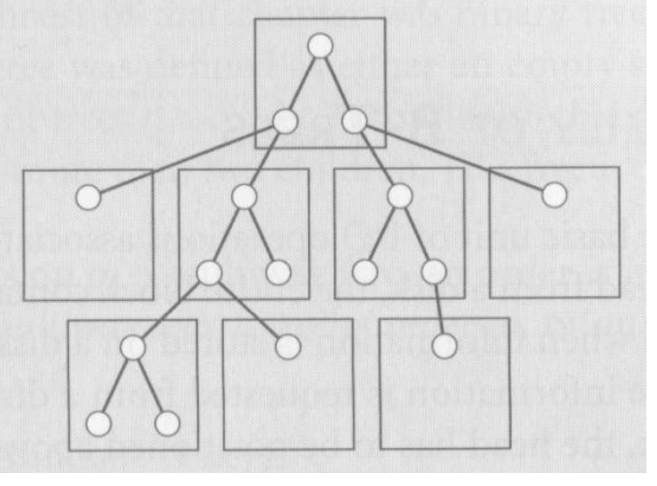
B-stabla (ponoviti gradivo iz Uvod u baze podataka)

Vanjska memorija (disk) još uvijek ima zamjetne nedostatke:

- pozicioniranje glave za čitanje/pisanje je relativno sporo (mehanička tromost)
- čitaju se cijeli blokovi podataka pa većinu pročitanih zapravo ne koristimo
- susjedni čvorovi (roditelji i djeca) stabala mogu biti "razasuti" u udaljene blokove pa je prijelaz iz roditelja u dijete spor usporkos njihovoj logičkoj blizini



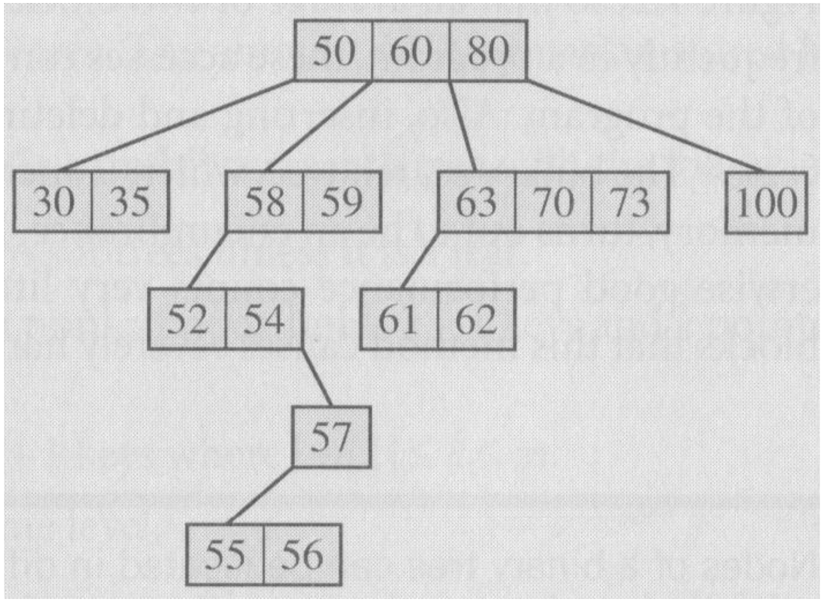
B-stabla ublažavaju posljedice ograničenja vanjskih jedinica povećanjem veličine čvorova u stablu koja se namješta na približno veličinu bloka na disku.
B-stabla su podvrsta M-stabala.

M (multiway) stabla - stabla u kojima čvorovi mogu imati proizvoljan broj djece

M-stablo m-tog reda: M-stablo u kojem čvorovi mogu imati najviše m djece

M-search-stablo m-tog reda je M-stablo sa sljedećim dodatnim svojstvima:

1. svaki čvor ima najviše m djece i m–1 podataka (ključeva)
2. ključevi u čvorovima su sortirani
3. ključevi u prvih ***i*** djece nekog čvora su manji od ***i***-tog ključa promatranog čvora
4. ključevi u zadnjih m–***i*** djece nekog čvora su veći od *i*-tog ključa promatranog čvora

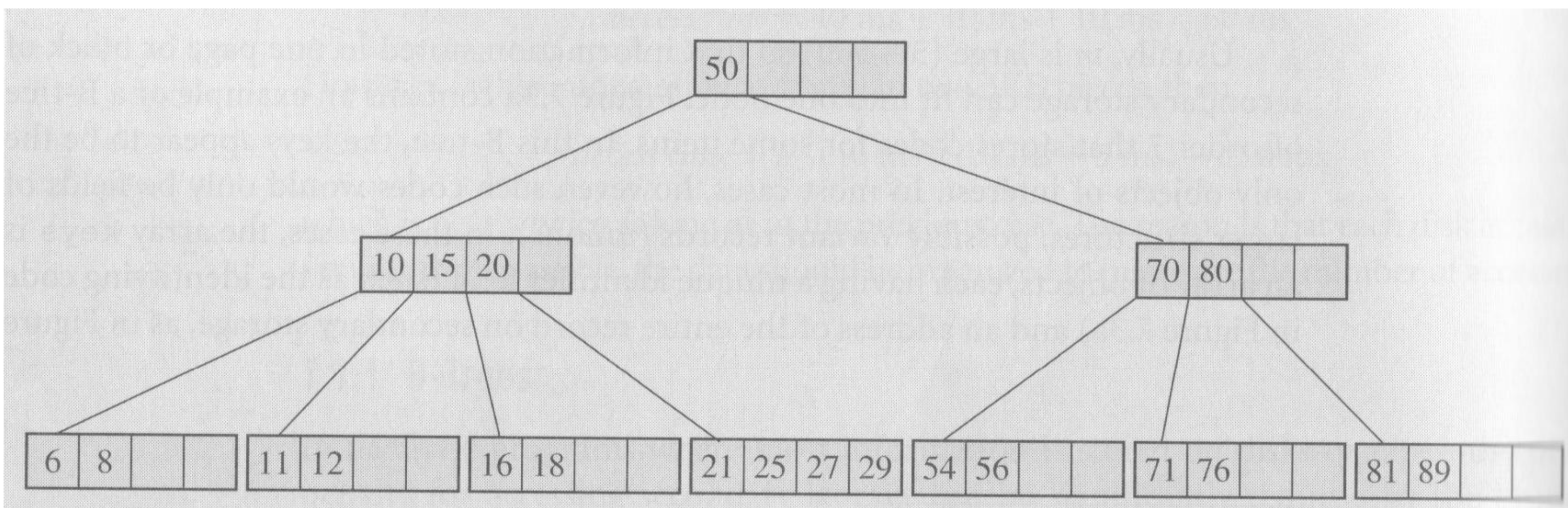


B-stablo m-tog reda je M-search-stablo sa sljedećim dodatnim svojstvima:

1. korijen ima najmanje dvoje djece, osim ako je ujedno i list (jedini čvor u stablu)
2. svaki čvor, osim korijena i listova, sadrži k–1 ključeva i k pokazivača na podstabla (ima k djece), pri čemu je $m/2 \le k \le m$ (ako rezultat dijeljenja nije cijeli broj, uzima se najmanji veći cijeli broj)
3. svi listovi sadrže k–1 ključeva, pri čemu je $m/2 \le k \le m$ (ako rezultat dijeljenja nije cijeli broj, uzima se najmanji veći cijeli broj)
4. svi listovi su na istoj razini

Sva ta svojstva ostvaruju se posebnim načinom održavanja B-stabla i nisu “prirodna” posljedica logičke apstrakcije.

Čvor B-stabla uobičajeno se realizira kao struktura (klasa) s poljem od m–1 ključeva, poljem od m pokazivača i još ponekim dodatnim podatkom za olakšavanje održavanja stabla, kao npr. brojem ključeva u čvoru ili oznake list/ne-list itd..
U prikazivanju B-stabala ti dodatni podatci se radi preglednosti izostavljaju.



Zbog definicijskih svojstava, B-stabla imaju dvije važne osobitosti:

- popunjenost im je barem 50 %
- savršeno su uravnotežena (to se postiže posebnim načinom dodavanja novih čvorova)

Algoritam pretraživanja B-stabla:

1. ući u čvor (na početku korijen) i redom pregledavati ključeve sve dok je trenutačni manji od traženog, a još ima neprovjerenih
2. ako je 1. korak završio zbog nailaska na ključ veći od traženog ili zbog dolaska do kraja čvora, spustiti se razinu niže (u odgovarajuće dijete) i nastaviti od koraka 1; u protivnom traženog ključa nema

Moguća realizacija (polja se koriste od indeksa =1, a ne =0):

```
SearchBTree (key, node)
if (node != NULL)
{  for (i=1; i<=Node->keyNum  //keyNum je član čvora
                     && node->keys[i]<key; ++i);
   if (i>Node->keyNum || node->keys[i]>key)
      SearchBTree (key, node->pointers[i]);
   else
      return node; }
else
   return 0;
```

Dodavanje čvora u B-stablo:
- za razliku od top-down izgradnje običnih stabala, B-stablo je jednostavnije graditi odozdo prema gore

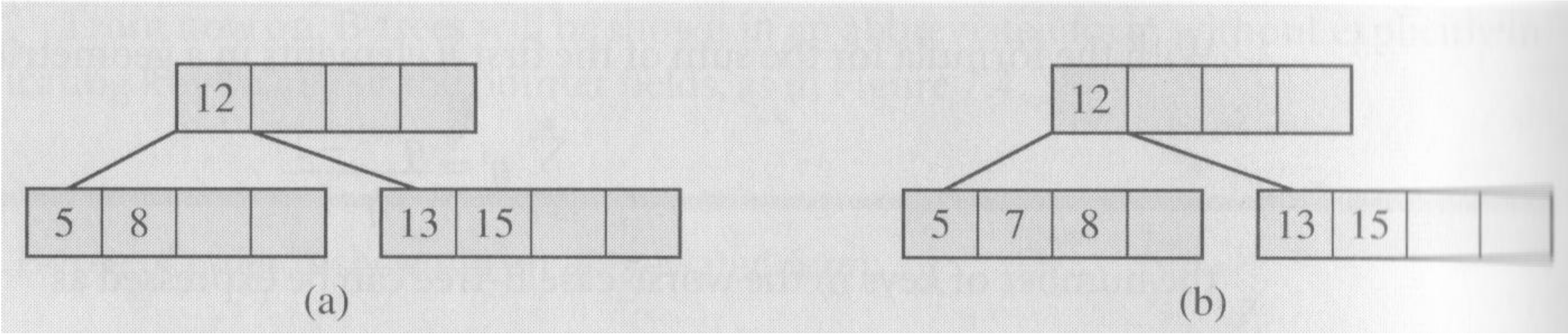
Algoritam dodavanja čvora u B-stablo:

1. pronaći list u koji bi trebalo smjestiti novi element
2. ako ima mjesta, upisati novi element
3. ako je taj list bio pun, napraviti novi list, razdijeliti elemente između ta dva čvora, a središnjeg upisati u roditelja ako u roditelju ima mjesta za njega
4. ako je i roditelj bio pun, ponavljati tu proceduru sve dok se ne dođe do korijena
5. ako je i korijen pun, napraviti novi korijen

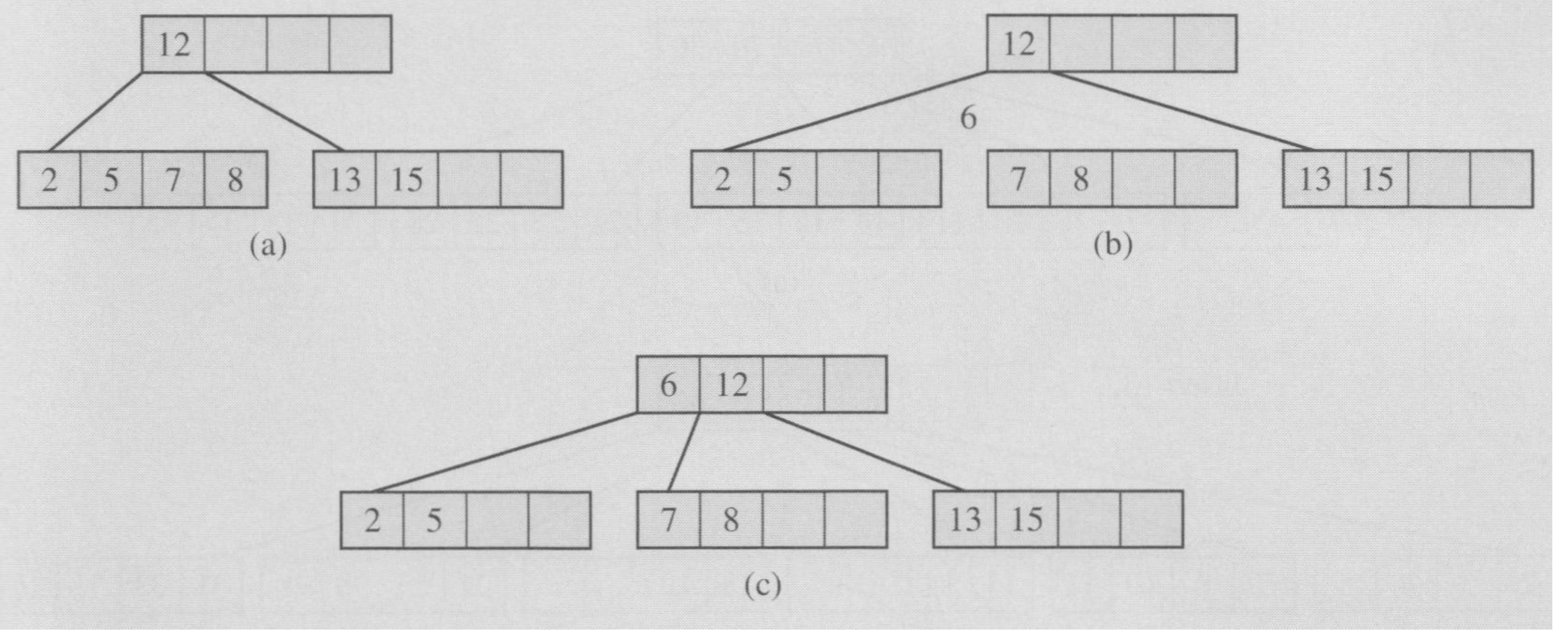
Prilikom ubacivanja novog čvora moguće su tri karakteristične situacije:

1. list u koji treba ići novi element nije pun
   - ubaciti novi element u taj list na odgovarajuće mjesto, pomičući po potrebi prethodni sadržaj

Primjer: dodavanje 7

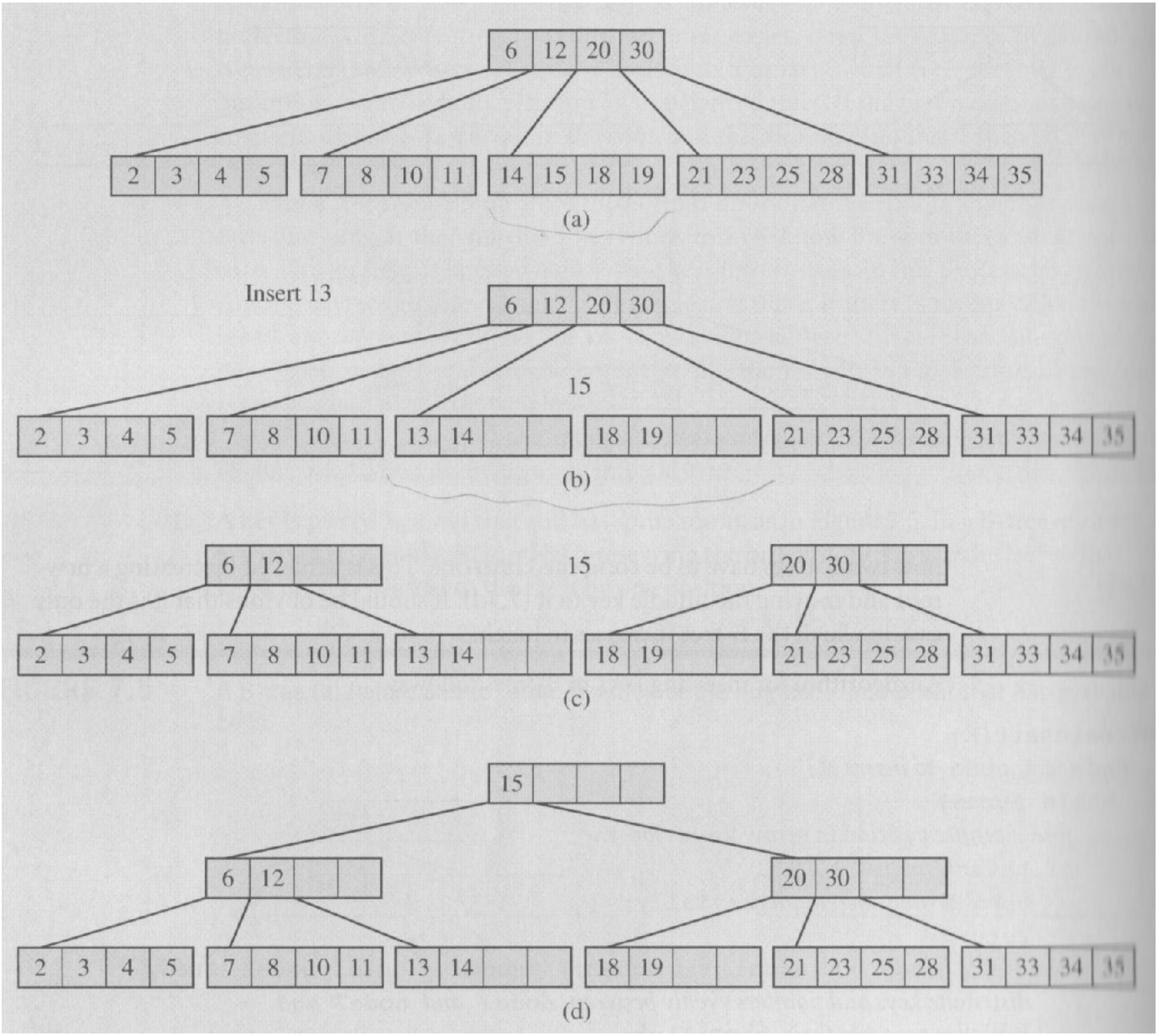


2. list u koji treba ići novi element je pun, ali korijen stabla nije
   - dovoljno je riješiti slučaj kad je list pun, a roditelj nije jer se to samo ponavlja, najviše do korijena
   - list se dijeli, tj. stvara se novi čvor i druga polovica elemenata iz popunjenog lista upisuje se u novi čvor, a središnji element se upisuje u roditelja

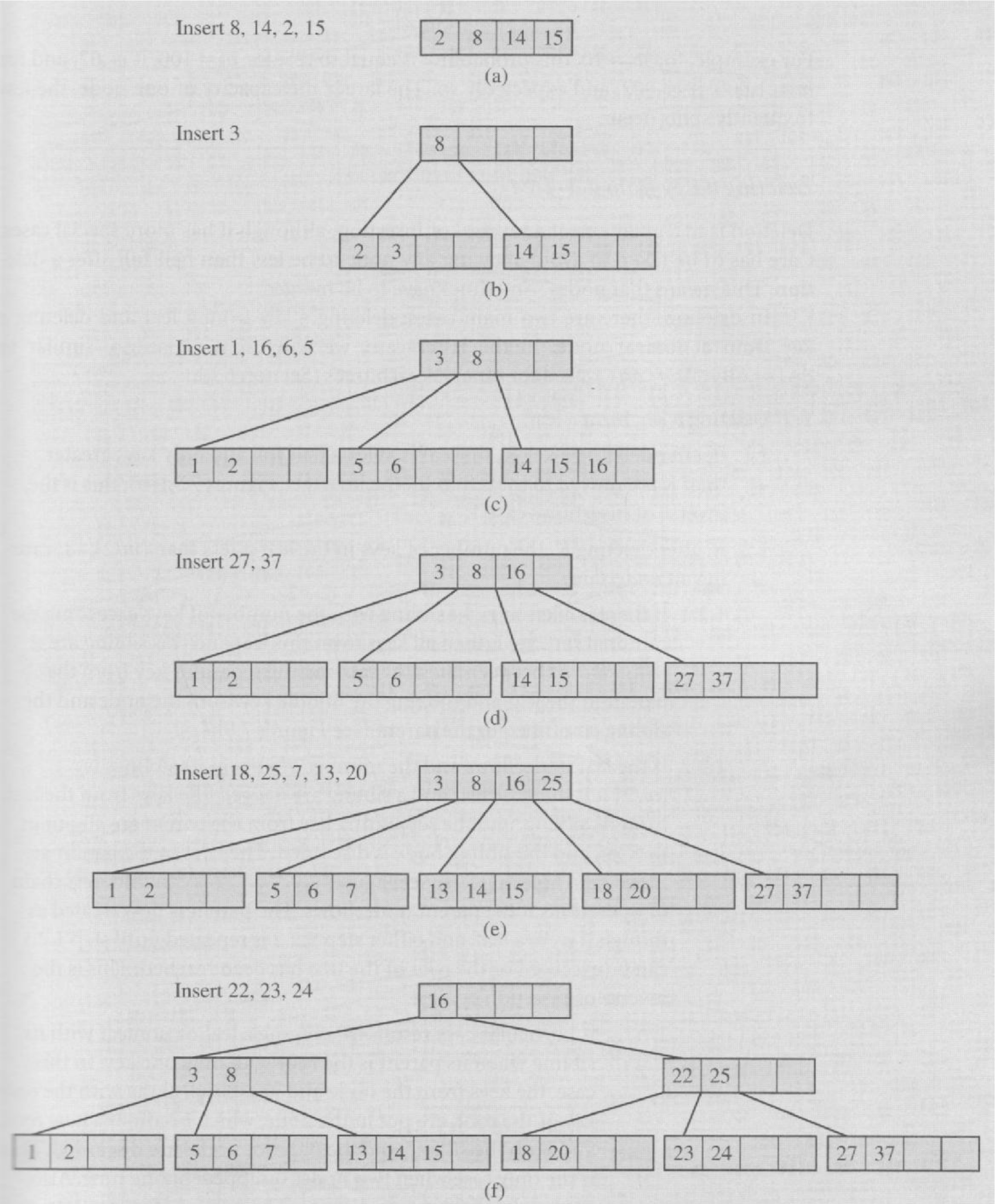


3. list u koji treba ići novi element je pun, a isto tako i korijen stabla
   - složenija inačica 2. slučaja; kad se razdijeli korijen nastaju dva B-stabla koja treba sjediniti
   - sjedinjenje se postiže stvaranjem još jednog čvora koji će biti novi korijen i upisivanjem središnjeg elementa u njega
   - to je jedini slučaj koji završava povisivanjem stabla i zahvaljujući takvom postupku, B-stablo je uvijek savršeno uravnoteženo

Primjer za 3. slučaj:



Primjer izgradnje B-stabla:
redom se dodaju 8, 14, 2, 15, 3, 1, 16, 6, 5, 27, 37, 18, 25, 7, 13, 20, 22, 23 i 24.



```
BTreeInsert (K)
find a leaf node to insert K;
while (true)
    find a proper position in array keys for K;
    if node is not full
        insert K and increment keyNum;
        return;
    else
        split node into node1 and node2;   // node1= node, node2 is new
        distribute keys and pointers evenly between node1 and node2
                                  and initialize properly their keyNum's;
        K = middle key;
        if node was the root
            create a new root as parent of node1 and node2;
            put K and pointers to node1 and node2 in the root,
                                                  and set its keyNum to 1;
            return;
        else
            node = its parent;    // and now process the node's parent
```

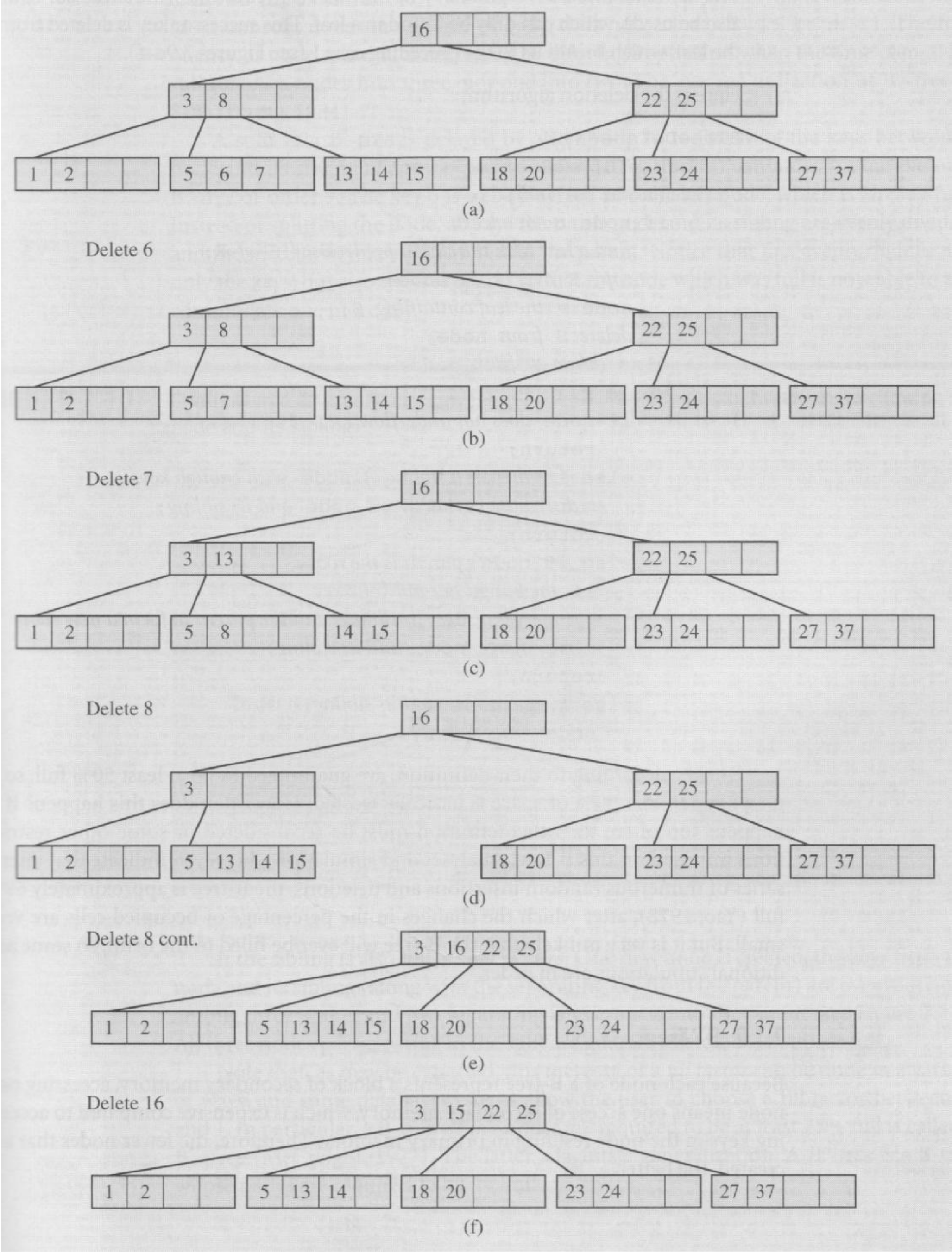
Brisanje elemenata u B-stablu: dva osnovna slučaja

1. brisanje elementa u listu
   - 1.1 list i nakon brisanja ima još barem m/2 ključeva; gotovo
   - 1.2 broj preostalih elemenata <m/2
     - 1.2.1 ako lijevo ili desno postoji susjed koji ima >m/2 ključeva, ravnomjerno rasporediti elemente iz lista i tog susjeda, pri čemu se diobeni element roditelja prepisuje u list, a središnji element ujedinjenih čvorova upisuje u roditelja (slika b-c); to je inverz 1. slučaja dodavanja elementa u B-stablo
     - 1.2.2 ako lijevo ili desno postoji susjed koji ima točno =m/2 ključeva, list i taj susjed se sjedinjuju; svi elementi lista, susjeda i diobeni element roditelja upisuju se u list, a susjed se briše iz stabla (slika c-d); to je inverz 2. slučaja dodavanja elementa u B-stablo i može izazvati lančano rasprostiranje te situacije ako sada roditelj ima <m/2 elemenata; u tom slučaju se ovaj postupak ponavlja tretirajući roditelja kao list sve dok se ne dođe u situaciju 1.2.1 ili do korijena stabla
       - 1.2.2.1 postupkom 1.2.2 došli smo do korijena koji sadrži samo jedan element; svi elementi lista, susjeda i korijena upisuju se u jedan čvor koji postaje novi korijen, a dva čvora se brišu iz stabla (slika c-e); to je inverz 3. slučaja dodavanja elementa u B-stablo

Brisanje elemenata u B-stablu: nastavak

2. brisanje elementa u čvoru koji nije list
    - samo po sebi komplicirano zbog restrukturiranja stabla
    - svodi se na brisanje elementa iz lista
        - na mjesto elementa koji treba izbrisati upisuje se njegov neposredni prethodnik, a taj može biti samo u listu
        - potom se u listu briše prepisani element postupkom u točki 1 (slika e-f)

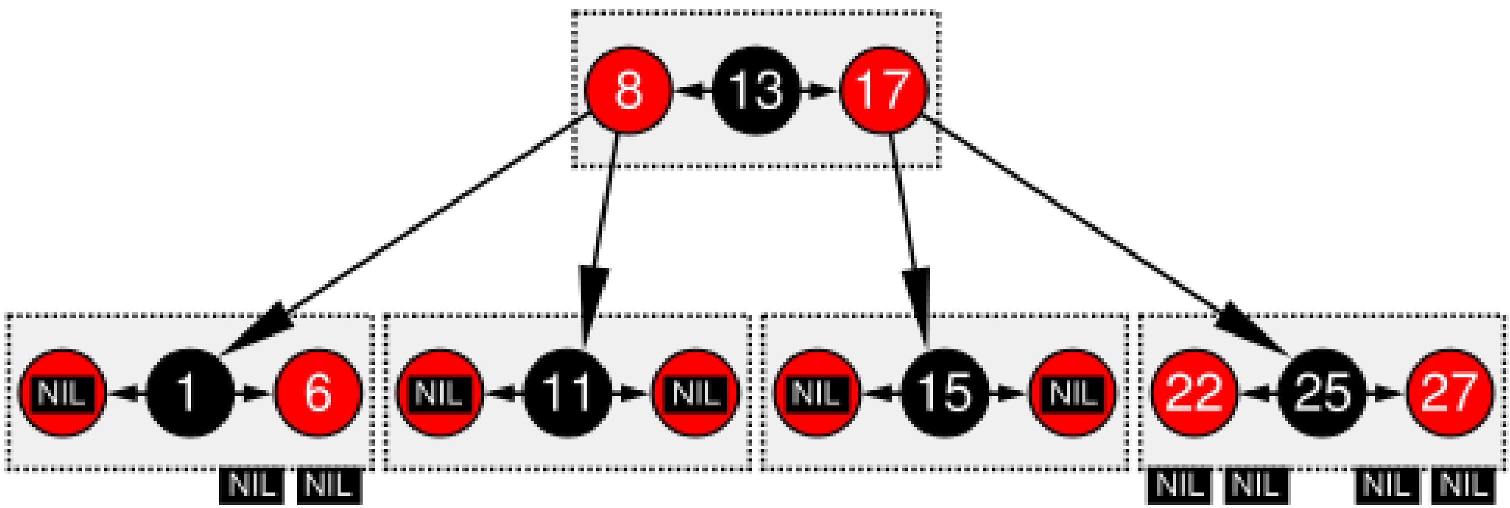
Primjer:


16
3 8
22 25
1 2
5 6 7
13 14 15
18 20
23 24
27 37
(a)
Delete 6
16
3 8
22 25
1 2
5 7
13 14 15
18 20
23 24
27 37
(b)
Delete 7
16
3 13
22 25
1 2
5 8
14 15
18 20
23 24
27 37
(c)
Delete 8
16
3
22 25
1 2
5 13 14 15
18 20
23 24
27 37
(d)
Delete 8 cont.
3 16 22 25
1 2
5 13 14 15
18 20
23 24
27 37
(e)
Delete 16
3 15 22 25
1 2
5 13 14
18 20
23 24
27 37
(f)

```
BTreeDelete (K)
node = SearchBTree (K, root);
if (node != NULL)
   if node is not a leaf
         find a leaf with the closest predecessor S of K;
         copy S over K in node;
         node = the leaf containing S;
         delete S from node;
   else
         delete K from node;
   while (1)
         if node does not underfiow                     //slučaj 1.1
               return;
         else if there is a sibling of node with enough keys redistribute
                     keys between node and its sibling;   //slučaj 1.2.1
               return;
         else if node's parent is the root               //slučaj 1.2.2.1
              if the parent has only one key
                 merge node, its sibling, and the parent to form a new root;
              else
                  merge node and its sibling;
                  return;
         else
             merge node and its sibling;                 //slučaj 1.2.2
             node = its parent;
```

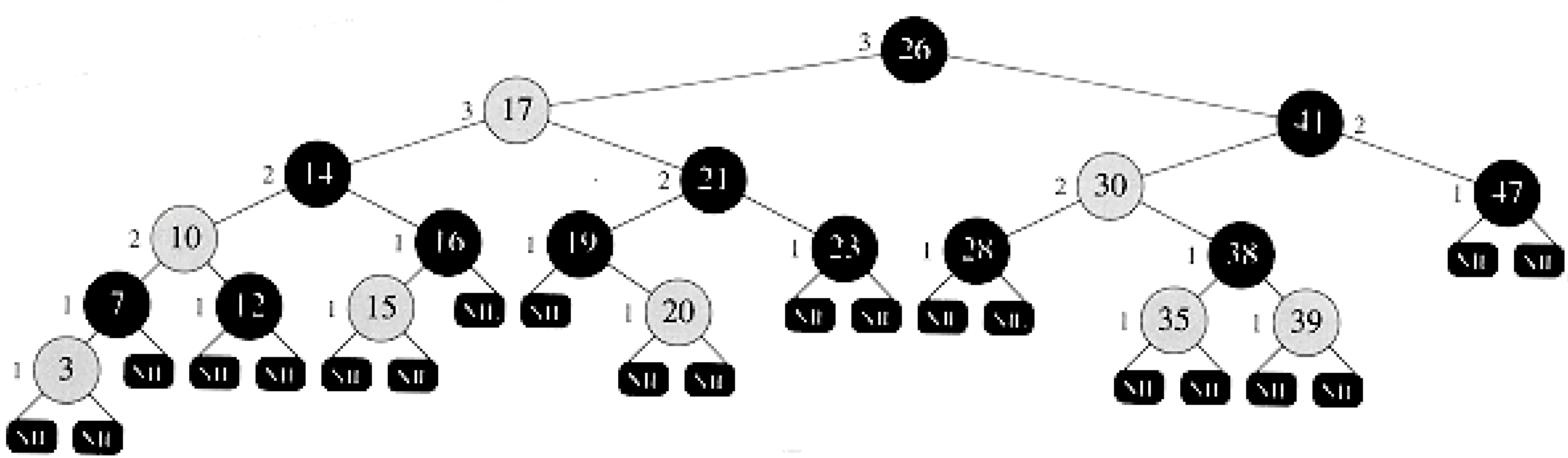
Crveno-crna stabla (Red-Black Trees)

- prilagodba B-stabla smještaju u memoriju računala
  - nema rasipanja memorije, a zadržava se uravnoteženost
- binarno stablo koje izravno proizlazi iz B-stabla 4.-tog reda ako mu se elementi čvorova smatraju obojanima prema strogim pravilima



Definicijska pravila (vidi npr. Cormen, Leiserson and Rivest):

1. svaki čvor je crven ili crn
2. svaki list (čvor koji ne sadrži informaciju!) je crn
3. oba potomka crvenog čvora su crna
4. svaka putanja od nekog čvora do (bilo kojeg) lista koji je njegov potomak prolazi istim brojem crnih čvorova



Listovi u crveno-crnom (RB) stablu ne sadrže informacije pa ne moraju ni postojati, nego roditelji mogu imati NULL pokazivače ili svi pokazivati isti poseban čvor, *sentinel*. Druga varijanta olakšava programiranje, npr. brisanje čvora.
Čvorovi koji nisu listovi nazivaju se *unutarnji* čvorovi.

Razlikujemo tzv. crvenu i crnu visinu stabla (*red and black height*; hr(x) i hb(x)) = broj crvenih (crnih)
čvorova na putu od čvora x (x se ne broji) do lista koji mu je potomak.

Crveno-crno (RB) stablo uravnoteženost nasljeđuje od B-stabla, a definicijska pravila osiguravaju brzinu pretraživanja.

Teorem: Visina RB-stabla s n unutarnjih čvorova je

$$h \le 2 \cdot \log_2(n+1). \quad \square$$

Dokaz: binarno stablo visine h ima najviše $n = 2^h - 1$ čvorova. Zbog 3. pravila, barem polovica visine je crna visina pa je $hb \ge h/2$. Budući da je n veći od broja crnih čvorova na putu od korijena do najnižeg lista, slijedi $n \ge 2^{hb} - 1 \ge 2^{h/2} - 1$, a iz toga izravno $h \le 2 \cdot \log_2(n+1)$. $\blacksquare$

Pretraživanje binarnog stabla je O(h) pa je pretraživanje RB-stabla $O(\log_2 n)$.

- to vrijedi i za dodavanje, odnosno brisanje čvorova
- AVL stabla su strože uravnotežena (niža) pa se RB stablo sporije pretražuje, ali zato lakše održava (brže dodaje/briše čvor)

Definicijska pravila osiguravaju još jedno svojstvo RB-stabala: najduži put od korijena do nekog lista najviše je dvostruko duži od najkraćeg puta od korijena do nekog lista. □

Dokaz: zbog 3. pravila, nijedan put ne može prolaziti uzastopno dvama crvenim čvorovima. Nadalje, najkraći put imao bi samo crne čvorove, dok bi se u najdužem crveni i crni izmjenjivali u svakom koraku. Budući da, zbog 4. pravila, najduži i najkraći put imaju jednak broj crnih čvorova, a najduži može imati još najviše onoliko crvenih čvorova koliko ima crnih, slijedi da najduži put može biti najviše dvostruko dulji od najkraćeg. ■

Dodavanje čvora u RB-stablo:

- radi lakše analize, uvode se pojmovi čvor-ujak (*uncle*) koji se označava s U, a znači *sibling* roditelja promatranog čvora (očev / majčin brat), i čvor-djed koji se označava s G (*grandfather*), a znači roditelj roditelja

1. ubaciti novi kao u svako drugo binarno search-stablo i pridijeliti mu crvenu boju
2. restrukturirati stablo da bi zadovoljavalo definicijska pravila (primjenom rotacija)

Koja pravila i kada se mogu narušiti?

- pravila 1 i 2 će uvijek biti zadovoljena
- pravilo 3: a) dodavanjem crvenog čvora

(Crveni ima crnu djecu.) b) mijenjanjem crni $\rightarrow$ crveni

c) u slučaju rotacije

- pravilo 4: a) dodavanjem crnog čvora

(Crne visine jednake.) b) mijenjanjem crveni $\rightarrow$ crni

c) u slučaju rotacije

Restrukturiranje stabla nakon dodavanja čvora

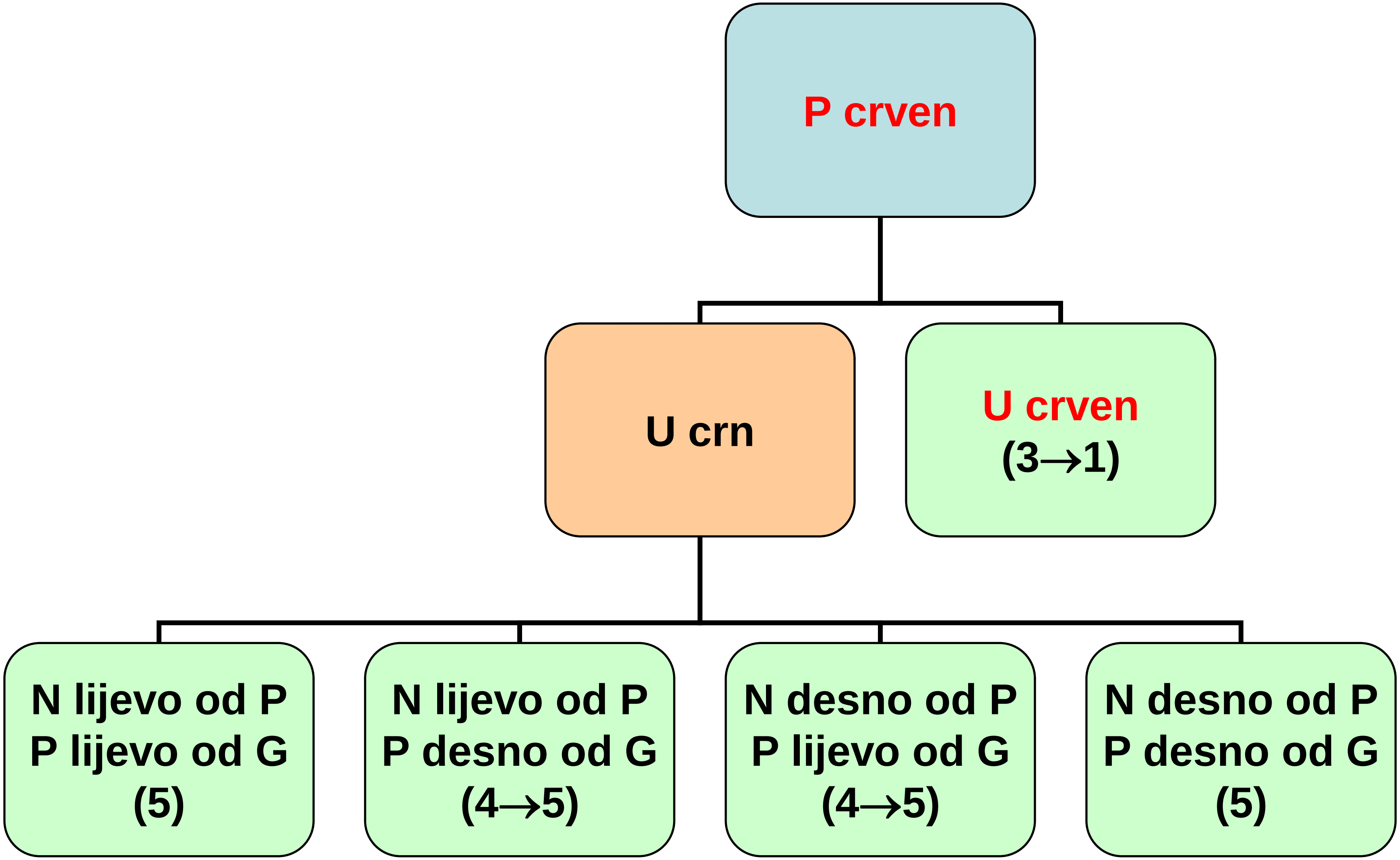
- novi čvor N, roditelj P, djed G, ujak U
- zadovoljenje definicijskih pravila osigurava se s pet provjera redom kako su navedene

1. novi čvor je korijen
   - samo ga prebojati u crno i gotovo;
     - 4. pravilo ostaje zadovoljeno jer je to dodatni crni čvor u svim putevima u stablu

2. roditelj novog čvora je crni čvor
   - budući da je novi crven, sve je ok; gotovo

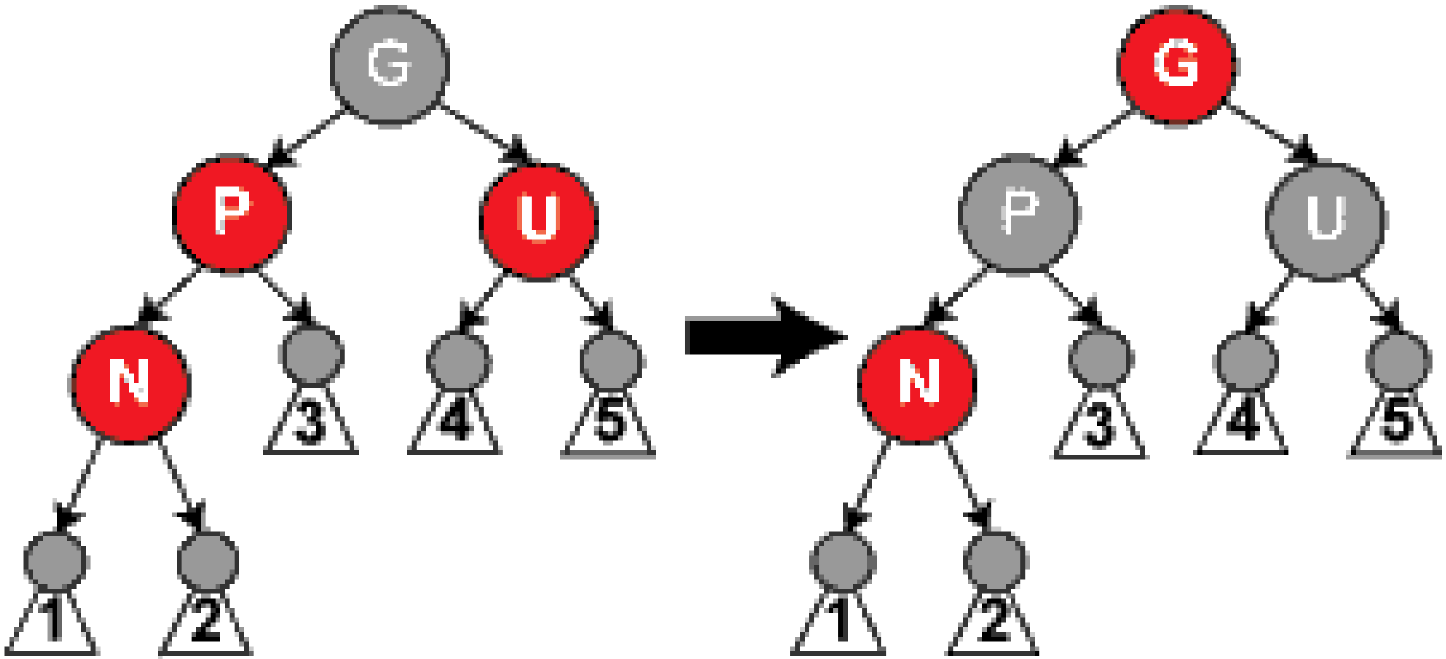
Dalje znamo da je roditelj crven, a budući da nije korijen, postoji i djed.

Ostaje razmotriti situacije u kojima je P crveni čvor:



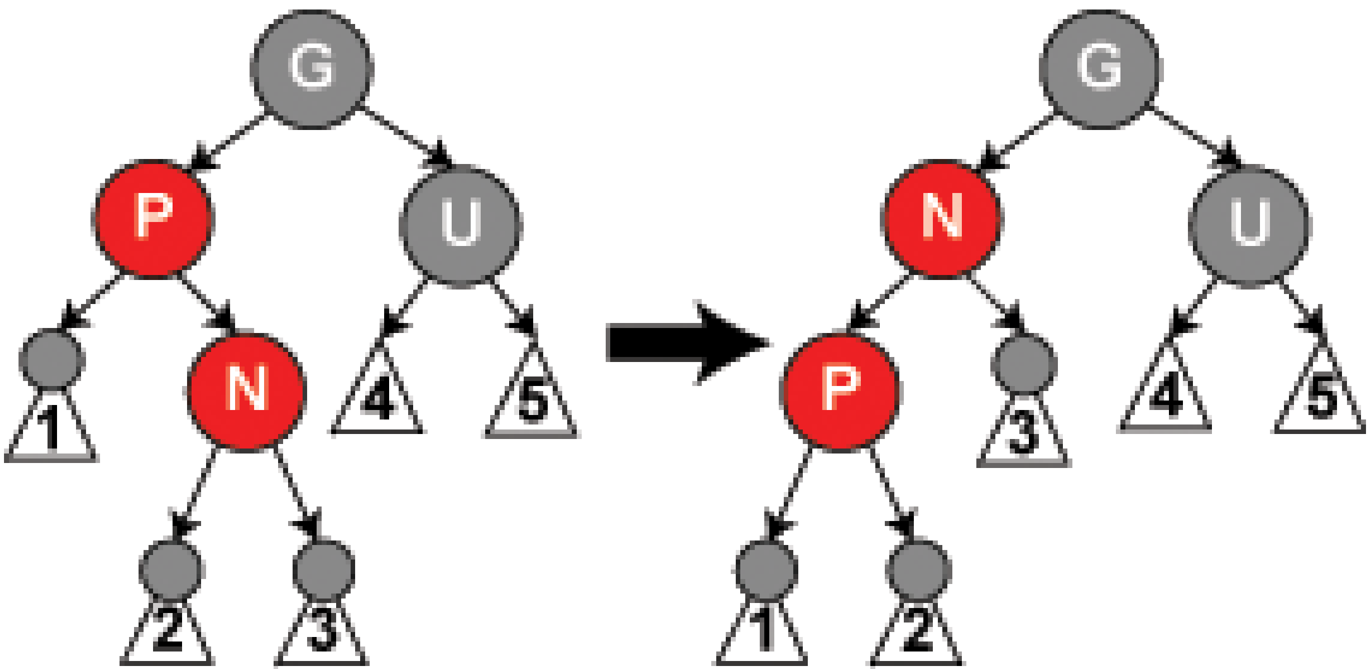
3. roditelj (sigurno) i ujak su crveni

- narušeno 3. pravilo (potomak od crvenog P je crveni N)
- prebojati P i U u crno, a G u crveno (radi očuvanja 4. pravila)
  - sada G može narušavati 3. pravilo ako ima crvenog roditelja; također, ako je G korijen, uobičajeno je crn
- vratiti se na korak 1 promatrajući G kao novi čvor (N)

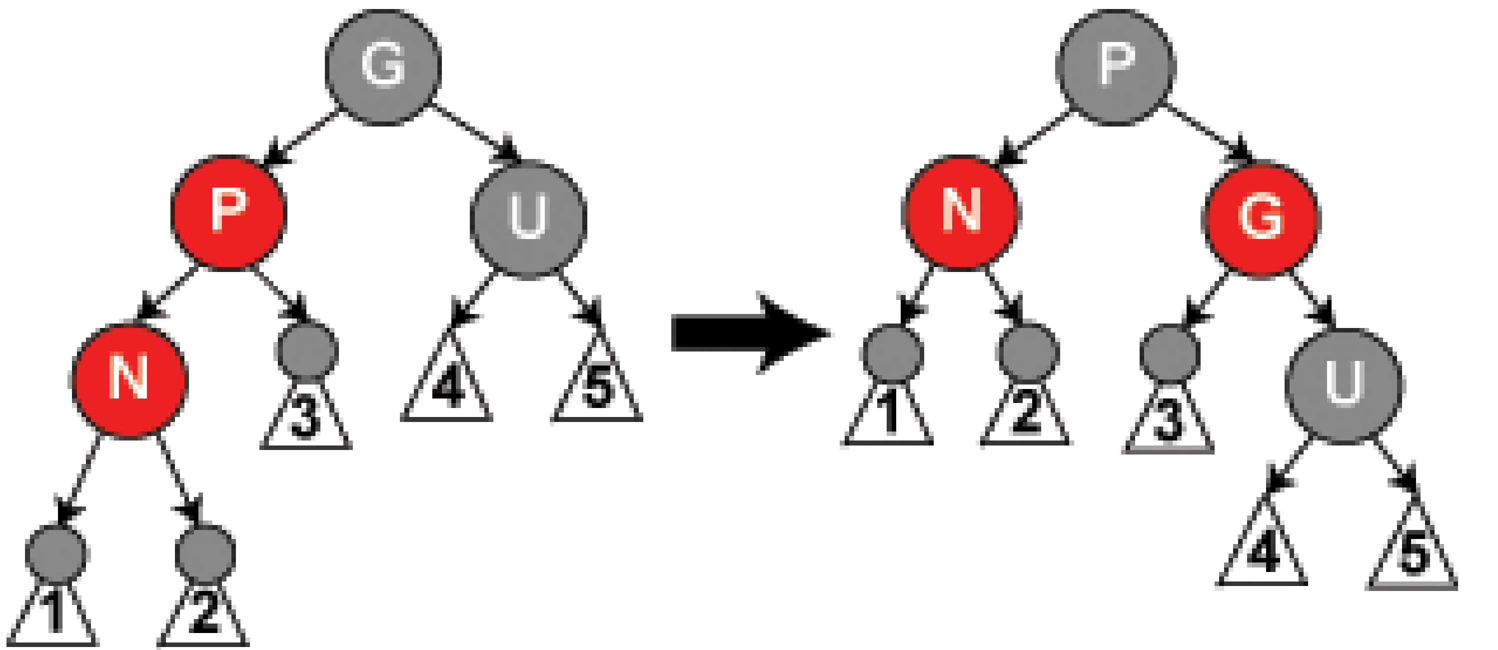


4. roditelj crven i ujak crn (“izlomljeni” poredak)

- idemo li navedenim redom, nakon treće provjere ovo je sigurno pa boje roditelja i ujaka ne treba ni provjeravati
- dva simetrična slučaja: N desno dijete od P i P lijevo dijete od G ili N lijevo dijete od P i P desno dijete od G
  - rotacija N oko P, čime se stanje prevodi u slučaj 5
  - nastaviti sa slučajem 5



5. roditelj crven i ujak crn (linijski poredak)
- idemo li navedenim redom, nakon treće provjere ovo je sigurno pa boje roditelja i ujaka ne treba ni provjeravati
- dva simetrična slučaja: N lijevo dijete od P i P lijevo dijete od G ili N desno dijete od P i P desno dijete od G
  - rotacija P oko G; gotovo



Brisanje čvora u RB-stablu:

- slično kao s B i AVL stablima, brisanje sjedinjenjem ne dolazi u obzir jer bi “razrušilo” cijelo stablo

1. prijepis podataka iz nabližeg prethodnika ili sljedbenika (zamjenski čvor)
2. ukloniti zamjenski čvor; on može imati najviše jedno dijete pa je problem pojednostavnjen (vidi bilješke!)

Dvije situacije su jednostavne:

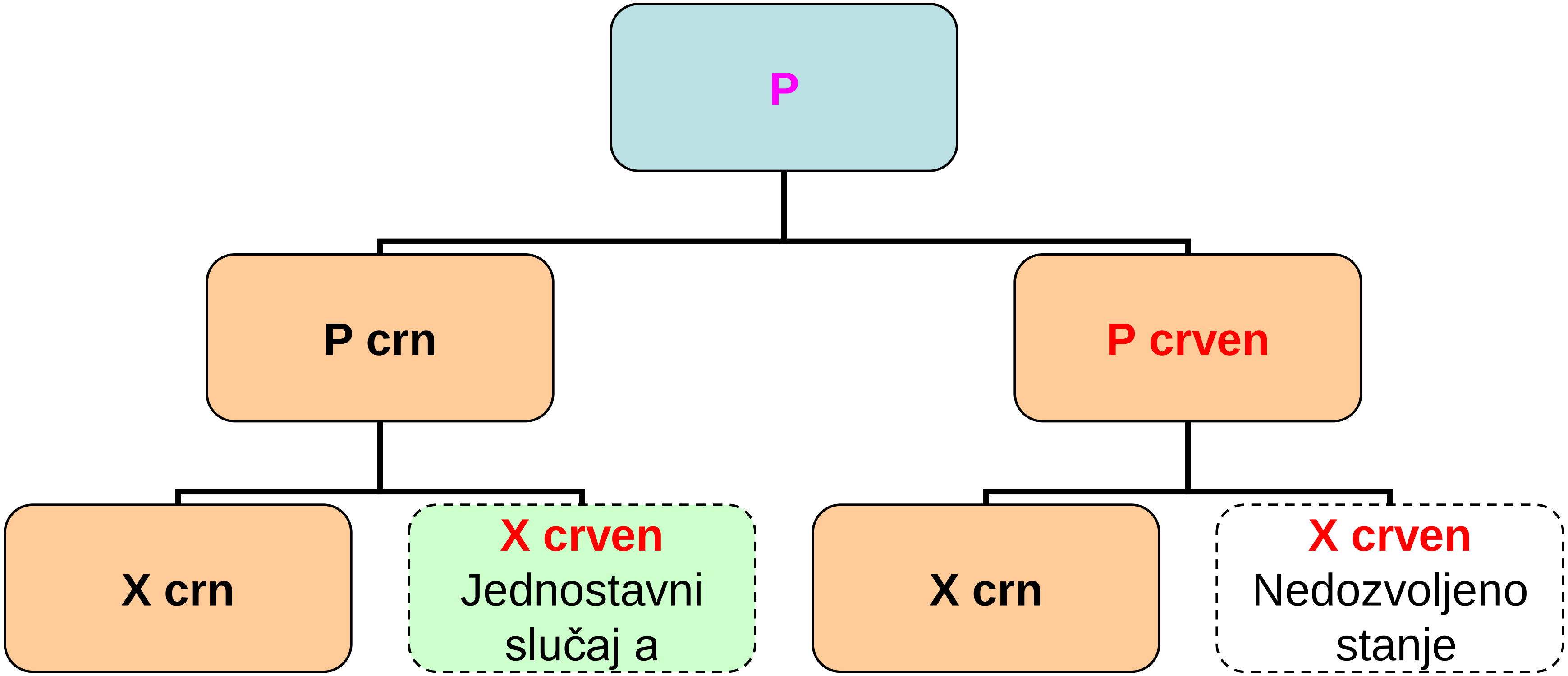
- označimo zamjenski čvor (koji se uklanja) s X

a) čvor X je crven
   - njegovo dijete, ako ga uopće ima, može biti samo crno
   - ubaciti dijete u stablo umjesto X; kraj; time se ne mijenja crna visina i ne može prekršiti nijedno definicijsko pravilo

b) čvor X je crn, a dijete crveno
   - ubaciti dijete u stablo umjesto X i prebojati ga u crno; kraj; crni čvor ostaje gdje je i bio pa se crna visina ne mijenja

Kratko razmišljanje - osjetno pojednostavnjenje!

- dvije jednostavne situacije i definicijska pravila znatno umanjuju broj slučajeva koje treba riješti
- u nastavku: čvor koji se uklanja X, dijete čvora koji se uklanja (već je na mjestu X-a) N, roditelj (od X) P, sibling od N (drugo dijete roditelja P) S, lijevo dijete siblinga SL, desno dijete siblinga SR (sibling ne može biti list!)



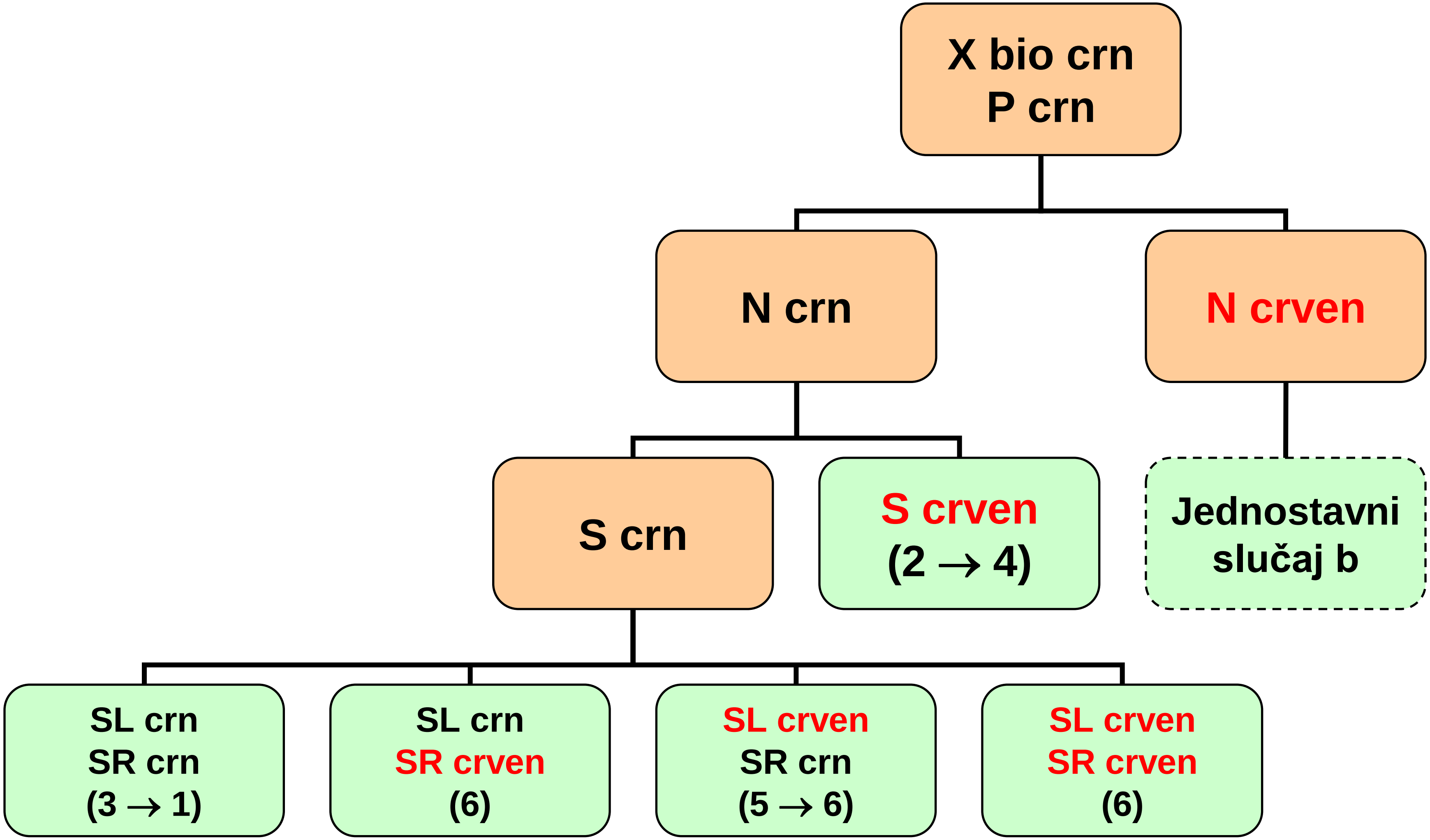
Zaključak: složenije situacije mogu nastati samo kada je X crni čvor.

```
DeleteRBNode (node)
copy content from X (the closest predecessor or successor
                                        of node) to node;
child = X's child;     //child must exist, at least as sentinel
replace X by child;    //incorporate child into the tree at
                       //the place of X; resolves the simple case a)
if (X->colour == BLACK)
    if (child->colour == RED)
        child->color = BLACK;    //simple case b)
    else
        RBProcessing(child);     //non-simple case
free(X);
```

Na slajdovima se `child`, koji je već na mjestu `X`, označava s N!

Znamo da je X (bio) crni čvor!

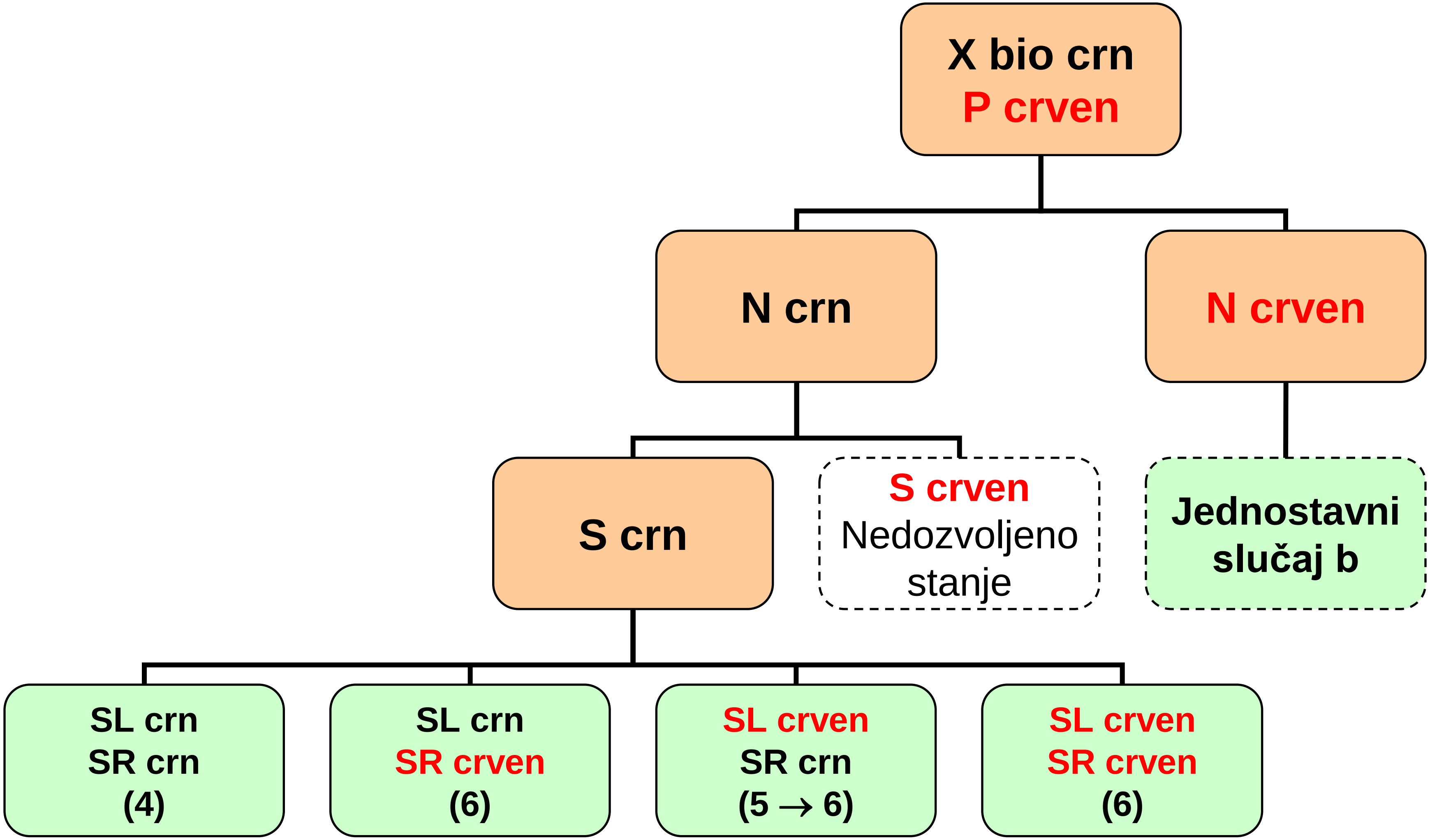
- N je već na mjestu X-a



Zaključak: složenije situacije mogu nastati samo kada je N crni čvor.

Znamo da je X (bio) crni čvor!

- N je već na mjestu X-a



Zaključak: složenije situacije mogu nastati samo kada je N crni čvor.

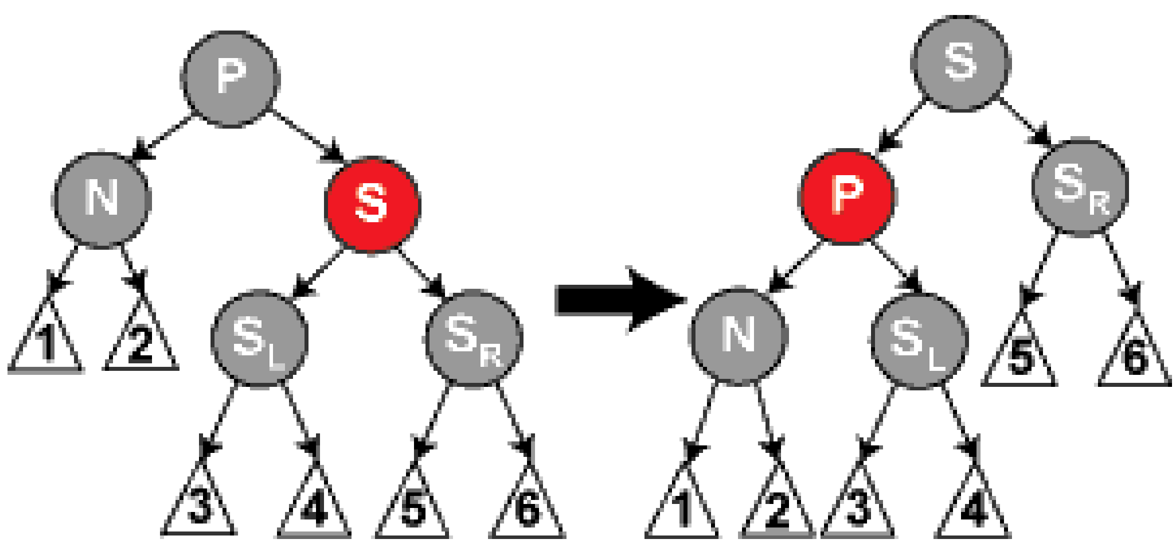
Uklanjanje zamjenskog čvora:

- s obzirom na dvije jednostavne situacije, brisanje zamjenskog čvora bezuvjetno započinje ubacivanjem djeteta na mjesto X (pokazivač u P usmjeriti na dijete - rješava situaciju *a*)
- ako je dijete N crveno, prebojati ga u crno (situacija *b*)
- inače problem: svi putevi koji su prolazili čvorom X sada imaju jedan crni čvor manje $\Rightarrow$ podstablo kojemu je N korijen ima manju crnu visinu od podstabla koje započinje S-om
- sve ovisi o tome što je u okolini od N; šest je slučajeva koje treba riješiti, pri čemu se redoslijed provjera mora poštivati

1. čvor N je novi korijen
   - gotovo; X je bio crni korijen pa je cijelom stablu crna visina umanjena za jedan i RB pravila su zadovoljena

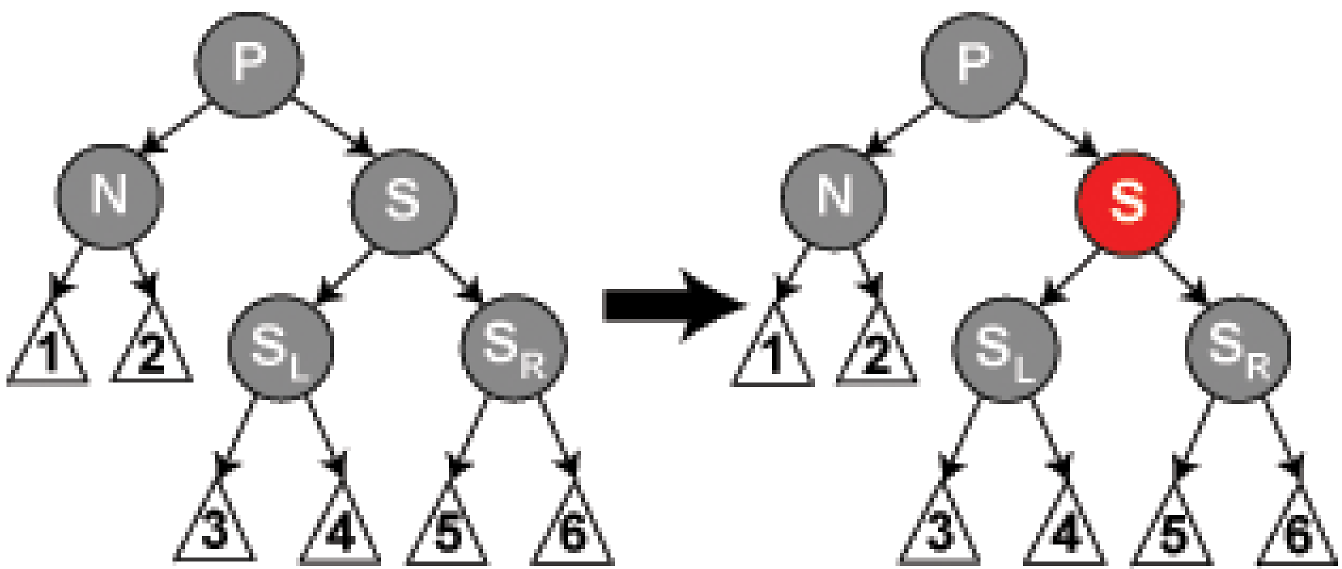
U primjerima za slučajeve 2, 5 i 6 pretpostavlja se da je N lijevo dijete od P, ali u realizaciji treba riješiti i simetrične sitacije.

2. sibling S je crven
   - P je sigurno crn jer ima crveno dijete
   - X i N su bili crni ⇒ nakon brisanja X crna visina lijevog podstabla od P za jedan manja od crne visine desnog
   - zamijeniti boje P i S pa rotirati S oko P (simetrija!)
     - gledano odozgo, iz ostatka stabla, putevi od S na niže imaju imaju isti broj crnih čvorova kao i prije
     - N sada sigurno ima crnog siblinga SL (jer je SL bio dijete crvenog čvora) i crvenog roditelja P → slučajevi 4 ili 5 i 6 (moramo dalje jer crne visine lijevo i desno od P nisu iste)

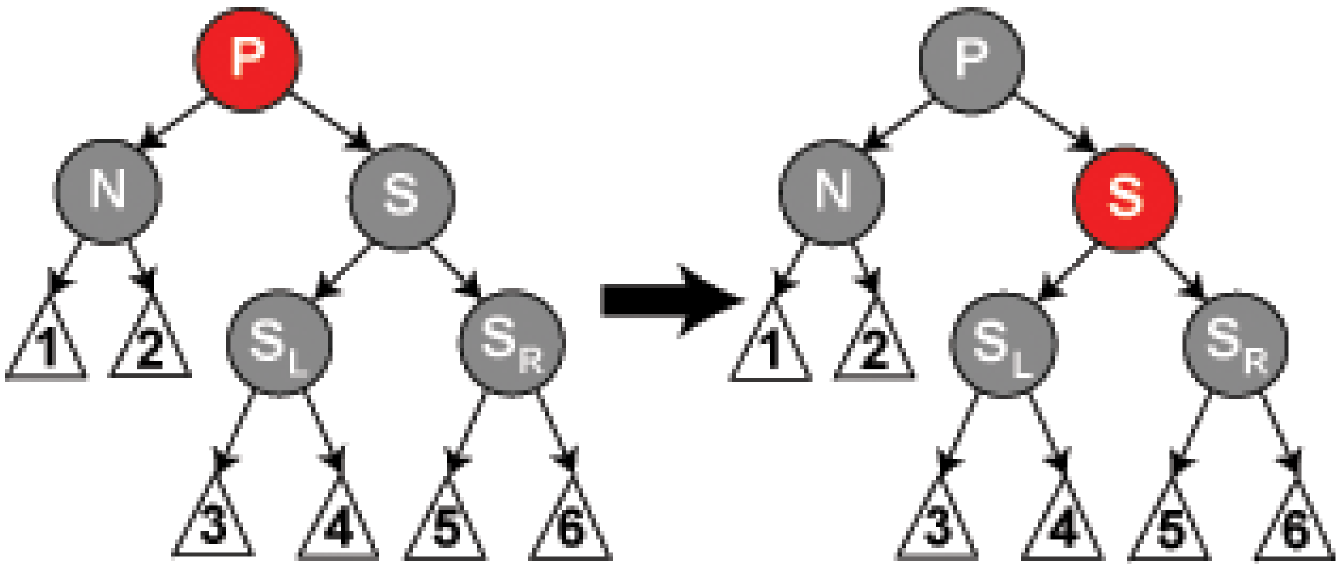


3. čvor P crn, S crn, djeca od S crna

- prebojati S u crveno
- svi putevi preko S gube jedan crni čvor, a to su upravo putevi koji ne prolaze čvorom N; brisanje X je smanjilo broj crnih čvorova u podstablu koje započinje s N za jedan pa se ovime crne visine N i S izjednačavaju
- sada svi putevi kroz P imaju jedan crni čvor manje od onih koji ne prolaze P-om → nazad na slučaj 1 s P kao N jer sada je P taj čije je podstablo niže (crna visina)

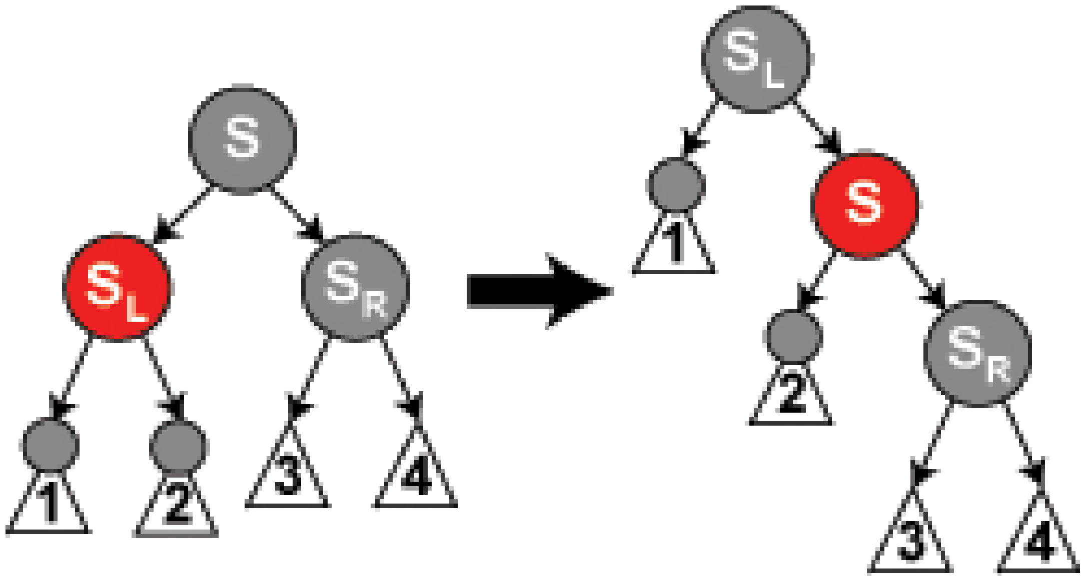


4. čvor S i djeca od S crna, a P crven
   - zamijeniti boje S i P
   - broj crnih čvorova desne grane ostaje isti, a lijevoj se povećava za jedan i time poništava gubitak u N uslijed brisanja X
   - kraj resturukturiranja



5. čvor S crn, SL crven, SR crn, a P nevažan

- S je N-ov sibling, a N je lijevo dijete od P (simetrija!)
- rotirati SL i S, tako da SL postane N-ov sibling, i potom im zamijeniti boje (S-u i SL-u)
- svi putevi i dalje imaju iste crne visine, ali sada N ima crnog siblinga kojemu je desno dijete crveno → slučaj 6
  - u primjeru 6, SL je preimenovan u S kao N-ov sibling



6. čvor S crn, SR crven, a P i SL nevažani
    - rotirati S oko P (simetrija!)
    - potom zamijeniti boje S i P, a SR prebojati u crno → kraj
        - korijen ostaje iste boje - dobro za roditelja korijena
        - gledano od P (kasnije S), putevi u podstabla 3, 4 i 5 zadržavaju isti broj crnih čvorova, a oni preko N dobivaju jedan novi i time kompenziraju raniji gubitak u N (P je ili bio crven pa postao crn zamjenom boje sa S ili je bio crn i prije)