalima.
  - Zatvoreni interval $[t_1, t_2]$ predstavlja skup $\{t \in \mathbb{R}: t_1 \leq t \leq t_2\}$ (mogu se posmatrati i otvoreni i polu-otvoreni intervali)
    - Primeri intervala su vremenski intervali
  - Interval $i$ ima parametre: $početak$ i $kraj$ $(i.početak = t_1, i.kraj = t_2)$
  - Intervali $a$ i $b$ se preklapaju kada je $a.početak \leq b.kraj$ i $b.početak \leq a.kraj$

- Primer implementacije:
  Čvor intervalnog stabla sadrži interval u polju $int$ sa dva broja $početak$ i $kraj$
  - Ključ je $int.početak$
  - Polje $max$ koje sarži maksimalnu vrednost kraja intervala iz podstabla u čijem korenu je $x$

$$x.max = \max(x.int.kraj, x.levo.max, x.desno.max)$$

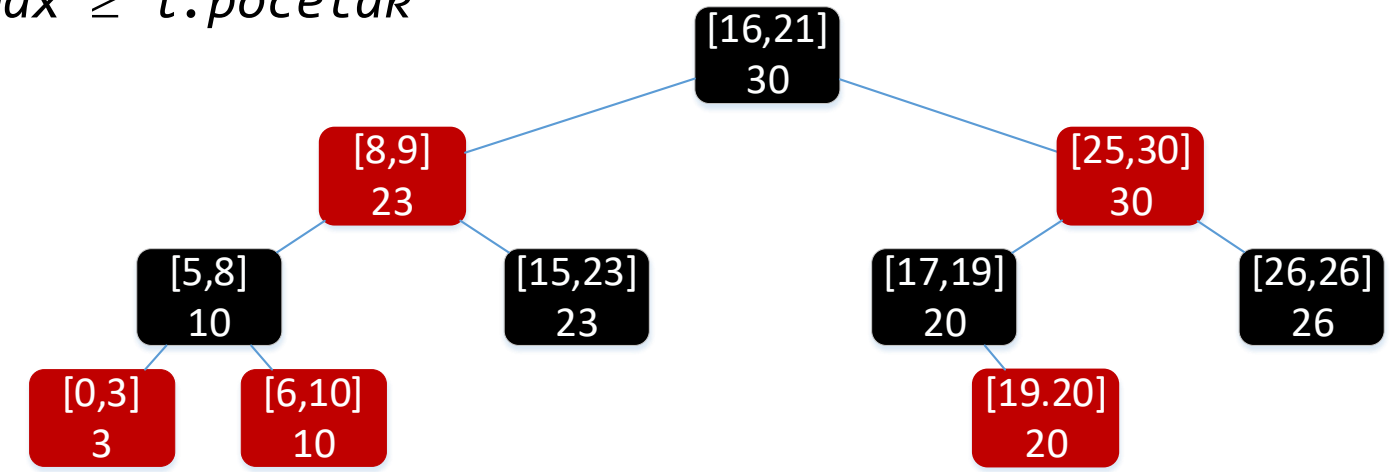
- Intervalno stablo ima osnovne operacije: dodavanja, brisanja i pretrage intervala, ali se mogu dodati i druge operacije, npr. presek sa datim intervalom itd.

# Operacija: Presek sa datim intervalom

Operacija se svodi na pretragu - pronalaženje intervala (čvora) koji se preklapa sa zadatim intervalom $i$

- Ukoliko nema takvog intervala u stablu vraća se T.$Nil$
- Vreme izvršavanja je $O(\log_2 n)$

```
PRESEK-SA-INTERVALOM(T, i)
1  x = T.koren
2  while x ≠ T.Nil & i ∩ x.int = ∅
3     if x.levo ≠ T.Nil & x.levo.max ≥ i.početak
4           x = x.levo
5     else x = x.desno
6  return x
```




Npr. traži se preklapanje sa intervalom [22, 25]?
Šta je sa intervalom [11, 14]?