14. ARBORI ECHILIBRAŢI

14.1 Echilibrarea structurilor arborescente

Pentru a descrie gradul de echilibru al structurilor arborescente sunt definite două abordări:

- arbori perfecţi echilibraţi în care pentru fiecare nod, diferenţa dintre numărul de noduri ale subarborelui drept şi stâng ia valori în mulţimea {-1 ; 0 ; +1}; într-un arbore perfect echilibrat de înălţime *h*, toate nodurile frunză sunt pe acelaşi nivel şi orice nod de pe nivelurile intermediare *1..h-2* are numărul maxim de fii; de exemplu, figura 14.1 descrie un arbore binar de căutare perfect echilibrat;

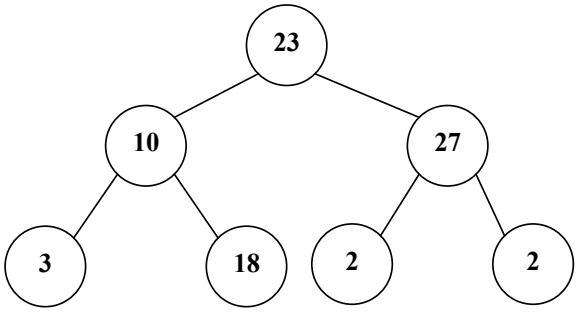

Figura 14.1 *Arbore binar perfect echilibrat*

cea mai simplă metoda de a obţine un astfel de arbore se bazează pe parcurgerea prin metoda *divide et impera* a şirului de chei ordonate crescător şi inserarea valorii din mijloc în arbore; aceasta metoda este ineficienta în practică deoarece presupune realizarea unui volum mare de calcule după fiecare operaţie de inserare sau ştergere ; efortul ridicat de prelucrare este dat de parcurgerea în inordine a arborelui pentru a obţine şirul sortat crescător al cheilor si de reconstrucţia structurii ; considerând un arbore binar de căutare, metoda utilizata în acest sens, *echilibrareArb*, are ca parametrii şirul sortat crescător al valorilor, vectorul *chei*, dimensiunea acestuia, *dim*, limitele intervalului curent, *stanga*, respectiv, *dreapta*, şi rădăcina arborelui ce va fi creat; menţinerea unei astfel de structuri reprezintă o operaţie cu grad de complexitate foarte ridicat, fapt care conduce la recrearea arborelui perfect echilibrat după fiecare operaţie de inserare sau ştergere cu metoda *echilibrareArb*;

```
void echlibrareArbore(int *chei, int dim, int stanga, int dreapta,
NodArbore *&radacina){
      if (dreapta>=stanga){
            int mijloc=(dreapta+stanga)/2;
            if (dreapta-stanga==1){
                  radacina =inserareArbore(radacina, chei[stanga]);
                  radacina = inserareArbore (radacina, chei[dreapta]);
            }
```

```
      else{
            if (dreapta==stanga)
                  radacina = inserareArbore (radacina, chei[stanga]);
            else{
            radacina = inserareArbore (radacina, chei[mijloc]);
            echlibrareArbore (chei,dim,stanga, mijloc -1, radacina);
            echlibrareArbore (chei,dim, mijloc +1,dreapta, radacina);
                  }
            }
      }
}
```

- imperfect echilibrat în care pentru fiecare nod, diferenţa dintre înălţimea subarborelui drept şi înălţimea subarborelui stâng ia valori în mulţimea {-1 ; 0 ; +1}; crearea unei astfel de structuri se bazează pe utilizarea metodei prezentate anterior pornind de la un set de valori sortate crescător sau descrescător; menţinerea gradului de echilibru al structurii după operaţiile de inserare sau ştergere este posibilă prin metode cu un grad de complexitate acceptabil şi care sunt specifice unor structuri arborescente echilibrate particulare, AVL, arbori B, arbori Rosu & Negru; aceste metode implică un efort de prelucrare mai mic decât volumul operaţiilor asociat reconstrucţiei arborelui prin metoda *echilibrareArb*.

Structurile arborescente sunt structuri de date dinamice în care elementele sunt poziţionate ierarhic în funcţie de legătura părinte – copil ce există între două elemente. Din punct de vedere al minimizării efortului de regăsire, această organizare este mai eficientă în raport cu structurile dinamice liniare, deoarece reduce numărul de comparări necesar identificării unui element. În cazul unei structuri de date liniare, cel mai nefavorabil caz descrie o complexitate egală cu O(m), unde *m* reprezintă numărul de elemente, şi este generat de căutarea ultimului element. Această situaţie este întâlnită şi în cazul structurilor arborescente ineficient construite, în care fiecare nod are maxim un fiu şi care au asociată imaginea unei liste.
Pentru a evita acest lucru şi pentru de a beneficia de efectele pozitive ale utilizării structurilor arborescente în operaţiile de căutare se definesc reguli stricte de realizare ale unei astfel de structuri. Prin prisma acestor reguli, structurile arborescente se diferenţiază pe mai multe tipuri. Dintre acestea, structurile arborescente echilibrate ocupă o pondere ridicată în dezvoltarea de soluţii eficiente deoarece descriu un nivel constant de efort apropiat de cel optim.

Se consideră şirul valorilor 23, 10, 27, 18, 3, 2 pe baza cărora se defineşte un arbore binar de căutare. Structura arborescenta obţinută este descrisă în figura 14.2.

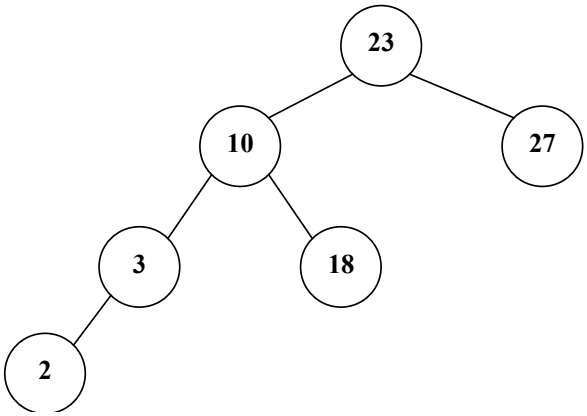

Figura 14.2 *Arbore binar de căutare*

Din analiza modului in care sunt poziţionate valorile, situaţia cea mai puţin favorabilă ce caracterizează acest arbore binar de căutare este data de cazul în care nodul ce conţine cheia cu valoarea 2 are o frecvenţă ridicată de utilizare. Situaţia este generată de faptul ca cel mai lung drum de la nodul rădăcină la un nod frunza este dat de drumul 23 – 10 – 3 – 2, de lungime 4. Dimensiunea este determinată de numărul de comparări necesare identificării nodului căutat.

Pe baza unei aranjări echilibrate a valorilor, acelaşi set de valori este reprezentat de structura arborescenta următoare

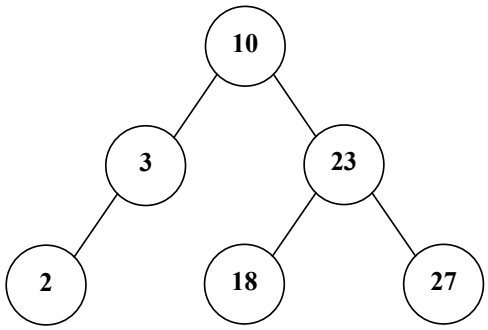

Figura 14.3 *Arbore binar de căutare echilibrat*

Prin prisma cazului cel mai puţin favorabil, arborele din figura 14.3 este mai eficient decât cel anterior deoarece numărul maxim de comparări necesare identificării oricărui nod din structura este egal cu 3.

În funcţie de numărul de elemente dintr-un arbore binar de căutare, *n*, o structură echilibrată descrie o complexitate egală cu $O(\log_2 n)$ pentru cel mai nefavorabil caz. În schimb, o structură arborescentă de acelaşi tip, dar care nu este echilibrată, este caracterizată pentru cel mai nefavorabil caz de o complexitate egală cu O(n).

Minimizarea efortului de regăsire a informaţiilor se obţine printr-o aranjare echilibrata a valorilor pe ambii subarbori ai fiecărui nod.

14.2 Caracteristici ale arborilor AVL

Un arbore AVL, definit prima dată de G.M. Adelson-Velskii şi E.M. Landis în [Ande62], este un arbore binar de căutare echilibrat pe înălţime. Un arbore binar de căutare este AVL dacă gradul de echilibru al fiecărui nod ia valori în mulţimea {-1,0,1}.

Pentru a măsura gradul de echilibru al unui nod se defineşte indicatorul GE ce descrie relaţia:

$$GE = H(SD) - H(SS) \qquad (14.1)$$

unde *H()* reprezintă funcţia de calcul a înălţimii unei structuri arborescente. Pentru a determina înălţimea unui arbore, al cărui nod rădăcină este *rad*, se utilizează formula:

$$H(rad) = 1 + max\ (H(subarbore\ drept), H(subarbore\ stâng)) \qquad (14.2)$$

în care funcţia *max*() este utilizată pentru a determina maximul dintre două valori.

```
int max(int valoare_1, valoare_2)
{
      return valoare_1 < valoare_2 ? valoare_2 : valoare_1;
}
```

Pentru valoarea indicatorului *GE = 0*, nodul este echilibrat, iar pentru valorile 1 si -1, nodul descrie un dezechilibru la dreapta, respectiv la stânga. Figura 14.4, descrie arborele binar de căutare pentru care s-a determinat gradul de echilibru.

Situaţiile în care GE are valoarea -1 sau 1 sunt acceptate deoarece, pentru un număr par de valori este imposibil sa se definească un arbore binar de căutare în care toate nodurile sunt perfect echilibrate.

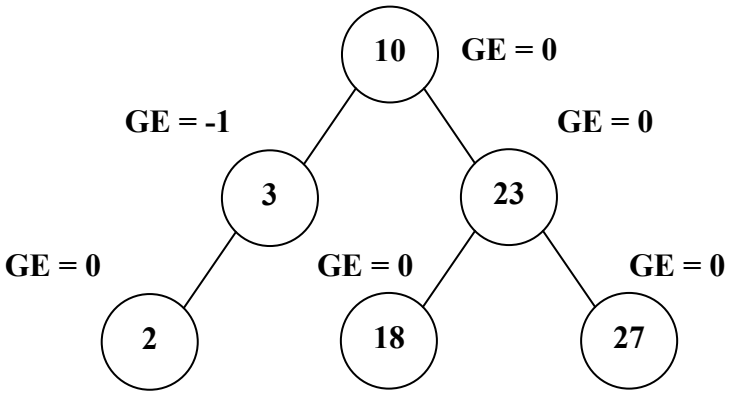

Figura 14.4 *Arbore binar de căutare echilibrat*

Arborele AVL, reprezintă un arbore binar de căutare echilibrat. Pornind de la această ipoteza, acest tip de arbore moşteneşte toate operaţiile implementate de arborii binari de căutare. Caracteristica de echilibru se gestionează prin verificarea atentă a gradului de echilibru, pentru fiecare nod în parte, în urma operaţiilor de inserare şi ştergere.

Aceste tipuri de prelucrări afectează structura arborelui şi conduc la situaţii de dezechilibru.

Pentru a menţine arborele AVL, după fiecare operaţie de inserare, respectiv ştergere, sunt căutate situaţiile de dezechilibru puternic, identificate prin intermediul nodurilor pentru care indicatorul GE ia valori în mulţimea {-2,2}.

Reechilibrarea arborelui binar de căutare şi păstrarea caracteristicilor aferente arborilor AVL se realizează prin operaţii de rotire:

- rotire simpla la stânga;
- rotire simpla la dreapta;
- dubla rotire la stânga ;
- dubla rotire la dreapta.

Este important de reţinut că printr-o singură rotaţie, selectată în funcţie de situaţie, un arbore AVL dezechilibrat în urma operaţiei de inserare va fi reechilibrat. În schimb, operaţie de reechilibrare în urma ştergerii unui nod este mult mai complexă, necesitând minim o rotaţie.

14.3 Operaţii pe arbori AVL

Determinarea metodei adecvate de reechilibrare se realizează prin analiza gradului de echilibru a nodurilor aflate pe drumul de la rădăcina arborelui la locaţia în care a fost inserat, respectiv şters, nodul.

Se considera situaţia din figura 14.4 în care se inserează nodul cu cheia 1. Arborele binar obţinut este:

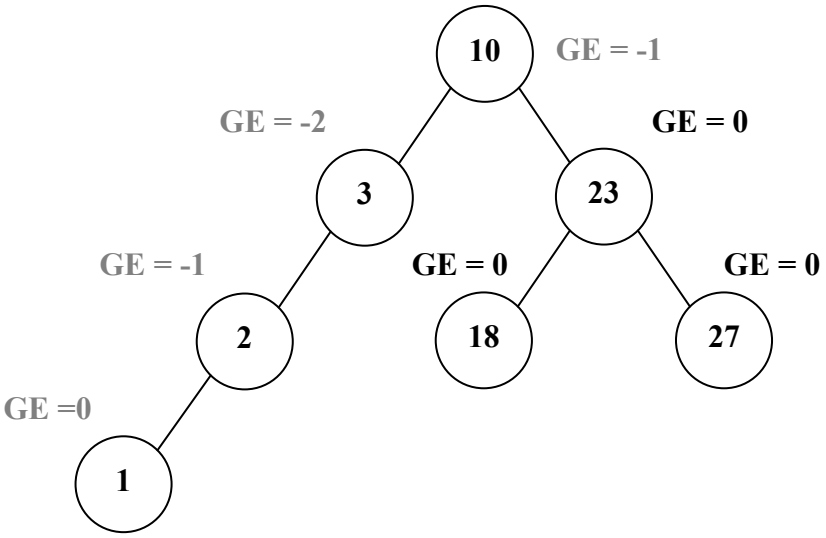

Figura 14.5 *Arbore binar AVL dezechilibrat*

În urma procesului de inserarea se recalculeaza gradul de echilibru al nodurilor ce sunt afectate. Aecstea noduri se gasesc in multimea nodurilor {10, 3, 2, 1}. Se observa ca arborele îşi pierde caracteristica de a fi AVL deoarece nodul cu cheia 3 are un grad de echilibru egal cu -2, ceea ce evidenţiază un dezechilibru puternic la stânga.

Pentru a aplica procesul de echilibrare, bazat pe operaţie de rotire, se identifica un nod, numit pivot, în care se realizează rotirea subarborelui.

Selectarea nodului pivot se face printr-o abordare jos-sus pornind de la locaţia nodului inserat, respectiv, şters către rădăcina arborelui.

Reechilibrarea arborelui se face cat mai aproape de locaţia care a generat dezechilibrul. Astfel, printr-un număr minim de rotaţii se reconstruieşte caracteristica arborelui AVL.

În figura, 14.5, nodul pivot este nodul a cărui cheie are valoarea 3. Pentru a reechilibra arborele este nevoie să transferăm o parte din greutatea subarborelui stâng către subarborele drept al nodului pivot.

Pentru a identifica operaţia de rotaţie corespunzătoare se analizează gradul de echilibru al nodului pivot si cel al nodului fiu de pe direcţia dezechilibrului. Analizând arborele din figura 14.4, se observa ca nodul pivot, cu cheia 3, are gradul de echilibru GE = -2, ceea ce implică un dezechilibru la stânga. Deoarece nodul fiu stânga, cu cheia 2, are dezechilibru simplu tot la stânga, reechilibrarea se realizează prin operaţia de rotire simplă la dreapta, figura 14.6.

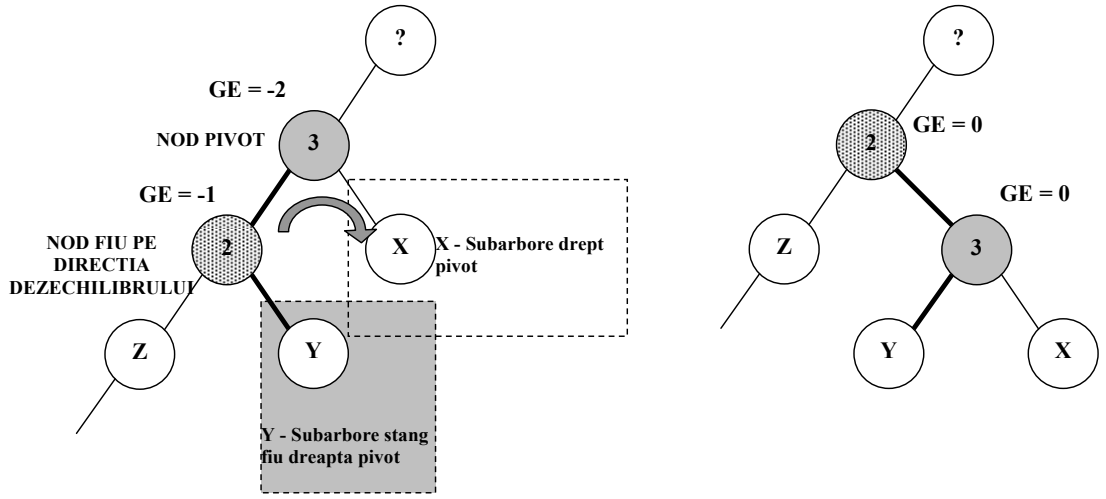

Figura 14.6 *Procesul de rotire simplă la dreapta*

Procesul de rotire simplă la dreapta implică existenţa a două elemente principale, nodul pivot, ce este dezechilibrat puternic la stânga şi fiul acestuia de pe direcţia dezechilibrului, care este la rândul său dezechilibrat slab tot la stânga. Pentru a reechilibra arborele în această situaţie este necesar şi suficient să scădem cu o unitate înălţimea subarborelui stâng al pivotului şi să creştem cu o unitate înălţimea subarborelui drept. Pentru a atinge acest obiectiv, se modifică cele două legături evidenţiate în figura 14.5**.** Se observă că subarborele Y devine subarbore drept pentru nodul pivot, fapt ce nu contrazice existenţa unui arbore binar de căutare deoarece toate valorile din acest subarbore sunt mai mici decât valoarea nodului pivot.

Prin aplicarea procesului de rotire, arborele se reechilibrează şi îşi păstrează caracteristicile specifice unui arbore AVL şi unui arbore binar de căutare. Se observă că prin rotaţia simplă la dreapta sunt afectate doar gradele de echilibru ale nodului pivot şi nodului fiu de pe direcţia dezechilibrului, nodul fiu stânga. Prin modificarea structurii arborescente, cele două noduri devin perfect echilibrate, GE = 0. Explicaţia este dată de faptul că nodul fiu stânga urcă pe nivelul superior în locul nodului părinte, iar acesta, fiind nod pivot, coboară în subarborele drept. Figura 14.7

utilizează ca reper înălţimea subarborelui Y pentru a descrie modul în care se ajunge la acest rezultat. Este evidenţiat modul de calcul al indicatorului GE pentru a sublinia modul în care se ajunge la rezultat.

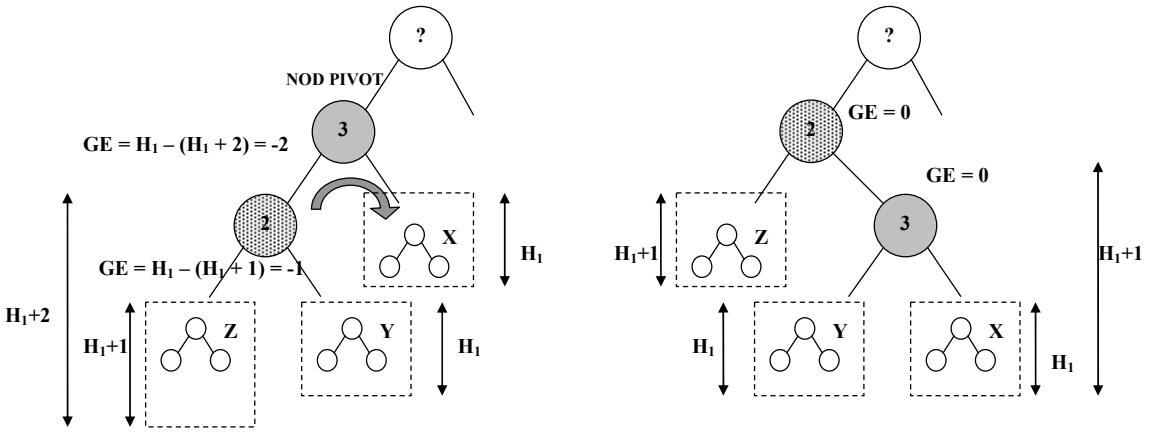

Figura 14.7 *Procesul de rotire simplă la dreapta*

Metoda clasei *AVLArbore* ce implementează această rotaţie simplă primeşte ca parametru referinţa la nodul pivot.

```
void AVLArbore::RotatieSimplaDreapta(AVLNod * &pivot)
{
      AVLNod *FiuStanga = pivot->st;
      pivot->st = FiuStanga->dr;
      FiuStanga->dr = pivot;

      pivot->Echilibru = 0;
      FiuStanga->Echilibru = 0;

      pivot = FiuStanga;
}
```

Aplicând această metodă arborelui analizat, rezultă structura arborescentă din figura 14.8.

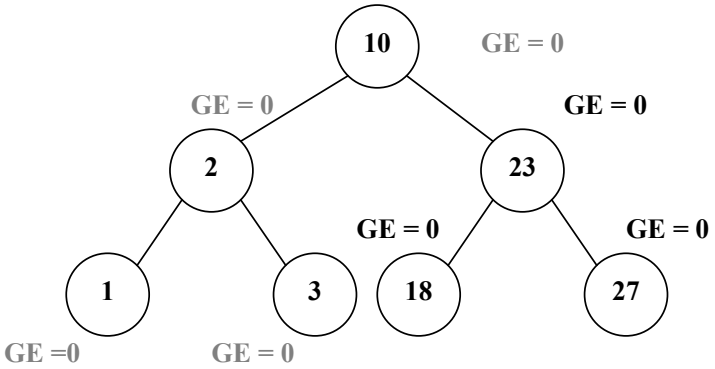

Figura 14.8 *Arbore AVL reechilibrat*

Dacă prin inserarea sau ştergerea unui nod se ajunge în situaţia din figura 14.9, reechilibrarea arborelui se realizează printr-o rotire simplă la stânga.

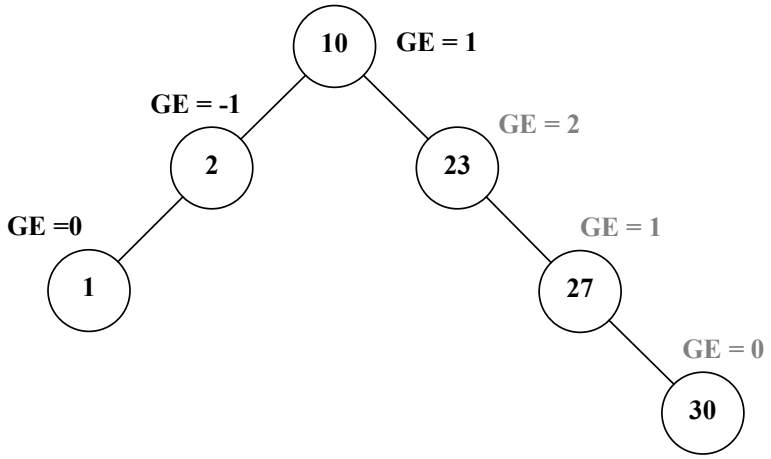

Figura 14.9 *Arbore AVL dezechilibrat*

În această situaţie, pivotul este dat de nodul cu valoarea 23, acesta fiind puternic dezechilibrat la dreapta, GE = 2. Deoarece nodul fiu dreapta, este dezechilibrat slab pe aceeaşi direcţie, reechilibrarea se realizează prin operaţia de rotire simplă la stânga.

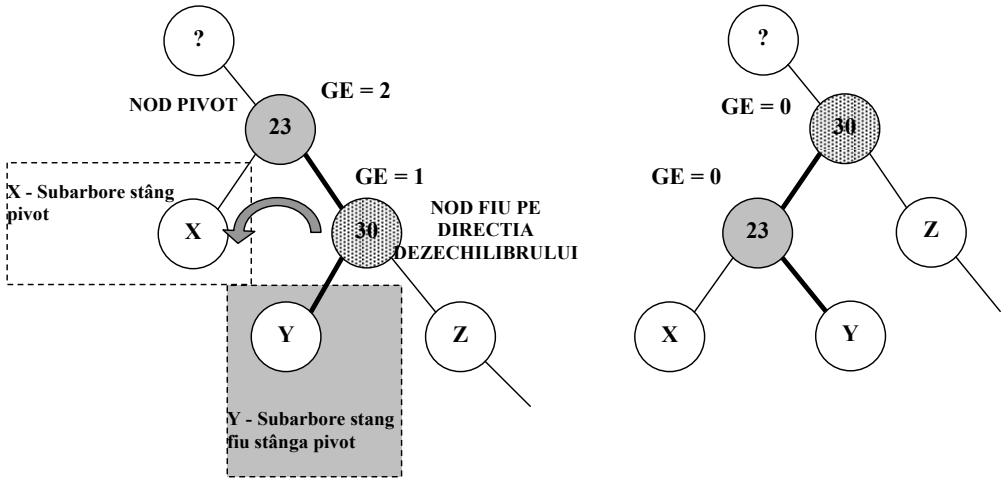

Figura 14.10 *Procesul de rotire simplă la stânga*

Asemănător operaţiei de rotaţie simplă la dreaptă, cele două noduri afectate direct de reorganizarea legăturilor, nodul pivot şi fiul acestuia din dreapta, au în final grade de echilibru egale cu valoarea zero. Reorganizarea celor două legături evidenţiate în figura 14.10 are ca efect reducerea cu o unitate a înălţimii subarborelui drept al nodului pivot şi creşterea cu o unitate a subarborelui stâng.

Metoda ce implementează această operaţie, *RotatieSimplaStanga*, are ca parametru de intrare referinţa nodului pivot.

```
void AVLArbore::RotatieSimplaStanga(AVLNod * &pivot)
{
      AVLNod *FiuDreapta = pivot->dr;
      pivot->dr = FiuDreapta->st;
      FiuDreapta->st = pivot;
```

```
        pivot->Echilibru = 0;
        FiuDreapta->Echilibru = 0;

        pivot = FiuDreapta;
}
```

Aplicând această operaţie arborelui din figura 14.10 se obţine arborele AVL:

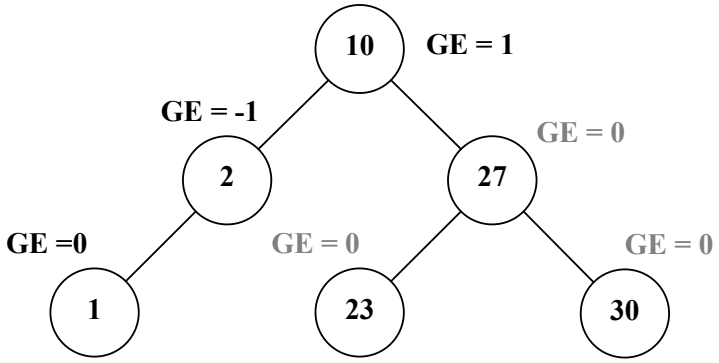

Figura 14.11 *Arbore AVL reechilibrat*

În cazul operaţiile de rotire duble, situaţia iniţială este caracterizată de sensuri opuse de dezechilibru pentru nodul pivot si pentru nodul său fiu de pe direcţia dezechilibrului. Pentru a exemplifica o astfel de situaţie se inserează în arborele AVL din figura 14.11, elementele cu valorile 16, 24, 26. Structura arborescentă obţinută este:

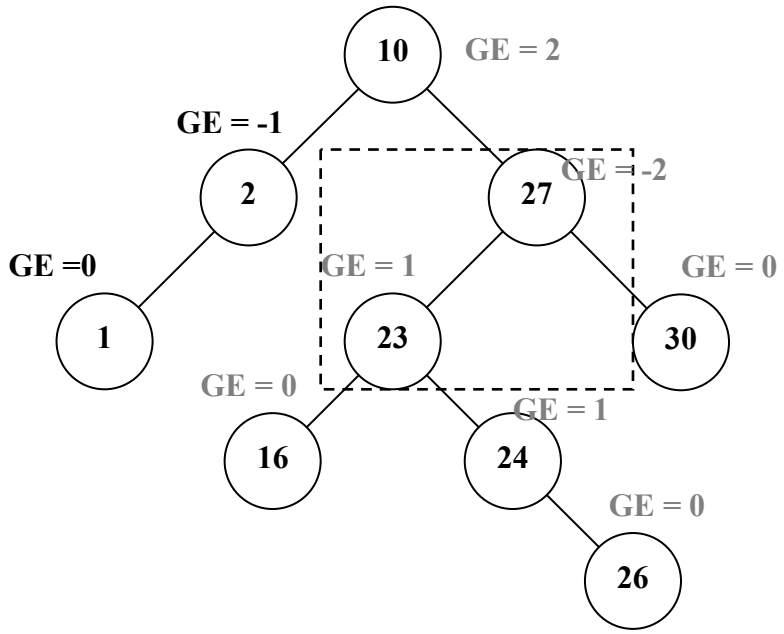

Figura 14.12 *Arbore AVL dezechilibrat*

Structura arborescentă din figura 14.12 nu este arbore AVL deoarece există noduri pentru care gradul de echilibru, GE, are valori în mulţimea {-2, +2}. Situaţia este generată de inserarea nodului cu valoarea 26, iar analiza drumului de la acest nod înapoi către nodul rădăcină conduce la

identificarea pivotului, nodul cu valoarea 27. Se observă că, acest nod este puternic dezechilibrat la stânga, iar nodul fiu de pe această direcţie, nodul 23, este dezechilibrat slab pe direcţia opusă. Soluţia pentru această problemă necesită o abordare diferită de cele două tipuri de rotiri simple descrise, deoarece acestea nu conduc la reechilibrarea arborelui. Pentru a exemplifica această abordare greşită se simulează o rotire simplă la dreapta aplicată pivotului. Rezultatul obţinut este:

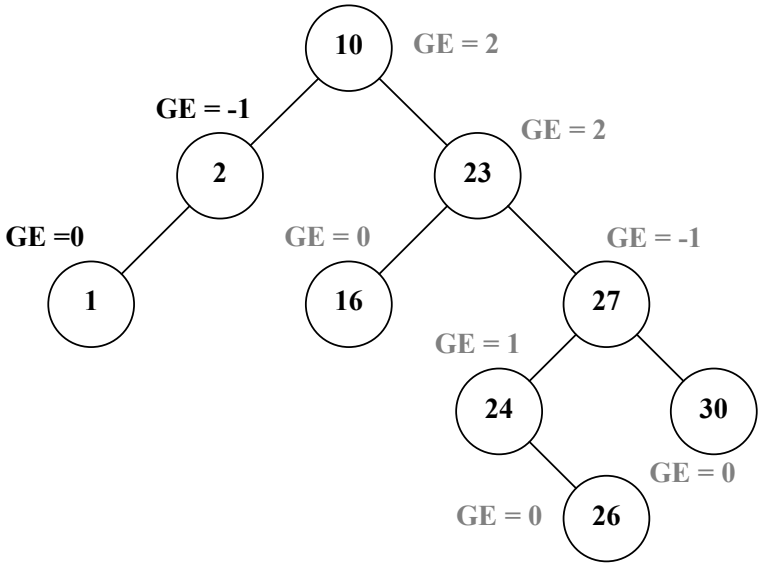

Figura 14.13 *Arbore AVL dezechilibrat*

Se observă că arborele, figura 14.13, este în continuare dezechilibrat, numai că de data aceasta, dezechilibrul este în sens opus. Încercarea reechilibrării, tot cu o rotire simplă, dar în sens opus, va conduce la obţinerea ipotezei iniţiale, descrisă în figura 14.12.

Soluţia eficientă a acestui tip de dezechilibru este dată de aplicarea unei rotiri duble, ce constă în aplicarea a două rotiri simple. Scopul primei rotiri este de a rearanja structura arborescentă astfel încât direcţiile dezechilibrului nodului pivot şi a fiului acestuia să aibă acelaşi sens. Cea de-a doua rotire are ca obiectiv reechilibrarea arborelui. Pe baza acestor motive, cele două rotaţii sunt aplicate unor noduri diferite. Prima rotaţie se aplică nodului fiu al nodului pivot, nod ce se găseşte pe direcţia dezechilibrului. Sensul acestei prime rotiri este identic cu direcţia dezechilibrului. A două rotire simplă se aplică nodului pivot şi are sens opus dezechilibrului.

Pentru structura arborescentă din figura 14.12, pivotul este dat de nodul cu valoarea 27 şi acesta este puternic dezechilibrat la stânga. Pentru a reechilibra arborele se parcurg următoarele etape:

- se analizează nodul fiu al nodului pivot pe direcţia dezechilibrului; acest nod are valoarea 23 şi este slab dezechilibrat la dreapta;
- deoarece pivotul şi nodul fiu sunt dezechilibrate pe direcţii diferite, reechilibrarea se realizează printr-o dublă rotaţie;
- prima rotaţie se aplică nodului fiu şi are sens identic cu dezechilibrul nodului pivot; se observă că această operaţie intermediară, figura 14.12.a, redefineşte situaţia aducând-o într-o formă specifică cazurilor în care se aplică rotaţii simple;

- a doua rotaţie se aplică nodului pivot şi are sens opus dezechilibrului.

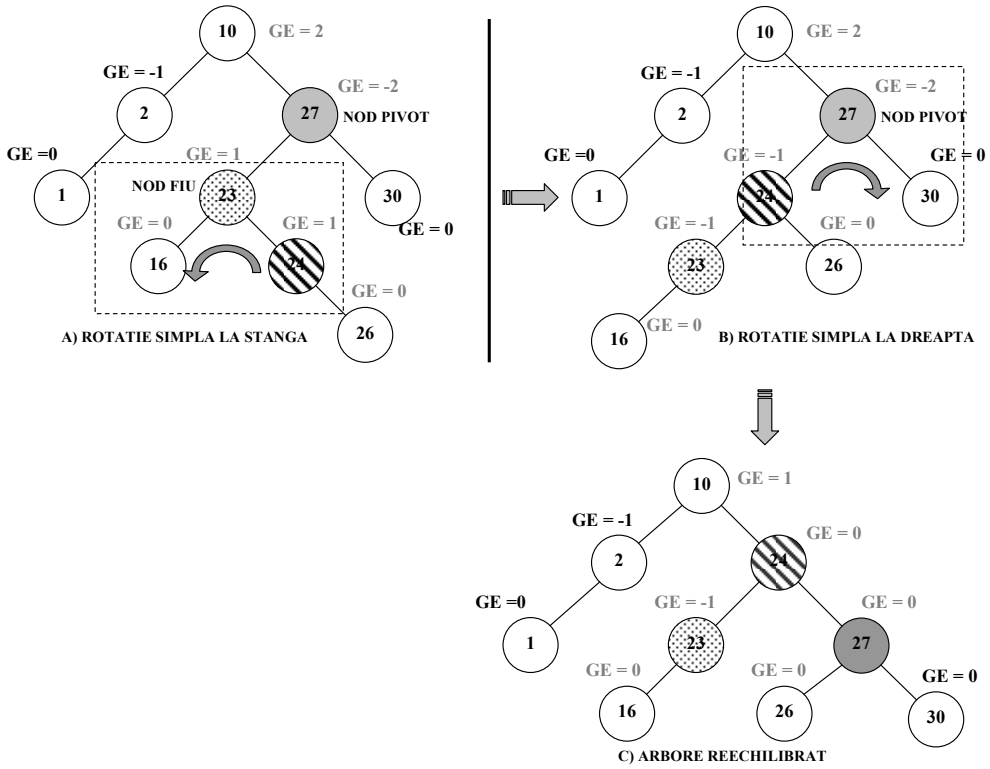

Figura 14.14 *Rotaţia dublă la dreapta*

Din analiza dublei rotaţii la dreapta sunt evidenţiate 3 elemente importante, în funcţie de care sunt reiniţializate o serie de legături:

- nodul pivot, puternic dezechilibrat la stânga, GE = -2;
- fiul stânga la pivotului, *FiuStanga* ,ce este slab dezechilibrat la dreapta, GE = 1;
- fiul dreapta al nodului *FiuStanga*, notat cu *FiuStanga_FiuDreapta*; în funcţie de situaţie, acest nod prezintă un grad de echilibru ce ia valori în mulţimea {-1, 0 , 1}.

Pentru a determina gradul de echilibru final al nodurilor afectate de dubla rotaţie, se analizează modul în care se distribuie înălţimea subarborilor în urma rotaţiilor. Tabelul 14.1 descrie situaţiile iniţiale şi rezultatele la care se ajunge în urma reechilibrării.

Tabelul nr. 14.1 *Rezultatul operaţiei de dublă rotaţie la dreapta*

Situaţie iniţială			**Situaţie finală**		
Pivot	**FiuStanga**	**FiuStanga_FiuDreapta**	**Pivot**	**FiuStanga**	**FiuStanga_FiuDreapta**
-2	+1	-1	1	0	0
-2	+1	0	0	0	0
-2	+1	+1	0	-1	0

Situaţia descrisă în tabelul anterior este utilizată pentru a defini mai eficient metoda care implementează acest tip de rotaţie. Astfel este evitat efortul suplimentar de a recalcula gradul de echilibru pentru cele trei noduri afectate.

Clasa *AVLArbore* implementează această operaţie prin intermediul metodei *RotatieDublaDreapta* ce primeşte ca parametrul referinţa nodului pivot.

```
void AVLArbore::RotatieDublaDreapta(AVLNod * &pivot)
{
	AVLNod *FiuStanga, *FiuStanga_FiuDreapta;
	FiuStanga = pivot->st;
	FiuStanga_FiuDreapta = FiuStanga->dr;

		//realizare rotatie 1 - simpla stanga
	FiuStanga->dr = FiuStanga_FiuDreapta->st;
	FiuStanga_FiuDreapta->st = FiuStanga;
		//realizare rotatie 2 - simpla dreapta
	pivot->st = FiuStanga_FiuDreapta->dr;
	FiuStanga_FiuDreapta->dr = pivot;

		//modificare grade de echilibru
	if(FiuStanga_FiuDreapta->Echilibru == 1)
	{
		pivot->Echilibru = 0;
		FiuStanga->Echilibru = -1;
	}
	else
		if(FiuStanga_FiuDreapta->Echilibru == 0)
		{
			pivot->Echilibru = 0;
			FiuStanga->Echilibru = 0;
		}
		else
		{
			pivot->Echilibru = 1;
			FiuStanga->Echilibru = 0;
		}

		FiuStanga_FiuDreapta->Echilibru=0;

	pivot = FiuStanga_FiuDreapta;
}
```

Structura arborescentă este modificată prin ştergerea nodului cu valoarea 16 şi prin adăugarea unei noi valori, 25. Arborele obţinut, descris în figura 14.15, încetează să mai fie AVL în urma aplicării ultimei modificări.

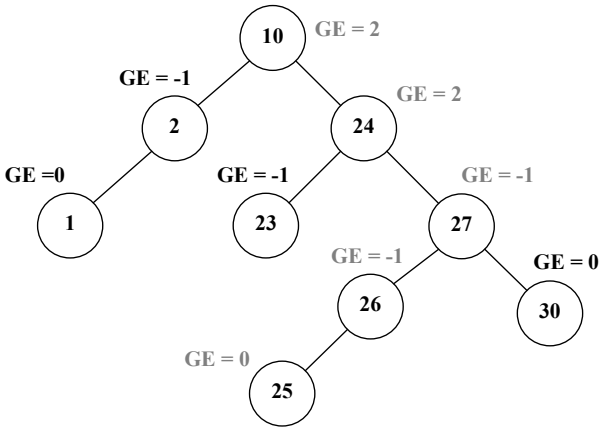

Figura 14.15 *Arbore AVL dezechilibrat*

Se observă că există două noduri, cu valoarea 24 şi 10, ce descriu dezechilibre puternice, GE = 2, la dreapta. Analizând, de jos în sus, drumul de la noul nod inserat la rădăcină arborelui, se stabileşte ca fiind pivot nodul cu valoarea 24. În mod asemănător cu situaţia descrisă anterior, fiul pivotului de pe direcţia dezechilibrului este dezechilibrat uşor în sens opus. Tentativa de a rezolva situaţia prin intermediul unei rotaţii simple nu conduce la soluţionarea problemei reechilibrării deoarece are ca rezultat mutarea dezechilibrului pe partea stângă.

Având în vedere condiţiile de lucru, reechilibrarea arborelui din figura 14.15 presupune:

- aplicarea unei rotaţii simple la dreapta în nodul fiu al pivotului; dacă pivotul are ambii fii atunci rotaţia se face în toate situaţiile asupra nodului fiu de pe direcţia dezechilibrului; deoarece pivotul este dezechilibrat puternic la dreapta, nodul fiu selecta este 27;
- aplicarea unei rotaţii simple la stânga, în sens opus dezechilibrului, în nodul pivot.

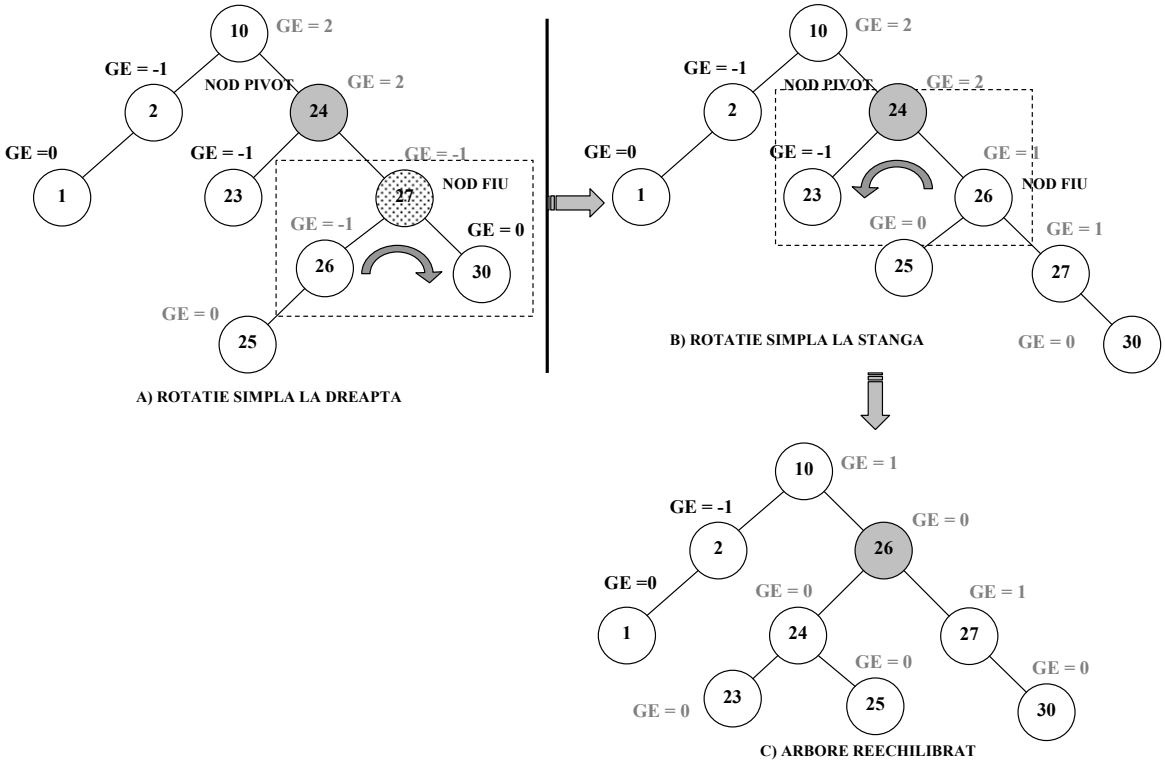

Figura 14.16 *Rotaţia dublă la stânga*

Pentru a implementa o soluţie software care să gestioneze datele prin intermediul unui arbore binar de căutare echilibrat de tip AVL, trebuie să fie dezvoltate rutine complementare operaţiilor de inserare şi ştergere în arbori binari de căutare care să reechilibreze structura arborescentă aflată în una din cele patru situaţii descrise.

Tabelul 14.2 sintetizează situaţiile de dezechilibru şi modul în care arborele AVL este menţinut în urma operaţiilor de inserare.

Tabelul nr. 14.2 *Situaţii dezechilibru arbori AVL*

Grad echilibru nod pivot	Nod fiu analizat	Grad echilibru nod fiu	Rotire
+2	dreapta	+1	Simplă la stânga
+2	dreapta	-1	Dublă la stânga: rotire simplă la dreapta în fiul din dreapta al pivotului; rotire simplă la stânga în pivot.
-2	stânga	-1	Simplă la dreapta
-2	stânga	+1	Dublă la dreapta: rotire simplă la stânga în fiul din stânga al pivotului; rotire simplă la dreapta în pivot.

Dezvoltarea de aplicaţii care implementează lucrul cu arbori de tip AVL se bazează pe dezvoltarea unei biblioteci de cod în care sunt definite clasele *AVLNod* şi *AVLArbore*. Clasa *AVLNod* descrie atributele şi metodele unui obiect ce reprezintă nodul unui arbore binar de căutare echilibrat.

```
class AVLNod
{
private:
      int Echilibru;
      int Info;
      AVLNod *st;
      AVLNod *dr;

public:
      //constructorii clasei
      AVLNod(void);
      AVLNod(int echilibru, int info, AVLNod * stanga, AVLNod *
dreapta);
      //destructorul clasei
      virtual ~AVLNod(void);

      //interfata pentru atributul Echilibru
      int GetEchilibru(void){return this->Echilibru;};

      // acces la atributele private din clasa AVLArbore
      friend class AVLArbore;
      //acces la atributele private din clasa AVLNodeStack
      friend class AVLNodeStack;
};
```

În comparaţie cu nodul unui arbore binar de căutare, această clasă defineşte o proprietate nouă, *Echilibru*, utilizată pentru a gestiona gradul de echilibru asociat fiecărui nod. Atributele *Info*, *st* şi *dr* sunt utilizate pentru a memora valoarea nodului curent şi pentru a face legătură între nodul părinte şi nodul fiu stânga, respectiv, dreapta.
Cele două metode constructor

```
AVLNod::AVLNod(void)
{
      Echilibru = 0;
      Info = 0;
      st = NULL;
      dr = NULL;
```

```
}
AVLNod::AVLNod(int echilibru, int info, AVLNod * stanga, AVLNod *
dreapta)
{
	Echilibru = echilibru;
	Info = info;
	st = stanga;
	dr = dreapta;
}
```

permit programatorilor crearea şi iniţializarea unui nod al arborelui cu valori implicite sau pe baza unor parametrii de intrare.

Clasa *AVLArbore* defineşte atributele şi metodele unui obiect de tip arbore AVL. Acesta gestionează structura dinamică de elemente prin intermediul referinţei către nodul rădăcină, *radacina*.

```
class AVLArbore
{
public:
	AVLNod *radacina;
public:

	//constructorul clasei
	AVLArbore(void);
	//constructorul de copiere al clasei
	AVLArbore(const AVLArbore & arbore);
	//destructorul clasei
	virtual ~AVLArbore(void);
	//operatorul =
	AVLArbore operator = (AVLArbore & arbore);

	//metodele clasei pentru inserare/stergere nod
	void Insert(const int info);
	void Delete(const int info);

	//metoda pentru afisarea arborelui
	static void AfisareArbore(AVLNod * rad);

	//metoda pentru stergerea arborelui
	void StergereArbore(AVLNod * &rad);

private:
	void AVLInsert(AVLNod* &arbore,AVLNod * nodNou, int &
echilibruNou);
	void AVLDelete(AVLNod* &arbore,const int Info,AVLNodeStack
&stiva);

	//rotatii simple utilizate la inserare
	void RotatieSimplaDreapta(AVLNod * &pivot);
	void RotatieSimplaStanga(AVLNod * &pivot);

	//rotatii simple utilizate la stergere
	void RotatieSimplaDreaptaStergere(AVLNod * &pivot);
	void RotatieSimplaStangaStergere(AVLNod * &pivot);

	void RotatieDublaDreapta(AVLNod * &pivot);
	void RotatieDublaStanga(AVLNod * &pivot);

	//metodele clasei pentru reechilibrarea arborelui
```

```
      void ReechilibrareSubarboreStang(AVLNod * &pivot, int
&echilibruNou);
      void ReechilibrareSubarboreDrept(AVLNod * &pivot, int
&echilibruNou);

      //metoda utilizata pentru copierea arborelui
      void CopiereArbore(AVLArbore &arboreNou, AVLNod * rad);

      //metoda inserare a unui arbore binar de cautare
      AVLNod * Inserare(AVLNod *rad, const int Valoare, int echilibru
= 0);
      static int Stergere(AVLNod*& Subarbore, AVLNodeStack &stiva);

      //metoda pentru determinarea inaltimii unui arbore
      int inaltime(AVLNod * radacina);

      //metoda ce determina maximul dintre doua valori
      int max(int a, int b){return a < b? b : a;}

      //metoda ce determina gradul de echilibru al nodului
      int CalculeazaEchilibru(AVLNod *& radacina);

      //metoda recalculeaza gradul de echilibru pentru toate nodurile
      void RecalculeazaEchilibrul(AVLNod *&rad);
};
```

O atenţie deosebită se acordă formei dată de programator a constructorului de copiere şi a operatorului =. Necesitatea este dată de existenţa atributului dinamic *AVLNod *radacina* şi de efectele negative pe care le au formele implicite ale acestor două metode asupra programului. Programatorul trebuie să se asigure că în situaţiile în care aceste două metode sunt apelate se vor crea structuri noi cu valori egale şi nu se vor face doar simple iniţializări de referinţe către aceeaşi zonă de memorie.

Copierea arborelui presupune parcurgerea structurii existente, cu păstrarea caracteristicilor acesteia. Din acest motiv, cele două metode se bazează pe o parcurgere în preordine a arborelui existent, completată de inserarea nodului curent în structura nou creată. Spre deosebire de parcurgere în inordine şi postordine, parcurgere în preordine asigură crearea unui nou arbore binar de căutare identic cu structura sursă şi cu minim de efort.

Se consideră structura arborescentă din figura 14.17 pentru care se obţin şirurile valorilor elementelor, parcurgând arborele prin cele trei metode cunoscute.

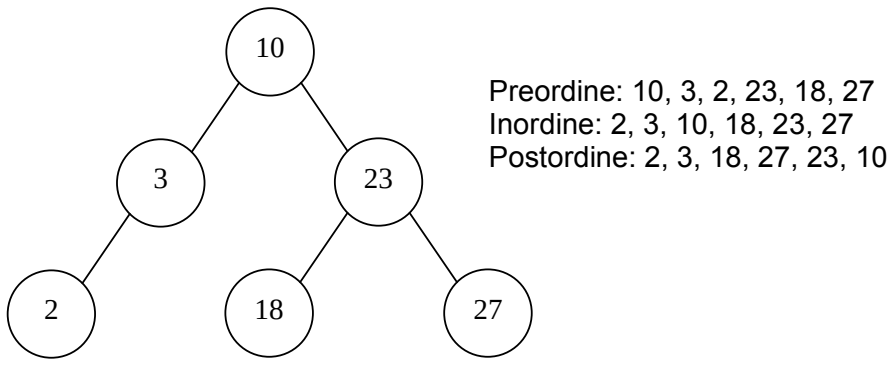

Figura 14.17 *Structură arborescenta de tip AVL*

Prin inserarea valorilor într-o nouă structură arborescentă, pe măsură ce acestea sunt accesate şi analizate, se obţin cei trei arbori binari de căutare din figura 14.18.

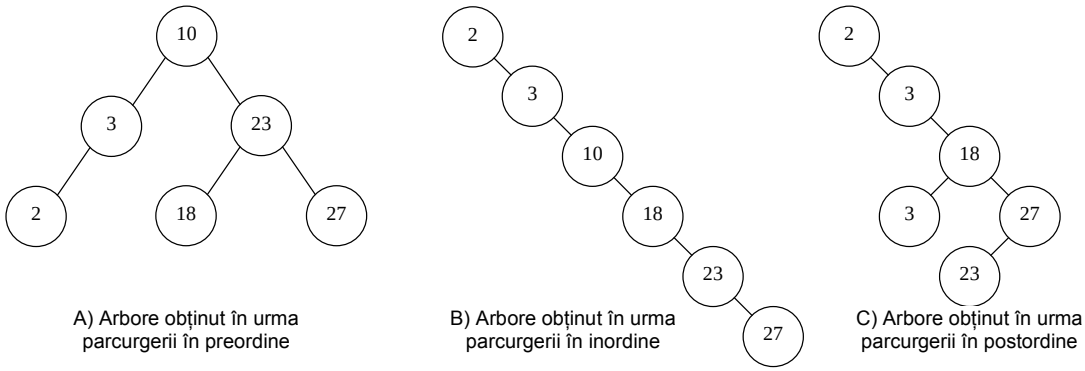

Figura 14.18 *Structuri arborescente binare*

Se observă că, dintre cele trei metode de parcurgere a unui arbore binar, cea mai potrivită pentru operaţia de copiere este abordarea în preordine. Celelalte două metode necesită un efort suplimentar de rearanjare a nodurilor şi nu asigură obţinerea unei arbore identic cu sursa. Din punct de vedere al reechilibrării, efortul este mult mai mare datorită prelucrărilor suplimentare.

Metoda *CopiereArbore* construieşte copia arborelui *radArboreVechi* parcurgând-ul în preordine.

```
void AVLArbore::CopiereArbore(AVLArbore &arboreNou, AVLNod
*radArboreVechi)
{
      if(radArboreVechi!=NULL)
      {
            arboreNou.radacina =
arboreNou.Inserare(arboreNou.radacina,radArboreVechi->Info,
radArboreVechi->Echilibru);
            CopiereArbore( arboreNou, radArboreVechi->st);
            CopiereArbore( arboreNou, radArboreVechi->dr);
      }
}
```

Metoda anterioră, se bazează pe parcurgerea recursivă a arborelui curent şi apelează rutina *Inserare* specifică arborilor binari de văutare pentru a insera o valoare într-o nouă structură arborescentă gestionată prin pointerul *arboreNou*.

```
AVLNod * AVLArbore::Inserare(AVLNod *rad, const int Valoare, int
echilibru)
{
      if(rad == NULL)
      {
            rad = new AVLNod(echilibru,Valoare, NULL, NULL);
      }
      else
            if(rad->Info<Valoare)
                  rad->dr = Inserare(rad->dr,Valoare,echilibru);
```

```
            else
                  if(rad->Info>Valoare)
                        rad->st = Inserare(rad->st,Valoare,echilibru);
      return rad;
}
```

Formele explicite ale constructorului de copiere şi a operatorului de egal implementează rutina de copiere a unui arbore pentru a genera noi structuri arborescente cu valori identice.

```
AVLArbore::AVLArbore(const AVLArbore &arbore)
{
      this->radacina = NULL;
      CopiereArbore((*this),arbore.radacina);
}
```

Spre deosebire de constructorul de copiere, operatorul = presupune ştergerea arborelui existent şi recrearea acestuia prin copierea valorilor structurii *arbore*.

```
AVLArbore AVLArbore::operator = (AVLArbore & arbore)
{
      StergereArbore(this->radacina);
      CopiereArbore((*this),arbore.radacina);
      return *this;
}
```

Metoda utilizată pentru ştergerea arborelui AVL este dată de operaţia specifică structurilor arborescente binare, ce realizează eliberarea memoriei de jos în sus, pornind cu nodurile frunză.

```
void AVLArbore::StergereArbore(AVLNod * &rad){
      if(rad!=NULL){
            StergereArbore(rad->st);
            StergereArbore(rad->dr);
            delete rad;
            rad = NULL;
      }
}
```

Operaţia de inserare în arborii AVL este derivată din metoda specifică arborilor binari de căutare. Operaţiile suplimentare sunt necesare procesului de reechilibrare şi de conservare a caracteristicii acestui tip de structură, menţinerea gradului de echilibru în mulţimea {-1; 0; 1} pentru toate nodurile arborelui.

Metoda AVLInsert parcurge o serie de etape necesare inserării unui nou nod, *nodNou*, într-un arbore de tip AVL, gestionat prin intermediul pointerului *arbore*:

- dacă arborele este vid, noul nod devine rădăcina arborelui AVL;
- dacă arborele există, se caută poziţia noului nod prin parcurgerea acestuia asemenea unui arbore binar de căutare; parcurgerea este recursivă, accesându-se nodul fiu stânga sau dreapta funcţie de rezultatul comparării valorii nodului nou cu valoarea nodului curent;

- se recalculează gradul de echilibru pentru toate nodurile parcurse; fiind un proces recursiv, revenirea din apelul rutinei asigură poziţionarea pe nodul anterior; variabilele *echilibruNou* şi *Reechilibrare* indică faptul că a avut loc o modificare de structură în apelul anterior, lucru care poate conduce la dezechilibre; în cazul în care aceste variabile sunt iniţializate cu valoare 1, este testat gradul de echilibru al nodului curent;

```
void AVLArbore::AVLInsert(AVLNod* &arbore,AVLNod * nodNou, int &
echilibruNou){
      int Reechilibrare;

      if(arbore == NULL){
            arbore = nodNou;
            arbore->Echilibru = 0;
            echilibruNou = 1;
      }
      else
            if(nodNou->Info<arbore->Info){
                  AVLInsert(arbore->st,nodNou,Reechilibrare);
                  if(Reechilibrare){
                        if(arbore->Echilibru == -1)

      ReechilibrareSubarboreStang(arbore,echilibruNou);

                        else
                              if(arbore->Echilibru == 0){
                                    arbore->Echilibru = -1;
                                    echilibruNou = 1;
                              }
                              else{
                                    arbore->Echilibru = 0;
                                    echilibruNou = 0;
                              }
                  }
                  else
                        echilibruNou = 0;
            }
            else{
                  if(nodNou->Info>arbore->Info){
                        AVLInsert(arbore->dr, nodNou, Reechilibrare);
                        if(Reechilibrare){
                              if(arbore->Echilibru == -1){
                              arbore->Echilibru = 0;
                              echilibruNou = 0;
                        }
                        else
                              if(arbore->Echilibru == 0){
                                    arbore->Echilibru = 1;
                                    echilibruNou = 1;
                              }
                              else

      ReechilibrareSubarboreDrept(arbore,echilibruNou);
                        }
                        else
                              echilibruNou = 0;
                  }
                  else
                        echilibruNou = 0;
```

```
}}
```

- identificarea nodului dezechilibrat, pivotul operaţiilor de rotire, este realizată doar dacă variabila *Reechilibrare* este setată, prin verificarea elementelor vizitate;
- dacă nodul curent are gradul de echilibru egal cu -1 iar nodul nou a fost inserat în subarborele stâng, are loc reechilibrarea acestuia prin apelul metodei *ReechilibrareSubarboreStang*;
- dacă nodul curent are gradul de echilibru egal cu 0 sau +1 iar nodul nou a fost inserat în subarborele stâng, atunci noul grad de echilibru al elementului curent este -1, respectiv 0; prin iniţializarea variabilei *echilibruNou* cu valoare 1 se continuă verificarea dezechilibrului la nodurile superioare; dacă nodul curent devine perfect echilibrat, se opreşte verificarea în acest punct, iar *echilibruNou* ia valoarea 0;
- dacă nodul curent are gradul de echilibru egal cu +1 iar nodul nou a fost inserat în subarborele drept, are loc reechilibrarea acestuia prin apelul metodei *ReechilibrareSubarboreDrept*;
- dacă nodul curent are gradul de echilibru egal cu 0 sau -1 iar nodul nou a fost inserat în subarborele drept, atunci noul grad de echilibru al elementului curent este +1, respectiv 0; asemenea situaţiei anterioare, variabila *echilibruNou* condiţionează prin valorile ei continuarea sau încetarea procesului de căutare;
- metoda *ReechilibrareSubarboreStang* ia în considerare toate situaţiile posibile de dezechilibru către stânga şi în funcţie de tipul acesteia reechilibrează subarborele cu rădăcina în nodul pivot prin rotaţie simplă la dreapta, metoda *RotatieSimplaDreapta*, sau prin rotaţie dublă la dreapta, metoda *RotatieDublaDreapta*; se observă caracterul general al acestei metode de reechilibrare ce este utilizată şi la ştergerea unui nod, procesul fiind descris în continuare;

```
void AVLArbore::ReechilibrareSubarboreStang(AVLNod * &pivot, int
&echilibruNou){
      AVLNod * FiuStanga = pivot->st;

      if(FiuStanga->Echilibru == -1){
            RotatieSimplaDreapta(pivot);
            echilibruNou = 0;
      }
      else
            if(FiuStanga->Echilibru == 1){
                  RotatieDublaDreapta(pivot);
                  echilibruNou = 0;
            }
            else
                  //situatie specifica operatiei de stergere
                  if(FiuStanga->Echilibru == 0){
                        RotatieSimplaDreaptaStergere(pivot);
                        echilibruNou = 0;
                  }
}
```

- metoda *ReechilibrareSubarboreDrept* analizează cazurile de dezechilibru la dreapta, reechilibrând pivotul prin una din cele două tehnici de rotaţie la stânga;

```
void AVLArbore::ReechilibrareSubarboreDrept(AVLNod * &pivot, int
&echilibruNou){
	AVLNod * FiuDreapta = pivot->dr;

	if(FiuDreapta->Echilibru == 1){
		RotatieSimplaStanga(pivot);
		echilibruNou = 0;
	}
	else
		if(FiuDreapta->Echilibru == -1){
			RotatieDublaStanga(pivot);
			echilibruNou = 0;
		}
		else
			//situatie specifica operatiei de stergere
			if(FiuDreapta->Echilibru == 0){
				RotatieSimplaStangaStergere(pivot);
				echilibruNou = 0;
			}
}
```

Metoda *AVLInsert* este o metodă internă clasei. Aceasta este epelată din programul principal de către metoda publică *Insert* ce primeşte ca parametru valoarea de inserat în arborele AVL.

```
void AVLArbore::Insert(const int info)
{
	AVLNod* RadacinaArbore = this->radacina;
	AVLNod* NodNou = new AVLNod(0,info,NULL,NULL);

	int EchilibruNou = 0;

	AVLInsert(RadacinaArbore,NodNou,EchilibruNou);

	this->radacina = RadacinaArbore;
}
```

Spre deosebire de operaţie de inserare, care necesită maxim o singură rotaţie pentru remedierea dezechilibrului, în cazul procedurii de ştergere a unui nod sunt necesare mai multe operaţii de rotaţie pentru a reechilibra arborele AVL şi pentru a conserva caracteristicile acestuia. Etapele parcurse se concentrează pe analiza tuturor nodurilor direct influenţate

- se identifică nodul de şters pe baza caracteristicilor arborilor binari de căutare;
- pe măsură ce se parcurge arborele, nodurile vizitate sunt salvate într-o structură de tip stivă; această operaţie suplimentară este necesară pentru a permite reconstruirea în sens invers a drumului parcurs de la rădăcina arborelui;

```
struct NodeStack
{
	AVLNod* Nod;
	NodeStack *next;
};

class AVLNodeStack
{
private:
	NodeStack * VarfStiva;
public:
	AVLNodeStack()
	{
		VarfStiva=NULL;
	}

	void PUSH(AVLNod* &NodNou){
		NodeStack *elementNou= new NodeStack;
		elementNou->Nod = NodNou;
		if(this->VarfStiva==NULL){
			this->VarfStiva = elementNou;
			elementNou->next=NULL;
		}
		else
		{
			elementNou->next = this->VarfStiva;
			this->VarfStiva = elementNou;
		}
	}

	AVLNod* POP(){
		if(this->VarfStiva==NULL)
			return NULL;
		else
		{
			NodeStack *elementSters = this->VarfStiva;
			AVLNod* NodAuxiliar = this->VarfStiva->Nod;
			this->VarfStiva = this->VarfStiva->next;
			delete elementSters;
			return NodAuxiliar;
		}

	}
	void AfiseazaStiva()
	{
		NodeStack *temp = this->VarfStiva;
		while(temp!=NULL)
		{
			printf("\n Nod in stiva este %d",temp->Nod->Info);
			temp=temp->next;
		}
	}
};
```

- nodul se şterge în mod asemănător cu operaţia asociată arborilor binari de căutare; dacă nodul este frunză se şterge efectiv; dacă nodul are un singur fiu, acesta îl înlocuieşte în structură; dacă nodul are cei doi fii, este înlocuit de nodul cu valoarea cea mai mare din subarborele drept, metoda *Stergere*;

```
int AVLArbore::Stergere(AVLNod*& SubarboreDrept, AVLNodeStack &stiva )
{
      if(SubarboreDrept->st)
      {
            stiva.PUSH(SubarboreDrept);
            return AVLArbore::Stergere(SubarboreDrept->st,stiva);
      }
      else
      {
            AVLNod * NodSters= SubarboreDrept;
            int valoare = SubarboreDrept->Info;
            SubarboreDrept = SubarboreDrept->dr;
            delete NodSters;
            return valoare;
      }
}
```

- sunt analizate toate nodurile parcurse şi sunt reechilibrate situaţiile de dezechilibru luând în calcul ipotezele de aplicare a celor patru tipuri de rotaţii; operaţia de ştergere se încheie în momentul în care sunt verificate toate locaţiile de dezechilibru posibil; pentru abordarea aleasă ca soluţie în acest capitol, operaţia se consideră încheiată în momentul în care stiva este golită;
- din analiza metodei *Stergere*, se observă că în etapa de identificare a nodului cu valoarea ce mai mare din subarborele drept, ce va lua locul nodului de şters, este completată de salvarea în stiva utilizată a nodurilor vizitate; necesitatea acestei operaţii suplimentare este dată de faptul că ştergerea unui nod poate conduce la dezechilibrarea nodurilor superioare aflate pe drumul de la rădăcină la poziţia lui; de asemenea, reechilibrarea unui nod părinte poate conduce la generarea unei alte situaţii de dezechilibru; pentru a exemplifica această situaţie, se ia în considerare arborele AVL din figura 14.19 în care se şterge nodul cu valoarea 50;

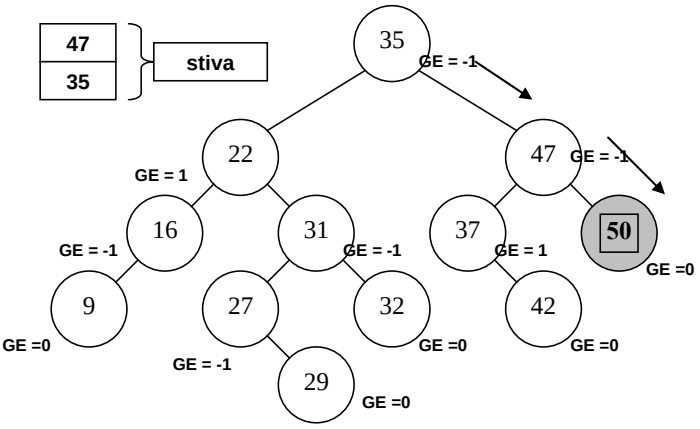

Figura 14.19 *Structură arborescenta de tip AVL*

Prin ştergerea nodului cu valoarea 50, se obţine stiva cu valorile 47 şi 35. Din analiza acestor noduri, se observă că arborele AVL, descris în figura 14.20, devine dezechilibrat în nodul cu valoarea 47.

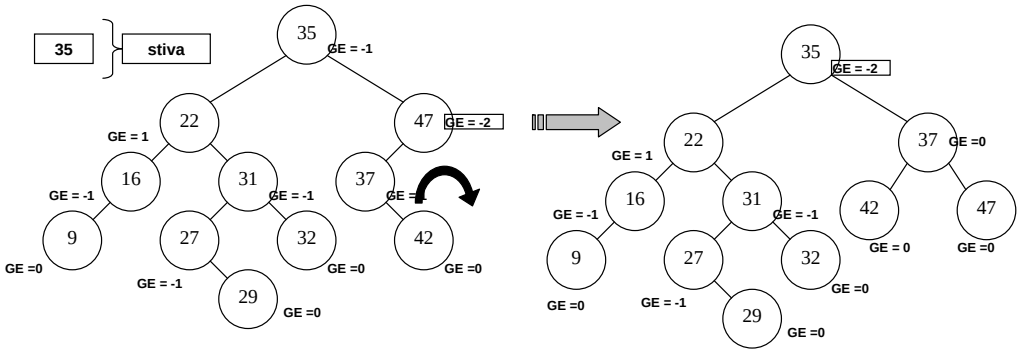

Figura 14.20 *Structură arborescenta de tip AVL dezechilibrată*

Prin reechilibrare, aplicând o rotaţie simplă la dreapta în pivot, se obţine o nouă situaţie de dezechilibru în următoare valoare din stivă, 35, figura 14.20. Printr-o rotaţie simplă la dreapta în nodul cu valoarea 35 considerat pivot, arborele AVL este reechilibrat. Deoarece stiva a fost golită, operaţie de ştergere se consideră încheiată, figura 14.21.

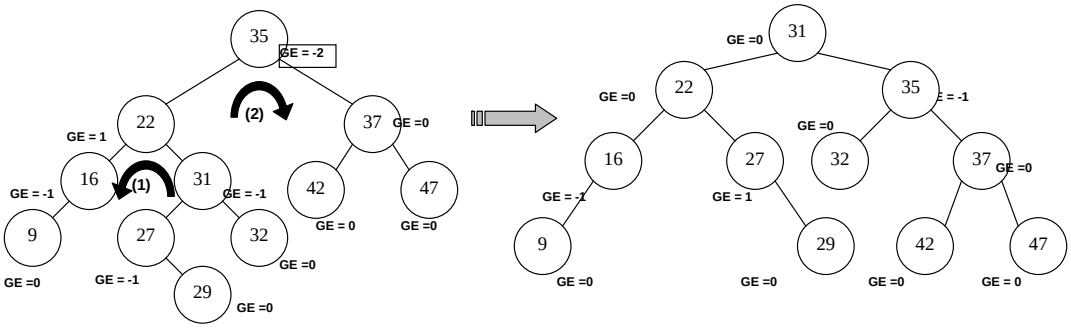

Figura 14.21 *Structură arborescenta de tip AVL*

Există cazuri în care prin ştergerea unui nod, se ajunge la situaţii de dezechilibru diferite de ipotezele analizate la operaţia de inserare. Luând în considerare arborele AVL din figura 14.22, se propune ştergerea nodului cu valoarea 16.

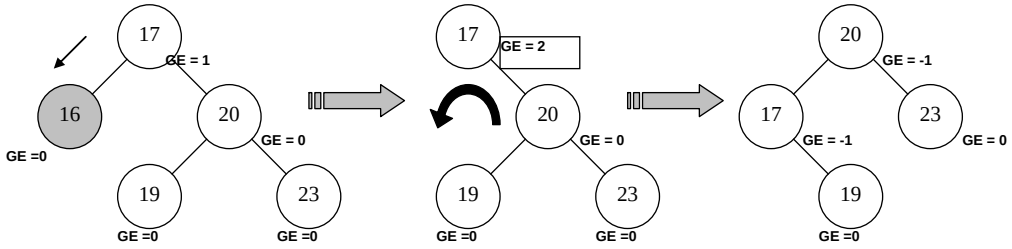

Figura 14.22 *Ştergere din structură arborescenta de tip AVL*

Situaţia diferă de cele întâlnite la inserare prin faptul că în acest dezechilibru pivotul are un grad de echilibru egal cu +2, iar nodul fiu de pe direcţia dezechilibrului are un echilibru egal cu 0. Soluţia acestui dezechilibru este dat de o rotaţie simplă în pivot la stânga.

Din acest motiv, metodele clasei *AVLArbore*, destinate analizei şi implementării tipului de rotaţie potrivit, sunt modificate în cazul operaţiei de ştergere. Cele două metode descrise anterior , *ReechilibrareSubarboreDrept* şi *ReechilibrareSubarboreStang* analizează şi situaţiile particulare în care nodul de pe direcţia dezechilibrului are gradul de echilibru egal cu zero, caz în care sunt apelate metodele *RotatieSimplaDreaptaStergere* şi *RotatieSimplaStangaStergere*.

```
void AVLArbore::RotatieSimplaDreaptaStergere(AVLNod * &pivot)
{
      AVLNod *FiuStanga = pivot->st;
      pivot->st = FiuStanga->dr;
      FiuStanga->dr = pivot;

      pivot->Echilibru += 1;
      FiuStanga->Echilibru += 1;

      pivot = FiuStanga;
}

void AVLArbore::RotatieSimplaStangaStergere(AVLNod * &pivot)
{
      AVLNod *FiuDreapta = pivot->dr;
      pivot->dr = FiuDreapta->st;
      FiuDreapta->st = pivot;

      pivot->Echilibru -=1;
      FiuDreapta->Echilibru -= 1;

      pivot = FiuDreapta;
}
```

Pentru a implementa operaţia de ştergere, se defineşte în clasa *AVLArbore* metoda *Delete*.

```
void AVLArbore::Delete(const int Info)
{
      int valTemp;

//definesc stiva nodurilor parcurse
      AVLNodeStack stiva;

//se sterge nodul
AVLDelete(this->radacina,Info,stiva);

//se analizeaza nodurile parcurse
AVLNod *temp = stiva.POP();
while(temp!=NULL){
      temp->Echilibru = this->CalculeazaEchilibru(temp);
      if(temp->Echilibru==2){
         AVLNod *parinte = stiva.POP();
         if(parinte!=NULL){
            if(parinte->dr==temp)
            this->ReechilibrareSubarboreDrept(parinte->dr,valTemp);
            else
            this->ReechilibrareSubarboreDrept(parinte->st,valTemp);
            parinte->Echilibru=this->CalculeazaEchilibru(parinte);
                  }
            }
```

```
      else
          if(temp->Echilibru==-2){
            AVLNod *parinte = stiva.POP();
            if(parinte!=NULL){
            if(parinte->dr==temp)
            this->ReechilibrareSubarboreStang(parinte->dr,valTemp);
            else
            this->ReechilibrareSubarboreStang(parinte->st,valTemp);
            parinte->Echilibru=this->CalculeazaEchilibru(parinte);
                        }
                  }
            temp=stiva.POP();
      }
}
```

Această metodă se bazează pe apelul metodei *AVLDelete* pentru a realiza ştergerea efectivă a nodului dorit, secvenţa de cod asociată fiind concentrată pe analiza nodurilor din stiva. Pentru fiecare din acestea, se recalculează gradul de echilibru prin intermediul metodei *CalculeazaEchilibru*.

```
int AVLArbore::CalculeazaEchilibru(AVLNod *& radacina)
{
      return inaltime(radacina->dr) - inaltime(radacina->st);
}
```

Metoda *AVLDelete* completează metoda întâlnită la ştergerea nodurilor din arbori binari de căutare prin gestiunea unei stive în care sunt inserate toate valorile întâlnite.

```
void AVLArbore::AVLDelete(AVLNod* &arbore,const int Info,AVLNodeStack
&stiva){
      AVLNod *NodAuxiliar;
      if(arbore){
            if(Info == arbore->Info){
                  NodAuxiliar = arbore;
                  if(!NodAuxiliar->dr){
                        arbore = NodAuxiliar->st;
                        delete NodAuxiliar;
                  }
                  else
                        if(!NodAuxiliar->st){
                              arbore = NodAuxiliar->dr;
                              delete NodAuxiliar;
                        }
                        else{
                        stiva.PUSH(arbore);
            arbore->Info = AVLArbore::Stergere(arbore->dr,stiva);
                        }
            }
            else
                  if(Info < arbore->Info){
                        stiva.PUSH(arbore);
                        AVLDelete(arbore->st,Info,stiva);
                  }
                  else{
                        stiva.PUSH(arbore);
                        AVLDelete(arbore->dr,Info,stiva);
                  }
```

```
        } }
```

În ciuda efortului asociat implementării şi executării secvenţelor de rotire ale structurii, arborii AVL oferă un ridicat nivel de eficienţă în ceea ce priveşte procesul de căutare în arbori binari de căutare. Structura arborescentă echilibrată

14.4 Caracteristici ale arborilor Roşu & Negru

Arborii Roşu & Negru reprezintă o altă tipologie de arbori binari de căutare echilibraţi, fiind prima dată definiţi de Rudolf Bayer în 1972 sub forma de arbori simetrici. Asemenea arborilor AVL, această structură este caracterizată de o complexitate a operaţiei de căutare egală cu O(log n), *n* fiind numărul de noduri din arbore, datorită modului în care nodurile sunt plasate în mod simetric în subarborii stângi sau drepţi.

Spre deosebire de arborele AVL, în care principala caracteristică se determină pe baza gradului de echilibru al fiecărui nod, în structurile arborescente de tip Rosu & Negru, factorul cel mai important este dat de culoarea fiecărui nod:

- fiecare nod are una dintre cele două culori, roşu sau negru;
- nodul rădăcină este întotdeauna negru;
- ambele noduri fiu ale unui nod părinte roşu sunt negre; un nod roşu nu poate avea ca părinte decât un nod negru;
- toate drumurile de la rădăcină la oricare din nodurile frunză conţin acelaşi număr de noduri negre.

Analizând aceste caracteristici sunt derivate proprietăţi noi care să fie utilizate în implementarea algoritmilor sau care să evidenţieze eficienţa acestui tip de structură faţă de un arbore binar de căutare:

- în arborele Roşu & Negru nu există pe un drum două noduri adiacente de culoare roşie deoarece orice nod roşu are ambii fii de culoare neagră;
- dacă se consideră că cel mai scurt drum din arbore are numai noduri negre în număr de *k*, atunci cel mai lung drum din arbore are maxim dublu noduri, 2 * *k*; ipoteza este demonstrată pe baza faptului că toate drumurile din acest tip de structură au acelaşi număr de noduri negre, fapt care conduce la concluzia că drumul cel mai lung poate fi format doar din perechi de noduri adiacente de culori opuse, figura 14.23.

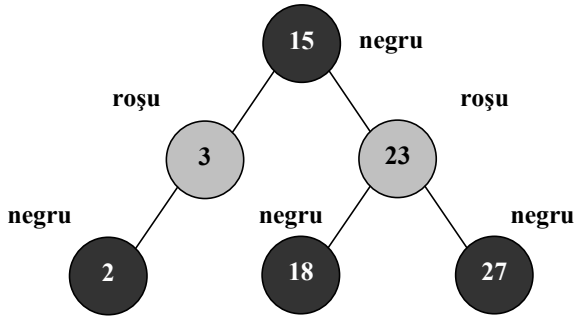

Figura 14.23 *Structură arborescenta de tip Roşu & Negru*

Pentru a facilita implementarea operaţiilor cu structuri arborescente de tip Rosu & Negru se propune o structură asociată nodului, clasa *RNNode*

```
class RNNod
{
      int Info;
      bool Culoare;
      RNNod *st;
      RNNod *dr;
      RNNod *parinte;

};
```

Elementele de tip *RNNode* includ pe lângă atributele întâlnite la toate structurile arborescente binare:

- informaţia utilă;
- cele două legături către nodurile fiu din stânga, respectiv, dreapta;

şi informaţia ce descrie culoare nodului, precum şi o legătură suplimentară către nodul părinte. Această abordare contribuie la implementarea mult mai facilă a operaţiilor de inserare sau ştergere, minimizând în timp real efortul de a identifica nodul părinte al nodului curent.

14.5 Operaţii pe arbori Roşu & Negru

Operaţiile pe arborii Roşu şi Negru descrise, inserare şi ştergere, sunt realizate asemenea arborilor binari de căutare deoarece acest tip de arbore este o structură binară particulară. Asigurarea caracteristicilor specifice acestui tip de structură arborescentă este realizată printr-o serie de operaţii auxiliare şi complementare procesului de inserare sau ştergere ce constau în rotiri sau modificări de culoare.

Pentru a descrie metodele specifice operaţiilor se defineşte ca nod bunic al nodului nou creat, nodul ce se găseşte pe al doilea nivel superior faţă de nodul analizat, figura 14.24. Se defineşte ca nod unchi al nodului analizat, al doilea nod fiu al nodului bunic.

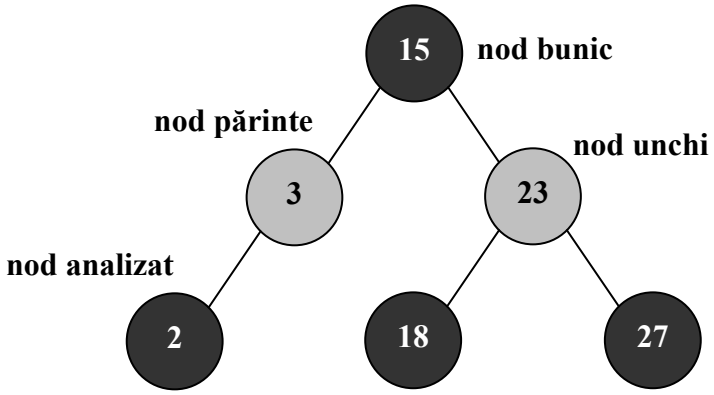

Figura 14.24 *Relaţii între noduri Roşu & Negru*

Pentru a determina poziţia acestor noduri particulare este utilizat atributul *RNNod *parinte* al fiecărui obiect de tip *RNNod*. De exemplu nodul bunic al nodului curent este determinat prin expresia NodCurent->parinte->parinte, iar nodul unchi este dat de *NodCurent->parinte->parinte->st* sau *NodCurent->parinte->parinte->st* în funcţie de poziţia acestuia relativă la nodul părinte al nodului curent.

Operaţia de inserare este analizată prin prisma cazurilor particulare. Acestea sunt definite de contextul în care se găseşte nodul nou creat şi de situaţiile de dezechilibru apărute.

Fiecare nod nou creat şi inserat în structura arborescentă de tip Roşu şi Negru are culoarea iniţială roşie. Astfel se încearcă evitarea situaţie în care este încălcată proprietatea că toate drumurile din arbore au acelaşi număr de noduri negre.

Se consideră arborele Roşu & Negru vid în care se inserează valoarea 43. prin inserare se obţine structura arborescentă din figura 14.25 ce trebuie reechilibrată prin modificarea culorii nodului rădăcină în negru. Astfel nodul rădăcină este negru.

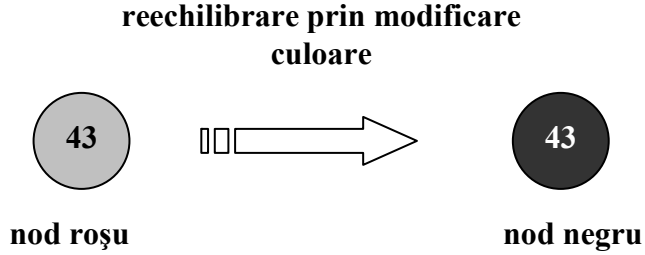

Figura 14.25 *Arbore Roşu & Negru cu un singur nod*

În arborele analizat se inserează valorile 25 şi 78. Nodurile nou create au culoare roşie, figura 14.26 şi nu este încălcată nici o proprietate a arborilor.

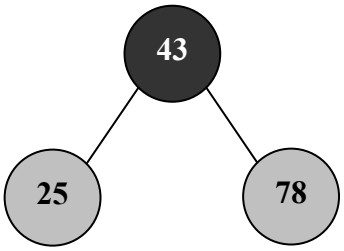

Figura 14.26 *Arbore Roşu & Negru echilibrat*

Se ia în considerare situaţia în care se inserează nodul cu valoarea 14. Nodul nou creat este roşu, fapt care încalcă proprietăţile arborilor Roşu & Negru, deoarece un nod roşu are întotdeauna un nod negru ca părinte. Dacă nodul este recolorat în negru atunci toate drumurile din arbore nu vor avea acelaşi număr de noduri negre. Situaţia este analizată prin prisma nodului părinte şi a nodului unchi. Dacă aceste două noduri sunt roşii atunci ele îşi schimbă culoarea în negru, iar nodul bunic, părintele celor două noduri, devine negru, figura 14.27. Dacă prin modificarea culorii nodului bunic, arborele este dezechilibrat atunci situaţia este remediată în manieră recursivă până se ajunge la rădăcina arborelui.

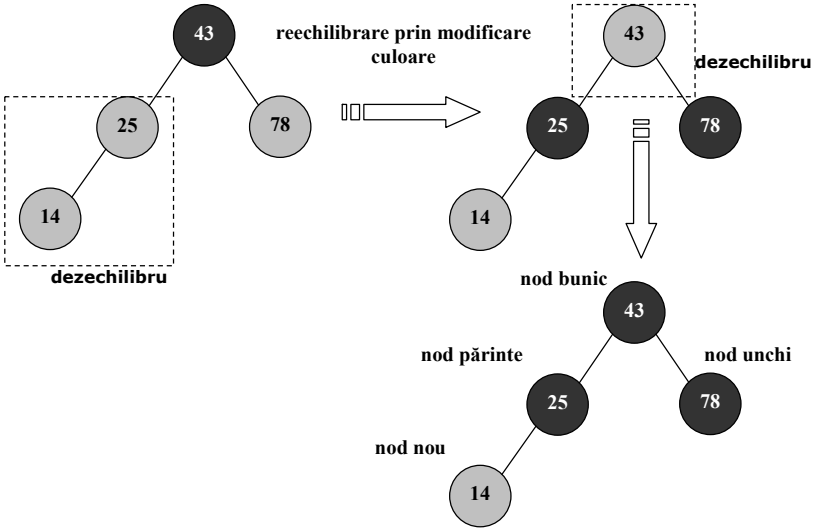

Figura 14.27 *Arbore Roşu & Negru reechilibrat*

În cazul arborelui din figura 14.27, nodul rădăcină devine negru la pasul următor, structura fiind reechilibrată.

Se consideră exemplul dat de inserarea valorii 17. Nodul nou are culoare roşie, fapt ce încalcă proprietatea acestui tip de arbore, toate nodurile fiu ale unui nod roşu sunt negre, figura 14.28.

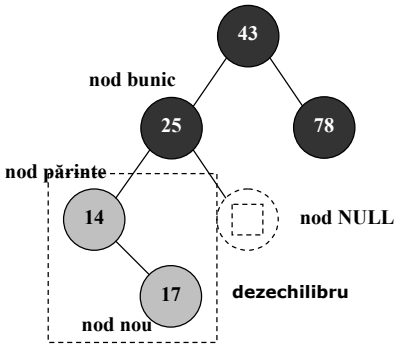

Figura 14.28 *Arbore Roşu & Negru dezechilibrat*

Reechilibrarea arborelui în această situaţie este realizată printr-o dublă rotaţie. Într-o primă fază, se realizează o simplă rotaţie la stânga în nodul părinte. Ipoteza de lucru este definită de faptul că:

- nodul părinte are culoare roşie, dar nodul unchi este fie negru, fie nod NULL;
- nodul nou creat este fiu dreapta pentru nodul părinte, care la rândul său este nod fiu stânga pentru nodul bunic.

Rotaţia este realizată asemenea arborilor AVL considerând pivot, nodul părinte. După această prima rotaţie se obţine arborele din figura 14.29.

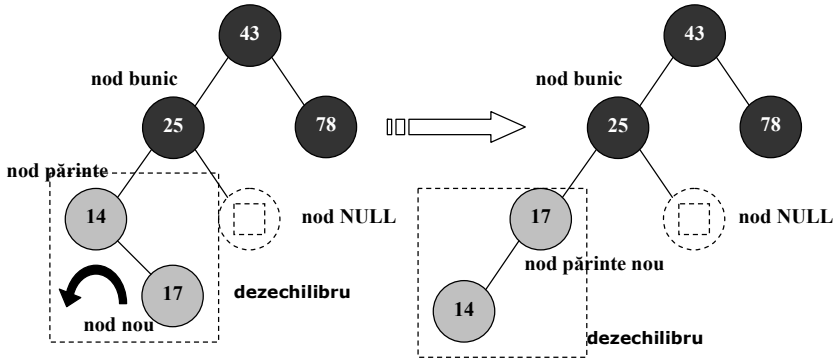

Figura 14.29 *Arbore Roşu & Negru dezechilibrat*

Structura arborescentă este dezechilibrată prin prisma aceleiaşi proprietăţi încălcate. Din acest motiv este necesară o a doua operaţie de rotaţie ce are ca pivot, nodul bunic. De data aceasta, rotaţie se realizează la dreapta, având sens opus cu direcţia nodului fiu faţă de nodul părinte. Ipoteza de lucru este definită de condiţiile:

- nodul părinte are culoare roşie, dar nodul unchi este fie negru, fie nod NULL;
- noul nod părinte este fiu stânga pentru nodul bunic şi nodul nou inserat este fiu stânga pentru acesta.

Rotaţia descrisă în figura 14.30 este însoţită şi de o recolorare a nodurilor, astfel încât nodul bunic devine roşu şi noul nod părinte devine negru.

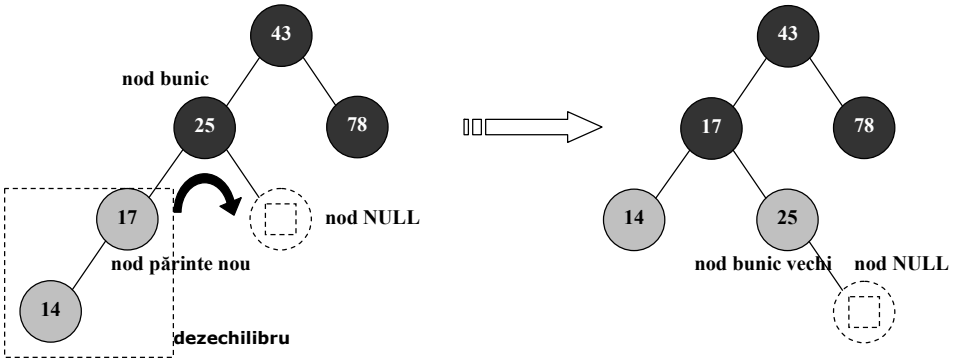

Figura 14.30 *Arbore Roşu & Negru reechilibrat*

În cazul în care, se insera valoarea 10 atunci erau atinse condiţiile implementării celei de a doua operaţie de rotaţie fiind evitată prima rotaţie la stânga.

Dacă nodul nou are ca părinte un nod de culoare roşie şi acesta este fiul din dreapta al nodului bunic, atunci situaţia reprezintă imaginea în oglindă a cazului anterior.

De exemplu, se inserează valorile 89 şi 95 în această ordine. Figura 14.31 descrie păşii parcurşi pentru reechilibrarea arborelui.

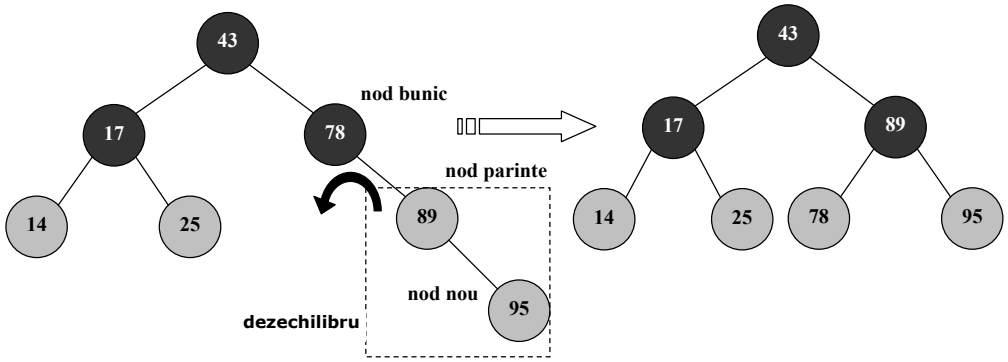

Figura 14.31 *Reechilibrare arbore Roşu & Negru*

Situaţia descrisă anterior este condiţionată de atingerea următoarelor condiţii de lucru:

- nodul părinte are culoare roşie, dar nodul unchi este fie negru, fie nod NULL;
- nodul părinte este fiu dreapta pentru nodul bunic şi nodul nou inserat este fiu dreapta pentru acesta.

În cazul în care ultima condiţie nu este îndeplinită, noul nod fiind fiu stânga, situaţia este ajustată prin operaţia de rotire la dreapta în nodul părinte.

Operaţia de ştergere în arbori Roşu şi Negru completează procesul întâlnit la arborii binari de căutare prin operaţii specifice de recolorare sau rotire a nodurilor astfel încât să fie păstrate caracteristicile acestei structuri arborescente.

În cazul în care nodul de şters are două noduri fiu atunci acesta este înlocuit de nodul cu valoarea cea mai mare din subarborele stâng sau de nodul cu valoarea cea mai mică din subarborele drept. Copierea de valoarea este însoţită de păstrarea culorii nodului şters astfel încât să nu fie afectat arborele. Oricare variantă se alege, nodul care va înlocui nodul de şters este la rândul său eliminat din structura arborescentă. Acesta este fie nod frunză, fie are maxim un fiu. De exemplu, se şterge nodul cu valoarea 43 din arborele descris în figura 14.32.

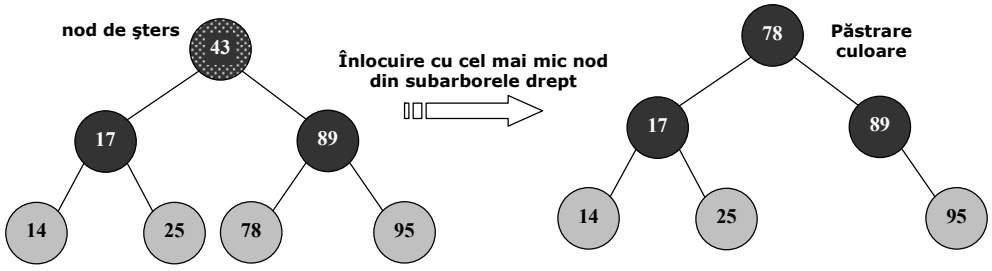

Figura 14.32 *Ştergere nod din arbore Roşu & Negru*

Prin prisma exemplului anterior, **problemele apărute la ştergerea unui nod dintr-un arbore Roşu şi Negru sunt concentrate în cazurile de ştergerea unui nod care are maxim un fiu**.

Dacă nodul de şters este de culoare roşie, figura 14.30, atunci nodul său fiu este de culoare neagră, aceasta fiind o caracteristică a arborilor Roşu şi Negru. Ştergerea nodului implică în această situaţie înlocuirea sa cu nodul

fiu. Arborele este în continuare Roşu şi Negru deoarece ştergerea unui nod roşu nu are implicaţii asupra numărului de noduri negre de pe fiecare drum.

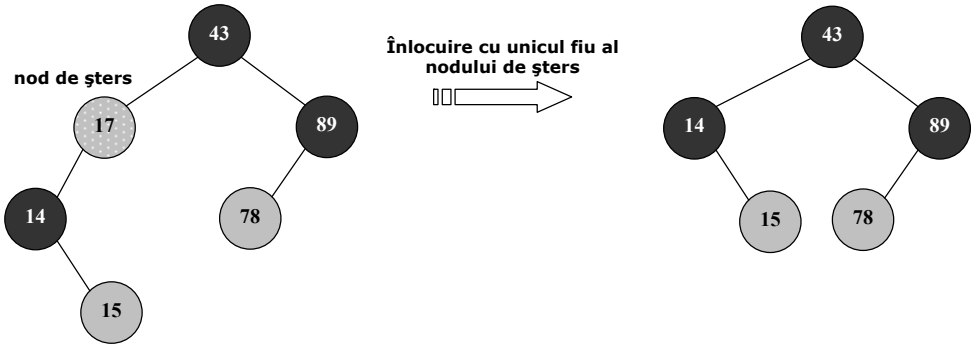

Figura 14.33 *Cazul 1 de ştergere nod din arbore Roşu & Negru*

Dacă nodul şters este de culoare neagră, iar fiul său este de culoare roşie, figura 14.34, atunci arborele devine dezechilibrat pe drumul care trece prin această zonă deoarece numărul de noduri negre este mai mic cu unul. Reechilibrarea structurii arborescente se face în această situaţie prin recolorarea în negru a nodului fiu. Astfel este refăcut numărul de noduri negre.

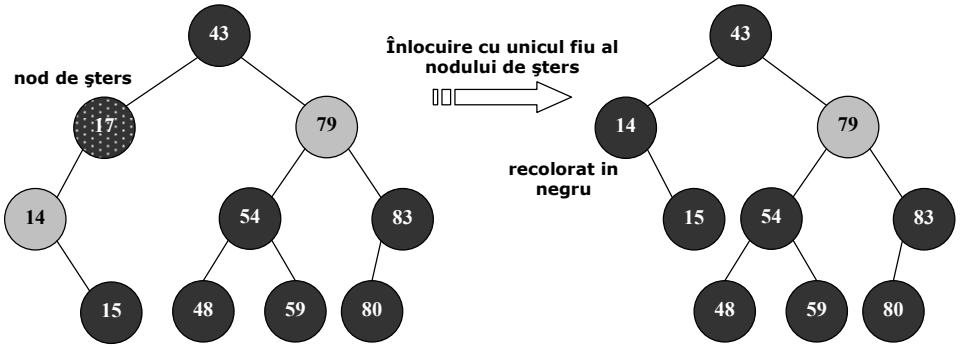

Figura 14.34 *Cazul 2 de ştergere nod din arbore Roşu & Negru*

Situaţiile complexe apărute la ştergerea unui nod dintr-un arbore de tip Roşu şi Negru sunt apar în cazul în care nodul de şters şi fiul său sunt de culoare neagră. Prin eliminarea nodului, arborele devine dezechilibrat deoarece o parte din drumuri conţin cu un nod negru mai puţin. Spre deosebire de cazurile prezentate anterior, nu mai este posibilă refacerea numărului de noduri negre prin recolorarea fiului deoarece acesta are deja culoarea neagră. Reechilibrarea arborelui este realizată printr-un număr fix de operaţii de rotire sau recolorare.

Pentru a descrie aceste cazuri particulare de dezechilibru şi soluţiile asociate, se fac o serie de notaţii care să ajute înţelegerea operaţiilor, figura 14.35. Se notează cu:

- P, nodul părinte al nodului de şters;
- F, nodul fiu al nodului de şters;
- B, nodul bunic al nodului de şters; acest nod este nodul părinte al nodului P;
- U, nodul unchi al nodului de şters; acest nod este reprezentat de al doilea fiu al nodului B;

- N_1 nodul nepot al nodului de şters; este reprezentat de fiul din stânga al nodului unchi;
- N_2 nodul nepot al nodului de şters; este reprezentat de fiul din dreapta al nodului unchi;

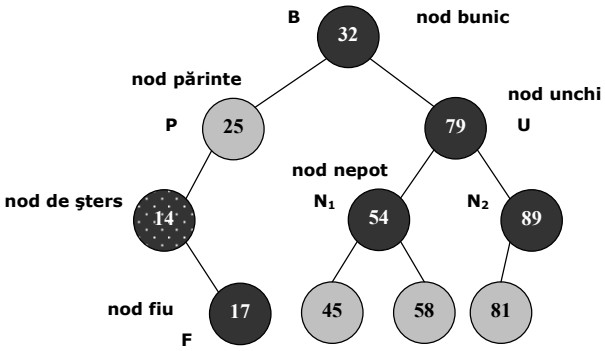

Figura 14.35 *Arbore Roşu & Negru*

Următorul caz analizat este dat de figura 14.36 în care nodul cu valoarea 25 este şters. Situaţia este descrisă de ipotezele:

- nodul de şters este negru;
- unicul fiu al nodului de şters este negru;
- nodul unchi al nodului de şters este negru;
- nodul părinte este negru;
- nodurile nepoţi sunt negre.

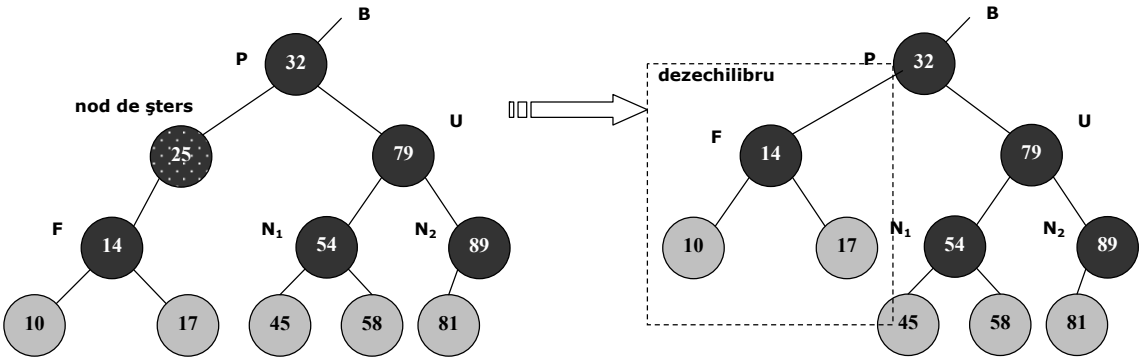

Figura 14.36 *Cazul 3 de ştergere nod din arbore Roşu & Negru*

Prin ştergerea nodului cu valoarea 25, arborele sau subarborele analizat ce are rădăcină pe nodul cu valoarea 32 este dezechilibrat la dreapta deoarece drumurile care pornesc din rădăcină şi continuă pe partea stângă au cu un nod negru mai puţin. Reechilibrarea arborelui se realizează prin modificarea culorii nodului unchi, valoarea 79, în roşu, figura 14.37. Astfel, este redus cu unu numărul de noduri negre din drumurile ce pornesc din rădăcina 32.

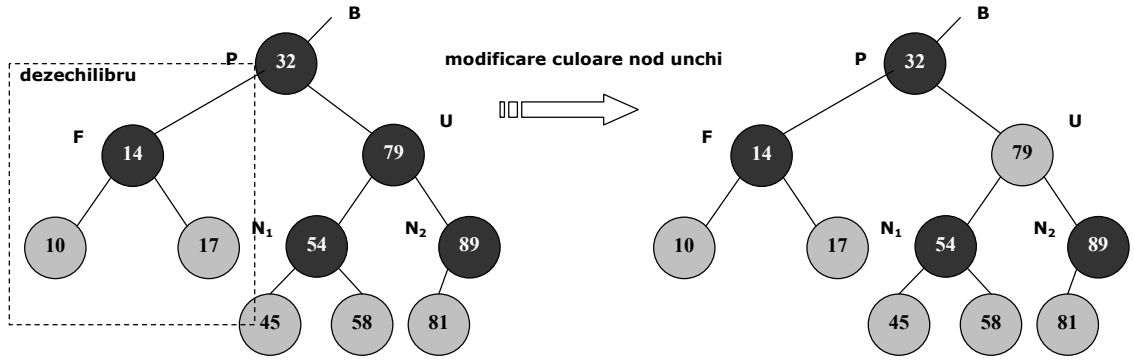

Figura 14.37 *Soluţie de reechilibrare caz 3 pentru arbore Roşu & Negru*

În cazul în care, nodul cu valoarea 32 reprezintă rădăcina unui subarbore, analiza se continuă în sus până când se atinge rădăcina arborelui sau până când arborele este reechilibrat pe baza unei soluţii din cele descrise.

Al patrulea caz de ştergere a unui nod dintr-un arbore Roşu şi Negru ia în considerare situaţia descrisă în figura 14.38:

- nodul de şters este negru;
- unicul fiu al nodului de şters este negru;
- nodul unchi al nodului de şters este negru;
- nodul părinte este roşu;
- nodurile nepoţi sunt negre.

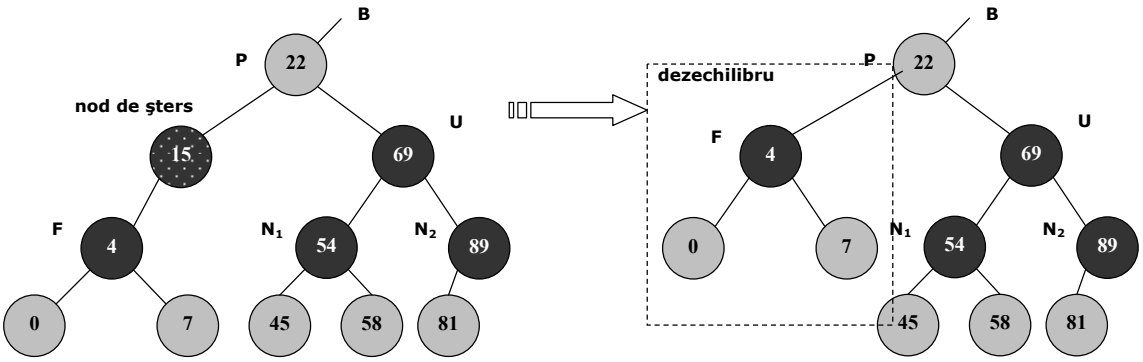

Figura 14.38 *Cazul 4 de ştergere nod din arbore Roşu & Negru*

Asemenea cazului anterior, arborele îşi pierde calitatea de a fi Roşu şi Negru în urma ştergerii deoarece nu toate drumurile de la rădăcină la nodurile frunză au acelaşi număr de noduri negre. Reechilibrarea este realizată prin interschimbarea culorilor nodului părinte şi nodului unchi, figura 14.39.

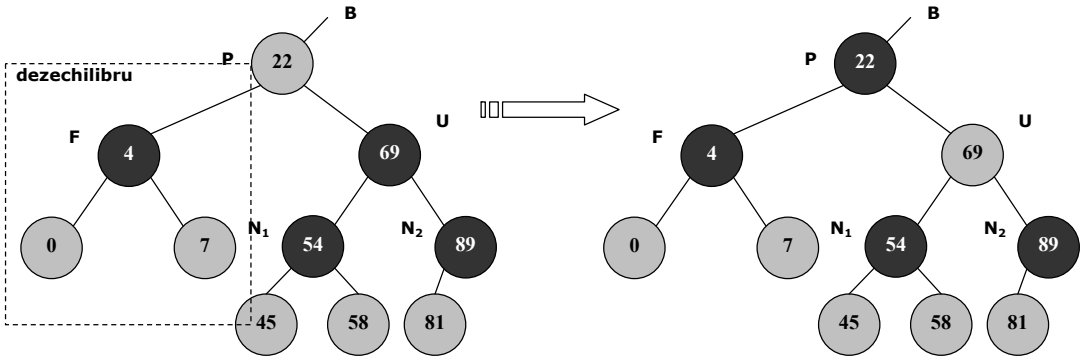

Figura 14.39 *Soluţie de reechilibrare caz 4 pentru arbore Roşu & Negru*

În situaţia în care arborele din figura 14.34 reprezintă un subarbore atunci soluţia de reechilibrare prezentată are doar efecte locale, deoarece lungimea măsurată în număr de noduri negre a tuturor drumurilor din acest subarbore este mai mică cu unu faţă de situaţia iniţială. Din acest motiv, reechilibrarea se continuă recursiv către rădăcina arborelui.

Cazul al cincilea de ştergere a unui nod ia în considerare ipotezele descrise în figura 14.40:

- nodul de şters este negru;
- unicul fiu al nodului de şters este negru;
- nodul unchi al nodului de şters este roşu;
- nodul părinte este negru;
- nodurile nepoţi sunt negre.

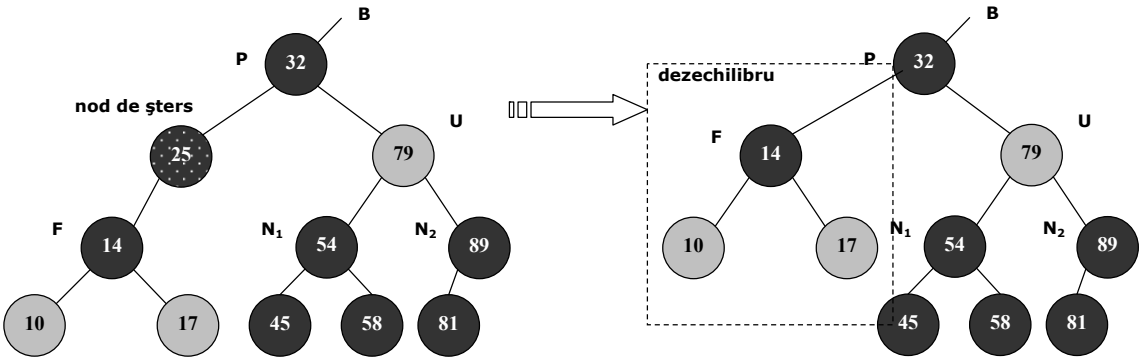

Figura 14.40 *Cazul 5 de ştergere nod din arbore Roşu & Negru*

Reechilibrarea arborelui pentru cazul 5 de dezechilibru se realizează prin interschimbarea culorilor nodului unchi şi nodului părinte, urmată de o rotaţie la stânga în nodul părinte, figura 14.41.

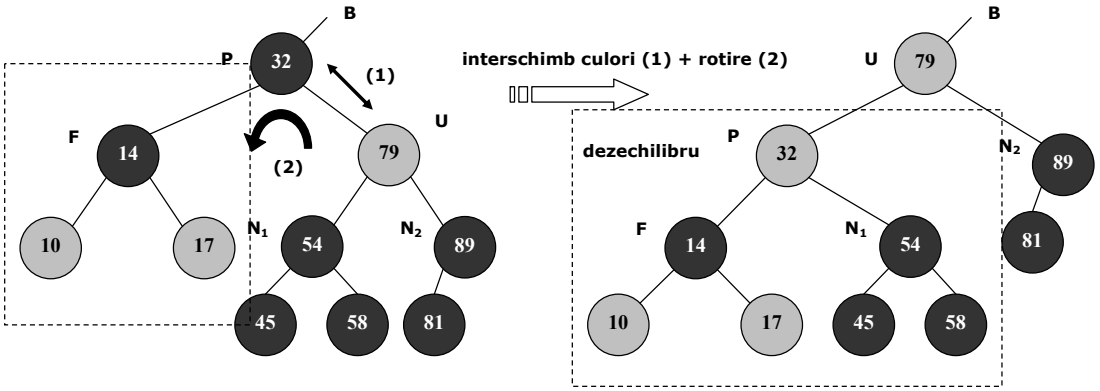

Figura 14.41 *Soluţie de reechilibrare caz 5 pentru arbore Roşu & Negru*

Analizând figura 14.41 se observă că soluţia cazului 5 nu conduce la reechilibrarea totală a arborelui. Zona de dezechilibru este modificată astfel încât să poată fi reechilibrată într-un număr finit de paşi. Această este analizată prin prisma cazului patru care a fost descris sau prin intermediul cazurilor şase şi şapte. De exemplu, arborele obţinut în figura 14.41 este reechilibrat, în figura 14.42 prin intermediul soluţie oferite în cazul patru, interschimbând culorile nodului cu valoare 32 şi nodului cu valoarea 54.

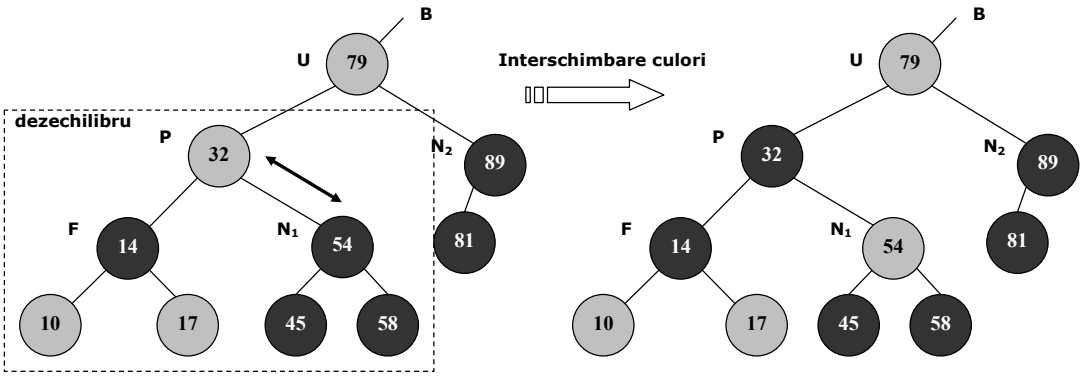

Figura 14.42 *Reechilibrare arbore Roşu & Negru din figura 14.41*

Următoarele două cazuri analizează culoarea nodurilor nepoţi luând în calcul situaţii derivate din cazul patru.

Cazul şase, descris în figura 14.43, este definit de următoarele ipoteze:

- nodul de şters este negru;
- unicul fiu al nodului de şters este negru;
- nodul unchi al nodului de şters este negru;
- nodul părinte este roşu sau negru;
- nodul nepot N_1 este roşu;
- nodul nepot N_2 este negru.

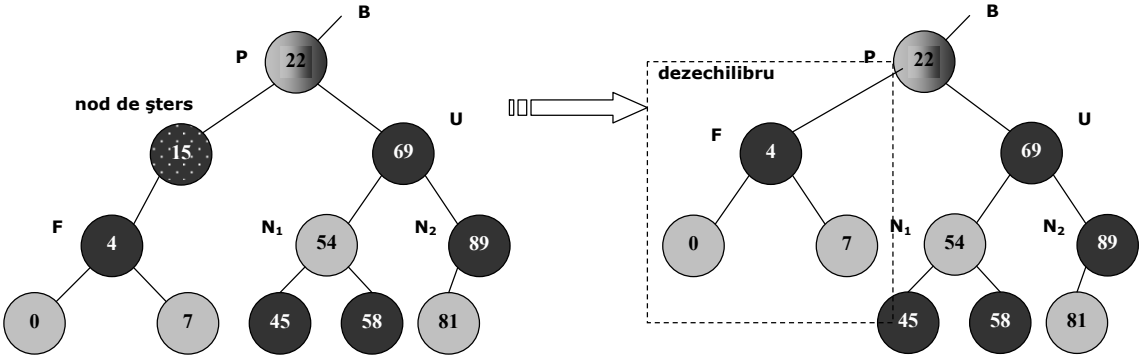

Figura 14.43 *Cazul 6 de ştergere nod din arbore Roşu & Negru*

Reechilibrarea arborelui din figura 14.43 este realizată prin:
- interschimbarea culorilor nodului părinte şi a nodului unchi;
- rotirea subarborelui cu rădăcină în nodul unchi la dreapta.

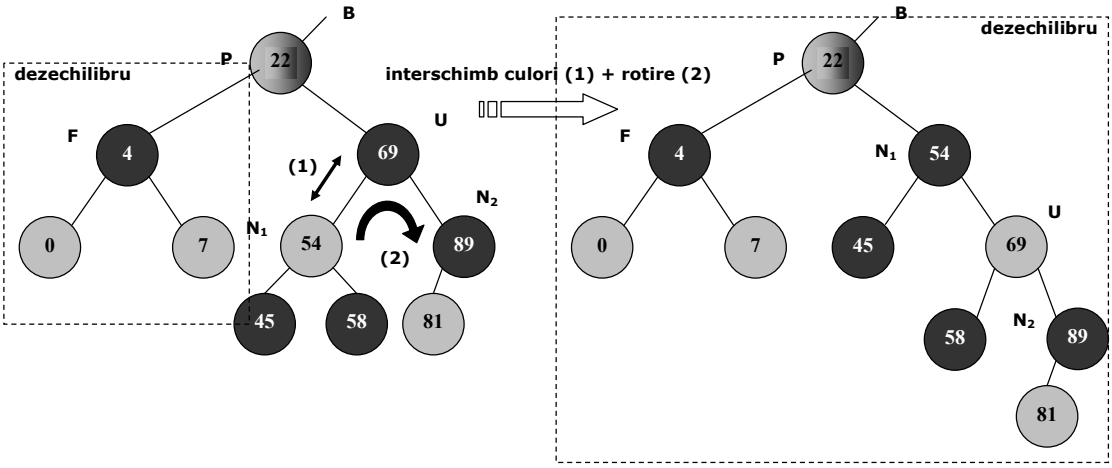

Figura 14.44 *Soluţie de reechilibrare caz 6 pentru arbore Roşu & Negru*

Rezultatul obţinut în urma operaţiei de schimbare a culorii şi de rotire nu conduce la reechilibrarea arborelui. Cu toate acestea, noua formă a subarborelui permite reechilibrarea la pasul următor, deoarece situaţia curentă descrie cazul şapte .

Cazul şase, descris în figura 14.45, este definit de următoarele ipoteze:
- nodul de şters este negru;
- unicul fiu al nodului de şters este negru;
- nodul unchi al nodului de şters este negru;
- nodul părinte este roşu sau negru;
- nodul nepot N_1 este roşu sau negru;
- nodul nepot N_2 este roşu.

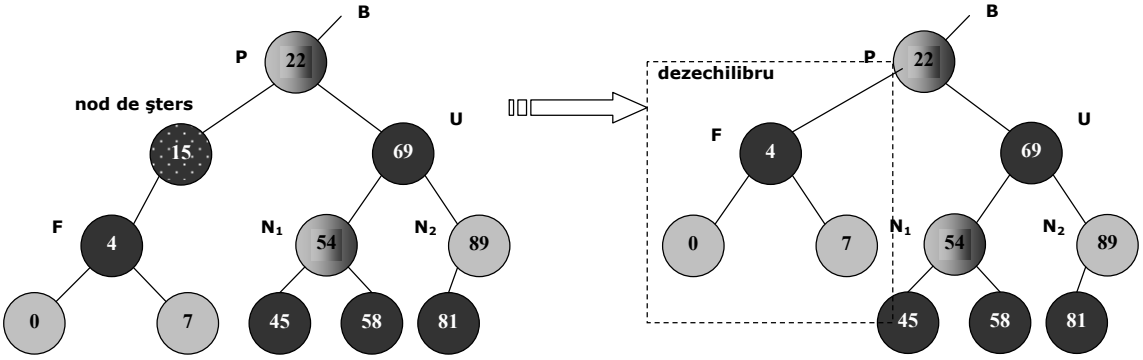

Figura 14.45 *Cazul 7 de ştergere nod din arbore Roşu & Negru*

Reechilibrarea situaţie descrise în figura 14.46 se realizează prin:
- interschimbare culoare nod părinte cu nodul unchi;
- rotire la stânga a arborelui în nodul părinte;
- modificare culoare nepot N_2 în negru.

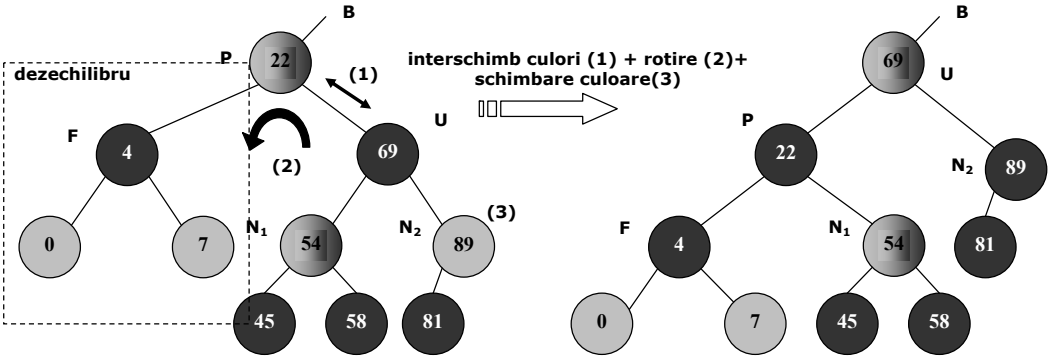

Figura 14.46 *Soluţie de reechilibrare caz 7 pentru arbore Roşu & Negru*

Figura 14.46 prezintă rezultatul obţinut în urma reechilibrării. Se observă eliminarea dezechilibrului din acest arbore sau subarbore.

Pentru exemplele analizate în acest capitol s-a considerat că nodul de şters se găseşte în partea stângă a nodului părinte. Pentru situaţia opusă, soluţiile descrise au aceleaşi efect dacă suferă mici modificări prin prisma noului reper de vizualizare a arborelui.

De asemenea, în exemplele prezentate reechilibrarea arborelui are un caracter local pentru a descrie tehnicile de reechilibrare, însă realizare unei aplicaţii trebuie să implementeze secvenţe care să parcurgă arborele de jos în sus, de la poziţia nodului de şters către rădăcina arborelui şi care să reechilibreze toată structura.