Strukture podataka: AVL stablo

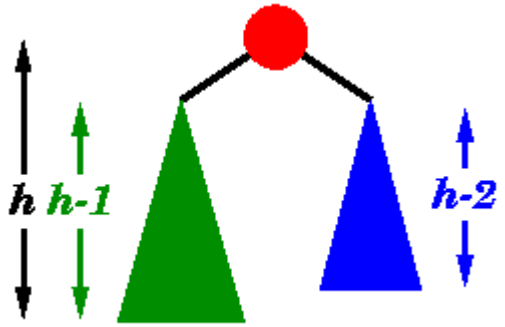
Nedostatak proizvoljnih binarnih stabala: umetanja i brisanja čvorova mogu dovesti do debalansiranosti stabla, čime se smanjuje efikasnost osnovnih operacija nad stablom (pretraga, umetanje, brisanje) => visnski balansirano, **AVL stablo (1962)**

Podsećanje: **e-skriptarnica studenata Matematičkog fakulteta**

AVL=Georgy Maximovich **A**delson-**V**elsky **(1922.–) ,** Yevgeniy Mikhailovich **L**andis **(1921–1997)**

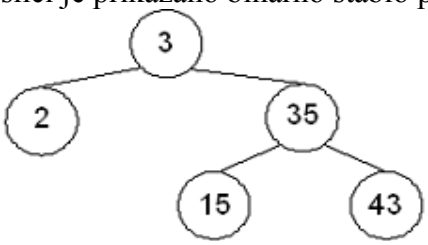
AVL stablo je binarno stablo pretrage kod koga je za svaki čvor apsolutna vrednost razlike visina levog i desnog podstabla manja ili jednaka od jedan.
Visina stabla je visina njegovog korena, tj. maksimalno rastojanje od korena do nekog čvora..

Visina čvora *x* je rastojanje od *x* do najdaljeg potomka.



Operacije pretrage, umetanja, brisanja nekog elementa obavljaju se za O(log n) vremena

Primer 1: Na slici je prikazano binarno stablo pretrage koje je balansirano prema (visinskom) AVL



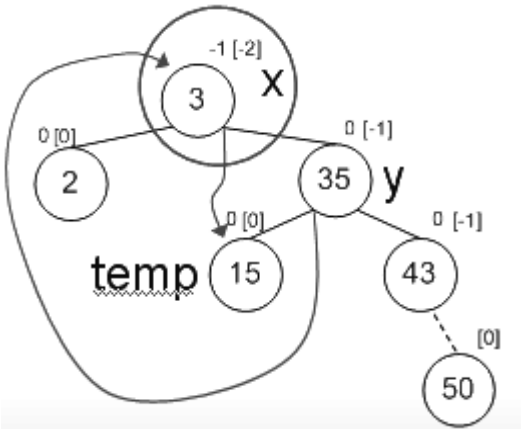
kriterijumu , ali nije balansirano prema opštem kriterijumu (razlika broja čvorova u levom i desnom podstablu svakog čvora može biti najviše 1)

Rebalansiranje stabla nakon umetanja lista

Potreba za rebalansiranjem AVL stabla se javlja kod umetanja lista, kada je neki njegov predak već nagnut (**kritičan čvor**) i nakon umetanja nagnuće prelazi dozvoljenu granicu . Postoje dva karakteristična slučaja:
a) kritični čvor i njegov sin na strani umetanja se naginju na istu stranu
b) kritični čvor i njegov sin na strani umetanja se naginju na suprotnu stranu
Rešenje: Rotacije (jednostuke ili dvostruke) oko kritičnog čvora ispravljaju balans i održavaju inorder poredak

Primer rebalansiranja umetanjem čvora 50 na slici u primeru 1:

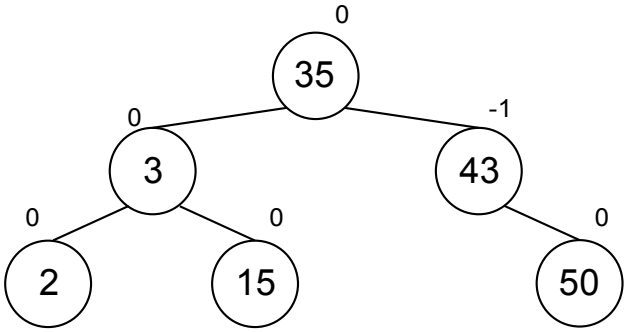
Nakon umetanja lista, sin kritičnog čvora naginje na istu stranu kao i kritični čvor (potrebna rotacija ulevo ili udesno)



Algoritam LEVE ROTACIJE

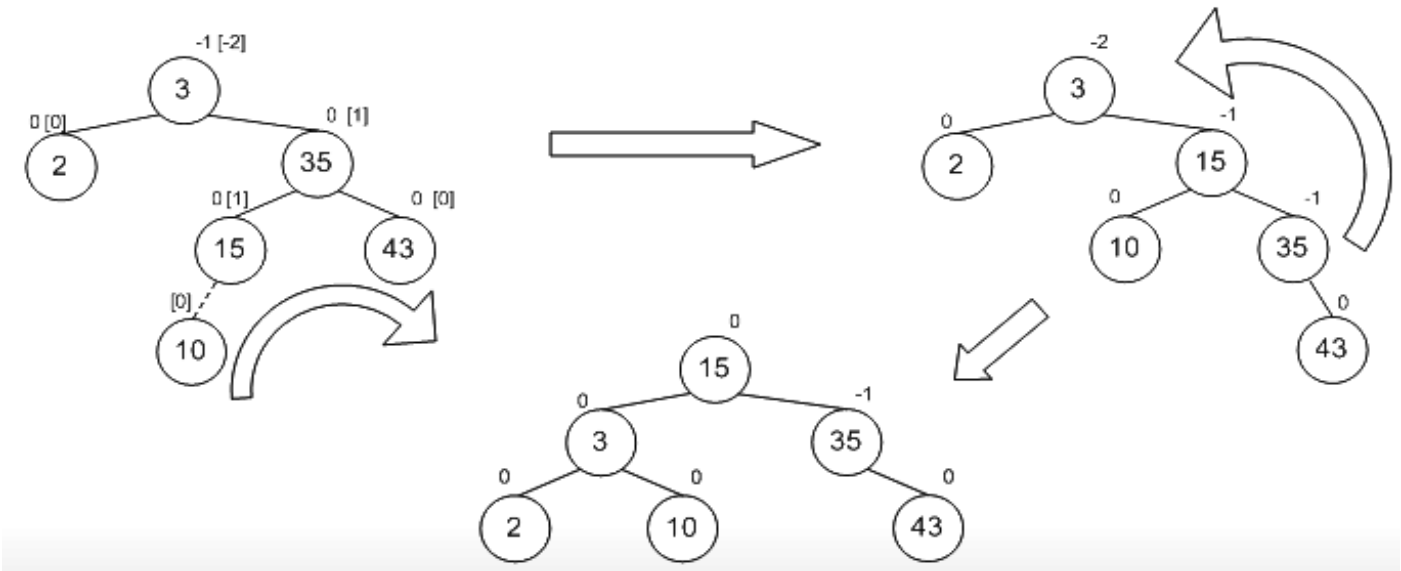
LEFT-ROTATION(*x*)

y = right(*x*)
temp = *left*(*y*)
left(*y*) = *x*
right(*x*) = *temp*



Primer rebalansiranja umetanjem čvora 10 na slici u primeru 1:

Nakon umetanja lista, sin kritičnog čvora naginje na suprotnu stranu od kritičnog čvora (potrebna dvostruka rotacija, u različitim smerovima)



Rotacije

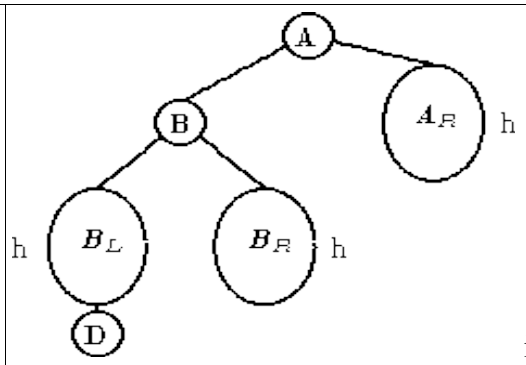
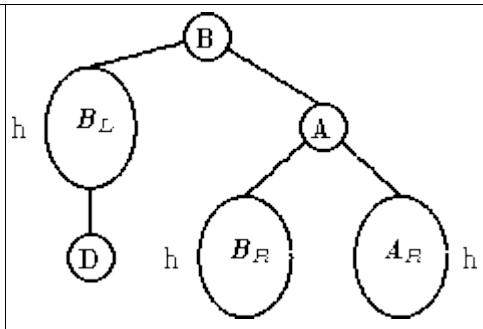
Pri umetanju čvorova u AVL stablo, najpre se pronalazi mesto čvoru tako da bude očuvano svojstvo stabala binarne pretrage o uređenosti ključeva, a potom se u stablo dodaje novi list sa ključem jednakim ključu novog čvora. Pri tom može da se desi da stablo postane neuravnoteženo, te se primenjuje rotacija. Rotacija je lokalna operacija i zahteva konstantno vreme izvršavanja.

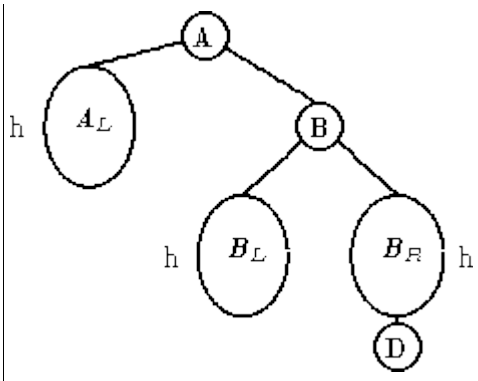
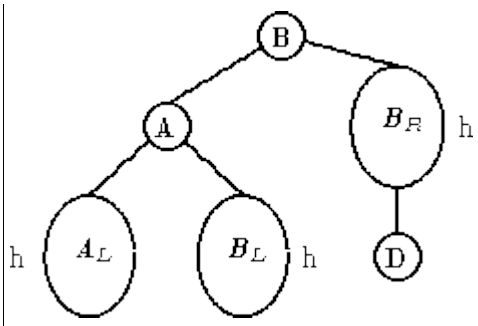
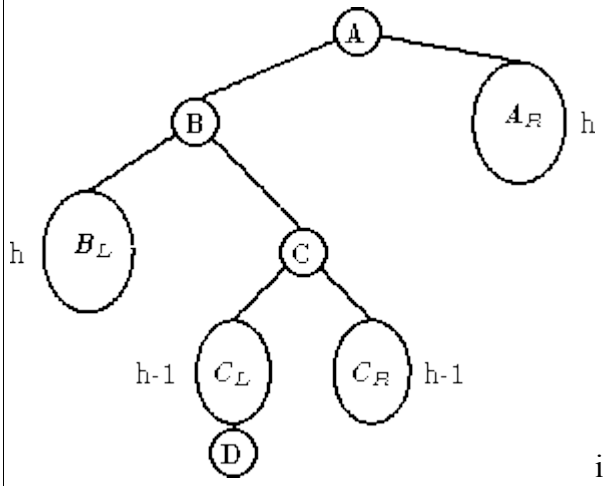
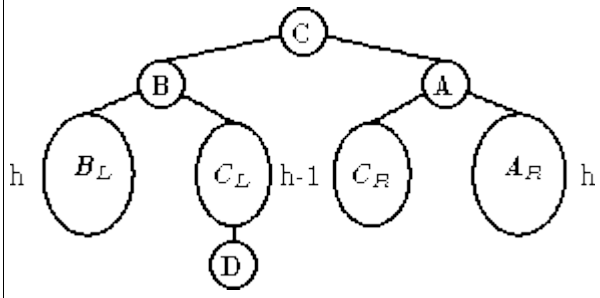
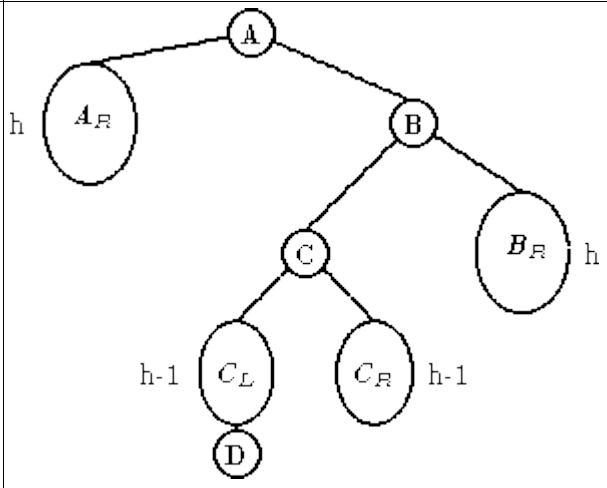
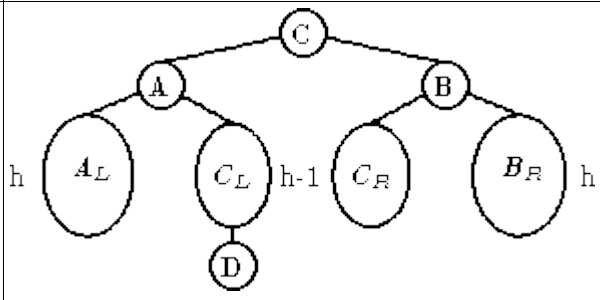
Razlikuju se

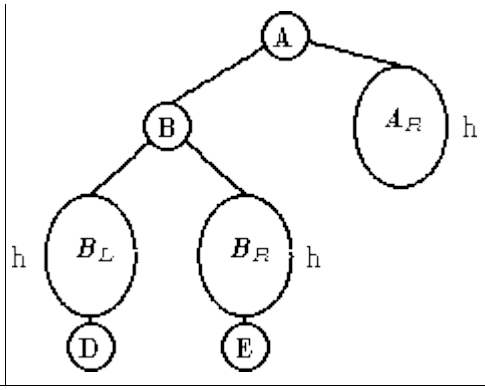
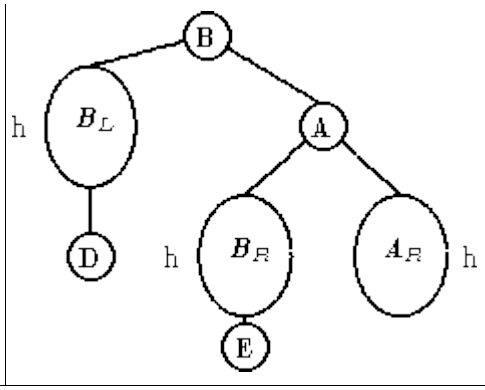
1. 1-struka rotacija (npr. koren levog podstabla podiže se i postaje koren novog stabla, a ostatak stabla se preuređuje tako da stablo ostane BSP stablo, npr. slika LL)

LL=left rotation, RR= right rotation

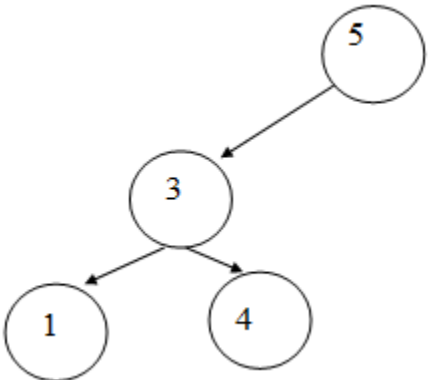
2. 2-struka rotacija i Leva(L) i Desna(R): LR, RL

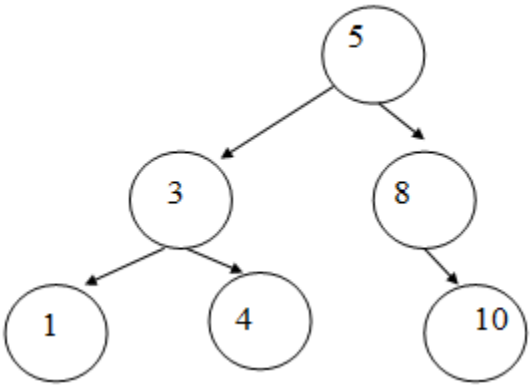
LL ako je levo podst. imalo veće h od desnog	 i čvor je umetnut u levo podst. levog podstabla	

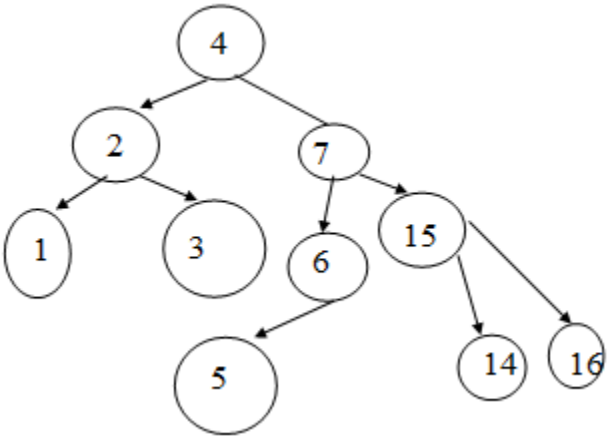
RR ako je desno podst. imalo veće h od levog	 i čvor je umetnut u desno podst. desnog podstabla	
LR ako je levo podst. imalo veće h od desnog	 i čvor je umetnut u desno podst. levog podstabla	
RL ako je desno podst. imalo veće h od levog	 i čvor je umetnut u levo podst. desnog podstabla	

LL & LR ⇒ LL		

1. Koja stabla na sledećim slikama NISU AVL stabla? Zašto?







2. Odrediti najmanji i najveći broj elemenata koje može imati AVL stablo visine h.
REŠENJE:

PRVI DEO ZADATKA je određivanje najvećeg broja čvorova za AVL stablo visine h.
Neka je G_h oznaka za najveći broj čvorova stabla visine h.

Važi

G_0=1 (samo koren čini AVL stablo visine 0),

G_1=3 (koren i levo i desno dete čine AVL stablo visine 1),

G_2=7 (koren, levo i desno dete, levi i desni unuk kao deca levog deteta, levi i desni unuk kao deca desnog deteta čine stablo visine 2)

i važi

$G_{h+1}=G_h+G_h+1$
$G_h=G_{h-1}+G_{h-1}+1$

Odavde sledi : $G_{h+1} - G_h = 2G_h + 1 - 2G_{h-1} - 1$
$G_{h+1} - 3G_h + 2G_{h-1}=0$

Karakteristična jednacina ove rekurentne veze je:
$t^2-3t+2=0$
odnosno $(t-1)(t-2)=0$
odakle sledi $t_1=1$, $t_2=2$.

Dakle, važi: $G_h= C_1 + C_2 * 2^h$
Kako je:
$1= C_1 + C_2$
$3= C_1 + 2* C_2$

Rešenja ovog sistema su $C_1 = -1$ i $C_2 = 2$.

Formula za određivanje najvećeg broja cvorova koje moze da ima AVL stablo visine h je:

$G_h= -1 + 2^{h+1}$

DRUGI DEO ZADATKA:

Neka je B_h oznaka za AVL stablo visine h sa najmanjim brojem čvorova sa i neka je $\mathbf{N_h}$ oznaka za broj čvorova stabla B_h .

B_h ima BAR JEDNO podstablo visine h-1 (stablo B_{h-1}), a zbog svojstva da B_h je AVL stablo sa najmanjim brojem čvorova, njegovo drugo podstablo je B_{h-2}, te važi:

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

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

$N_{h+1} - 2N_h + N_{h-2} = 0$

Karakteristična jednačina ove rekurentne veze je

$t^3 - 2t^2 + 1 = 0$

odnosno $(t-1)(t^2-t-1)=0$

odnosno njeni koreni su $t_1=1$, $t_2 = \frac{1+\sqrt{5}}{2}$, $t_3 = \frac{1-\sqrt{5}}{2}$

Zato je opsti član niza N_h jednak

$N_h = C_1 + C_2 * \left(\frac{1+\sqrt{5}}{2} \right)^h + C_3 * \left(\frac{1-\sqrt{5}}{2} \right)^h$

Zbog $N_0=1$, $N_1=2$, $N_2=4$

dobija se sistem:

$1 = C_1 + C_2 + C_3$

$2 = C_1 + C_2 * \frac{1+\sqrt{5}}{2} + C_3 * \frac{1-\sqrt{5}}{2}$

$4 = C_1 + C_2 * \left(\frac{1+\sqrt{5}}{2} \right)^2 + C_3 * \left(\frac{1-\sqrt{5}}{2} \right)^2$

Rešenja sistema su:

$C_1 = -1$

$C_2 = \frac{2+\sqrt{5}}{\sqrt{5}}$

$C_3 = \frac{\sqrt{5}-2}{\sqrt{5}}$

++++++++++++++++++++++++++++++++

Važna posledica II dela zadatka:

uočite da smo broj čvorova u AVL stablu B_h (AVL stablo visine h sa najmanjim brojem čvorova) opisali kao:

$N_h = N_{h-1} + N_{h-2} + 1 \quad /+1$

tj. $N_h + 1 = (N_{h-1}+1) + (N_{h-2}+1)$ tj. Brojevi N_h+1 zadovoljavaju istu rekrentnu jednačinu kao i članovi Fibonačijevog niza.

Indukcijom se može pokazati da važi $N_h + 1 = F_{h+2}$

Dakle, za broj n unutrašnjih čvorova AVL stabla visine h važi nejednakost $n >= F_{h+2} - 1$

Ali, kako važi da $F_n > \frac{\phi^n}{\sqrt{5}} - 1$, gde važi da $\phi = [1+\sqrt{5}]/2$ i kako važi da $N_h + 1 = F_{h+2,}$ onda važi da

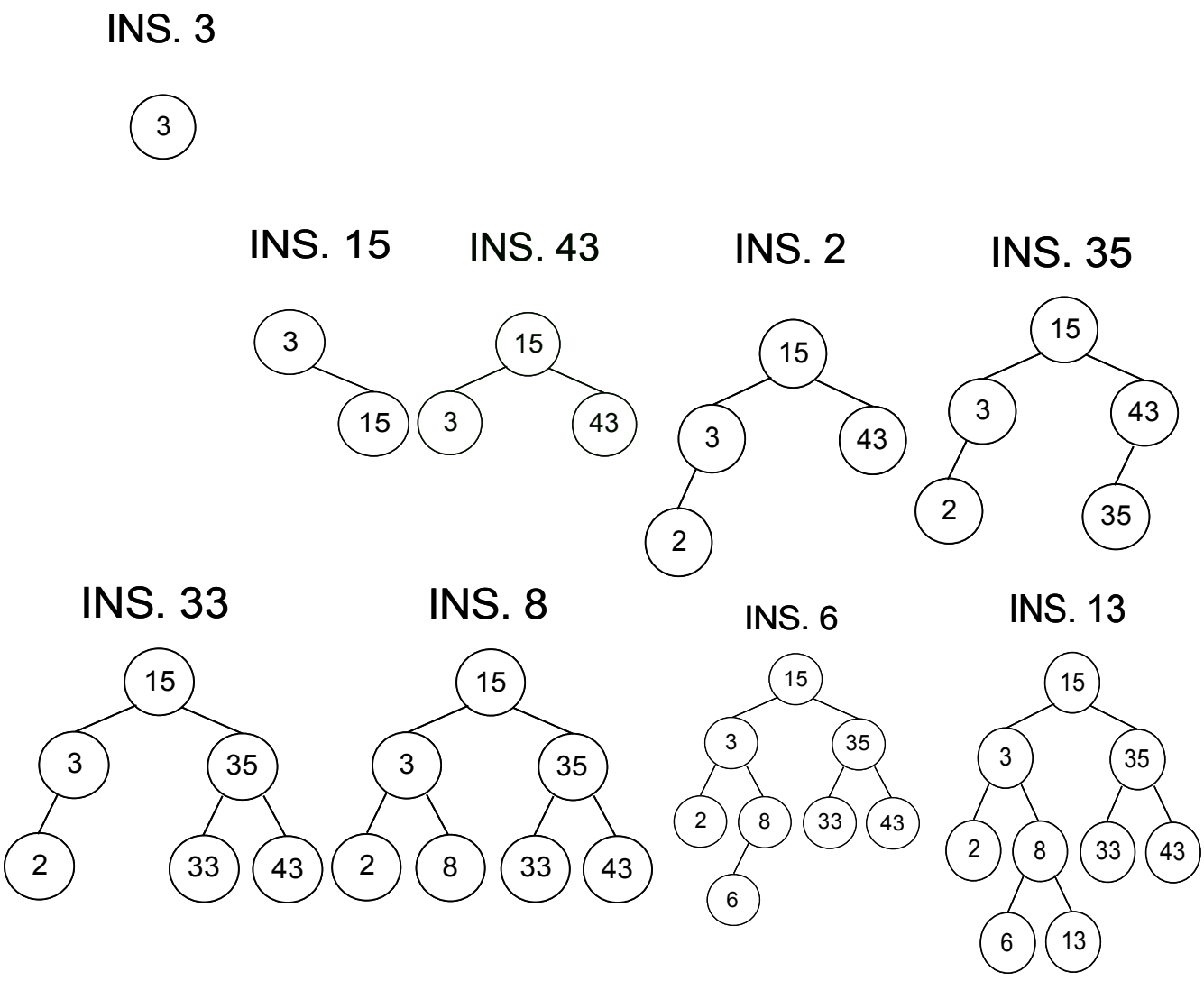
$n+2 > \frac{\phi^{h+2}}{\sqrt{5}}$ =>

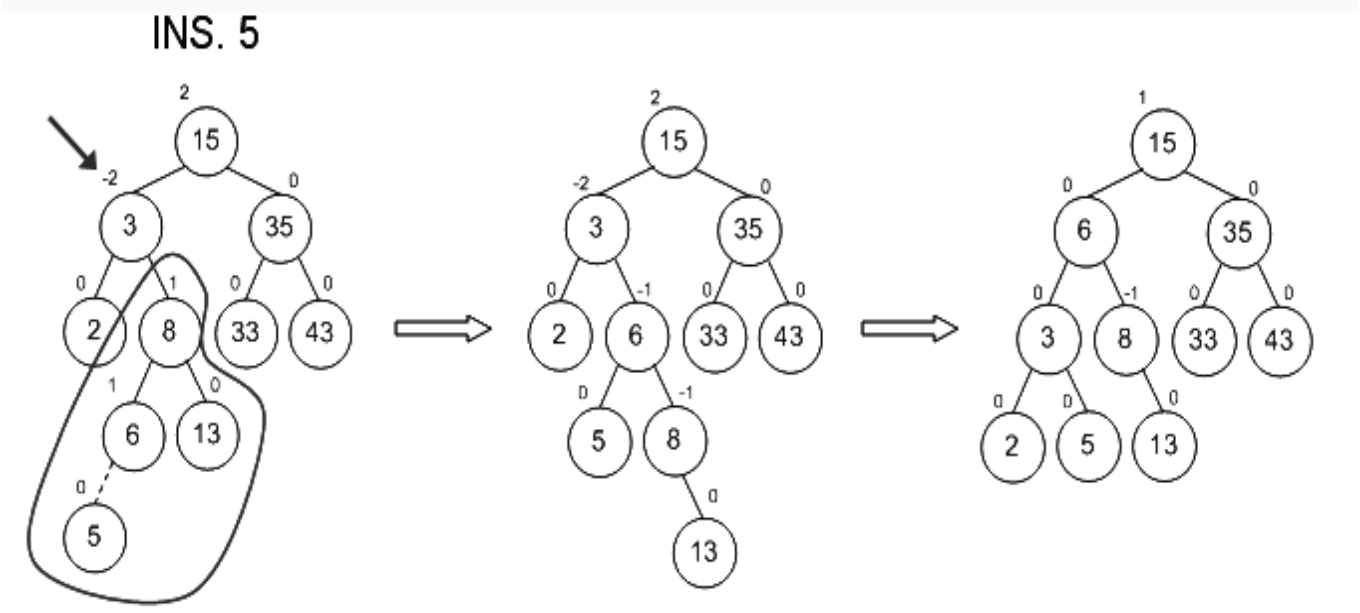
$$\log_\phi(n+2) > \log_\phi\left(\frac{\phi^{h+2}}{\sqrt{5}}\right) = h + 2 - \log_\phi(\sqrt{5})$$

Odatle imamo važan rezultat teoreme 3.1 iz udžbenika *Algoritmika* da $h<1.44 \log_2 (n+2) -0.328$ tj. h~O(log n) tj. visina AVL stabla je O(log n)

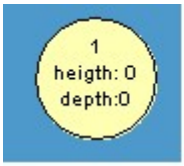
3. Odrediti izgled AVL stabla dobijenog umetanjem redom brojeva 3, 15, 43, 2, 35, 33, 8, 6, 13, 5

REŠENJE:

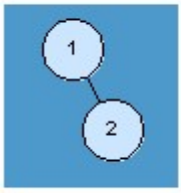




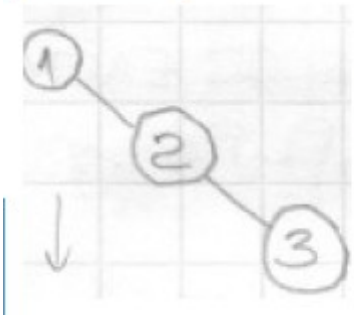
4. Odrediti izgled AVL stabla dobijenog umetanjem redom brojeva 1,2,3,4,5,6,7,8
Podsećanje: Kako bi izgledalo binarno stablo pretrage za ovaj ulazni skup podataka?
REŠENJE:



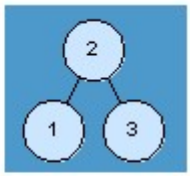
insert 1

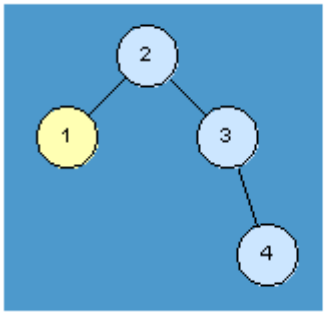


insert 2

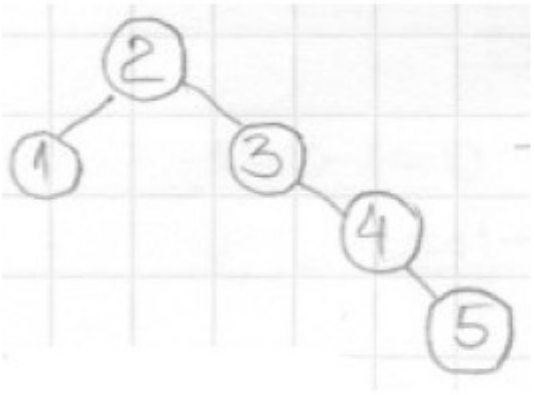


insert 3

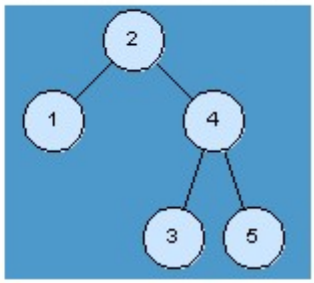


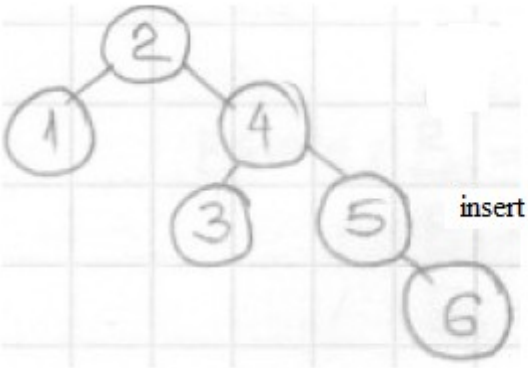


insert 4

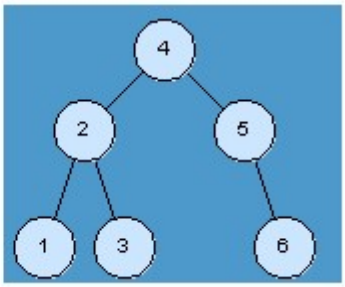


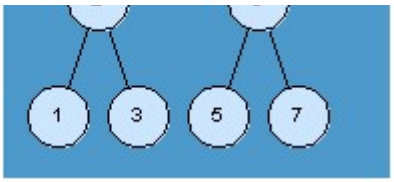
insert 5

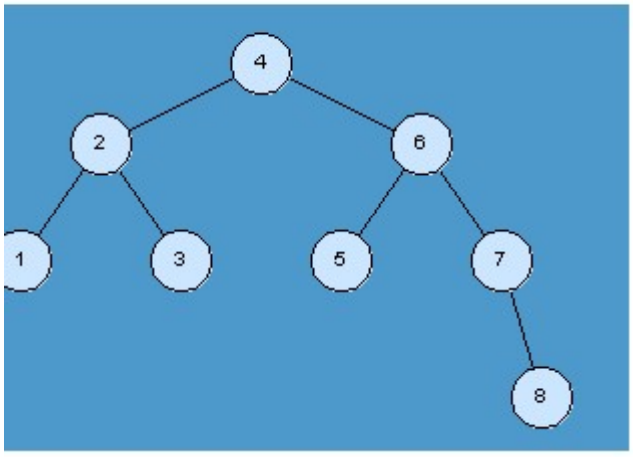




insert 6







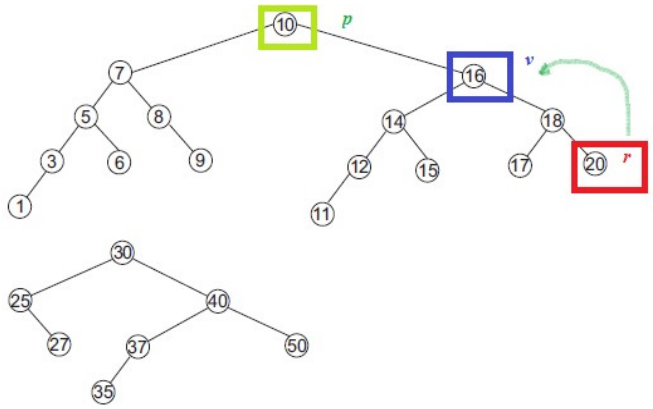
insert 8

5. *Konkatenacija* je operacija nad dva skupa koji zadovoljavaju uslov da su svi ključevi u jednom skupu manji od svih ključeva u drugom skupu; rezultat konkatenacije je unija skupova. Konstruisati algoritam za konkatenaciju dva AVL stabla u jedno. Složenost algoritma u najgorem slučaju treba da bude O(h), gde je h veća od visina dva AVL stabla.

REŠENJE:

Neka su zadata dva AVL stabla T, G, tako da svi ključevi stabla G su veći od svih ključeva stabla T.

Uz čvorove stabla T i G čuvaće se i informacija o visini, jer je to svojstvo značajano za AVL stabla, kao tip balansiranih stabala i neka je, na primer, visina h_2 stabla G sada manja ili jednaka od visine h_1 stabla T. Drugi slučaj rešava se analogno.



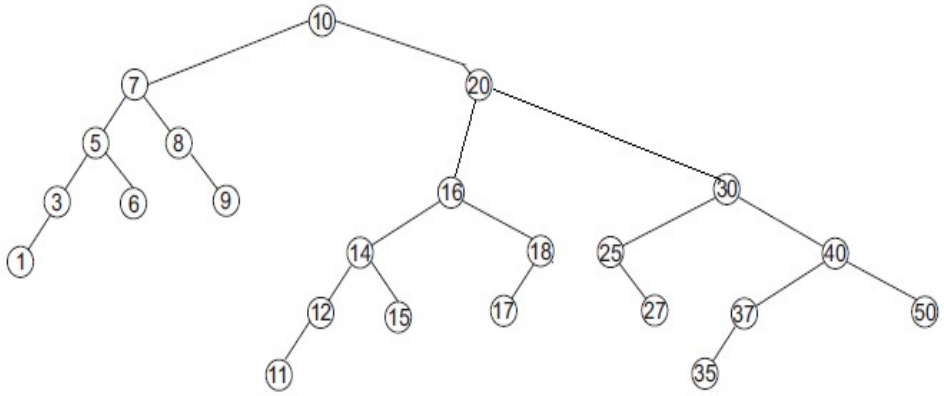
1. **Ukloniti najveći čvor iz stabla T, na primer čvor *r*** i on će biti koren stabla preko kog se stablo G privezuje za T. Evo kako.

2. Spuštajući se u stablu T samo desnim granama, **naći čvor *v*** visine h_2 ili h_2-1 (a to je moguće jer $h_1>=h_2$). Neka **otac čvora *v* jeste čvor *p***.

3. Postaviti da **desni sin čvora *p*** ne bude čvor *v*, već čvor *r* iz koraka 1. (što je održava poredak, jer čvor *r* je sa najvećim ključem, te može biti desni sin čvora *p*)

4. Postaviti da **levi sin čvora *r*** bude čvor ***v* sa svojim podstablom iz T**, a **desni sin je koren stabla G**. To je u redu sa stanovista BSP svojstva.

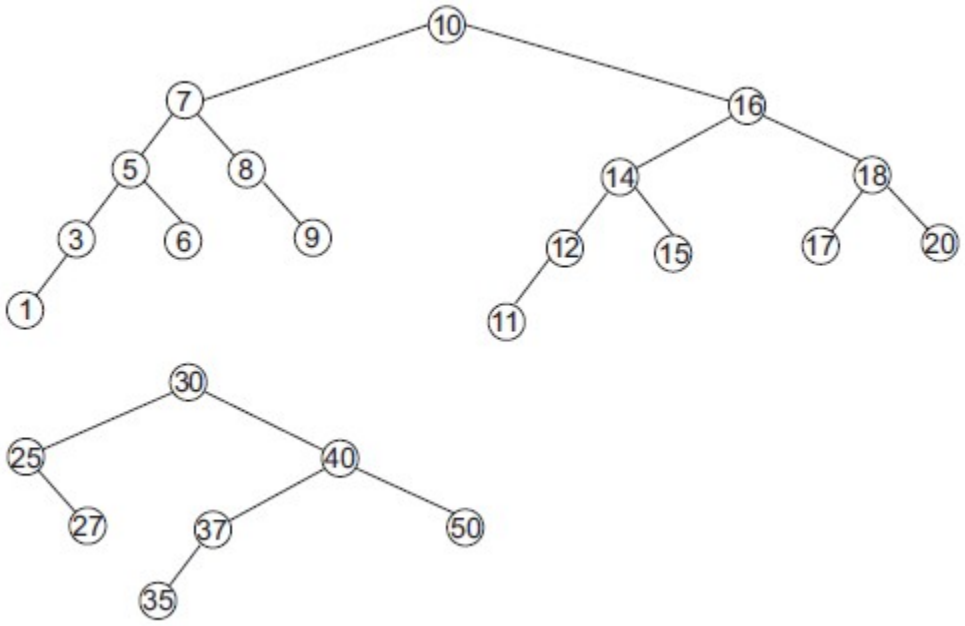
OBRAZLOŽENJE: *r* je najveći čvor u T, te ključ čvora *v* nije veći od njega i može čvor *v* biti levi sin, a kako svaki čvor stabla G, te i koren je veći od najvećeg ključa iz T, onda koren iz G može biti desni sin sa ciljem da i novoformirano stablo bude binarno stablo pretrage (BSP).



5. U koracima 3 i 4 formirano je stablo koje ima BSP svojstva, ali možda nije uravnoteženo, tj. visina čvora *r* može da postane veća za 1 od visine čvora *v* koji je bio na tom mestu, te je nužno obaviti eventualno uravnoteženje pomoću rotacije.

Sve operacije traju <=$O(h_1)$ koraka zbog svojstva AVL stabla.

6. Izvršiti konkatenaciju stabala sa slike.



7. Opisati realizaciju apstraktnog tipa podataka koji podržava sledeće operacije:

Umetni (x) umetanje ključa *x* u strukturu podataka (ako već ne postoji)

Obrisi (x) brisanje ključa *x* iz strukture podataka (ako postoji u strukturi)

Naredni (x) pronalaženje najmanjeg ključa u strukturi podataka koji je veći od *x*

Izvršavanje svake od ovih operacija treba da ima vremensku složenost *O(log n)* u najgorem slučaju, gde je *n* broj elemenata u strukturi podataka.

Rešenje:

Za realizaciju navedenih operacija, pogodna struktura podataka je *AVL* stablo u kome se umetanje i brisanje izvršava za *O(log n)* koraka.

Za izvođenje operacije **Naredni (x)** koristi se

- najpre algoritam pronalaska ključa *x* (O(log n))
- **potom se kao u već rađenom zadatku iz binarnih stabala pretrage (zadatak 9), pronalazi sledbenik u stablu (O(log n)).**

8. Opisati realizaciju apstraktnog tipa podataka koji podržava sledeće operacije:

Umetni (x) umetanje ključa *x* u strukturu podataka (ako već ne postoji)

Obrisi (x) brisanje ključa *x* iz strukture podataka (ako postoji u strukturi)

Naredni (x,k) pronalaženje najmanjeg k-tog ključa u strukturi podataka među onim koji su veći od *x*

Izvršavanje svake od ovih operacija treba da ima vremensku složenost *O(log n)* u najgorem slučaju,

gde je *n* broj elemenata u strukturi podataka.

Rešenje:

Pogodna struktura podataka za rešavanje ovog problema je AVL stablo, jer je za njega složenost opercija umetanja i brisanja *O(log n)*

Ideja je da svakom čvoru *v* dodamo novo polje *v.D* koje će sadržati broj potomaka čvora *v* računajući i sam čvor *v*.

Označimo sa Rang(v) broj elemenata stabla čiji je ključ manji ili jednak od v.Kljuc

Na osnovu polja D može se odrediti Rang(v) u toku traženja čvora v na sledeći način (indukcijom):

baza: Ako koren ima levog sina, onda je njegov rang za 1 veći od vrednosti D njegovog levog sina, u protivnom je rang korena 1

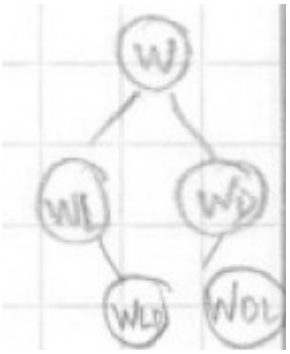
IH: pretpostavimo da znamo rangove svih čvorova na putu od korena do nekog čvora *w*

Neka je w_L levi sin čvora w

Neka je w_D desni sin čvora w

Neka je w_{LD} desni sin čvora w_L

Neka je w_{DL} levi sin čvora w_D



Važi da

$Rang(w_L)=Rang(w)-1-w_{LD}.D$

$Rang(w_D)=Rang(w)+1+w_{DL}.D$

Ako neki od sinova ne postoji, uzima se da je odgovarajuca vrednost polja D jednaka 0.

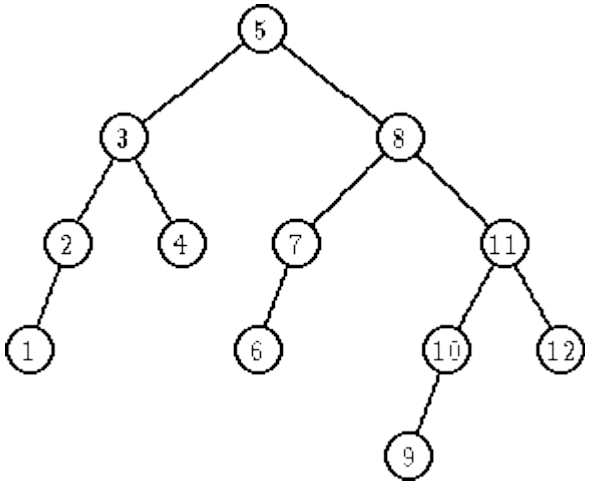
Da bi se izvršila operacija Naredni(x,k) treba prvo pronaći čvor v takav da je v.Kljuc=x.
Ako takav čvor ne postoji, pretraga je neuspešna

Potom se traži čvor sa rangom Rang(v)+k. Traženje prema rangu iste je složenosti kao i traženje prema ključevima. Rangovi se računaju usput, a redosled u stablu im je isti kao redosled ključeva.

Ostaje nam da (pošto smo uveli novo polje D) posle svake operacije(umetanje, brisanje) ažuriramo vrednost polja D.

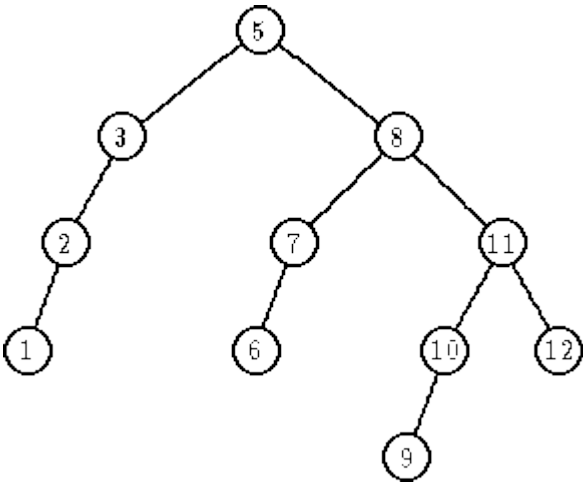
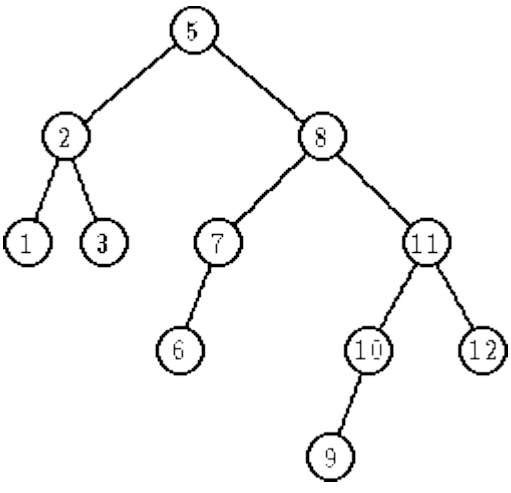
Zašto je uvedeno polje D umesto polja Rang? Zato što umetanje/brisanjeu najgorem slučaju menja rangove O(n) elemenata (ako brišemo najmanji element)

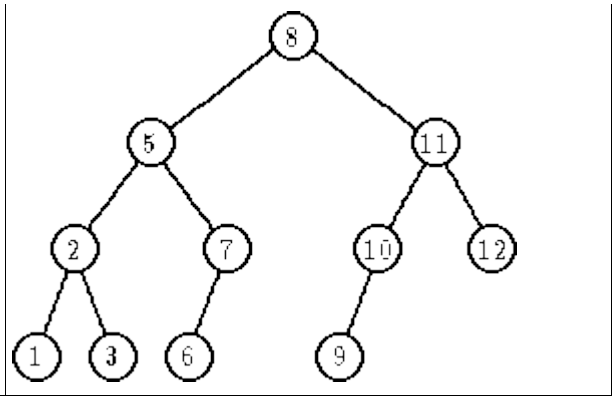
9. Demonstrirati brisanje čvora 4 iz datog AVL stabla.



Rešenje:

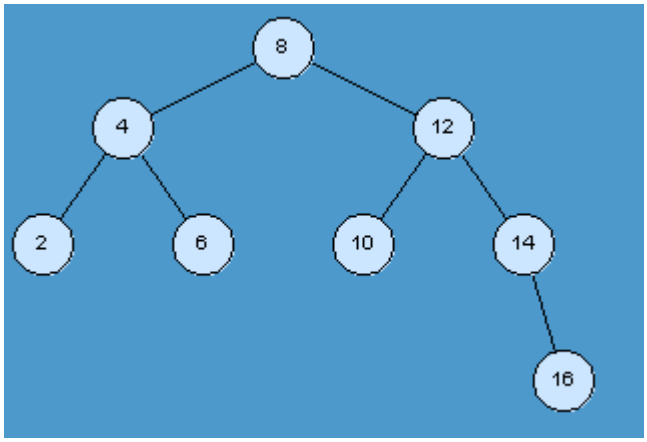
Uklanjanje elementa iz AVL stabla je komplikovanije neko kod binarnog stabla pretrage, jer je potrebno više rotacija. Na primer, da bi se uravnotežilo Fibonačijevo (potpuno) AVL stablo posle uklanjanja "pogodno" izabranog čvora, nužno je izvršiti h-2 rotacije, gde h je visina stabla.

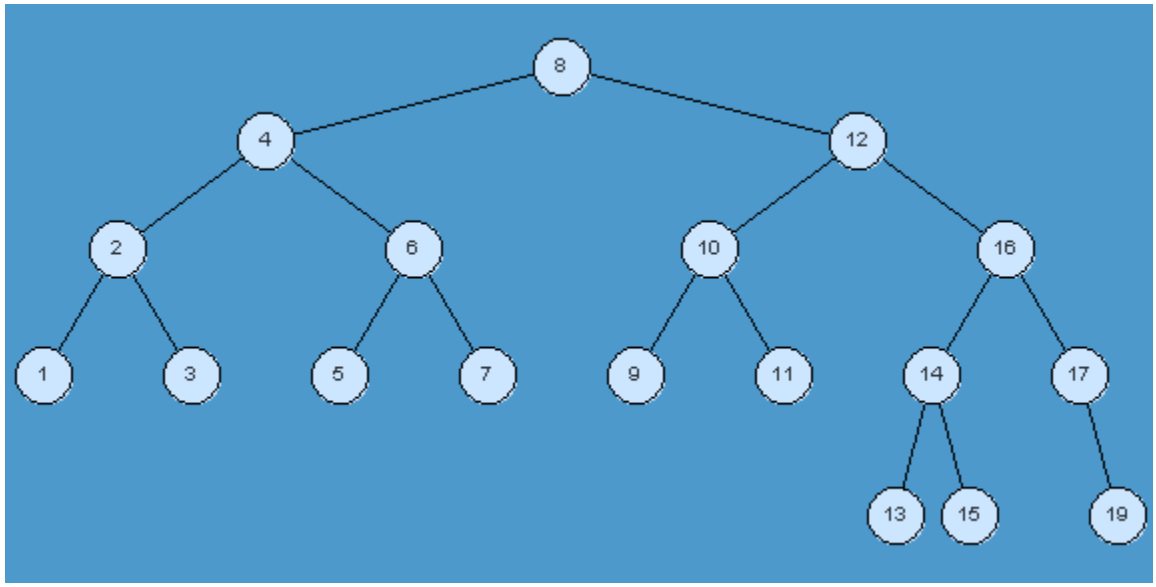
Ukloni 4	
debalans sa 3, LL rotacija sa 2	

deabalans sa 5, RR rotacija sa 8	

10. Neka su data dva AVL stabla na slici. Smatrati da čvor binarnog stabla je **S-AVL čvor** ako nije list i ako razlika visina njegovog levog i desnog podstabla je jednaka 0.
Pronađi:

a) sve S-AVL čvorove na stablu b) sve S-AVL čvorove čiji niti jedan potomak nije S-AVL čvor





11. Konstruisati algoritam čija vremenska složenost je linearna po broju čvorova binarnog stabla i koji u stablu markira (označava) sve S-AVL čvorove čiji niti jedan potomak nije S-AVL čvor.

Algoritam SAVL (stablo x)

ulaz: stablo x

izlaz: dubina stabla kao oznacen broj

/* CILJ: ako algoritam vrati dubinu <0, to znači da se u stablu kao potomak pojavljuje SAVL cvor

ako algoritam vrati dubinu 0, to znači da stablo je prazno

algoritam vrati dubinu >0, to znači da SAVL čvor jos nije nađen

IDEJA:

1. Ako su dubine levog i desnog podstabla jednake i ako ni u levom ni u desnom podstablu se ne nalazi SAVL čvor (jer je vracena dubina >0), tada se taj čvor markira,
jer je SAVL čvor (npr. sa x->oznaka=1;). U tom slucaju vraća se negativna dubina stabla.

2. Ako nisu dubine levog i desnog podstabla jednake ili ako se u levom ili u desnom podstablu nalazi neki SAVL čvor (jer je vracena dubina <0),

tada se vraća dubina drveta sa predznakom, zavisno od toga da li je nađen SAVL čvor ili ne.

```
{
    if (x==NULL)  return 0;
   else
     {
        levo=SAVL(x->levo);
        desno=SAVL(x->desno);
      if (levo >0 && desno >0)
         if (levo==desno)  /*situacija 1*/
             { x->oznaka=1; return -(levo+1);}
         else return max(levo, desno) +1;
     else return  -( 1+ max (abs(levo), max(abs(desno));
}
```

Složenost algoritma:

Svako čvor se jednom obrađuje u const vremenu, te važi:

za n=1, T(n) =c1 (const za obradu stabla sa jednim čvorom)

za n>1, T(n)=n*c1*c2 (c2 je cena vremena koje obuhvata rekurzivni poziv SAVL algoritma i povratka) => T(n)=O(n) q.e.d.

12. Napisati funkciju koja za dato binarno stablo pretrage proverava da li je AVL. Napisati potom i program koji testira ovu funkciju.

```
#include<stdio.h>
#include<stdlib.h>


typedef struct cvor{
   int vrednost;
   struct cvor *levi;
   struct cvor *desni;
} Cvor;

/* pratece funkcije za rad da binarnim stablom. Njih vec imamo */

Cvor *napravi_cvor(int broj){
  Cvor *novi=malloc(sizeof(Cvor));
  if(novi==NULL){
    printf("Greska pri alokaciji\n");
    exit(1);
  }
  novi->vrednost=broj;
  novi->levi=NULL;
  novi->desni=NULL;
  return novi;
}

Cvor *dodaj_u_stablo(Cvor *koren, int broj){
  if(koren==NULL)
    return napravi_cvor(broj);

  if(broj<koren->vrednost)
    koren->levi=dodaj_u_stablo(koren->levi, broj);
  else if(broj>koren->vrednost)
    koren->desni=dodaj_u_stablo(koren->desni, broj);

  return koren;
}

Cvor *pretrazi_stablo(Cvor *koren, int broj){
  if(koren==NULL)
    return NULL;

  if(broj<koren->vrednost)
    return pretrazi_stablo(koren->levi, broj);
  else if(broj>koren->vrednost)
    return pretrazi_stablo(koren->desni, broj);
  return koren;
}
```

```
void prikazi_stablo(Cvor *koren){
  if(koren==NULL)
    return;

  prikazi_stablo(koren->levi);
  printf("%d ", koren->vrednost);
  prikazi_stablo(koren->desni);
}

void oslobodi_stablo(Cvor *koren){
  if(koren==NULL)
    return;

  oslobodi_stablo(koren->levi);
  oslobodi_stablo(koren->desni);
  free(koren);
}



/* pomocna funkcija, vec smo je ranije pisali */
/* visina stabla */
int visina(Cvor *koren){
   int l, d;
   if(koren==NULL)
     return -1;

   l=visina(koren->levi);
   d=visina(koren->desni);

   return (l>d)? l+1: d+1;
}


void koeficijenti_ravnoteze(Cvor *koren){
  if(koren==NULL)
    return;
  koeficijenti_ravnoteze(koren->levi);
  printf("Cvor %d, koeficijent_ravnoteze: %d\n", koren->vrednost,
                    visina(koren->levi)-visina(koren->desni));
  koeficijenti_ravnoteze(koren->desni);


}

/* TRAZENA FUNKCIJA */
/* AVL stablo: svaki cvor ima koef. ravnoteze 1, 0 ili -1 */
int AVL(Cvor *koren){
  int d;
  if(koren==NULL)
    return 1;

  d=visina(koren->levi)-visina(koren->desni);

  return (d==0 || d==1 || d==-1) && AVL(koren->levi) && AVL(koren->desni);

}
```

```
int main(){
    Cvor *koren=NULL;
    int broj;
    Cvor *sledb;

    printf("Unesite sledeci broj (Ctrl+D za kraj)\n");
    while(scanf("%d", &broj)==1){
      koren=dodaj_u_stablo(koren, broj);
      printf("Unesite sledeci broj (Ctrl+D za kraj)\n");
    }

    printf("Uneto je stablo: ");
    prikazi_stablo(koren);

    printf("\n");

   koeficijenti_ravnoteze(koren);

   if(AVL(koren))
     printf("JESTE AVL\n");
   else
     printf("nije AVL\n");

   oslobodi_stablo(koren);

   return 0;
}
```

13. Podvucite tačne iskaze:

a) Vremenska složenost pretraživanja u AVL stablu sa n čvorova je O(log N) pošto su AVL stabla uvek visinski balansirana.

b) Vremenska složenost umetanja i brisanja u AVL stabu stablu sa n čvorova je uvek O(logn).

c) Mana AVL stabla je da čuvanje faktora ravnoteže zahteva dodatan memorijski prostor.

d) AVL stabla jesu asimptotski brža od binarnih stabla pretrage, ali pošto rebalansiranje zahteva dodatno vreme, onda se u sistemima za upravljanjem bazama podataka koriste druge strukture (npr. B-stabla) podataka, jer se obimna pretraživanja uglavnom rade na disku.

e) Vremenska složenost rotacije AVL stabla u najboljem slučaju je O(log N).

14.

a) Kreirajte binarno stablo pretrage umetanjem brojeva 10, 8, 44, 4, 21, 68, 2, 12, 30, 75, 26, 40, 22.

b) Da li je dobijeno stablo visinski balansirano?
c) Balansirajte binarno stabla pretrage dobijeno pod a) primenom metoda za balansiranje AVL stabla

d) Kreirajte AVL stablo umetanjem brojeva 10, 8, 44, 4, 21, 68, 2, 12, 30, 75, 26, 40, 22.

e) Uporedite rezultat dobijen pod c), d).