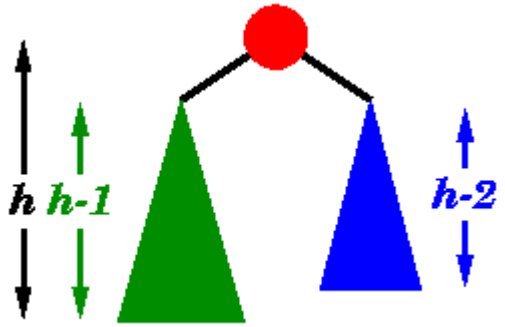
Strukture podataka: AVL stablo

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)**

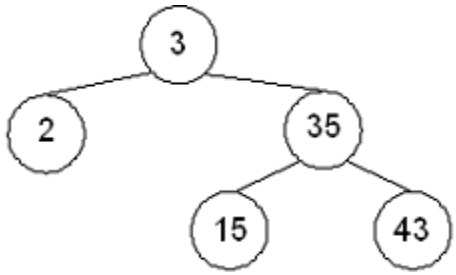
AVL stablo je binarno stablo pretrage kod koga je za svaki čvor apsolutna vrednost razlike visina levog i desnog podstabla manja ili jednaka od jedan.
Visina stabla je visina njegovog korena, tj. maksimalno rastojanje od korena do nekog čvora..

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)

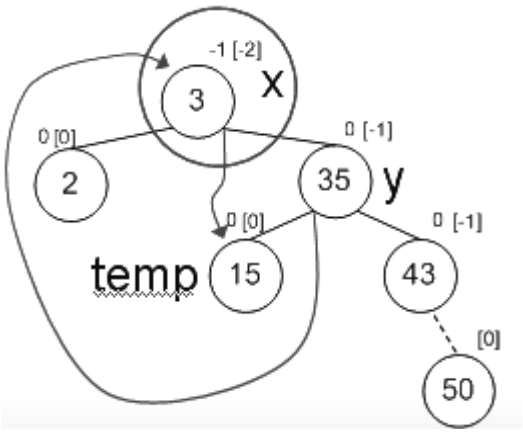


Rebalansiranje stabla nakon umetanja lista

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

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

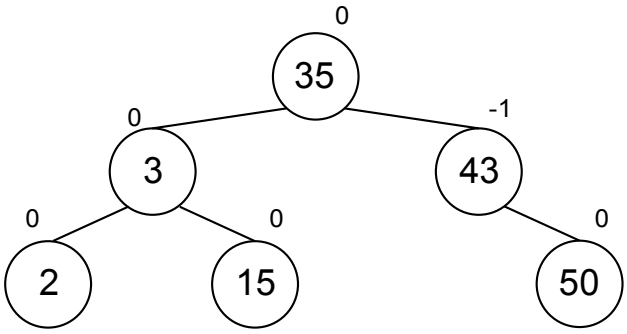
LEFT-ROTATION(*x*)

y = right(*x*)

temp = *left*(*y*)

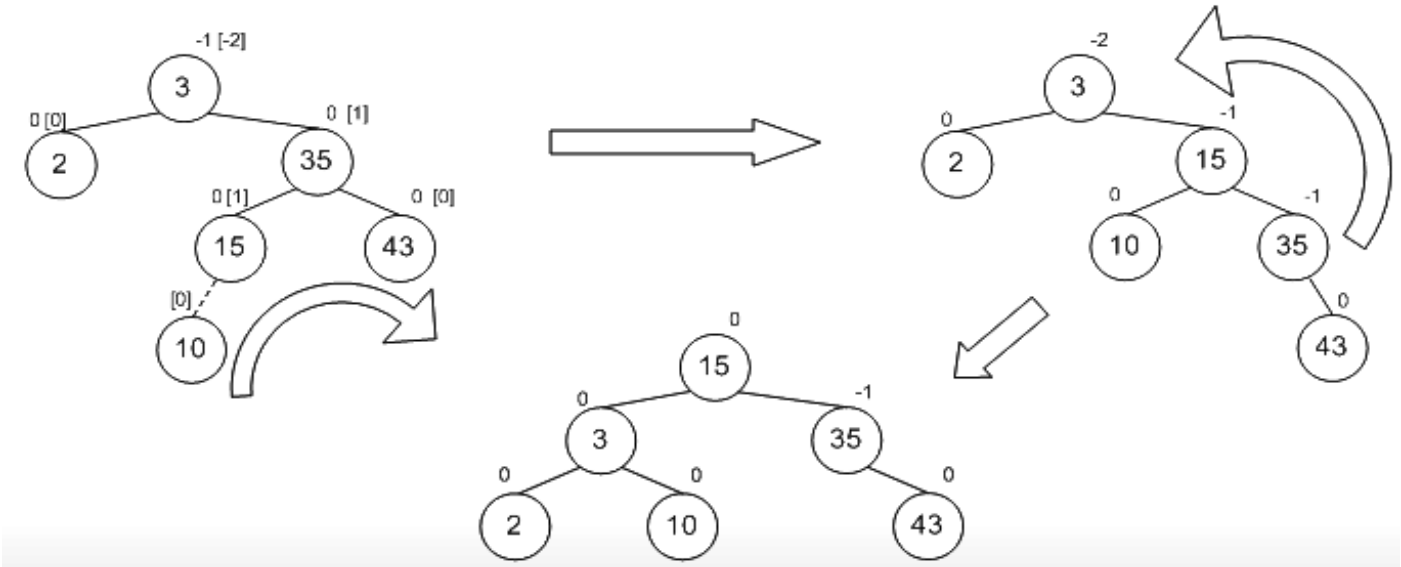
left(*y*) = *x*

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)



1. 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:

$1=C_1 + C_2 + C_3$

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

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

Rešenja sistema su:

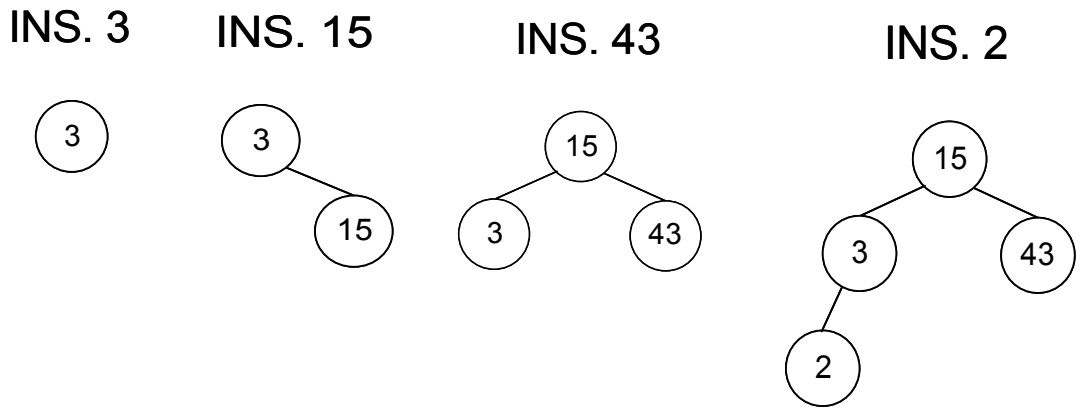
$C_1 = - 1$

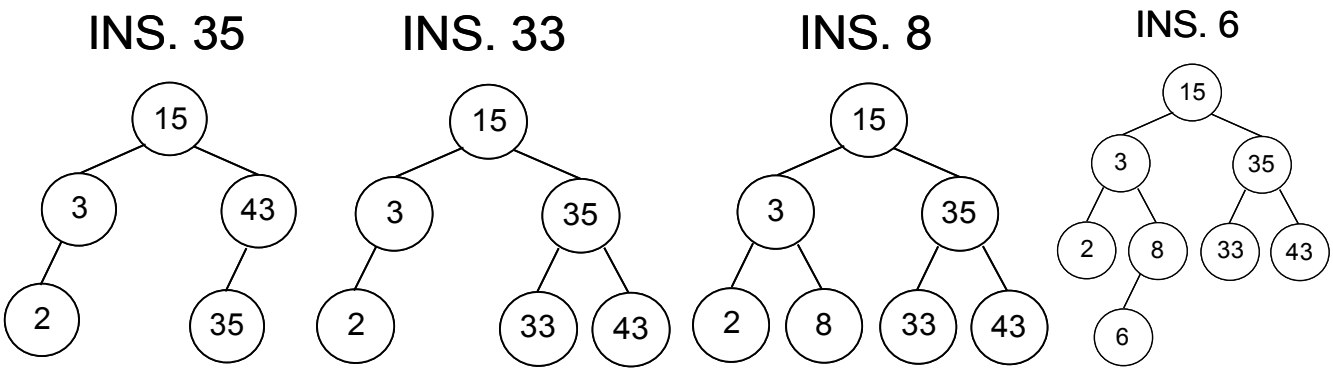
$C_2 = \frac{2+\sqrt{5}}{\sqrt{5}}$

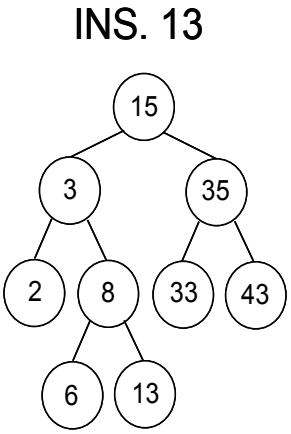
$C_3 = \frac{\sqrt{5}-2}{\sqrt{5}}$

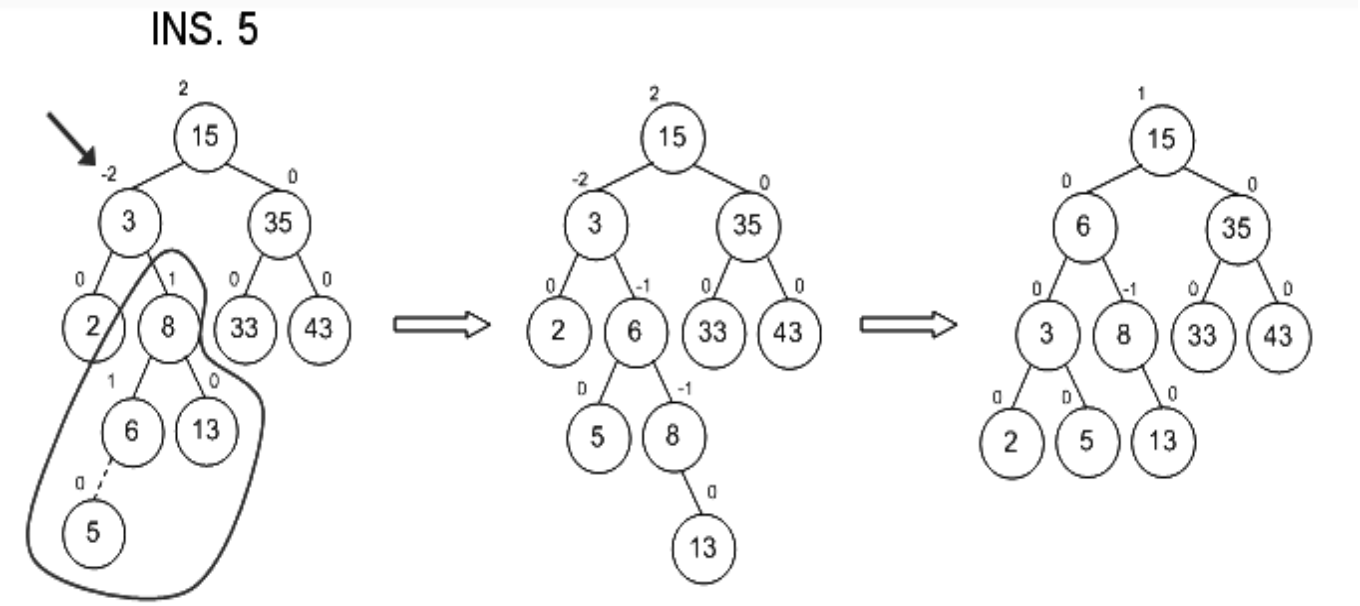
2.Odrediti izgled AVL stabla dobijenog umetanjem redom brojeva 3, 15, 43, 2, 35, 33, 8, 6, 13, 5

REŠENJE:









3. *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 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 cvor *v*, vec 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 cvor *v* sa podstablom T, a desni sin koren G. To je u redu sa stanovista BSP svojstva, jer *r* je najveci čvor u T. te je ključ čvora *v* ne veći od njega i može 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 nuzno obaviti eventualno uravnotezenje pomocu rotacije.

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

4. 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 najamanjeg 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 8), pronalazi sledbenik u stablu (O(log n)).

5. 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. Smatrati da čvor binarnog stabla je S-AVL čvor ako nije list i ako razlika visina njegovog levog i desnog podstabla je jednaka 0.

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 list

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

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

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.

Podsećanja:

ZADATAK iz lekcije Hash tabele: Pretpostavimo da se u hash tabeli umesto povezanih listi koriste binarna stabla pretraživanja za razrešavanje kolizija odvojenim ulančavanjem.
(a) Koliko je vreme izvršavanja operacija umetanja i pretraživanja u najgorem slučaju?
(b) Koliko je očekivano vreme izvršavanja istih operacija?
REŠENJE:

(a) Najgori slučaj se dešava kada se svih n ključeva u heš tabeli preslikaju heš funkcijom na istu poziciju u heš tabeli. Na taj način se na toj poziciji dobija binarno stablo pretraživanja od n elemenata. Takođe, tih n ključeva treba u heš tabelu da se umetnu u rastućem/opadajućem poretku kako bi binarno stablo pretraživanja dobilo oblik liste. U tom slučaju vreme izvršavanja operacija umetanja i pretraživanja iznosi $T(n) = O(n)$.

(b) Očekivane performanse heš tabele zavise od odnosa broja ključeva n koji su smešteni u heš tabelu i veličine heš tabele m. Ova veličina se zove faktor opterećenja (engl. *load factor*) heš tabele i označava se kao $\alpha = n/m$. Faktor opterećenja heš tabele u kojoj se za razrešavanje kolizija koristi odvojeno ulančavanje može biti manji, jednak ili veći od 1. On označava prosečan broj ključeva koji se očekuje na nekom indeksu heš tabele (u ovom slučaju to je broj elemenata u binarnom stablu pretraživanja) pod pretpostavkom da je heširanje uniformno. Da bismo videli zašto, označimo sa n_i, $i = 0, 1, 2, ..., m-1$, broj ključeva u i-tom binarnom stablu pretrživanja. Tada je ukupan broj ključeva u heš tabeli $n = n_0 + n_1 + ... + n_{m-1}$ i zbog toga

$$\frac{n_0 + n_1 + ... + n_{m-1}}{m} = \frac{n}{m} = \alpha$$

U ovom slučaju vreme izvršavanja operacija umetanja i pretraživanja iznosi $T(n) = O(\log \frac{n}{m})$.