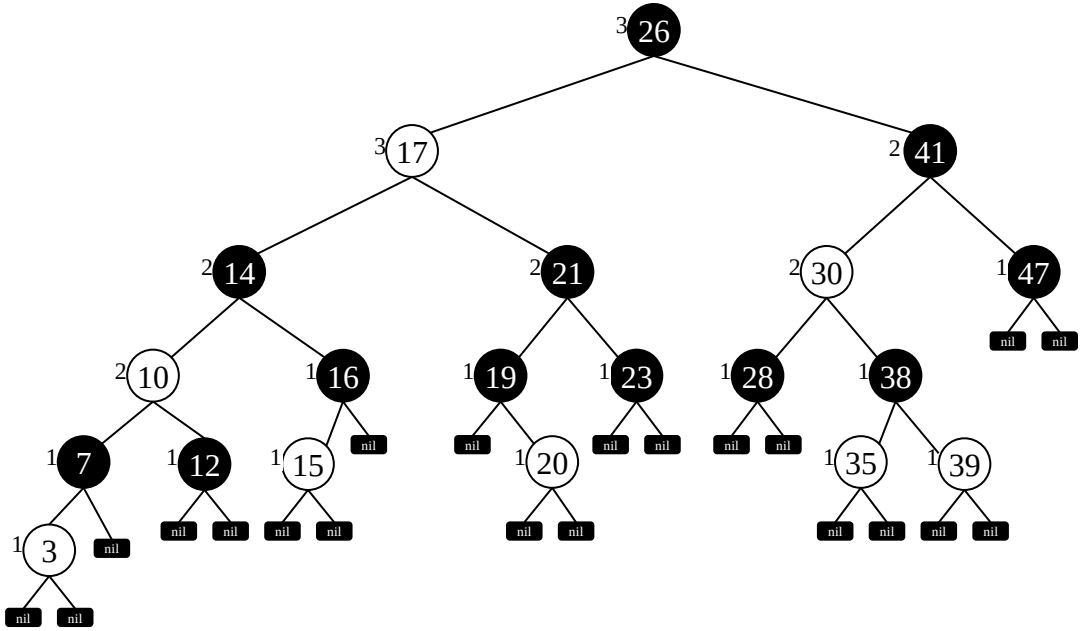
Crveno-crna binarna stabla

SPA2

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Osobine crveno-crnog stabla

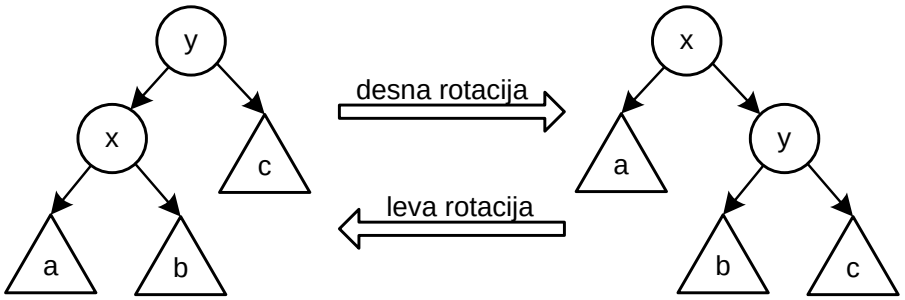
- Svaki čvor je crven ili crn
- Koren stabla je crn
- Svaki list (NIL) je crn
- Ako je čvor crven on ima dva crna naslednika
- Za svaki čvor važi da put od tog čvora do svakog lista naslednika ima jednak broj crnih čvorova (crna visina čvora)



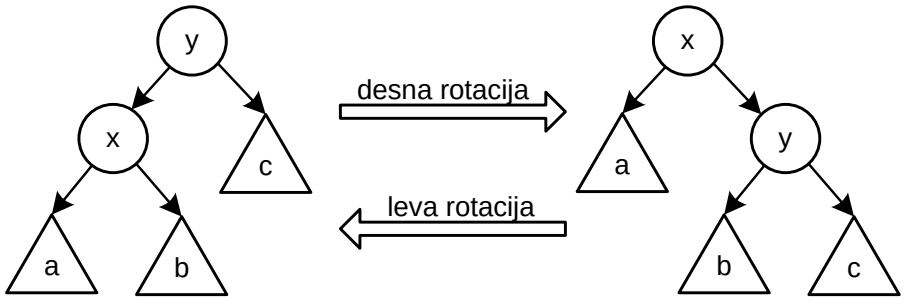
```
#include <stdio.h>
#include <stdlib.h>

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;
}
```

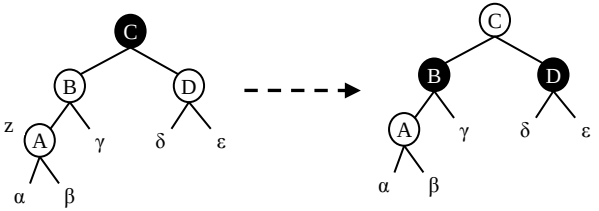


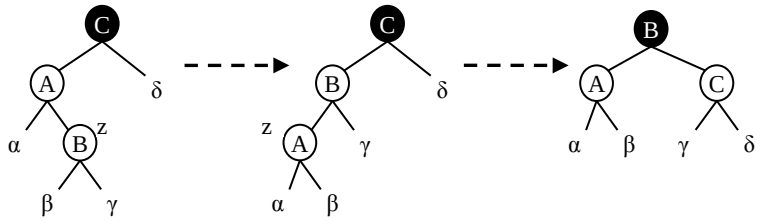
```
void drotacija(struct drvo **t){
   struct drvo *x,*y;
   y = *t;
   x=y->levi;
   y->levi=x->desni;
   if(x->desni) x->desni->roditelj=y;
   x->roditelj=y->roditelj;
   if(y->roditelj)
        if(y==y->roditelj->levi) y->roditelj->levi=x;
        else y->roditelj->desni=x;
   x->desni=y;
   y->roditelj=x;
   *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) && (z->roditelj->boja==CRVENA)){
        if (z->roditelj==z->roditelj->roditelj->levi) {
              y=z->roditelj->roditelj->desni;
              if ((y) && (y->boja==CRVENA)) {
                    z->roditelj->boja=CRNA;
                    y->boja=CRNA;
                    z->roditelj->roditelj->boja=CRVENA;
                    z=z->roditelj->roditelj;
                   }
                 else {
                      if (z==z->roditelj->desni){
                            z=z->roditelj;
                            lrotacija(&z);
                            if(!z->roditelj) *p=z;
                            z=z->levi;
                            }
                       z->roditelj->boja=CRNA;
                       z->roditelj->roditelj->boja=CRVENA;
                       t=z->roditelj->roditelj;
                       drotacija(&t);
                       if(!t->roditelj) *p=t;
                      }
            }
```





```
        else{
                y=z->roditelj->roditelj->levi;
                if ((y) && (y->boja==CRVENA)) {
                        z->roditelj->boja=CRNA;
                        y->boja=CRNA;
                        z->roditelj->roditelj->boja=CRVENA;
                        z=z->roditelj->roditelj;
                          }
                else {
                        if (z==z->roditelj->levi){
                                z=z->roditelj;
                                drotacija(&z);
                                if(!z->roditelj) *p=z;
                                z=z->desni;
                                   }
                        z->roditelj->boja=CRNA;
                        z->roditelj->roditelj->boja=CRVENA;
                        t=z->roditelj->roditelj;
                        lrotacija(&t);
                        if(!t->roditelj) *p=t;
                    }
            }
    }
    (*p)->boja=CRNA;
}
```

Sve dok je roditelj od novog cvora razlicit od NULL i dok je CRVEN radi

{ 1. ako je roditelj levo dete svom roditelju

1.1 ako je ujka (brat mom roditelju) razlicit od NULL i ako je CRVEN

{ stavi roditelja i ujku na CRNO a dedu na CRVENO i trenutni pokazivac prelazi na dedu.

}

1.2 ako nemamo ujku ili je CRN

{ ako je dete desni roditelju

{

trenutni pokazivac postavi na roditelja i rotiraj u levo i ako je posle rotacije njegov roditelj NULL, on postaje KOREN i trenutni prelazi na levo dete

}

stavi roditelja na CRNO, dedu na CRVENO, rotiraj dedu u desno, ako je roditelj od dede NULL, stavi KOREN na dedu

}

}

2. Suprotno od ovog... :)}

Brisanje čvora iz crveno-crnog binarnog stabla

SPA2

```
void brisi(struct drvo **p,struct drvo *q, int x)
 {
  struct drvo *pom;
  if (q!=NULL)
  if (q->broj != x)
   {
     novi(pom);
     pom->broj=q->broj;
     pom->boja=CRVENA;
     pom->roditelj=NULL;
     pom->levi=NULL;
     pom->desni=NULL;
     dodaj(p,pom);
     brisi(p,q->levi,x);
     brisi (p,q->desni,x);
   }
  else {brisi(p,q->levi,x); brisi (p,q->desni,x); }
  else return;
 }
```

```
void ispis(struct drvo *Koren)
 {
  if(Koren != NULL)
   {
       ispis(Koren->levi);
       printf("%d, ",Koren->broj);
       if(Koren->boja == CRVENA)
       printf("(CRVENA)");
       else
       printf("(CRNA)");
       ispis(Koren->desni);
   }
  putchar('\n');
 }
```

```
main(){
    struct drvo *p,*q, *b=NULL;int k,x;
    p=form();
    ispis(p); printf("\n");
    printf("Unesi sta se brise:");
    scanf("%d",&x);
    brisi(&b,p,x);
    ispis(b);

}
```