Алгоритми и структури на податоци

– белешки од предавања –

6. Пребарувачки стебла

6.1 Врска помеѓу бројот на јазли во бинарно стебло и неговата висина

Доколку бинарното стебло не е празно и е максимално пополнето, на ниво 0 тоа има еден јазел, на ниво 1, има два јазли, на ниво 2 има четири јазли или во општ случај, на ниво k има 2^{k-1} јазли. Доколку сите јазли се соберат добиваме дека за максимално пополнето бинарно стебло важи дека доколку има n јазли, тој број е еднаков на сумата

$$1 + 2^1 + 2^2 + \cdots + 2^{d-1} = 2^d - 1$$

каде d е длабочината на стеблото. Од тука може да заклучиме дека за секое стебло со n јазли важи

$$n \leq 2^d - 1$$

од каде пак следи дека минималната длабочина (висина) на бинарно стебло со n јазли (која се добива кога е тоа максимално пополнето) е

$$d_{\min} = \lceil \lg (n + 1) \rceil$$

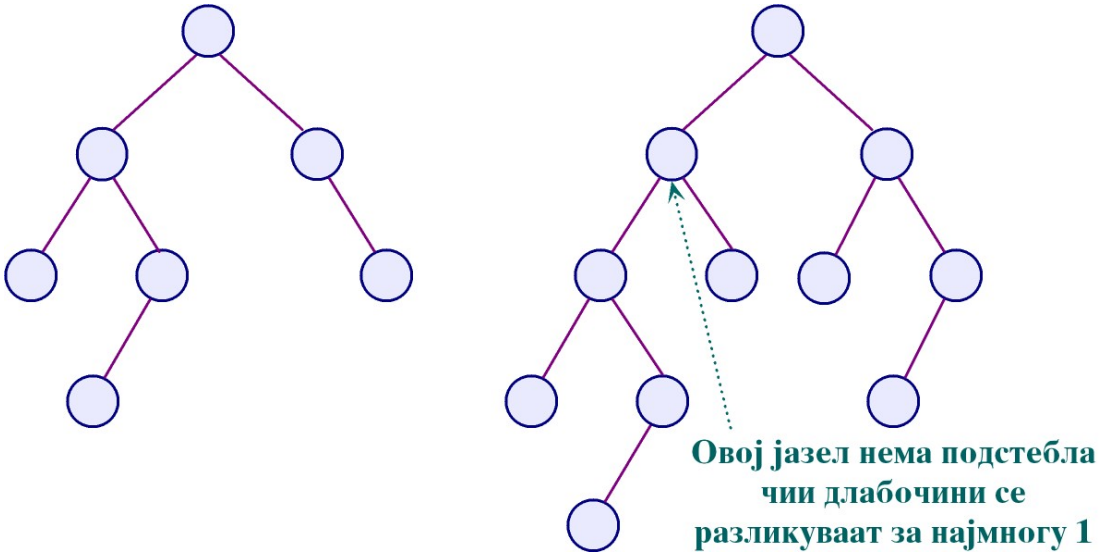
каде “ $\lceil A \rceil$ ” го означува најмалиот цел број еднаков или поголем од A.

Од тука следи дека $n = 20$, имаме $d_{\min} = \lceil \lg 21 \rceil = \lceil 4.39 \rceil = 5$; додека за n =1000 се добива $d_{\min} = \lceil \lg 1001 \rceil = 10$. Ова е значајно затоа што најголемата висина на стебло со 1000 јазли е 1000 (во случај кога е секогаш активна само една врска).

Доколку прептпоставиме дека стеблото како податочна структура се користи за складирање на некоја информација и под претпоставка дека бараната информација се наоѓа во листовите на стеблото (на пример при градење на индекс), од интерес ќе биде нивото на секој јазел во стеблото да биде еднаквао или приближно еднакво. Таквите бинарни стебла се нарекуваат балансирани бинарни стебла.

6.2 Балансирани бинарни стебла

Балансирано бинарно стебло е она бинарно стебло каде што за секое подстебло на произволен јазел важи дека висините не се разликуваат за повеќе од еден. На слика 6а, е претставено балансирано бинарно стебло. На слика 6б е прикажано небалансирано бинарно стебло кое е такво поради тоа што за посочениот јазел не важи дека висините на неговото лево и десно подстебло се еднакви или се разликуваат за еден. Во конкретниот случај висините на левото и десното подстебло на посочениот јазел се разликуваат за два.

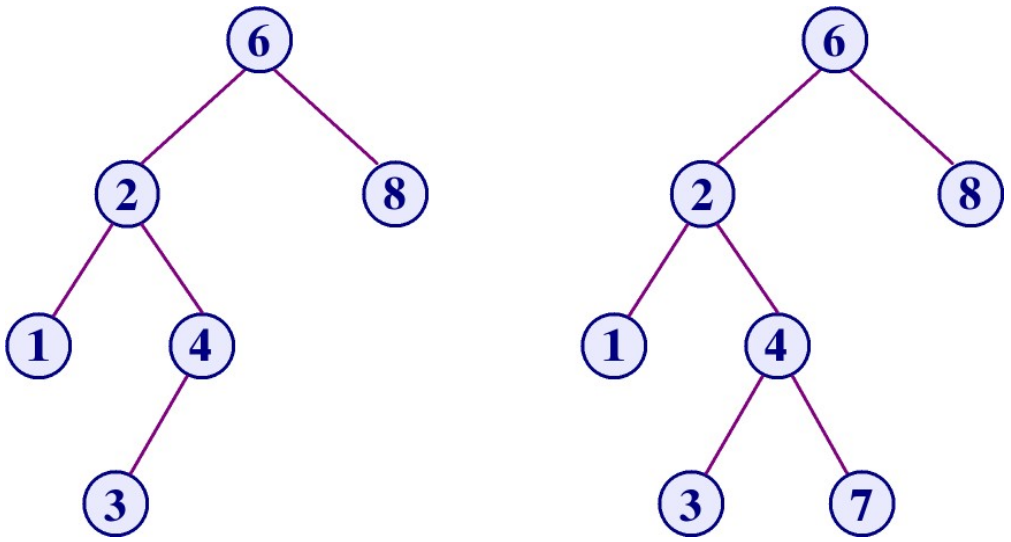


(а) (б)

слика 6.1 Балансирано (а) и небалансирано бинарно стебло (б)

6.3 Бинарни пребарувачки стебла

Една важна примена на стеблата е во областа на пребарувањето. Да претпоставиме дека на секој јазел од бинарното стебло е доделена една вредност наречена клуч. Бинарното стебло се нарекува пребарувачко доколку за секој јазел од левото подстебло на јазелот со клуч X содржи вредности (клучеви) помали од X и секој јазел од десното подстебло на јазелот со клучот X, содржи јазли со клучеви поголеми од X (за сега се претпоставува дека не постојат дупликати на клучевите). На слика 6.2а е прикажано бинарно пребарувачко стебло и обично бинарно стебло (слика 6.2б). Второто стебло не е пребарувачко поради тоа што јазелот со клуч 7 не се наоѓа во десното подстебло на јазелот со клуч 6.



(а) **(б)**

слика 6.2 Бинарно пребарувачко стебло (а) и обично бинарно стебло (б)

Интересно е да се забележат неколку својства на бинарните пребарувачки стебла. Јазелот со најмал клуч е јазелот до кој се доаѓа доколку од коренот се оди само по левите врски се додека не се дојде до листот на стеблото. Јазелот со најголем клуч е листот до кој се доаѓа со одењето по само десните врски почнувајќи од коренот на стеблото. Доколку стеблото се измине во инордер, се добива подредена низа од клучеви и тоа во растечки редослед.

Една можна дефиниција на јазелот од бинарно стебло е дадена во продолжение

```
typedef struct tree_node *tree_ptr;
struct tree_node
{
element_type element;
tree_ptr left;
tree_ptr right;
};
typedef tree_ptr SEARCH_TREE;
```

Како што се гледа од кодот, оваа дефиниција е во принцип иста како дефиницијата на јазелот за обично бинарно стебло. Од тука следи дека апликативно мора да се реши дефиницијата за пребарувачко бинарно стебло. Тоа значи дека при пишување на кодот за креирање на бинарно стебло мора да се извршат сите проверки кои ќе гарантираат дека е задоволена дефиницијата на бинарно пребарувачко стебло.

Типична операција која се извршува над бинарното пребарувачко стебло е операцијата на пребарување. Таа операција може да се реализира со кодот на следната рекурзивна функција:

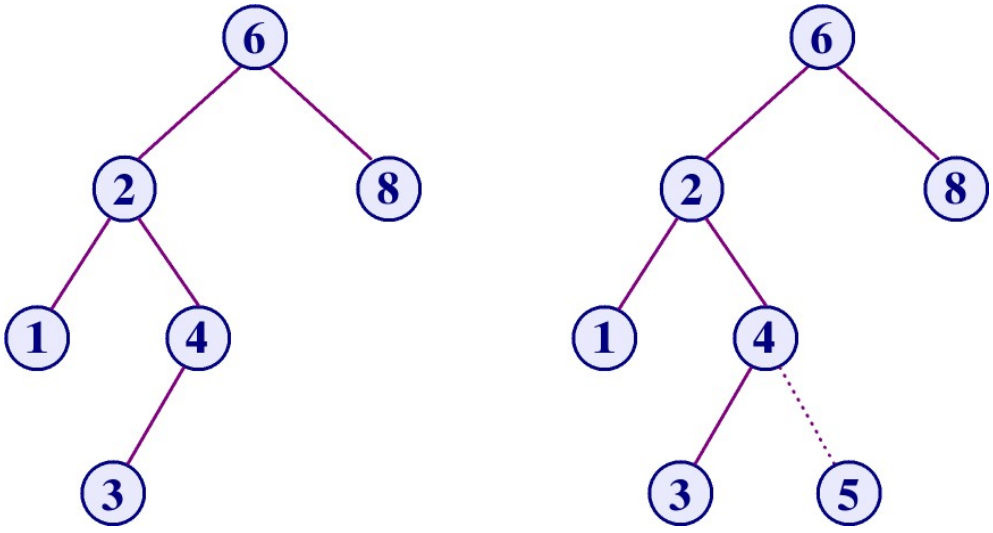
// Daden e pokazuvac kon korenot na stebloto i kluc *k*, `TREE-SEARCH` vrakja pokazuvac kon jazel so kluc *k* ako takov postoi; vo obraten slucaj, vrakja `NIL`. //

```
TREE-SEARCH (x, k)
1 if x = NIL or k = key[x]
2     then return x
3  if k < key[x]
4     then return TREE-SEARCH (left[x], k)
5     else return TREE-SEARCH (right[x], k)
```

Истата функција во нерекурзивна изведба би била

```
ITERATIVE-TREE-SEARCH (x,k)
1  while x ≠ NIL and k ≠ key[x]
2      do if k < key[x]
3            then x ← left[x]
4            else x ← right[x]
5  return x
```

Кодот за градење (вметнување на јазел на бинарно пребарувачко стебло) е релативно едноставен затоа што се сведува на движење низ јазлите на стеблото се додека не се дојде до јазел кој е најблизок (помал или поголем) од клучот од јазелот што треба да се вметне. Јазелот што се вметнува е секогаш терминален јазел (види слика 6.3).



(а) **(б)**

слика 6.3 Бинарно пребарувачко стебло пред (а) и по (б) вметнување на јазел со клуч 5

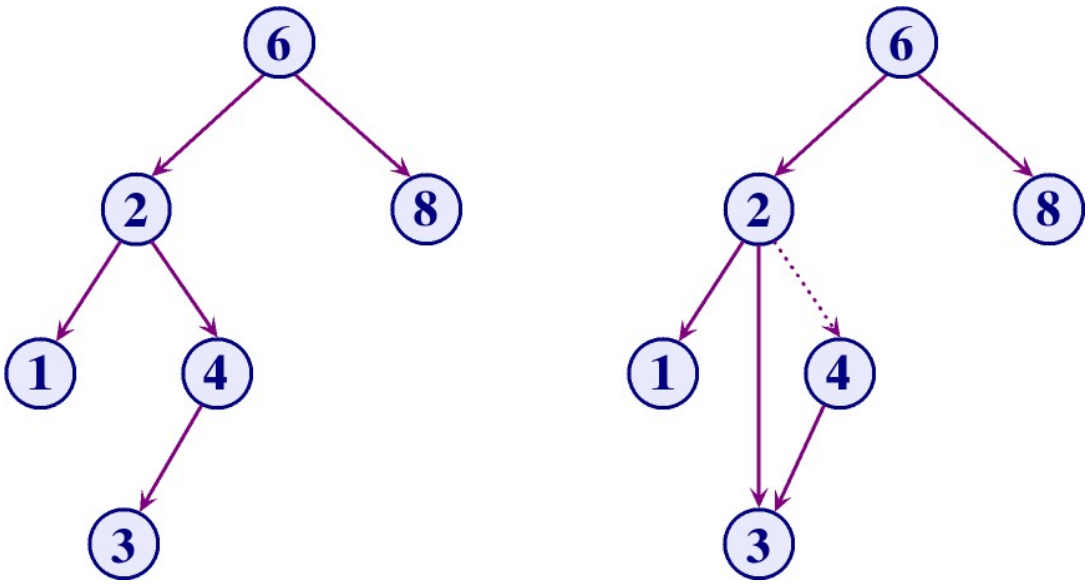
Кодот за вметнување на јазел во бинарно пребарувачко стебло може да се реализира на следниот начин:

// Na procedurata i se predava jazel *z* za koj *key*[*z*] = *v*, *left*[*z*] = NIL i *right*[*z*] = NIL. Go modificira *T* i del od polinjata na *z* na takov nacin sto *z* se vmetnuva vo soodvetna pozicija od drvoto.//

```
TREE-INSERT(T,z)
1  y ← NIL
2  x ← root[T]
3  while x ≠ NIL
4        do begin y ← x
5           if key[z] < key[x]
6              then x ← left[x]
7              else x ← right[x]
8        end do //y stores the parent of x//
9 if y = NIL
10       then root[T] ← z
11       else if key[z] < key[y]
12               then left[y] ← z
13               else right[y] ← z
```

Операцијата на бришење на клуч (односно јазел што го содржи клучот) од бинарно пребарувачко стебло е покомплексна операција од операцијата вметнување. Тоа се должи на фактот што и по бришењето на саканиот јазел, стеблото мора да ги задоволува условите кои се барани од него за тоа да биде пребарувачко. Поради тоа, при бришењето на јазел од бинарно стебло ќе бидат разгледани неколку случаи:

Доколку јазелот што треба да се избрише е лист, бришењето е едноставно и се врши со едноставно премостување на покажувачот на јазелот родител (покажувачот кон јазелот што треба да се ибрише станува нулев покажувач).

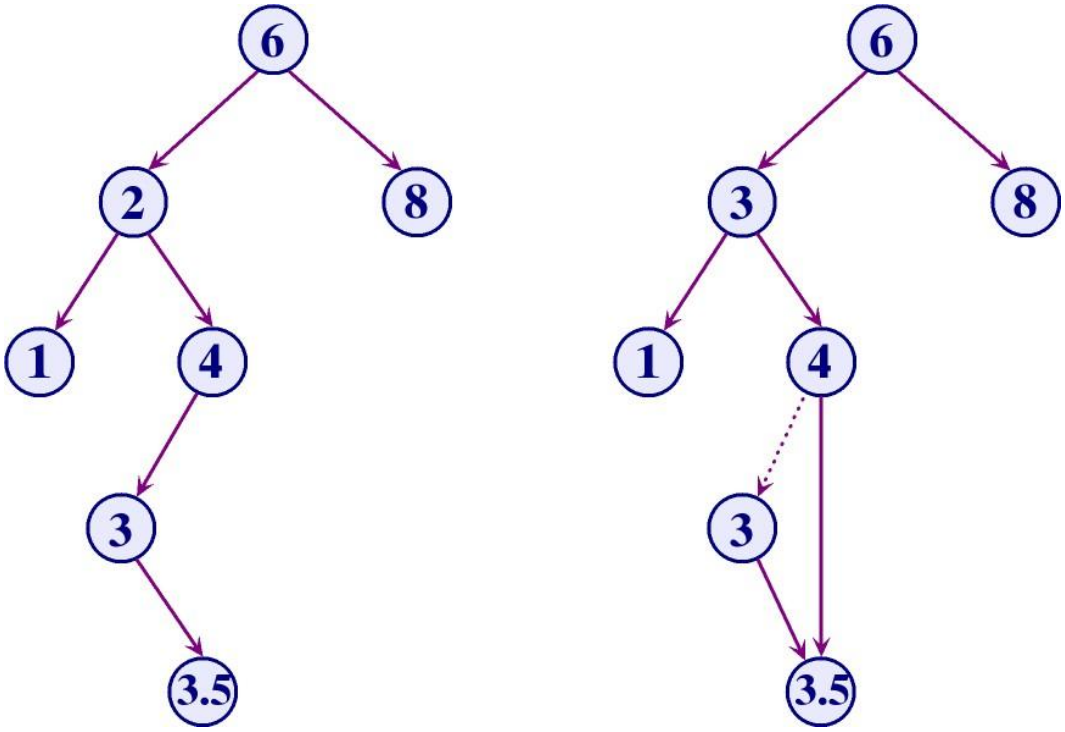


(а) **(б)**

слика 6.4 Бришење на јазелот со едно дете (клуч 4) пред (а) и после операцијата (б)

Најкомплексен е случајот кога јазелот што треба да се избрише има две деца. Алгоритмот на бришење кој треба да се реализира се состои од два чекора. Во првиот чекор се наоѓа најмалиот клуч од десното подстебло на јазелот што треба да се избрише. Во вториот чекор тој се заменува со клучот од јазелот кој треба да се избрише. Во третиот чекор се брише јазелот од кој се искористил клучот во предходниот чекор. Овој случај на бришење е едноставен затоа што јазелот што го содржи најмалиот клуч во некое подстебло мора да има нулева лева врска, па согласно предходната анализа бришењето се сведува на премостување. Важно е да се разбере дека со оваа замена стеблото останува балансирано, поради тоа што сите јазли од десното подстебло се поголеми од јазелот што сакаме да го избришеме, па согласно тоа, по неговото бришење тие ќе бидат поголеми од најмалиот јазел во тоа подстебло.

Илустрација на овој случај на бришење е дадена на слика 6.5.



(а) **(б)**

слика 6.5 Бришење на јазелот со две деца (клуч 2) пред (а) и после операцијата (б)

Треба да се забележи дека случајот на бришење на јазел со две деца работи и кога клучот на јазелот што треба да се избрише ќе се замени со најголемиот клуч од неговото лево подстебло. Зошто?

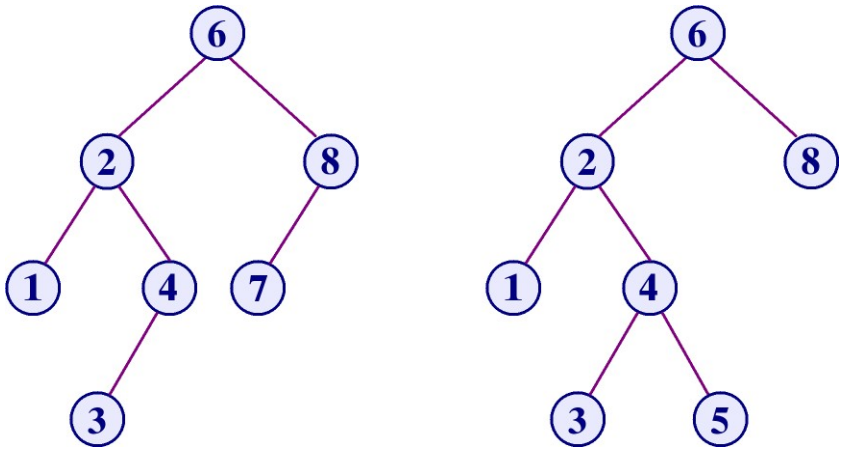
Една можна реализација на операцијата бришење е дадена со следната рекурзивна функција:

```
delete( element_type x, SEARCH_TREE T )
{
tree_ptr tmp_cell, child;
if( T == NULL )
        error("Elementot ne e pronajden");
else
        if( x < T->element )  /* Odi levo */
                T->left = delete( x, T->left );
        else
                if( x > T->element )  /* Odi desno */
                        T->right = delete( x, T->right );
                else      /* Najdeniot element da se izbrise */
                        if( T->left && T->right )  /* Dve deca */
                {       /* Zameni so najmaliot od desnoto podsteblo */
                        tmp_cell = find_min( T->right );
                        T->element = tmp_cell->element;
                        T->right = delete( T->element, T->right );
                        }
                        else      /* Edno dete */
                                {
                                tmp_cell = T;
                                if( T->left == NULL )
                                /* Samo desno dete */
                                child = T->right;
                                if( T->right == NULL )
                                /* Samo levo dete */
                                child = T->left;
                                free( tmp_cell );
                                return child;
                                }
return T;
}
```

Времетраењето на сите разгледани операции (пребарување, вметнување на јазел, бришење на јазел) кај бинарните пребарувачки стебла зависи од висината на јазелот кој треба да се обработи. Како што утврдивме на почетокот на ова поглавје нивото на јазлите во стебло со n елементи може значајно да варира во интервалот од $\log_2 n$ до n. Доколку сакаме подобри перформанси стеблата треба да бидат балансирани.

6.4. AVL Стебла

AVL (Adelson-Velskii & Landis) стебло е бинарно пребарувачко стебло кое е дополнително и балансирано стебло. Со тоа се осигурува дека длабочината на стеблото (а со тоа и комплексноста на најчестите операции) е од редот $O(\log n)$. На слика 6.6 е даден пример за AVL стебло (а) и бинарно пребарувачко стебло (б). Стеблото на слика 6.6б не е AVL стебло, затоа што, иако е пребарувачко висината на левото и десното подстебло на јазелот со клуч 6 (коренот) се разликуваат за повеќе од еден.

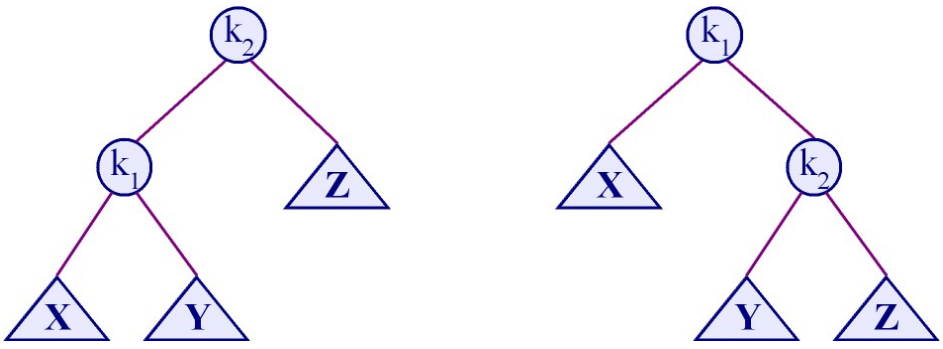


(а) **(б)**

слика 6.6 AVL стебло (а) и бинарно пребарувачко стебло (б)

Со користење на AVL стеблата се добива подобри перформанси. За жал, реализацијата на овие стебла (како и во случајот на обичните пребарувачки стебла) повторно мора да се изврши програмски. Поради ова, се зголемува комплексноста на кодот на операциите на вметнување на јазел и бришење на јазел. Поради природата на операциите, овие операции може да предизвикаат нарушување на балансираноста на стеблото.

Двете стебла дадени на слика 6.7 се бинарни пребарувачки стебла доколку за нив важи дека k2 > k1. Трансформацијата со која се постигнува конверзија помеѓу овие две стебла се нарекува еднократна ротација, и може да биде корисна доколку се примени за исполнување на условот за балансираност по некоја операција со AVL стеблата.



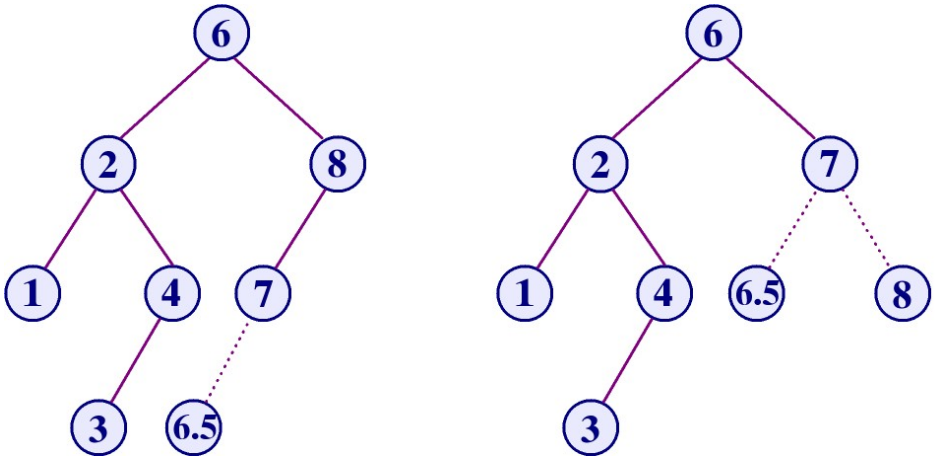
(а) **(б)**

слика 6.7 Еднократна ротација на десно од (а) кон (б) и на лево (обратно)

Треба да се укаже дека еднократната ротација може да се изврши врз било кој јазел од стеблото. Таа се извршува врз јазелот во кој е утврдено нарушување на балансираноста. Постапката за проверка на балансираноста по операција над некој јазел вообичаено почнува од тој јазел, при што се проверува неговата балансираност и ако е потребно се балансира. Доколку балансираноста на тој

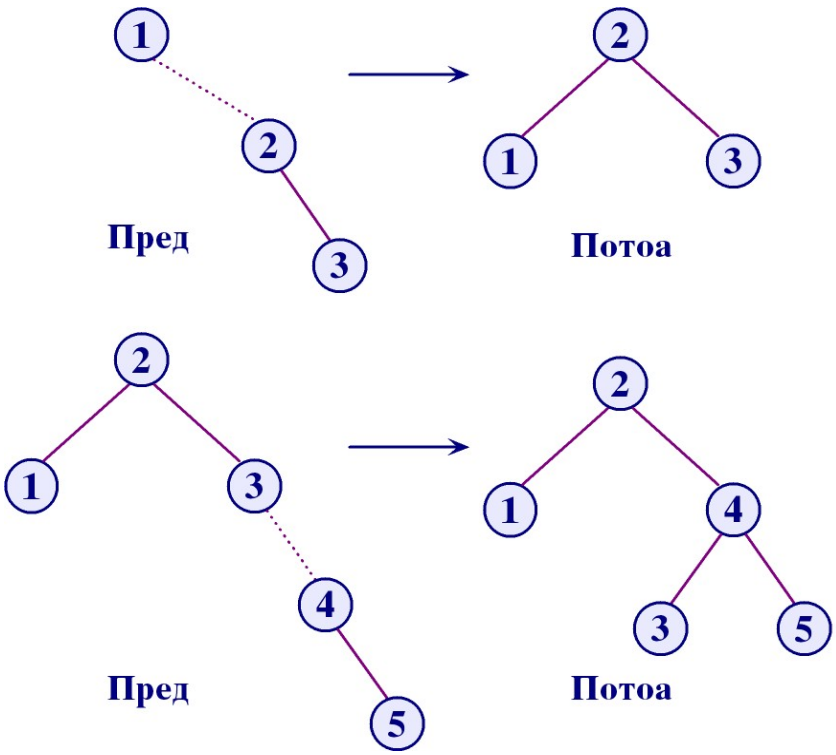
јазел не е нарушена се проверува балансираноста на неговиот родител. Оваа постапка се повторува се додека не се дојде до коренот на стеблото.

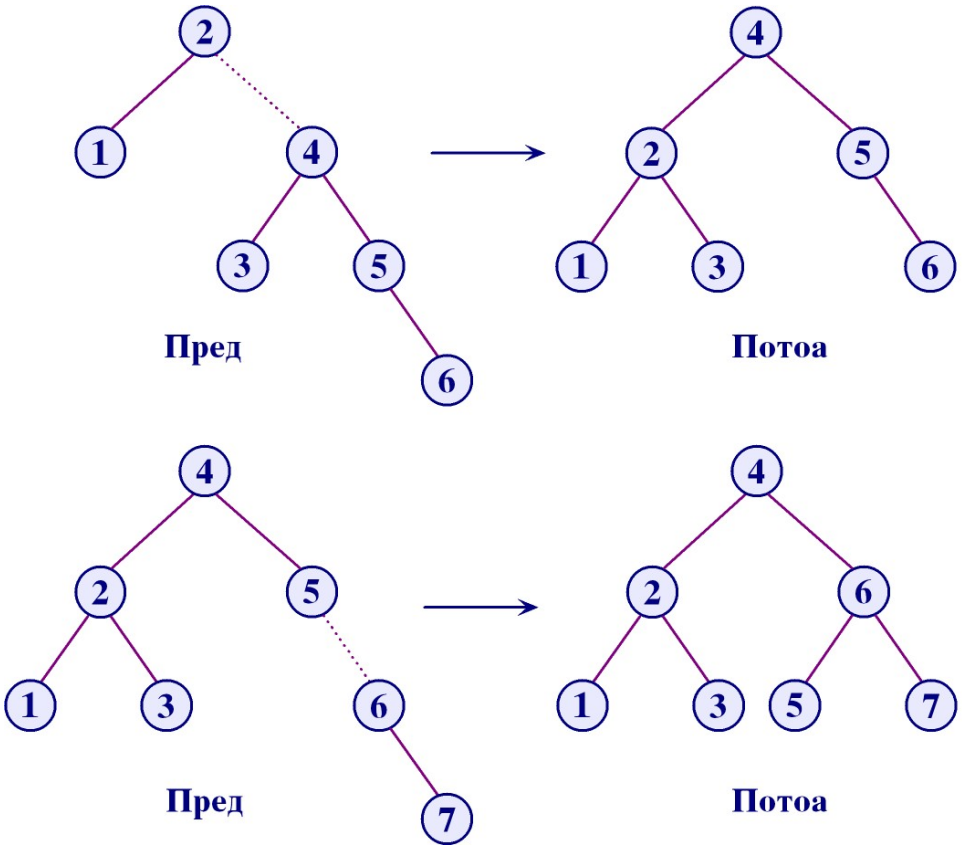
На примерот даден на слика 6.8 по вметувањето на јазелот 6,5, нарушена е балансираноста на јазелот 8 и таа се корегира со еднократна ротација на десно.



слика 6.8 Пример за балансирање на стебло со помош на еднократна ротација на десно

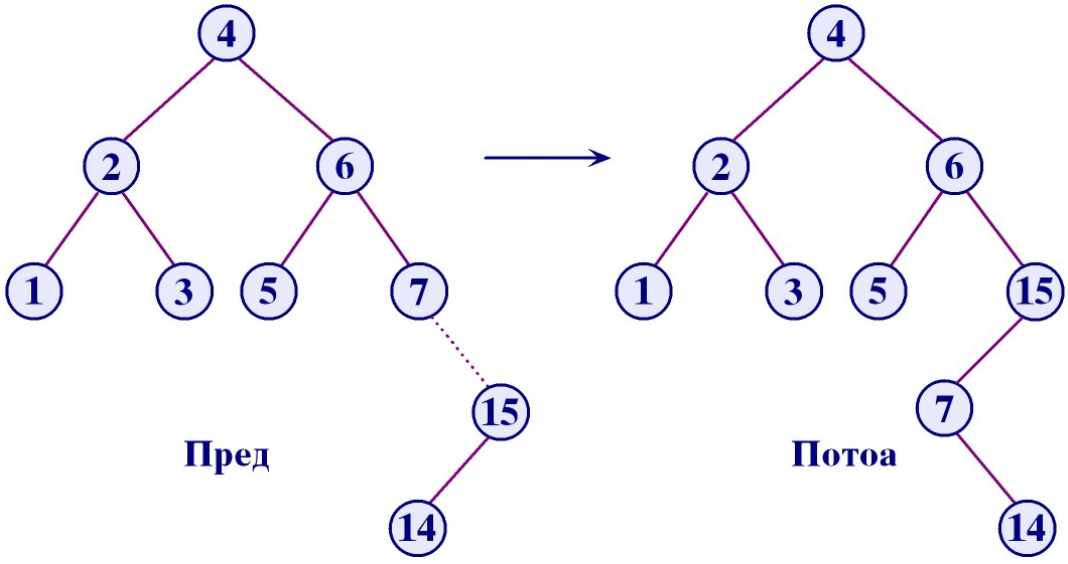
Во следниот пример (види слика 6.9) е покажана постапката на креирање на AVL стебло кога во него се внесуваат последователно клучеви со вредности: 1, 2, 3, 4, 5, 6 и 7. При тоа по потреба се вршат еднократни ротации на лево. Јазлите кај кои е утврдена нарушеност на балансираноста имаат една врска означена со испрекината линија.





слика 6.9 Пример за балансирање на стебло со помош на еднократна ротација на лево

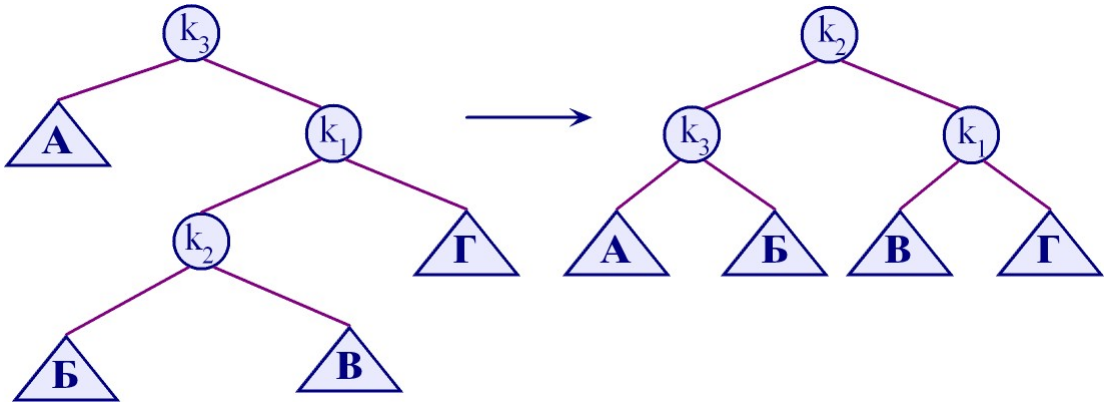
Предходниот алгоритам не дава секогаш задоволителни резултати. За илустрација да го продолжиме внесувањето во предходното AVL стебло со клучевите 15 и 14 (види слика 6.10).



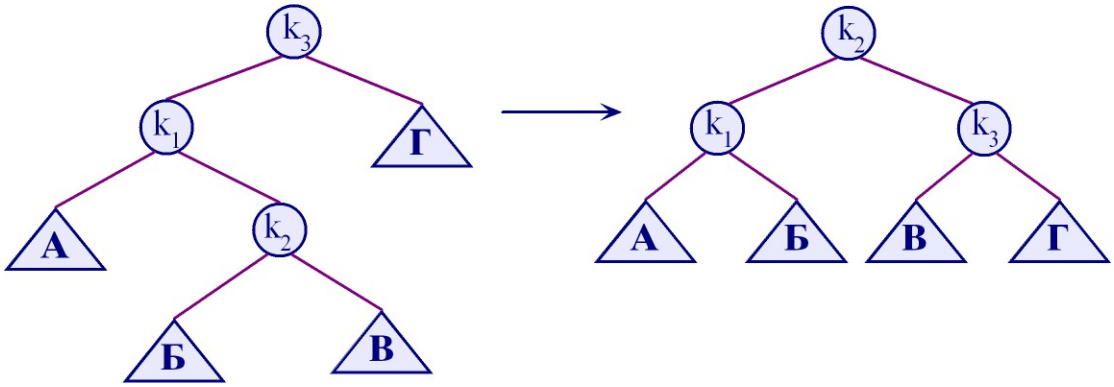
слика 6.10 Пример кога еднократната ротација не помага во балансирањето на стеблото

Внесувањето на клучот 14 предизвикува нарушување на балансираноста, но со примена на еднократна ротација на лево таа не се разрешува. Тоа е поради тоа што нововнесениот клуч (14) припаѓа на подстеблото У од слика 6.7 Тоа се елементи кои се вметнати како поголеми елементи од некое лево подстебло или помали елементи од некое десно подстебло (елементи кои визуелно одат кон средината на подстеблото). Во тој случај треба да се примени двојната или двократна ротација.

Двократната ротација е графички опишана на слика 6.11. Постојат два симетрични типа на двократна ротација: двократна ротација десно-лево и двократна ротација лево-десно.



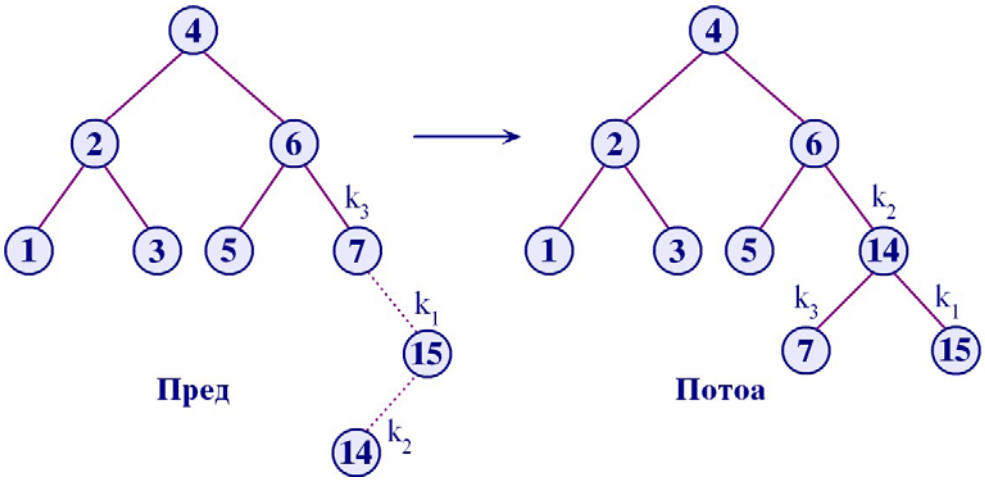
(а)



(б)

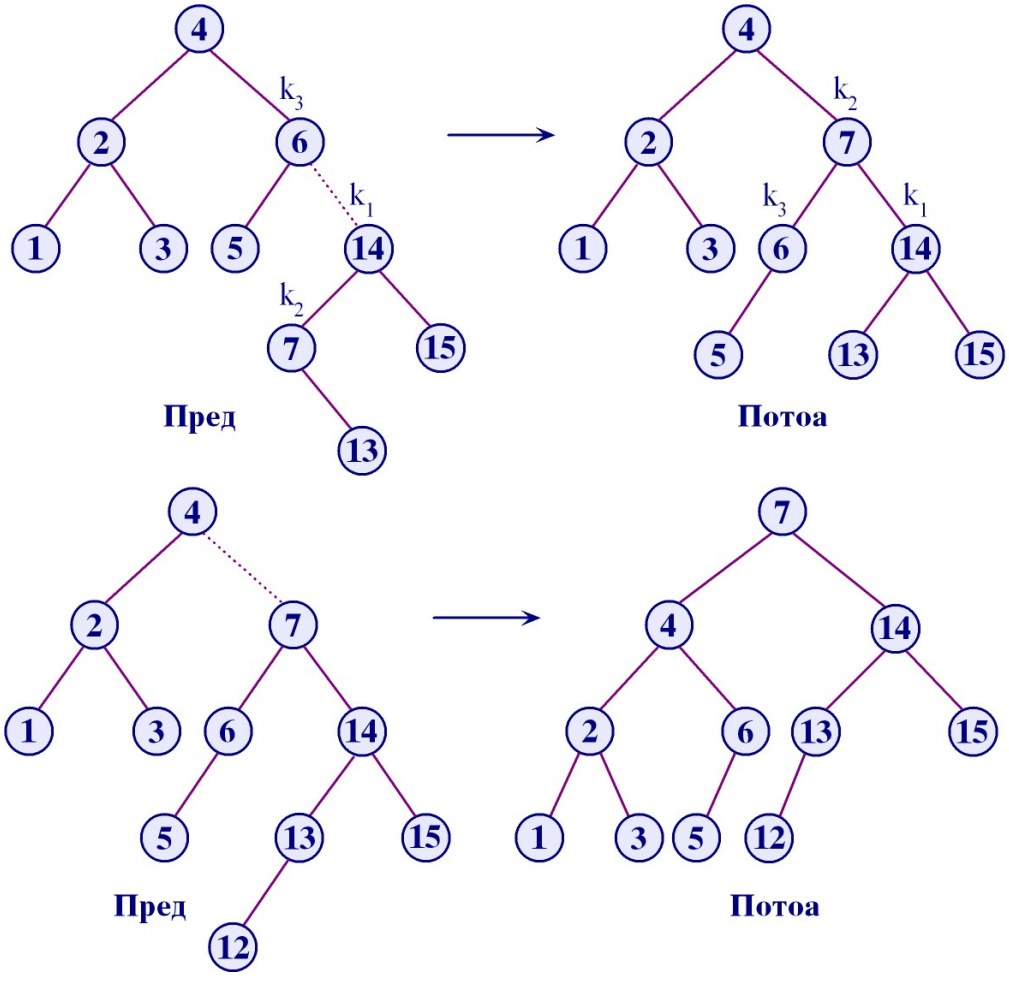
слика 6.11 Двократна ротација десно-лево (а) и лево десно (б)

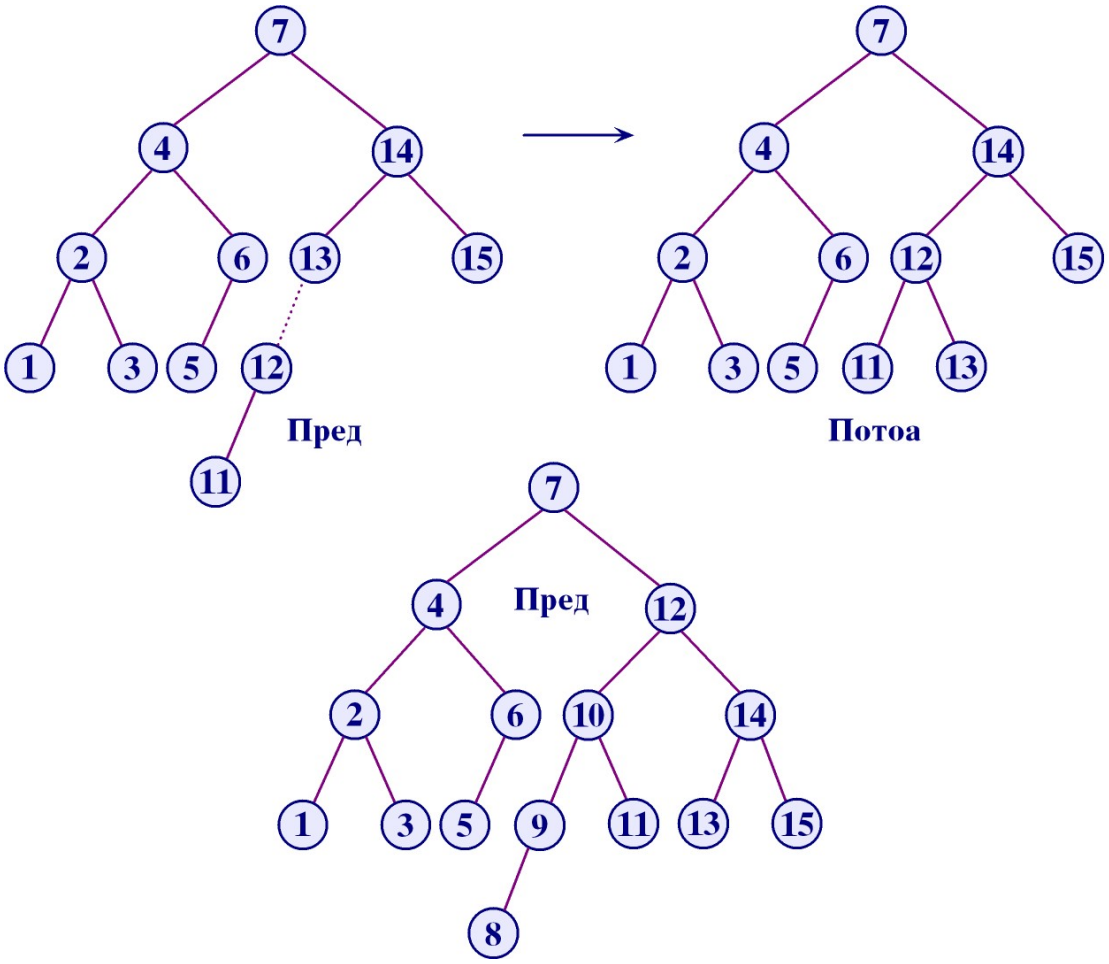
Со примена на двократната ротација десно-лево, во примерот од слика 6.10 се добива балансираното стебло како во слика 6.12.



слика 6.12 Примена на двократна ротација десно-лево за балансирање на стебло

Да продолжиме со примерот внесувајќи ги последователно клучевите 13, 12, 11, 10, 9 и 8. Дел од овие клучеви ќе предизвикаат нарушување на балансираноста кое треба да се реши со двојна ротација (13), еднократна ротација (12, 11, 10 и 9), додека внесувањето на клучот 8 не ја нарушува балансираноста на стеблото. Постпката на градење на стеблото е дадена на слика 6.13.





слика 6.13 Постапка на градење на AVL стебло

Бришењето на јазли во AVL стебла е малку покомплексно од вметнувањето. Постојат повеќе стратегии за бришење на јазли од ваквите стебла. Во случај кога нема честа потреба од бришење на јазли најчесто се користи таканаречената "мрзливо" бришење кое претпочитува само означување на јазлите кои не се користат, но не и нивно физичко отстранување.

6.5 B - Стебла

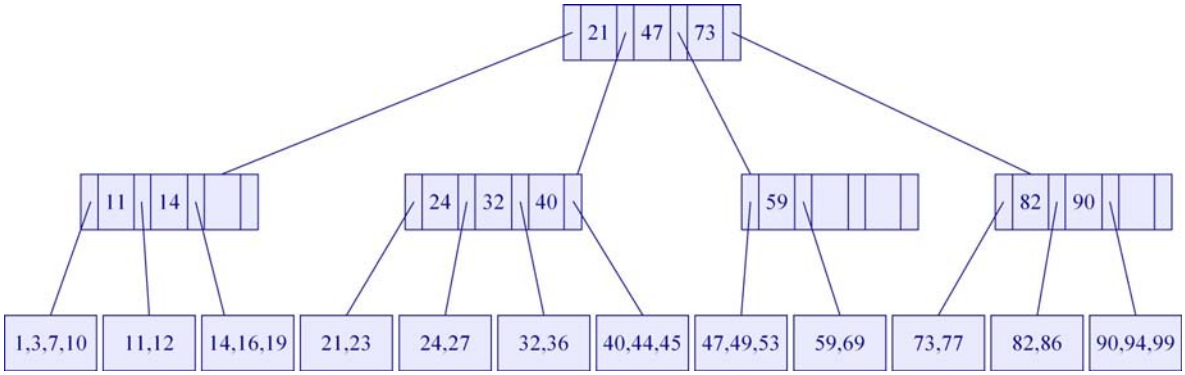
Постојат и повеќе типови на небинарни пребарувачки стебла. Во рамките на овој курс ќе бидат обработени само B стеблата. Останатите (B+, B*, R итн) претставуваат подобрување на B стеблата но, во суштина се засноваат на истата идеа.

B-стебло од ред *m* е стебло со следните структурни својства:

- Коренот е или лист или има помеѓу 2 и m деца.
- Сите внатрешни јазли (освен коренот) имаат помеѓу $\lceil m/2 \rceil$ и *m* деца.
- Сите листови се на исто ниво.

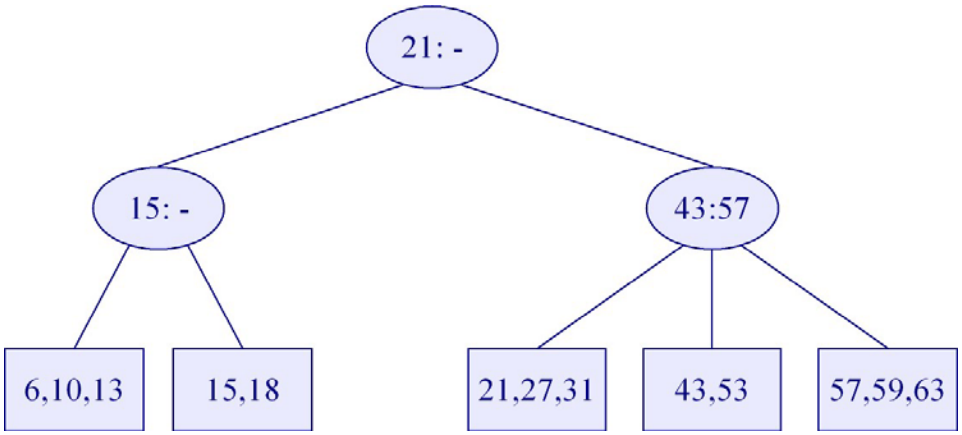
Сите податоци се сместени во листовите. Внатрешните јазли содржат покажувачи кон децата ($p_1, p_2, \ldots, p_m$) и вредности - клучеви ($k_1, k_2, \ldots, k_{m-1}$) кои ги означуваат најмалите клучеви кои постојат во подстеблата одредени со ($p_2, p_3, \ldots, p_m$) соодветно. Доколку не постојат вредности за клучевите во внатрешните јазли, соодветните покажувачи се нулеви. Нека (иако тоа не мора да биде случај) бројот на информации и во листовите се движи во опсегот од $\lceil m/2 \rceil$ и m.

На слика 6.14 е прикажано B стебло од ред 4.



слика 6.14. B стебло од ред 4

На слика 6.15 е дадено B стебло од ред 3 (секој јазел освен коренот мора да има 2 или 3 покажувачи), каде што нетерминалните јазли се означени со елипси, додека терминалните јазли со квадрати. Нетерминалните јазли содржат еден или два клуча. И во двата случаја на стебла се забележува дека информациите во листовите се подредени.

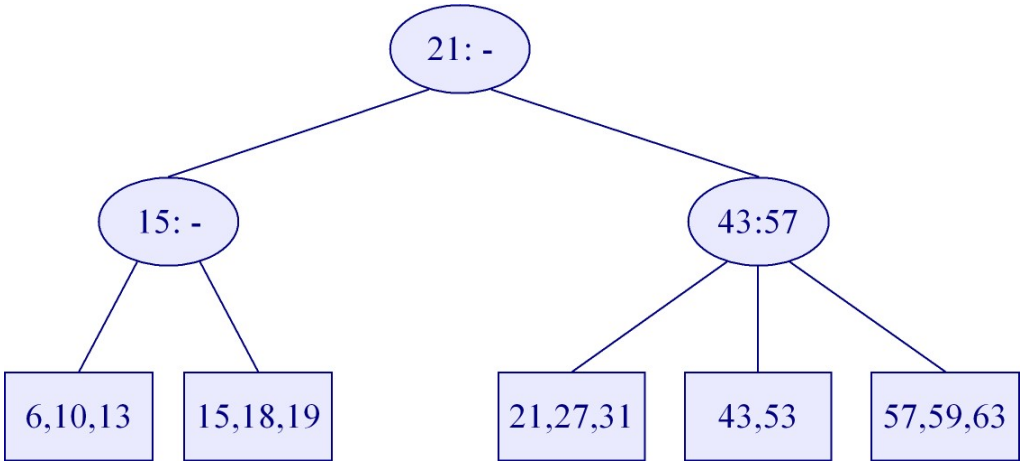


слика 6.15. B стебло од ред 3

Операцијата на пребарување е релативно едноставна. Се поаѓа од коренот и се движиме по нетерминалните јазли споредувајќи ја вредноста на клучот што го бараме, на начин многу сличен со движењето по бинарните пребарувачки стебла (одиме по покажувачот лево или десно од клучот што го споредуваме со бараниот клуч, по правилото "сите клучеви на лево се помали"). Еднаш кога ќе дојдеме до лист се движиме секвенцијално (како во низа).

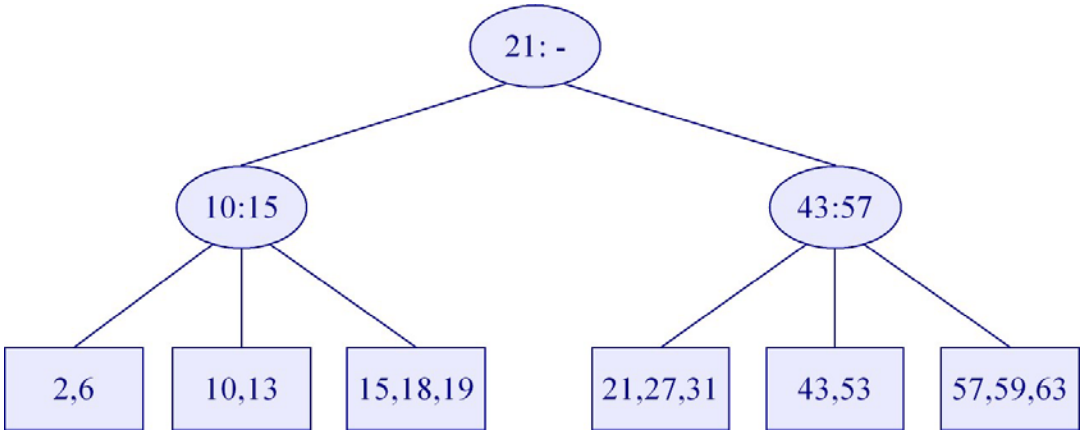
Внесувањето на вредност на клуч се сведува на внесување на вредност на клучот во еден терминален јазел и евентуална промена на клучевите во нетерминалните јазли. Да го разгледаме тој процес, на пример во кој на предходното B стебло ќе ги внесеме клучевите: 19, 2, 20 и 29.

Внесувањето на клучот со вредност 19, во првиот чекор одредува дека тој треба да се смести во непополнет краен јазел. Тогаш имаме едноставен случај на негово внесување на правото место (види слика 6.16).



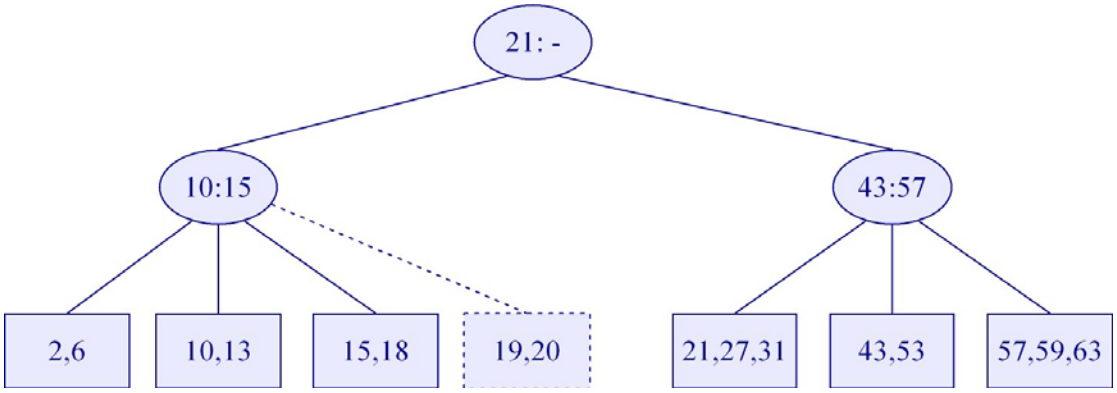
слика 6.16. Внесување на клуч во непополнет јазел од B стебло од ред 3

Внесувањето на клучот со вредност 2 открива дека тој треба да се смести во терминален јазел кој е веќе полн. Ставањето на уште еден клуч во полн јазел би предизвикало креирање на јазел со 4 елементи што не е дозволено со дефиницијата на B стеблото. Во конкретниот случај така добиениот јазел со 4 елементи може да го разделиме во два јазла со по 2 елементи, при што ќе ја ажурираме вредноста на клучевите и покажувачите во родителскиот јазел. Со тоа доаѓаме во ситуација претставена на слика 6.17. Оваа постапка може да се повтори и по останатите родители, доколку за тоа има потреба.



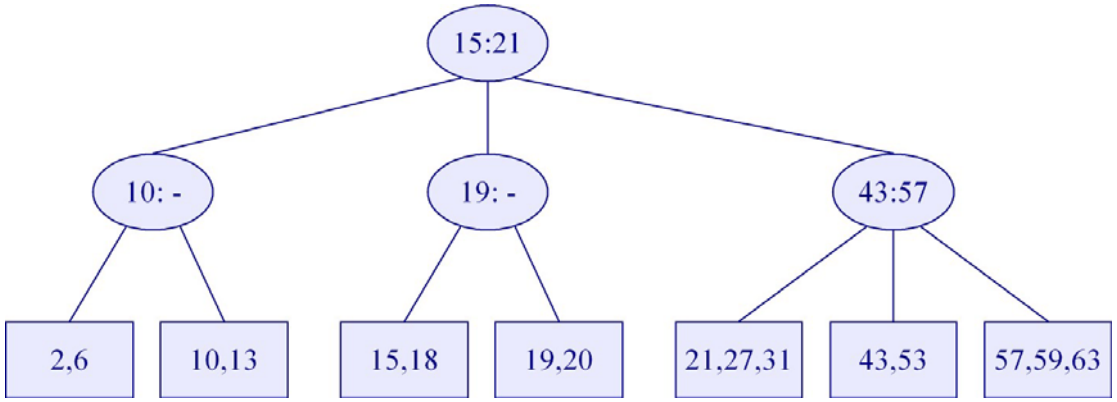
слика 6.17. Внесување на клуч во пополнет јазел од B стебло од ред 3 со помош на делење на терминален јазел

Доколку на сличен начин сакаме да го внесеме и клучот 20 ќе предизвикаме недозволена ситуација прикажана на слика 6.17. Иако терминалните јазли не се пополнети, нивниот родител содржи повеќе од дозволениот број на покажувачи.



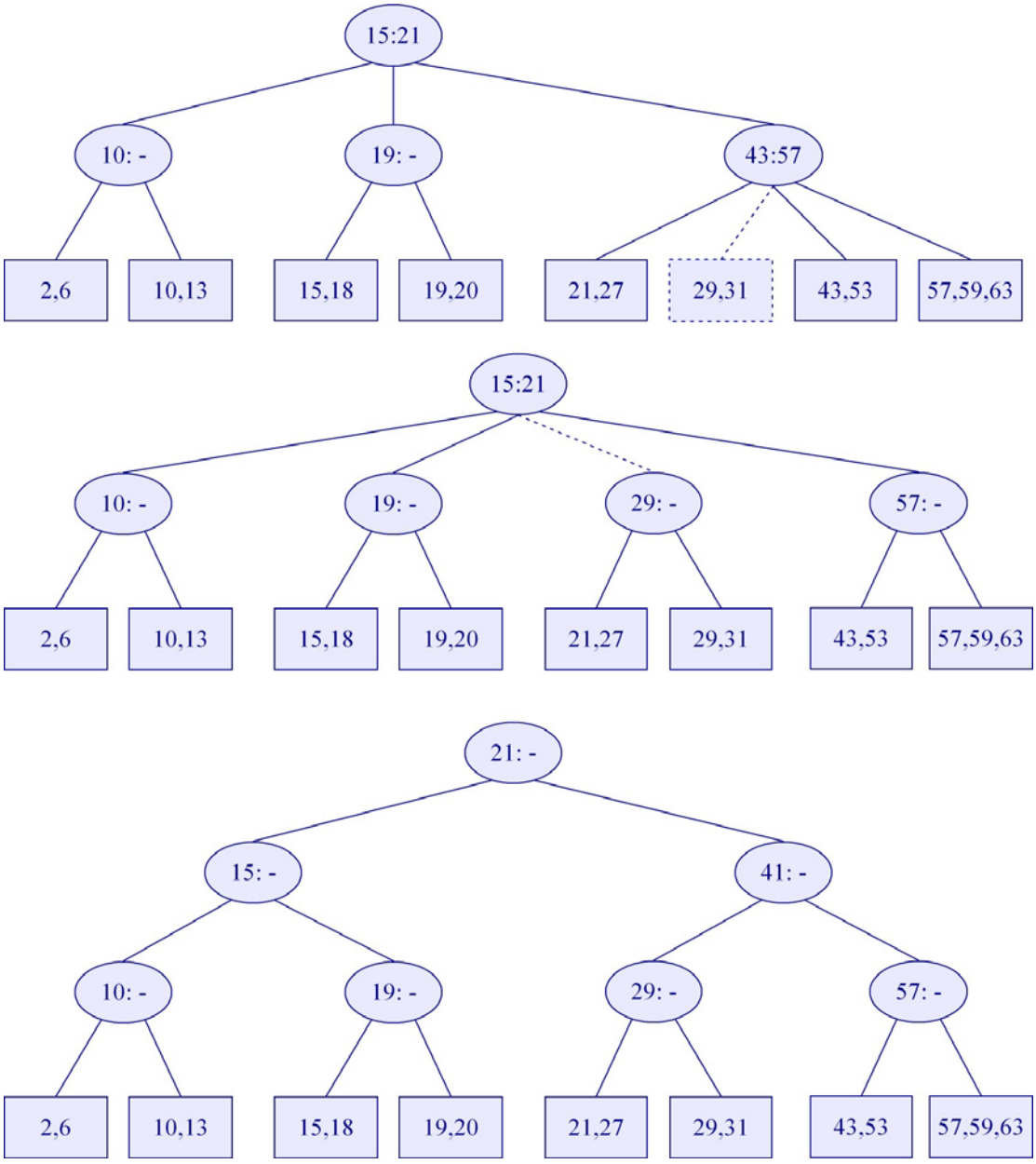
слика 6.18. Недозволена состојба за B стебло од ред 3

Во тој случај внатрешниот јазел кој е "преоптоварен" се дели на два јазли со по две деца (види слика 6.19). Тоа може да предизвика рекурзивно ажурирање на неговиот родител и така се до коренот.



слика 6.19. Делење на внатрешен јазел кај B стебло од ред 3

Итеративното креирање на нови внатрешни јазли се добива при внесувањето на клучот 29. Во првата итерација родителот на листот се дели на два нови внатрешни јазли. Со тоа се добиваат четири јазли на ниво 1, односно од коренот треба да излегуваат четири покажувачи што не е дозволено. Тоа предизвикува делење на коренот на два нови јазли, кои пак се обединуваат со воведување на нов корен со што се зголемува вкупната висина на стеблото. Целата постапка на делење е претставена на слика 6.20.



слика 6.20. Делење на внатрешен јазел кај B стебло од ред 3 со што се предизвикува зголемување на висината на стеблото

Доколку сакаме да го спречиме брзото "растење" на стеблото може да разгледаме стратегии кои ќе проверат дали со прераспределба на клучевите во листовите (и ажурирање на клучевите на внатрешните јазли) може да се спречи растењето на стеблото. Така на пример, доколку сакаме да внесеме клуч со вредност 70, наместо да воведеме нов терминален јазел, може да провериме дали соседниот терминален јазел има место (а има) и да извршиме поместување на клучот 57 во соседниот терминален јазел (на лево) со што ни се ослободува место во крајниот десен терминален јазел во кој можеме да го сместиме клучот 70. При тоа очигледно треба да направиме ажурирање на вредноста на клучот и кај родителот на двата јазли (57 да стане 59) што може каскадно да предизвика и други ажурирања во внатрешноста на стеблото. Очигледно на овој начин

заштедуваме мемориски простор, но ја правиме операцијата вметнување на клуч многу покомплицирана.

Операцијата на бришење се сведува на наоѓање на клучот и негово отстранување. Доколку со тоа отстранување терминалниот јазел остане непополнет (во нашиот пример само со еден клуч) можеме да извршиме спојување со соседен терминален јазел (брат) при што има два случаја: да се добие јазел со три или четири елементи. Во вториот случај, слично на предходно опишаното јазелот со четири клуча се дели на два јазла со по два клуча. Во првиот случај, бидејќи родителот губи едно дете тоа може да предизвика негова непополнетост, па постапаката итеративно се повторува одејќи нагоре низ стеблото. Во одреден случај може да се јави потреба од намалување на висината на стеблото.

Примената на B стеблата (или m патните пребарувачки стебла) главно се наоѓа во градењето на индексите кај системите за управување со бази на податоци.