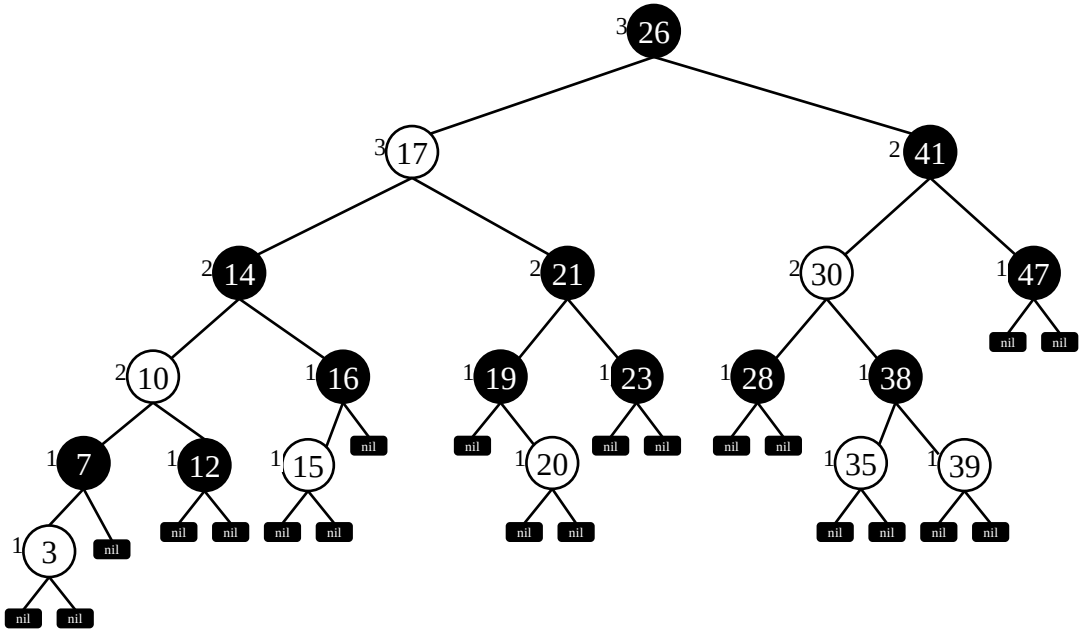
Crveno-crna binarna stabla

SPA2

Osobine crveno-crnog stabla

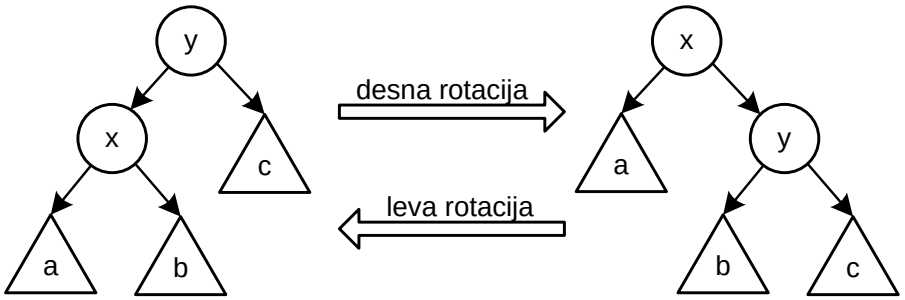
- Svaki čvor je crven ili crn
- Koren stabla je crn
- Svaki list (NIL) je crn
- Ako je čvor crven on ima dva crna naslednika
- Za svaki čvor važi da put od tog čvora do svakog lista naslednika ima jednak broj crnih čvorova (crna visina čvora)



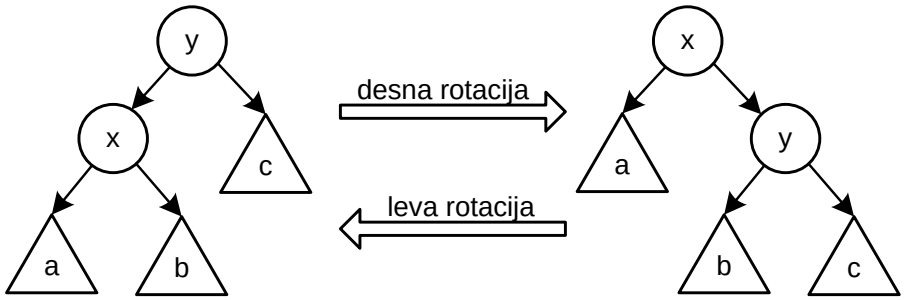
```
#include <stdio.h>
#include <stdlib.h>

enum {CRVENA,CRNA};

struct drvo{
   int broj,boja;
   struct drvo *roditelj,*levi,*desni;
};

#define novi(x) x=(struct drvo *) malloc(sizeof(struct drvo))

void dodaj(struct drvo **,struct drvo*);
void bojefix(struct drvo **,struct drvo *);
struct drvo* form();
void ispis(struct drvo*);
void lrotacija(struct drvo**);
void drotacija(struct drvo**);
```

```
main(){
    struct drvo *p,*q;int k;
    p=form();
    ispis(p); printf("\n");
}

struct drvo* form(){
  struct drvo *koren,*cvor;   int k;
  koren=NULL;
  scanf("%d",&k);
  while(k) {
       novi(cvor);
       cvor->broj=k;   cvor->boja=CRVENA;
       cvor->levi=cvor->desni=NULL;
       dodaj(&koren,cvor);
       scanf("%d",&k);
  }
  return koren;
}
```

```
void lrotacija(struct drvo **t){
  struct drvo *x,*y;
  x = *t;
  y = x->desni;
  x->desni = y->levi;
  if (y->levi) y->levi->roditelj=x;
  y->roditelj=x->roditelj;
  if (x->roditelj)
       if (x==x->roditelj->levi) x->roditelj->levi=y;
       else x->roditelj->desni=y;
  y->levi=x;
  x->roditelj=y;
  *t=y;
}
```

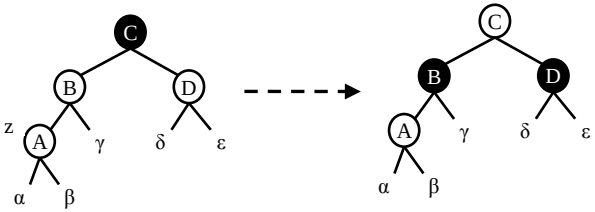


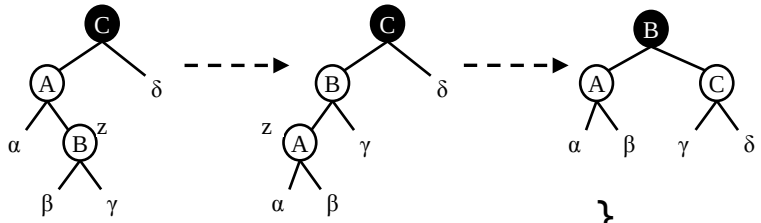
```
void drotacija(struct drvo **t){
  struct drvo *x,*y;
  y = *t;
  x=y->levi;
  y->levi=x->desni;
  if(x->desni) x->desni->roditelj=y;
  x->roditelj=y->roditelj;
  if(y->roditelj)
       if(y==y->roditelj->levi) y->roditelj->levi=x;
       else y->roditelj->desni=x;
  x->desni=y;
  y->roditelj=x;
  *t=x;
}
```



```
void dodaj(struct drvo **p, struct drvo *z){
  struct drvo *y,*x;
  x=*p;
  y=NULL;
  while(x){
       y=x;
       if (z->broj<x->broj) x=x->levi;
       else x=x->desni;
  }
  z->roditelj=y;
  if(!y) *p=z;
  else{
       if (z->broj<y->broj) y->levi=z;
       else y->desni=z;
  }
  bojefix(p,z);
}
```

```
void bojefix(struct drvo **p, struct drvo *z){
   struct drvo *y,*t;
   while(z->roditelj->boja==CRVENA){
         if (z->roditelj==z->roditelj->roditelj->levi) {
                y=z->roditelj->roditelj->desni;
                if (y->boja==CRVENA) {                          //
                                                                //
                      z->roditelj->boja=CRNA;                   //
                      y->boja=CRNA;                             // I slucaj
                      z->roditelj->roditelj->boja=CRVENA;       //
                      z=z->roditelj->roditelj;                  //
                }
                else {
                      if (z==z->roditelj->desni){               //
                            z=z->roditelj;                      //
                            lrotacija(&z);                      // II slucaj
                            if(!z->roditelj) *p=z;              //
                            z=z->levi;                          //
                      }
                      z->roditelj->boja=CRNA;                   //
                      z->roditelj->roditelj->boja=CRVENA;       //
                      t=z->roditelj->roditelj;                  // III slucaj
                      drotacija(&t);                            //
                      if(!t->roditelj) *p=t;                    //
                }
         }
```





```
        else{
                y=z->roditelj->roditelj->levi;
                if (y->boja==CRVENA) {                          //
                        z->roditelj->boja=CRNA;                 //
                        y->boja=CRNA;                           // I' slucaj
                        z->roditelj->roditelj->boja=CRVENA;     //
                        z=z->roditelj->roditelj;                //
                }
                else {
                        if (z==z->roditelj->levi){              //
                                z=z->roditelj;                  //
                                drotacija(&z);                  // II' slucaj
                                if(!z->roditelj) *p=z;          //
                                z=z->desni;                     //
                        }
                        z->roditelj->boja=CRNA;                 //
                        z->roditelj->roditelj->boja=CRVENA;     // III' slucaj
                        t=z->roditelj->roditelj;                //
                        lrotacija(&t);                          //
                        if(!t->roditelj) *p=t;                  //
                }
        }
}
(*p)->boja=CRNA;
}
```

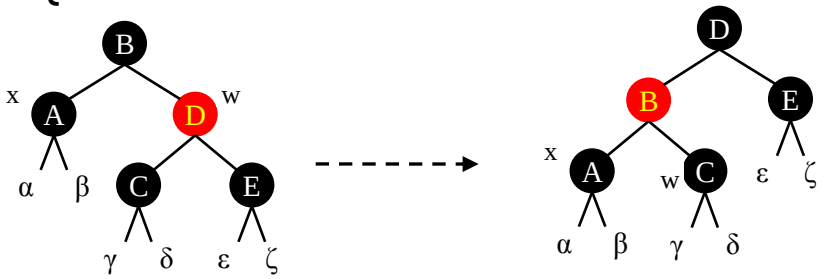
Brisanje čvora iz crveno-crnog binarnog stabla

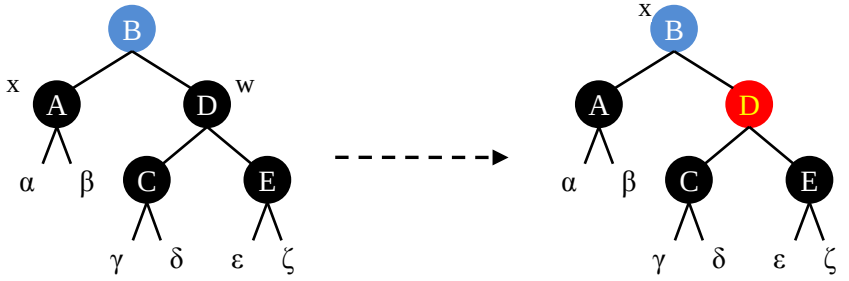
SPA2

```
void obrisi(struct drvo **p,int k){
   struct drvo *x;
   x=pronadji(*p,k);
   if (x) obrisi_cc(p,x);
}

struct drvo* pronadji(struct drvo *p,int k){
   struct drvo *t=NULL;
   if (p) {
      if (p->broj==k) return p;
      if (p->broj>=k) t=pronadji(p->levi,k);
      else t=pronadji(p->desni,k);
   }
   return t;
}

struct drvo* naslednik(struct drvo *x){
   struct drvo *y;
   if(x->desni) {
      y=x->desni;
      while (y->levi) y=y->levi;
   }
   return y;
}
```

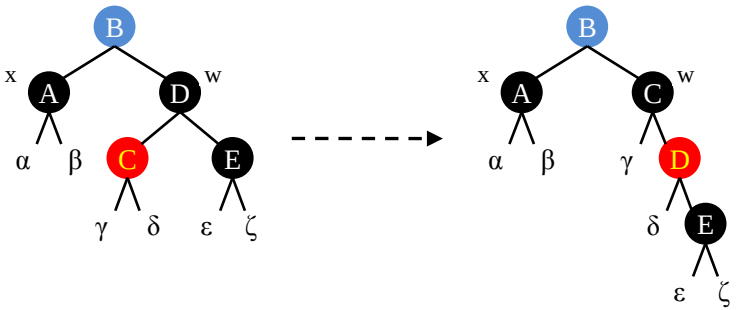
```
void obrisi_cc(struct drvo **p,struct drvo *z){
   struct drvo *x,*y;
   if((!z->levi) || (!z->desni)) y=z;
   else y=naslednik(z);
   if (y->levi) x=y->levi;
   else x=y->desni;
   if (x) x->roditelj=y->roditelj;
   if (!y->roditelj) *p=x;
   else {
      if (y==y->roditelj->levi) y->roditelj->levi=x;
      else y->roditelj->desni=x;
   }
   if(y!=z) z->broj=y->broj;
   if ((x) && (y->boja==CRNA)) BojaDelFix(p,x);
}
```

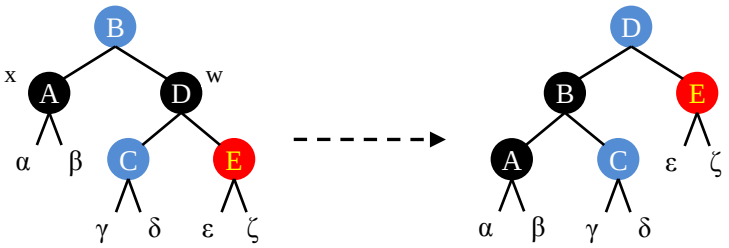
```
void BojaDelFix(struct drvo **p, struct drvo *x){
   struct drvo *w,*t;
   while ((x!=(*p)) && (x->boja==CRNA)) {
      if (x==x->roditelj->levi) {
         w=x->roditelj->desni;
         if ((w) && (x->boja==CRVENA)) {
            w->boja=CRNA;
            x->roditelj->boja=CRVENA;
            t=x->roditelj;
            lrotacija(&t);
            if(!t->roditelj) *p=t;
            w=x->roditelj->desni;
         }
         if ((w->levi) && (w->desni)) {
            if((w->levi->boja==CRNA) && (w->desni->boja==CRNA)){
               w->boja=CRVENA;
               x=x->roditelj;
            }
...
```





```
...
        else {                          //if((w->levi->boja==CRNA) && (w->desni->boja==CRNA))
            if (w->desni->boja==CRNA) {
               w->levi->boja=CRNA;
               w->boja=CRVENA;
               drotacija(&w);
               if(!w->roditelj) *p=w;
               w=x->roditelj->desni;
            }
            w->boja=x->roditelj->boja;
            x->roditelj->boja=CRNA;
            w->desni->boja=CRNA;
            t=x->roditelj;
            drotacija(&t);
            if(!t->roditelj) *p=t;
            x=*p;
         }
      }
   }
...
```





```
...
      else{                                          // if (x==x->roditelj->levi)
         w=x->roditelj->levi;
         if ((w) && (x->boja==CRVENA)) {
            w->boja=CRNA;
            x->roditelj->boja=CRVENA;
            t=x->roditelj;
            drotacija(&t);
            if(!t->roditelj) *p=t;
            w=x->roditelj->levi;
         }
         if ((w->levi) && (w->desni)) {
            if((w->levi->boja==CRNA) && (w->desni->boja==CRNA)){
               w->boja=CRVENA;
               x=x->roditelj;
            }
...
```

```
            else {                          //if((w->levi->boja==CRNA) && (w->desni->boja==CRNA))
               if (w->levi->boja==CRNA) {
                  w->desni->boja=CRNA;  w->boja=CRVENA;
                  lrotacija(&w);
                  if(!w->roditelj) *p=w;
                  w=x->roditelj->levi;
               }
               w->boja=x->roditelj->boja;
               x->roditelj->boja=CRNA;  w->levi->boja=CRNA;
               t=x->roditelj;
               lrotacija(&t);
               if(!t->roditelj) *p=t;
               x=*p;
            }
         }
      }
   }
   x->boja=CRNA;
}
```