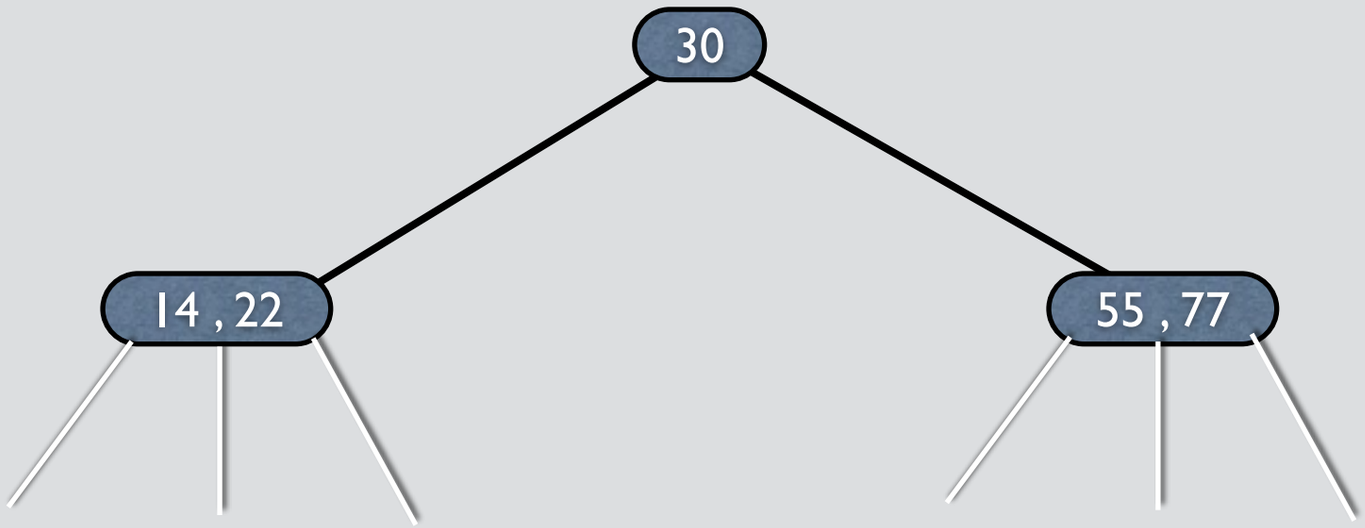
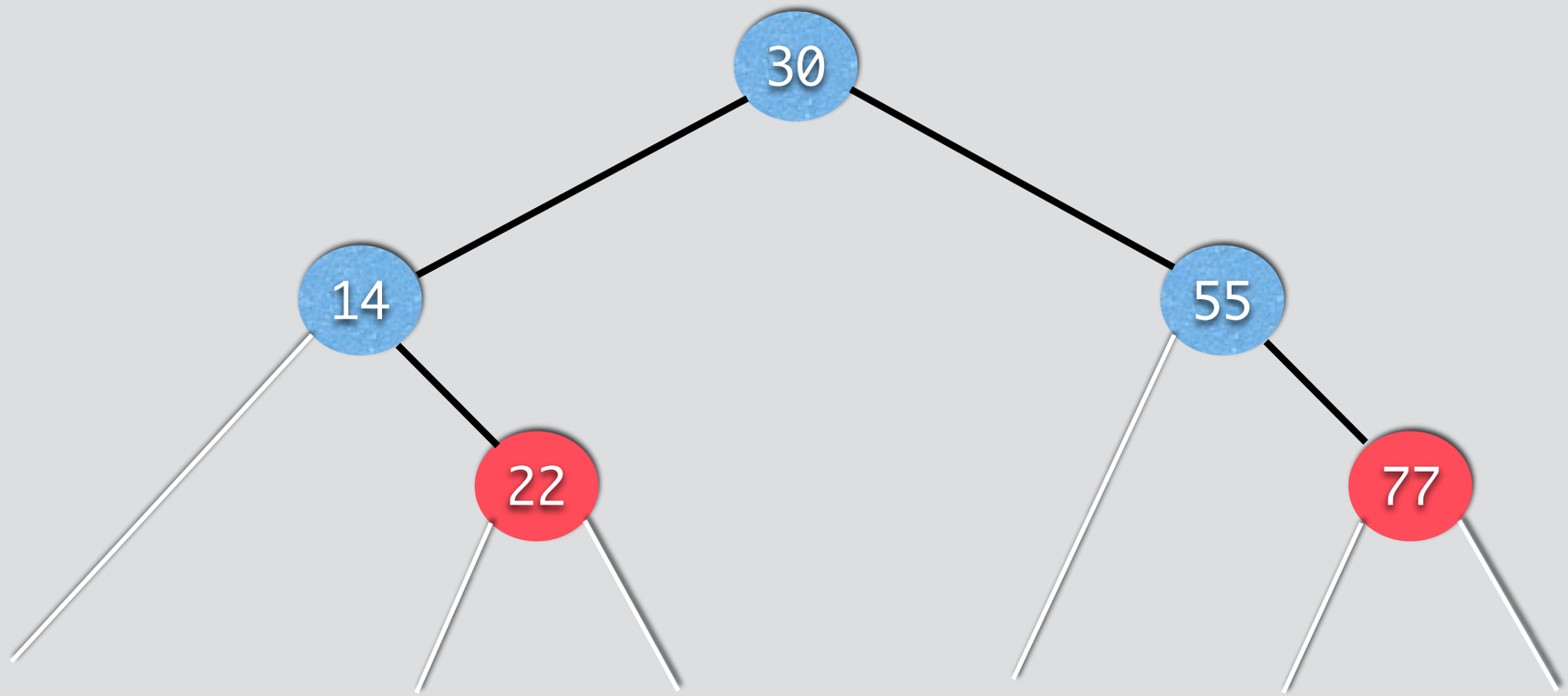
Piros-Fekete fák

Példa

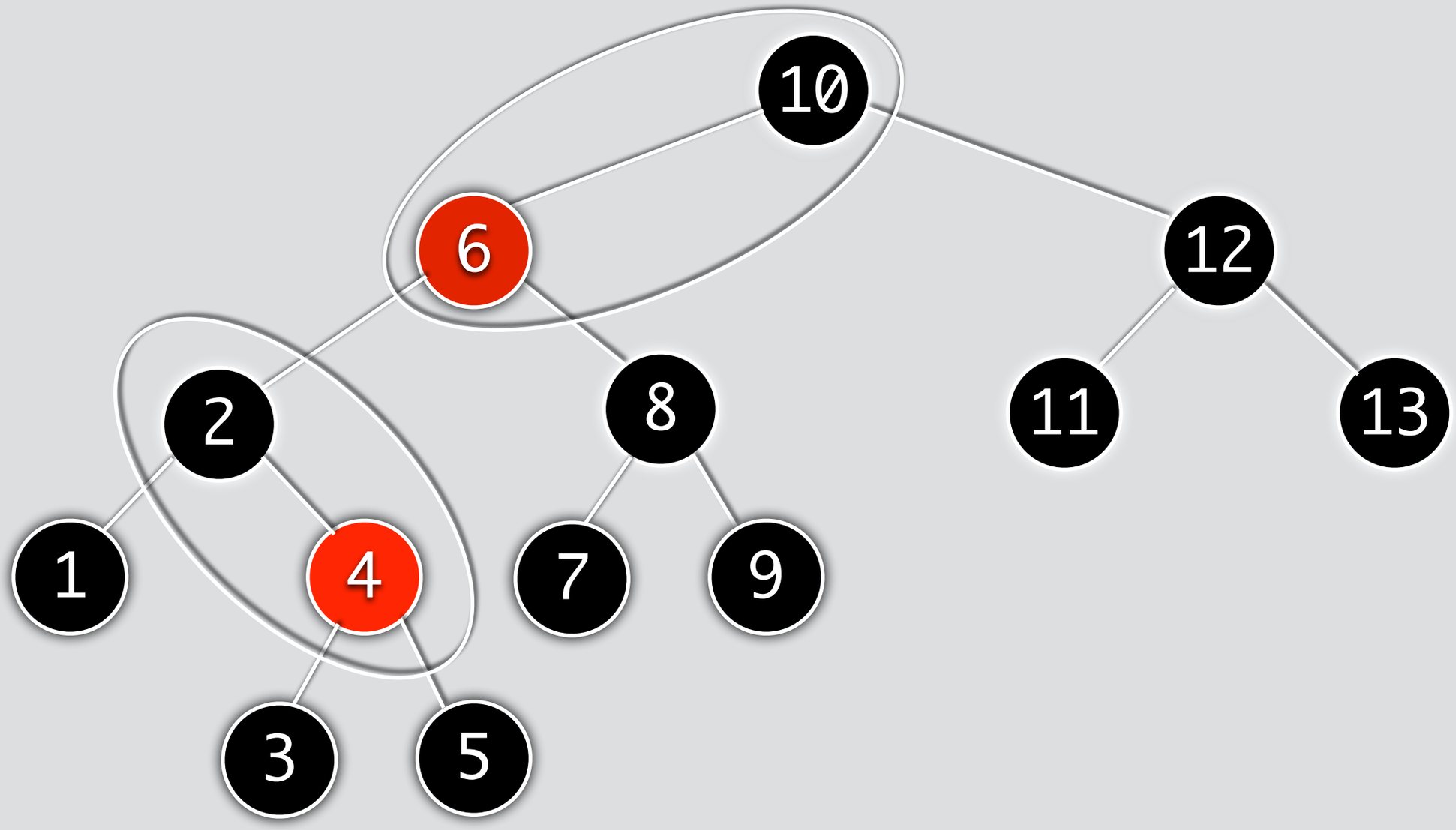




Piros-Fekete fák

A piros-fekete fa olyan bináris keresőfa, amelynek minden pontja egy extra bit információt tartalmaz, ez a pont színe, amelynek értékei: **PIROS** vagy **FEKETE**.

Tehát a fa minden pontja tartalmazza a szín, kulcs, bal, jobb és apa mezőket. Ha egy ponthoz tartozó fiú vagy apa nem létezik, akkor a megfelelo mező a NIL értéket tartalmazza. Úgy tekintjük, hogy az ilyen NIL mutató értékek a bináris keresőfa külső (levél) pontjaira mutatnak, míg a fa kulcsot tartalmazó pontjai a belső pontok. Megvalósítás során ezeket a külső pontokat egyetlen őrszem ponttal ábrázoljuk, amelyet NIL jelöl.


10
6
12
2
8
11
13
1
4
7
9
3
5

Piros-Fekete fák

A piros-fekete fák azok, amelyekre teljesülnek a következő tulajdonságok:

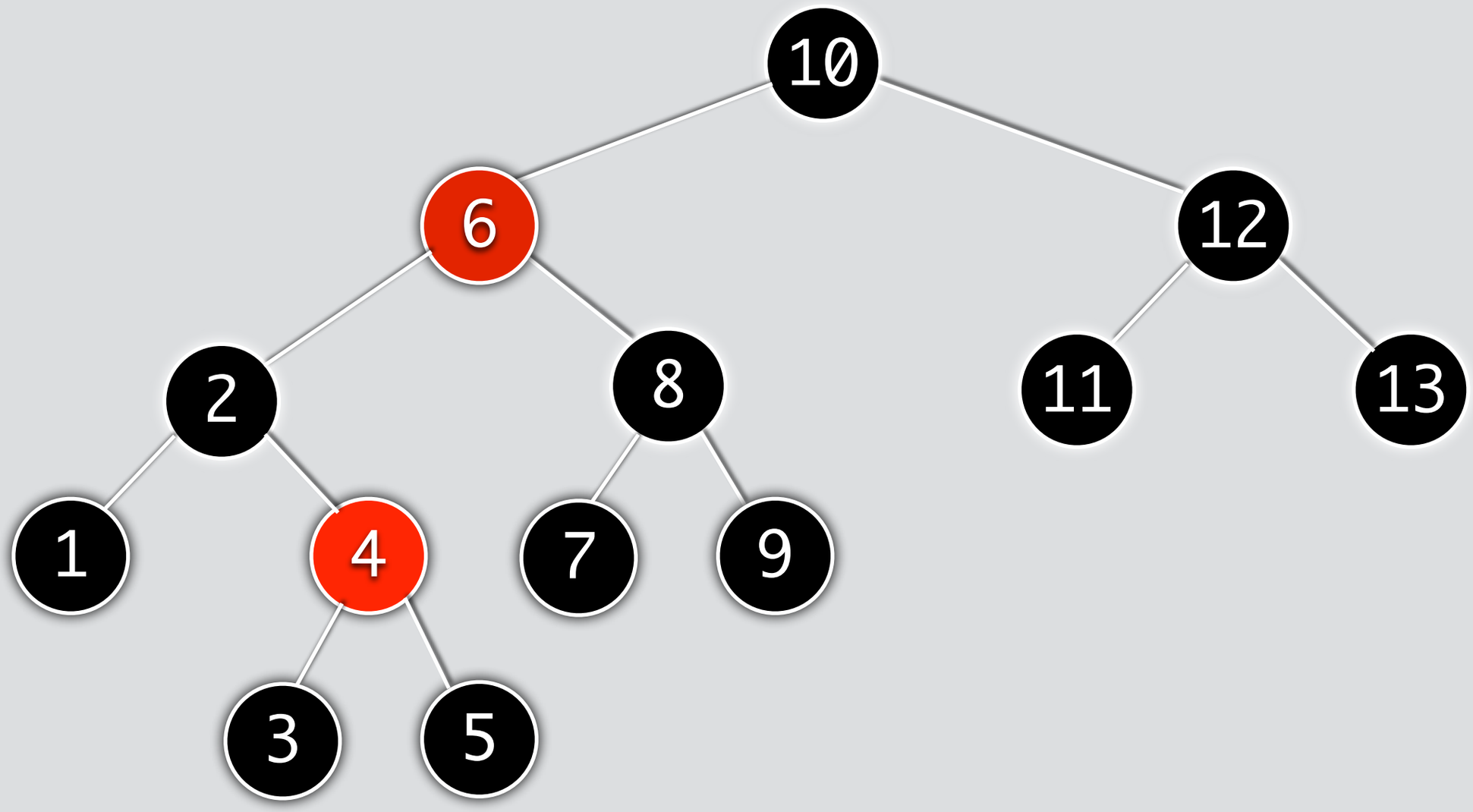
1. Minden pont színe vagy **PIROS** vagy **FEKETE**.

2. A gyökérpont színe **FEKETE**.

3. Minden levél (a NIL pontokat tekintjük levélnek) színe **FEKETE**.

4. Minden **PIROS** pontnak mindkét fia **FEKETE**.

5. Bármely pontból bármely levélig vezető úton ugyanannyi **FEKETE** pont van.


10
6
12
2
8
11
13
1
4
7
9
3
5

Egy x pont fekete-magasságának nevezzük az x pontból induló, levélig vezető úton található, x-en kívüli fekete pontok számát, és fm(x)-szel jelöljük.
Az 5. tulajdonság miatt a fekete-magasság jól definiált, mivel minden ilyen út azonos számú fekete pontot tartalmaz.
Egy piros-fekete fa fekete-magasságát a fa gyökérpontjának fekete-magasságaként definiáljuk.
A piros-fekete fákra teljesül a következő állítás:

Tétel: Bármely n belső pontot tartalmazó piros-fekete fa magassága legfeljebb 2log(n+1)

Biz(1). Teljes indukcióval igazolható, hogy a fa minden x gyökerű részfája legalább $2^{fm(x)}-1$ belső pontot tartalmaz.
Ha x magassága 0, akkor x levél (nil), tehát az x gyökerű részfának valóban 0 belső pontja van.
Tegyük fel, hogy x magassága pozitív, és két fia van. Mindkét fiú fekete-magassága vagy fm(x), vagy fm(x)-1 attól függően, hogy a színe piros vagy fekete.
Mivel x fiainak magassága kisebb, mint x magassága, így az indukciós feltevés alapján mindkét részfa legalább $2^{fm(x)-1} - 1$ belső pontot tartalmaz. Tehát az x gyökerű részfa belső pontjainak száma legalább $(2^{fm(x)-1}-1) + (2^{fm(x)-1}-1) + 1 = 2^{fm(x)}-1$.

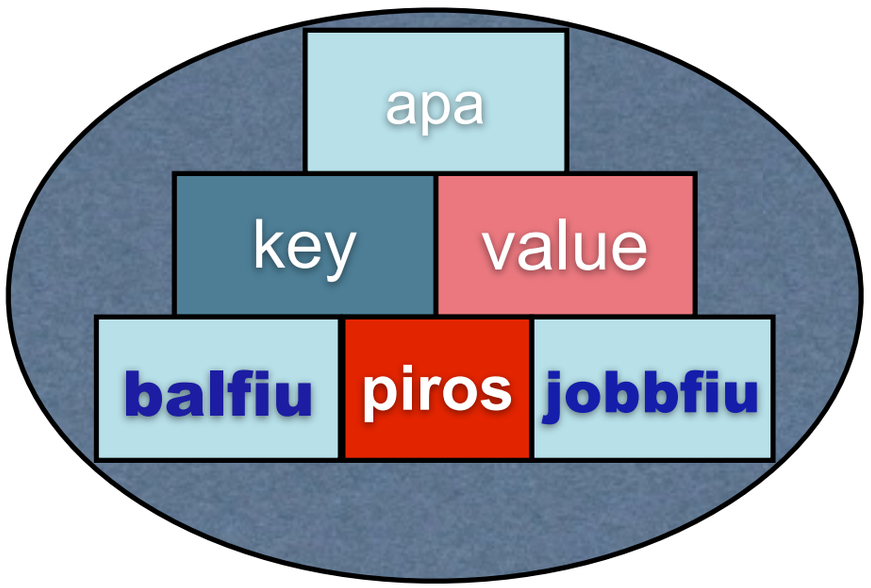
Legyen x magassága h.
A 4. tulajdonság szerint minden a gyökértől levélig haladó út, legalább feleannyi fekete pontot tartalmaz, mint ezen út pontjainak száma, nem számítva a gyökeret.
Tehát a gyökér fekete-magassága legalább h/2, így $n \geq 2^{h/2} - 1$.
Tehát azt kapjuk, hogy $\log(n+1) \geq h/2$, azaz $h \leq 2\log(n+1)$. □

Következésképpen a keresőfa műveletek időigénye O(logn), ahol n a pontok száma.
Azt kell megvizsgálni a piros-fekete fa tulajdonságok karbantartásának mekkora az időigénye.

Fapont tárolása

első megközelítés, a gyakorlatban nem így tároljuk!

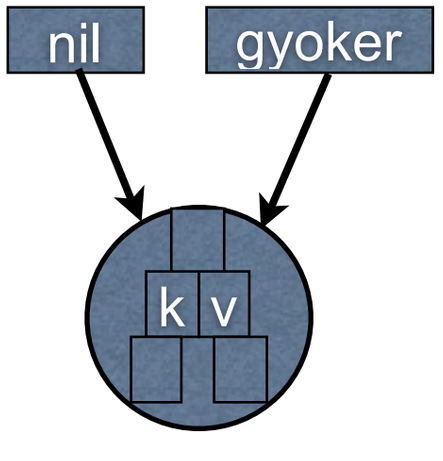
```
struct fapont{
    int key, value ;
    boolean piros ;
    fapont bal, jobb, apa ;
}
```



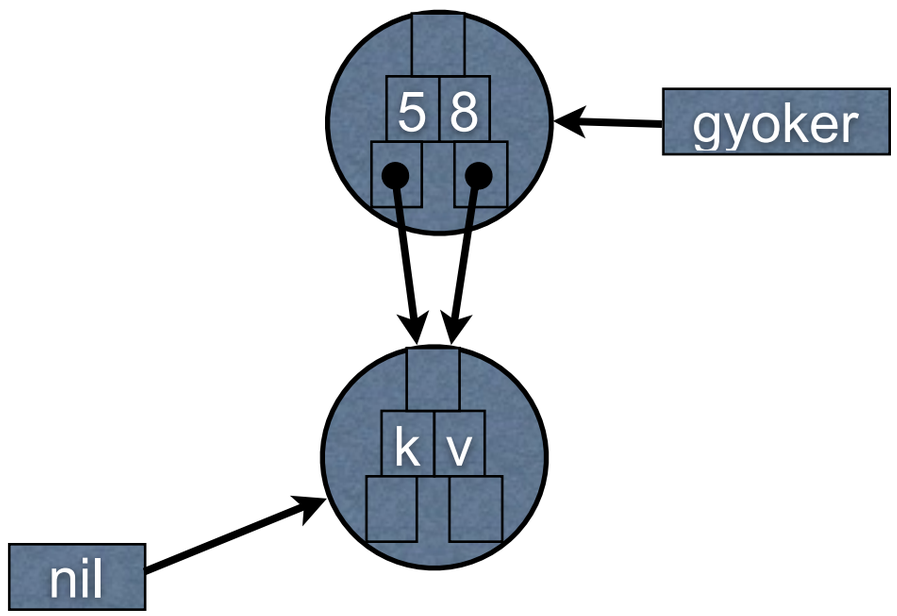
Piros-fekete fa tárolása

első megközelítés, a gyakorlatban nem így van!

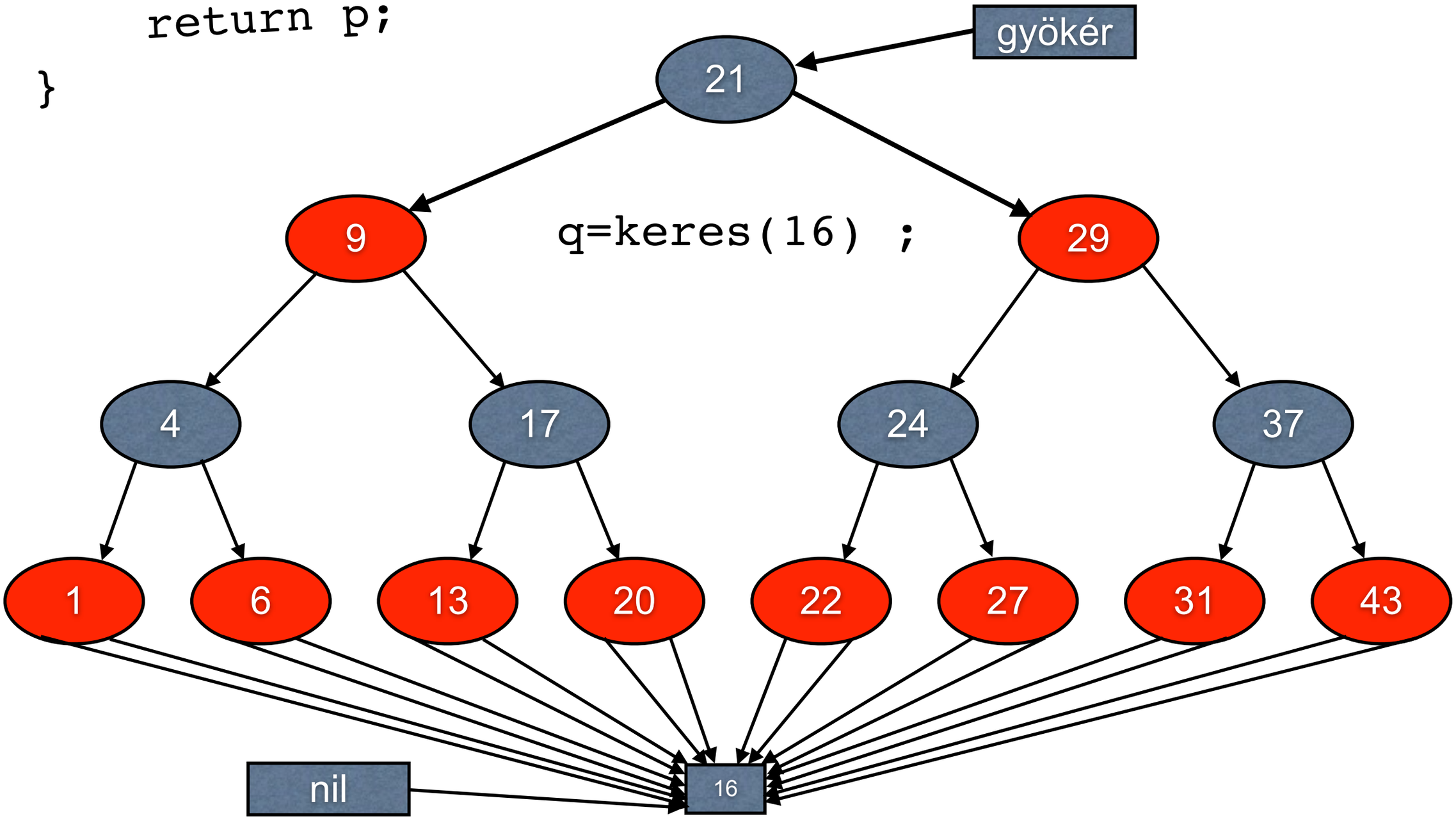
```
public class fa {
   int elemszam ;
   struct fapont{
      int key, value ;
      boolean piros ;
      fapont bal, jobb, apa ;
   }
   fapont nil, gyoker ;
   void fa() {
      fapont q = new fapont() ;
      nil=q ;
      gyoker=q ;
      elemszam=0 ;
      q.piros=false ;
   }
   boolean beszur(int x, int v) {
    ...
  }
  ...
}
```



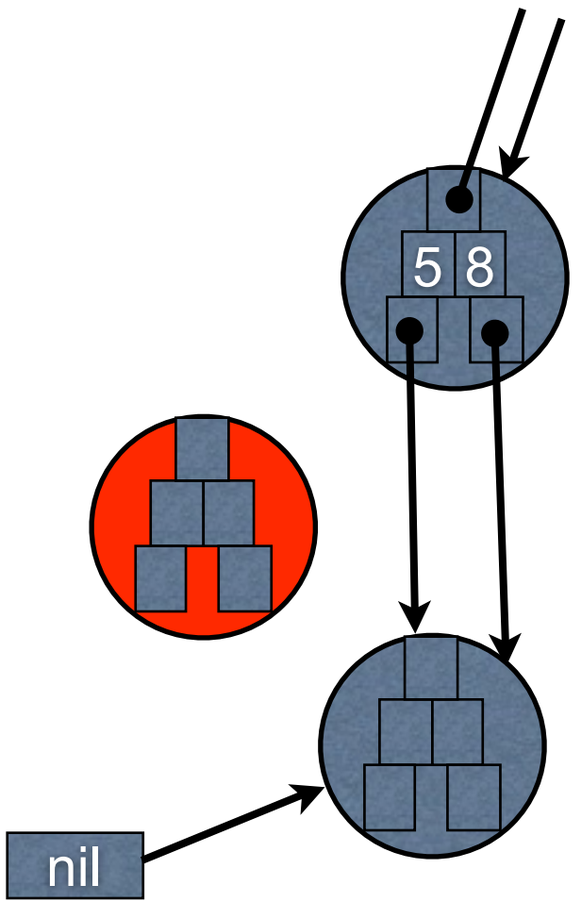
beszur(5, 8)

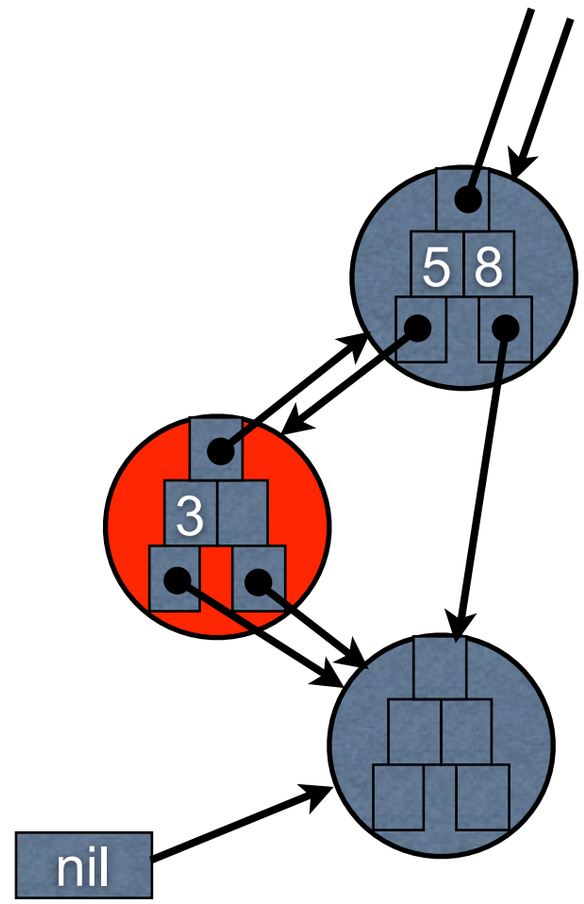


```
fapont keres(int x) {
     fapont p=gyoker ;
     nil.key=x ;
     while (p.key!=x) p=x<p.key?p.bal:p.jobb ;
     return p;
}
```

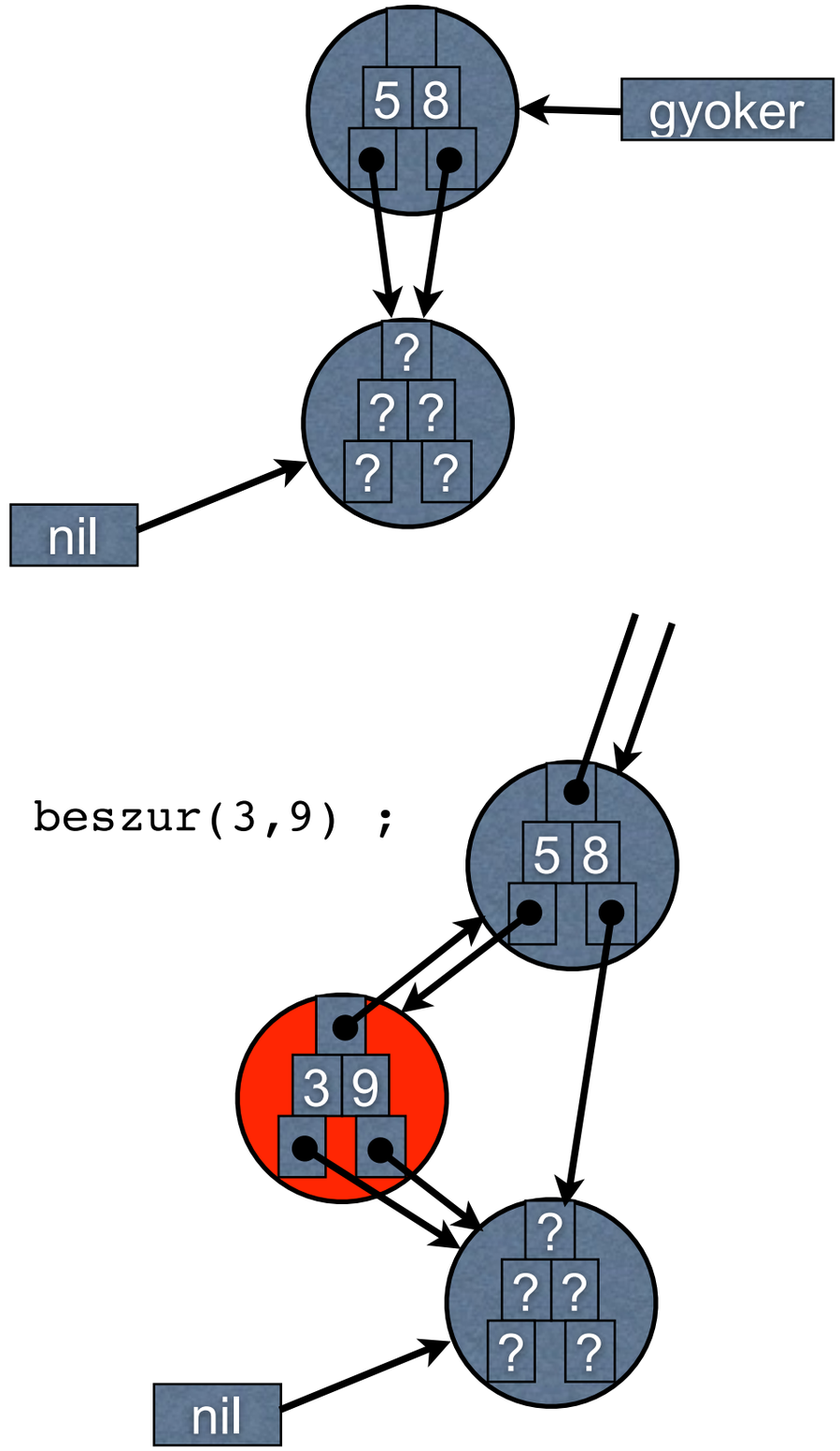


q=keres(16) ;

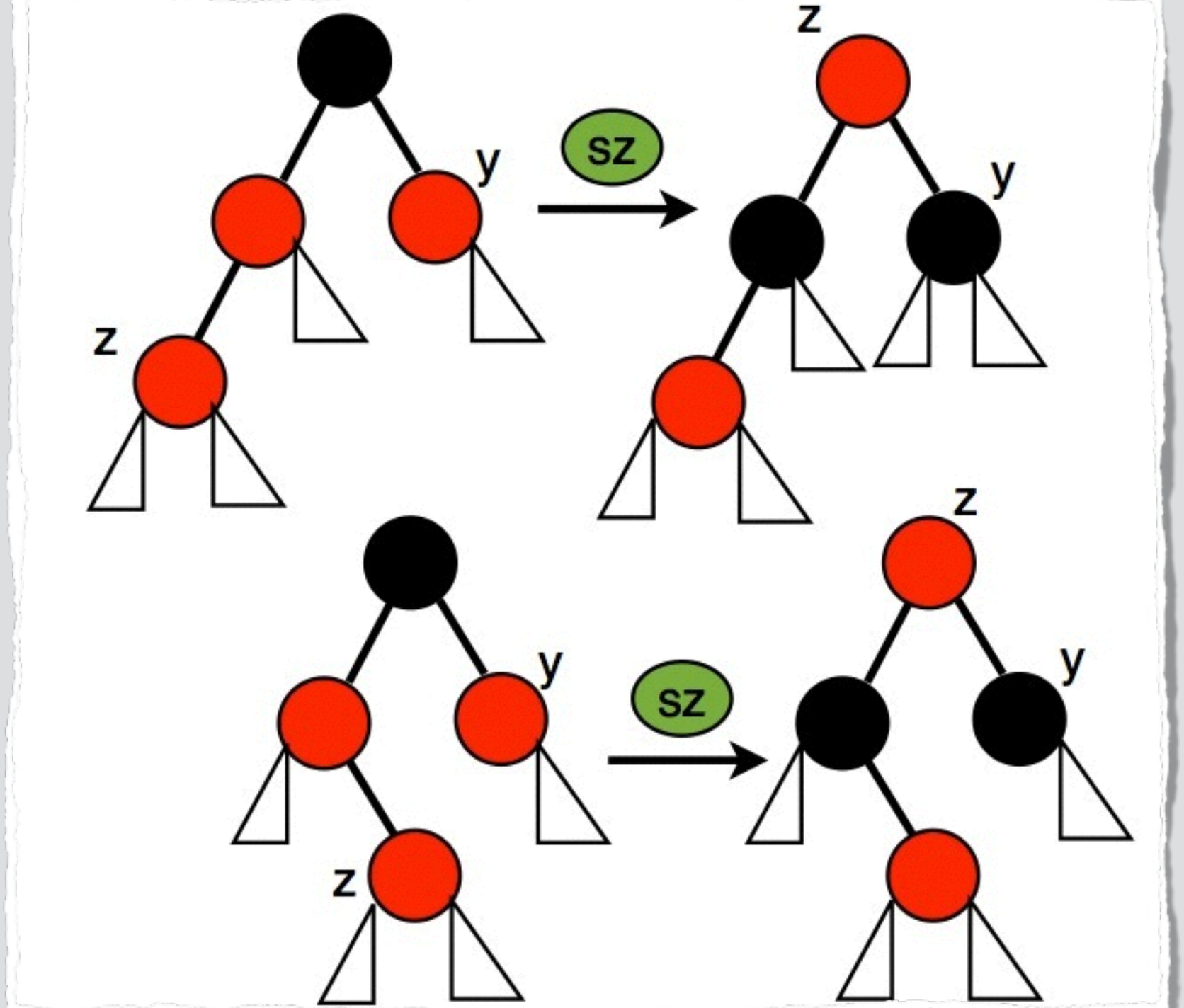

5 8
nil


5 8
3
nil

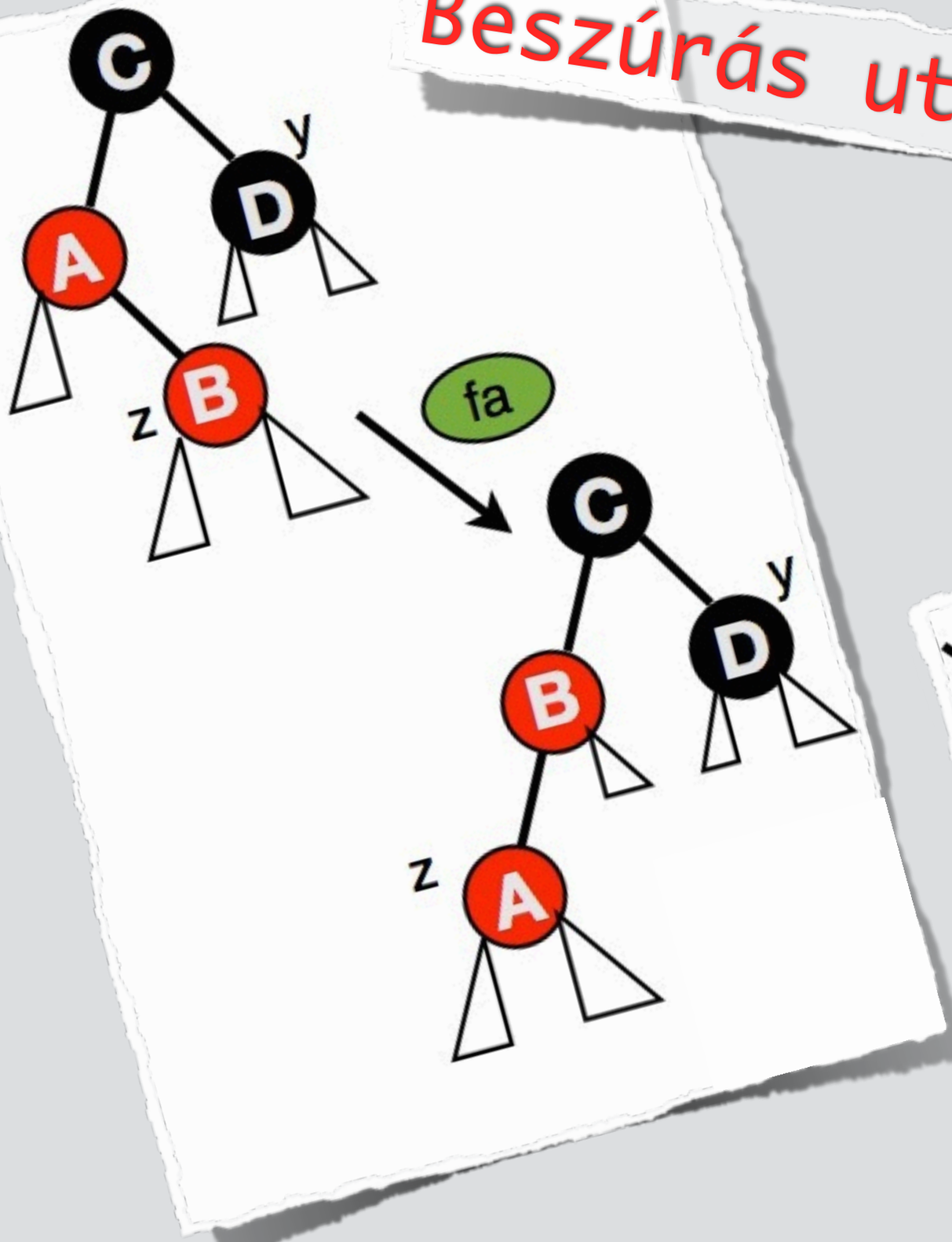
```
boolean beszur(int x, int v) {
      fapont p,papa ;
      p=gyoker ;
      papa=nil ;
      while (p!=nil && p.key!=x) {
         papa=p ;
         if (x<p.key) p=p.bal;
         else p=p.jobb;
      }
      if (p==nil) {
         fapont ujpont = new fapont();
         if (gyoker==p) gyoker=ujpont ;
         ujpont.key=x ;
         ujpont.value=v ;
         ujpont.bal=nil;
         ujpont.jobb=nil;
         ujpont.apa=papa;
         ujpont.piros=true ;
         if (papa!=nil) {
            if (x>papa.key) papa.jobb=ujpont;
            else  papa.bal=ujpont;
         }
         elemszam++ ;
         beszurjavit(p);
         return true ;
      } else {
         p.value=v ;
         return false ;
      }
   }
```

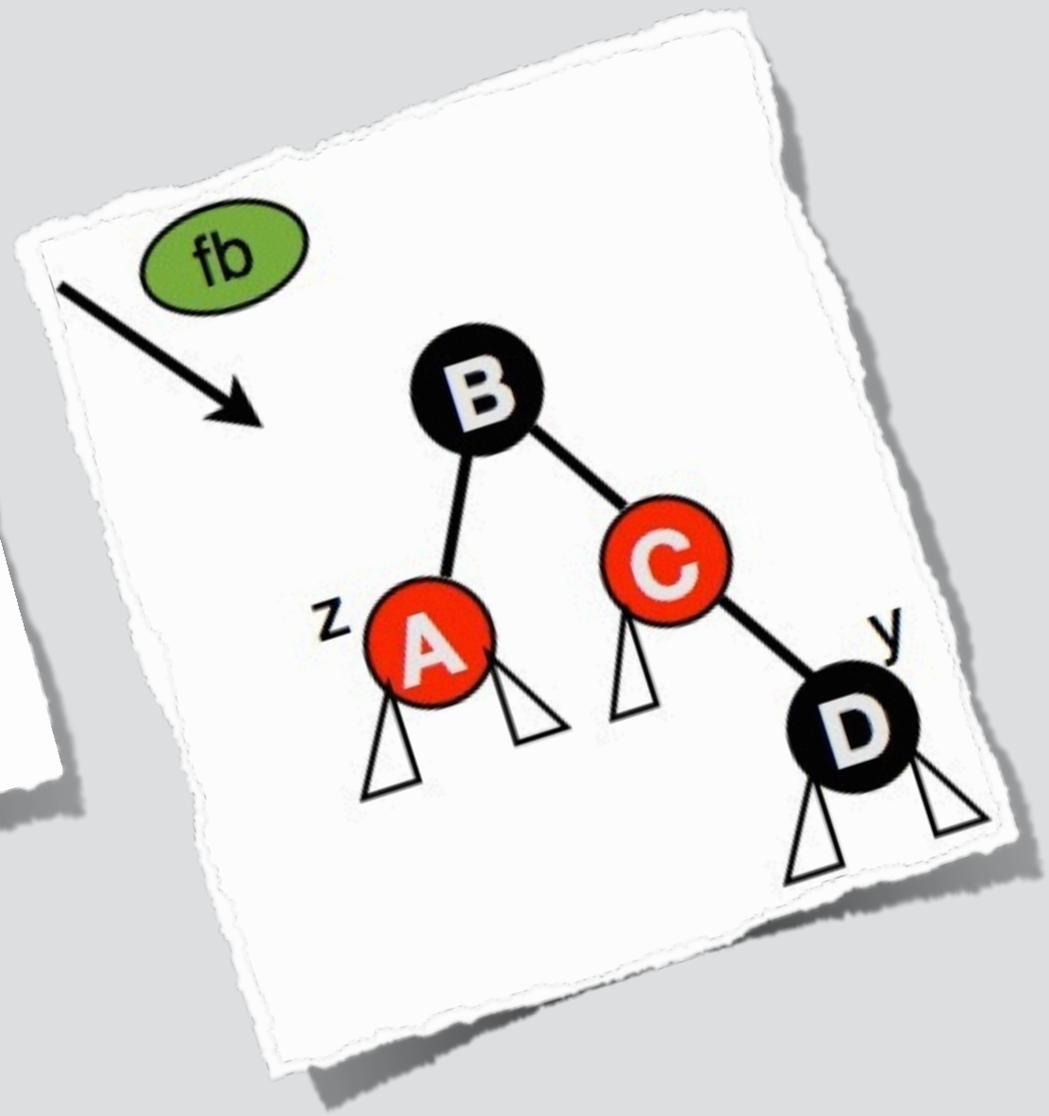


Beszúrás utáni javítás I.



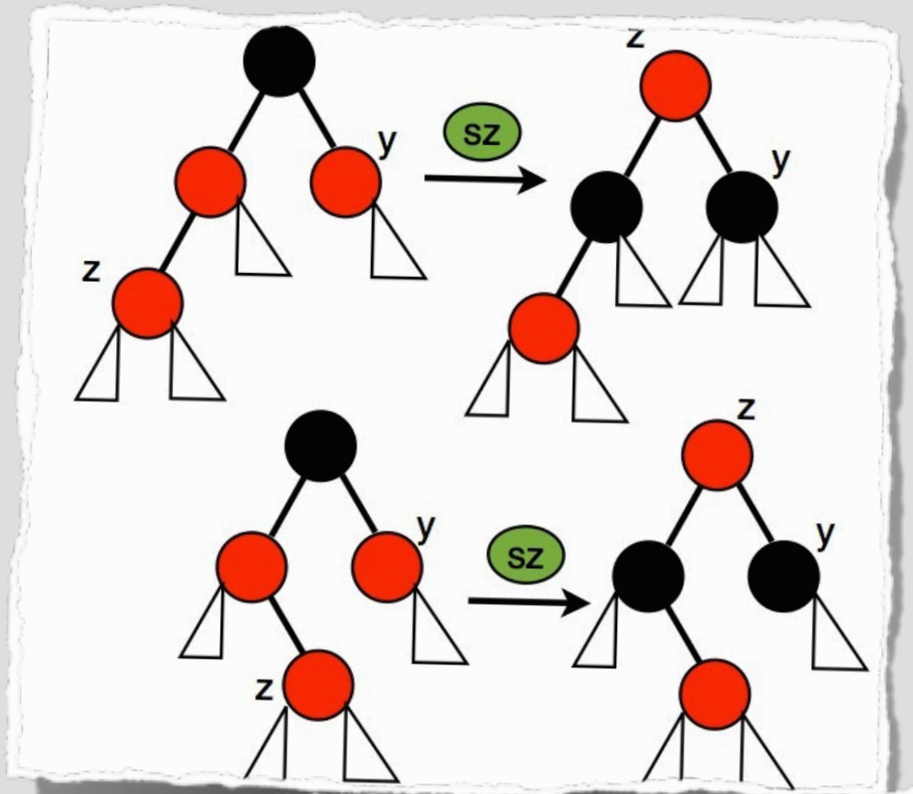
Beszúrás utáni javítás II.



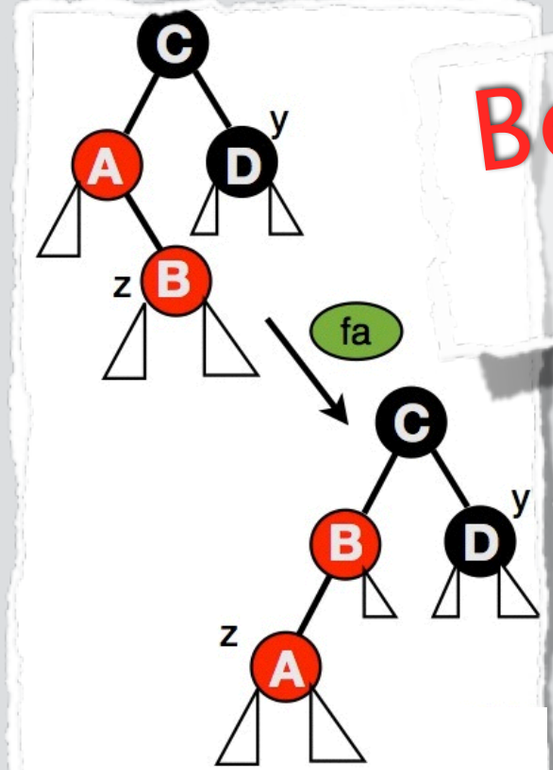


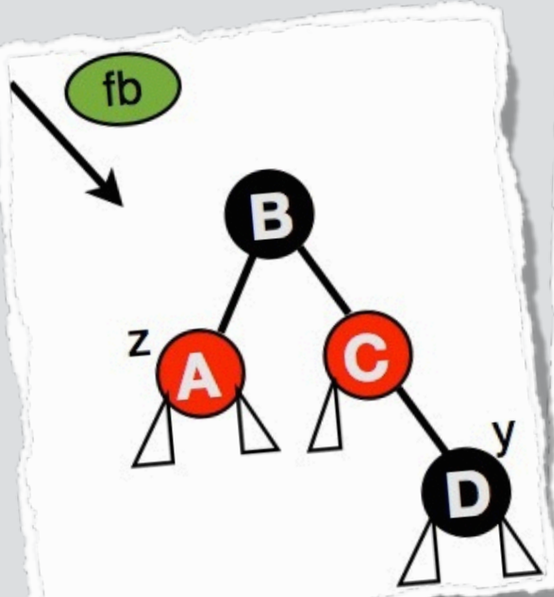
```
void balraforgat(fapont x) {
    fapont y ;
    y=x.jobb ;
    x.jobb=y.bal ;
    if (x.jobb!=nil) x.jobb.apa=x;
    y.apa=x.apa;
    if (y.apa==nil) gyoker=y ;
    else {
      if (x==y.apa.bal) y.apa.bal=y ;
      else y.apa.jobb=y ;
    }
    y.bal=x ;
    x.apa=y ;
}
```

0(1)



Beszúrás utáni javítás






```
void pf-beszurjavit(fapont z) {
  while (z.apa.piros) {
    if (z.apa==z.apa.apa.bal) {
      y=z.apa.apa.jobb ;
      if (y.piros) {
        z.apa.piros=false ;
        z.apa.apa.piros=true ;
        y.piros=flase ;
        z=z.apa.apa ;
      } else {
        if (z==z.apa.jobb) {
          z=z.apa ;
          balraforgat(z) ;
        }
        z.apa.piros=false ;
        z.apa.apa.piros=true ;
        jobbraforgat(z.apa.apa) ;
      }
    } else {
      //szimmetrikus eset
    }
  }
  gyoker.piros=false ;
}
```

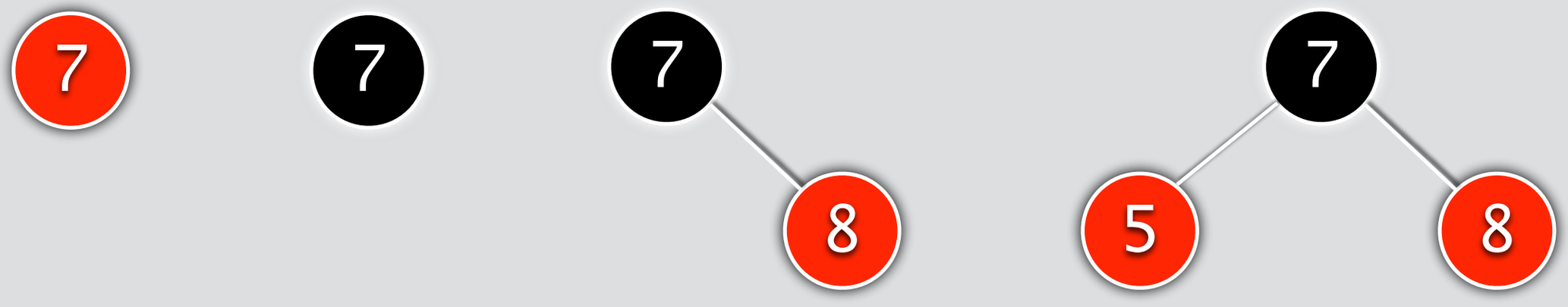
SZ

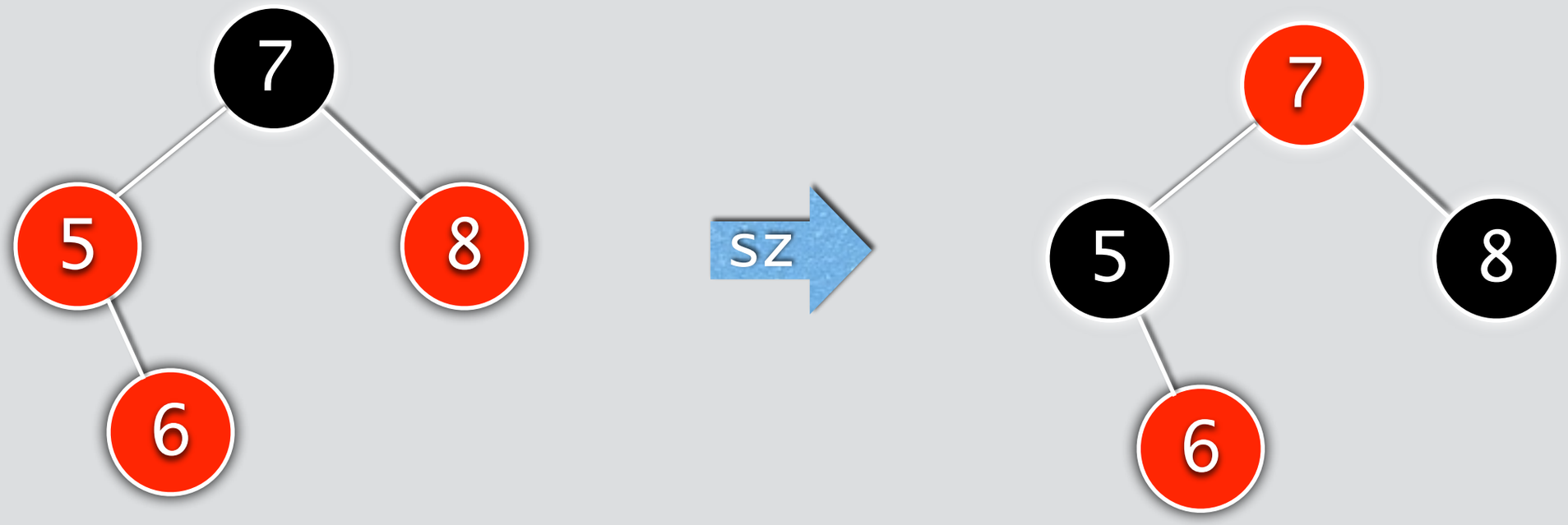
fa

fb

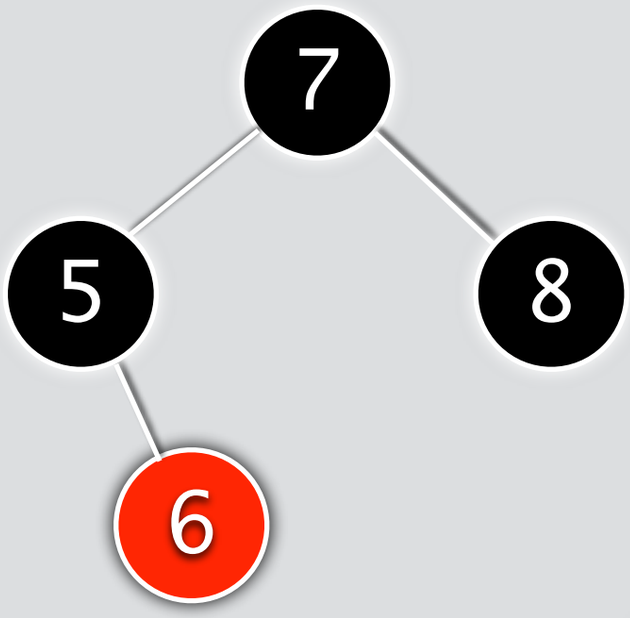
O(logn)

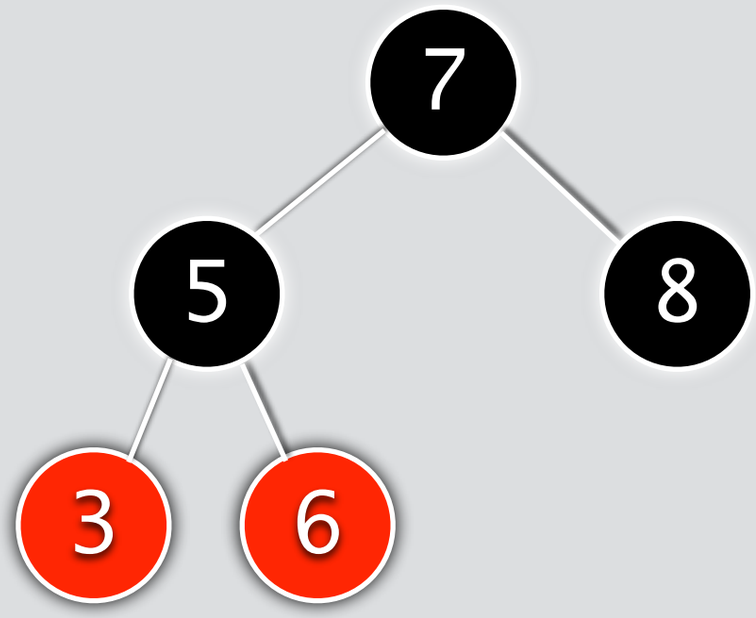
példa: 7, 8, 5, 6

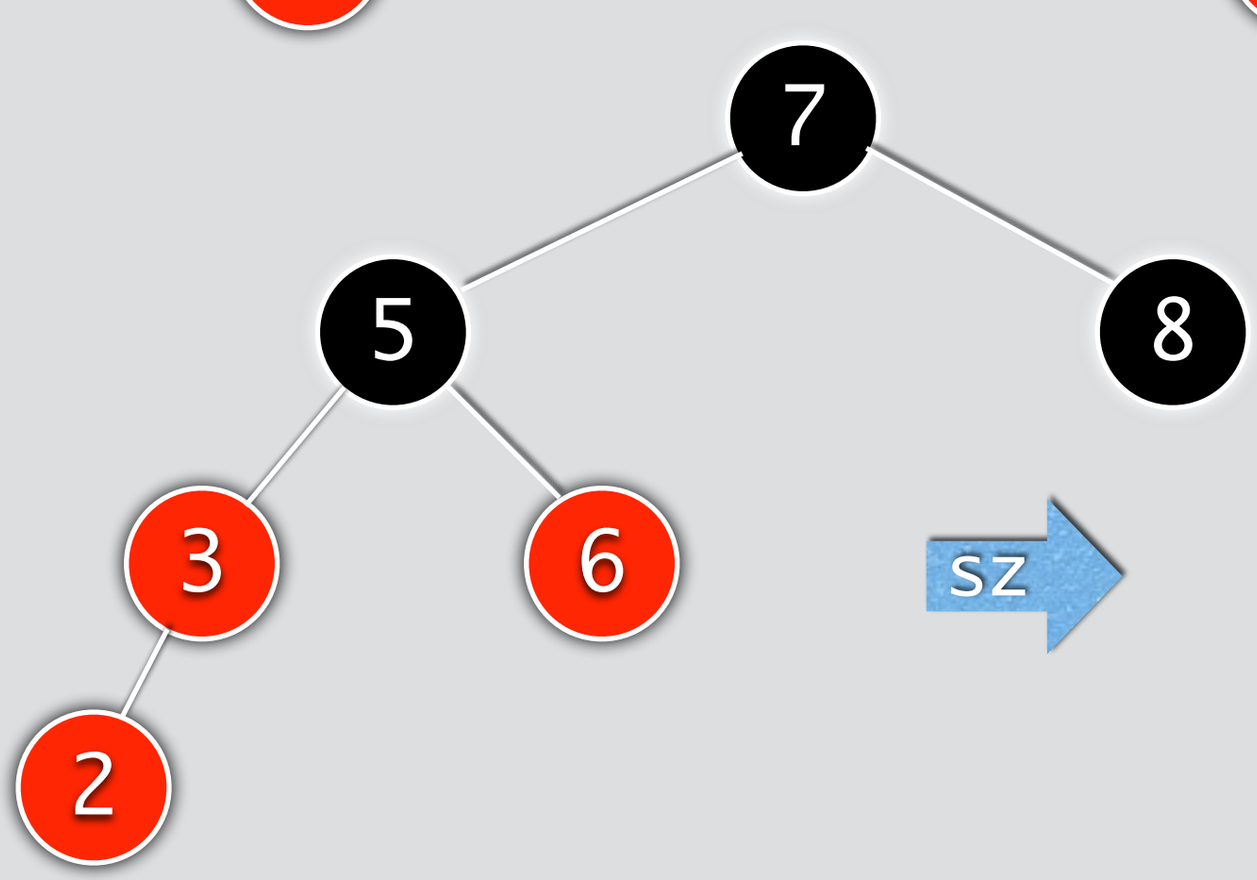


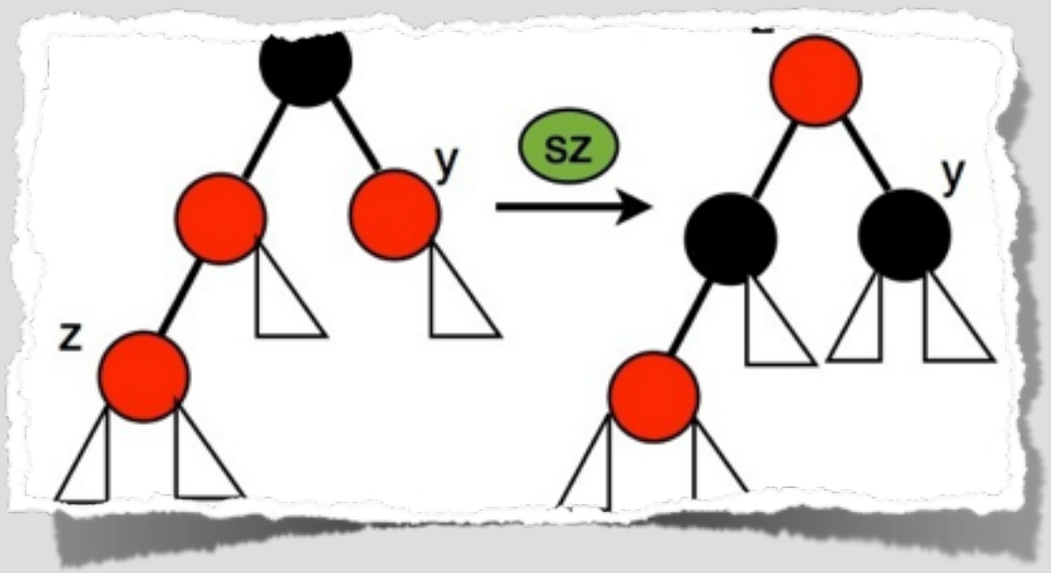


példa: 7, 8, 5, 6, 3, 2

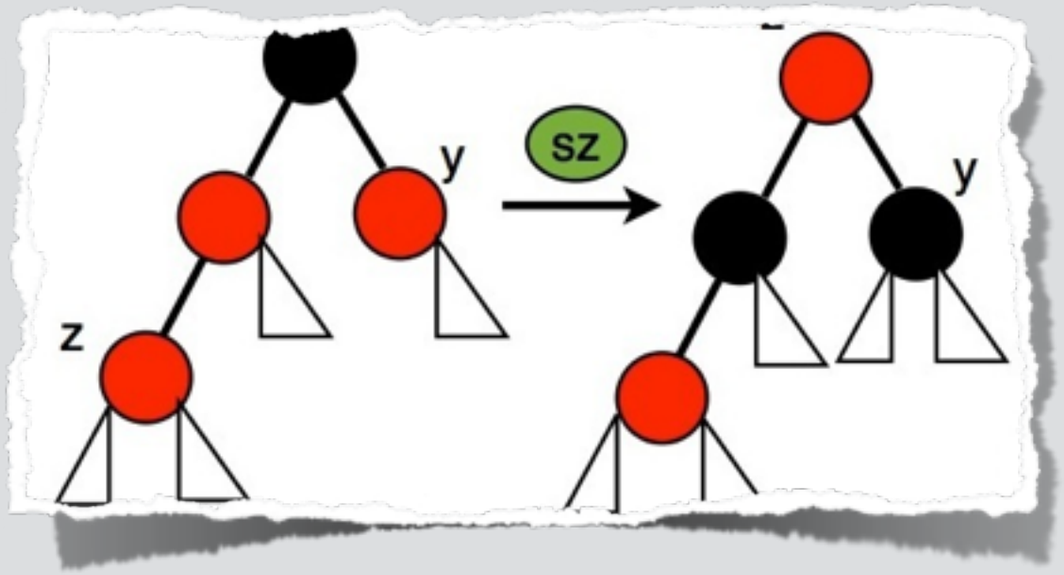


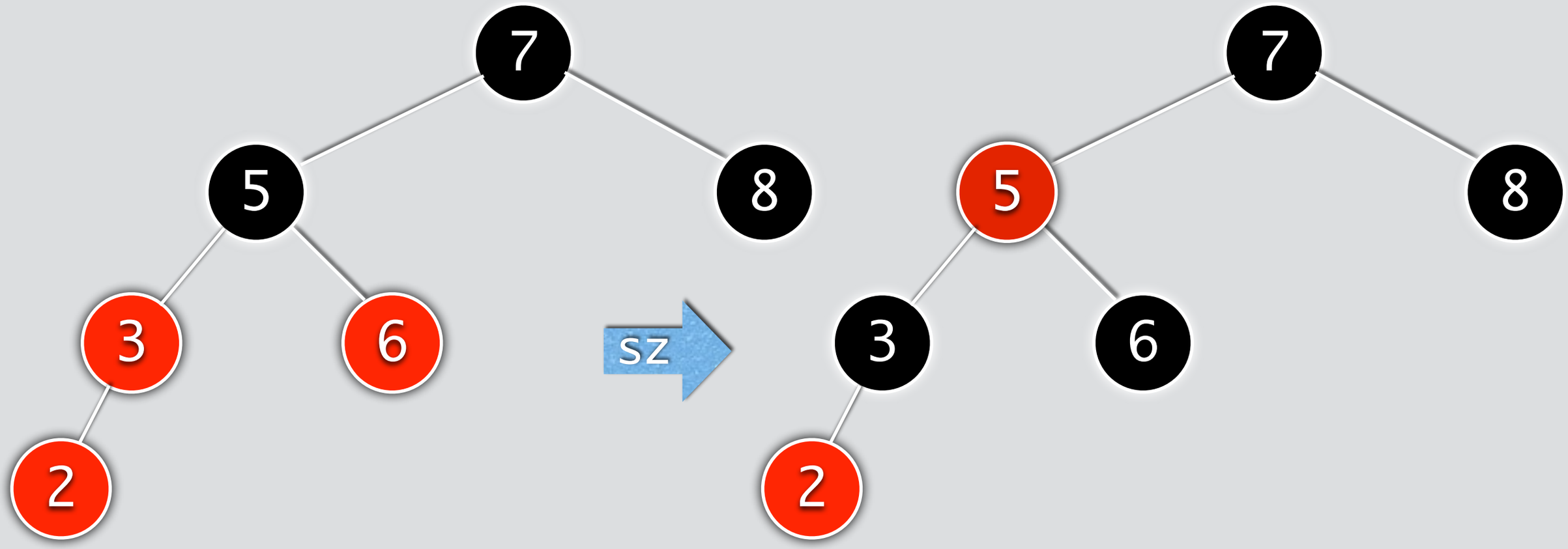




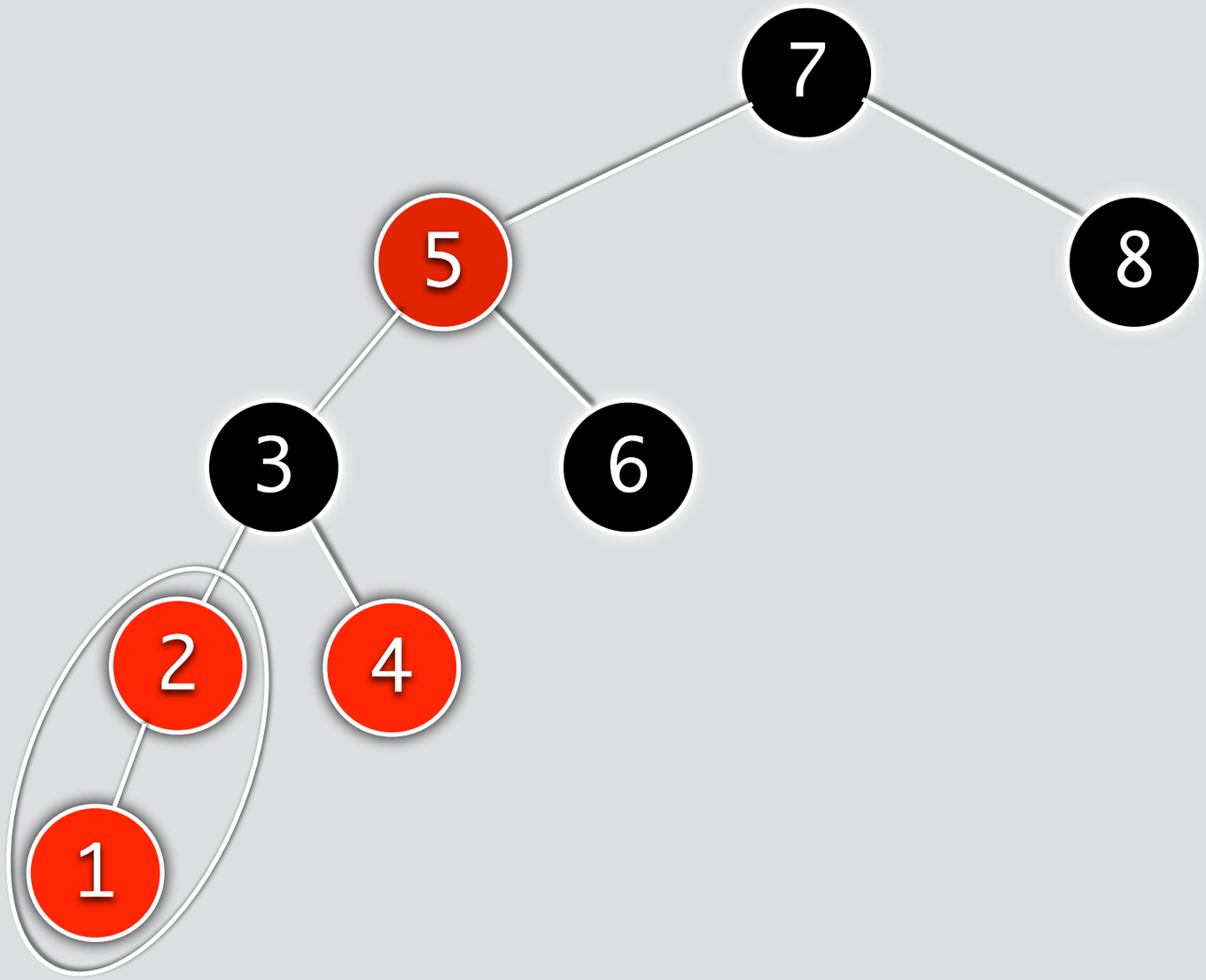


példa: 7, 8, 5, 6, 3, 2

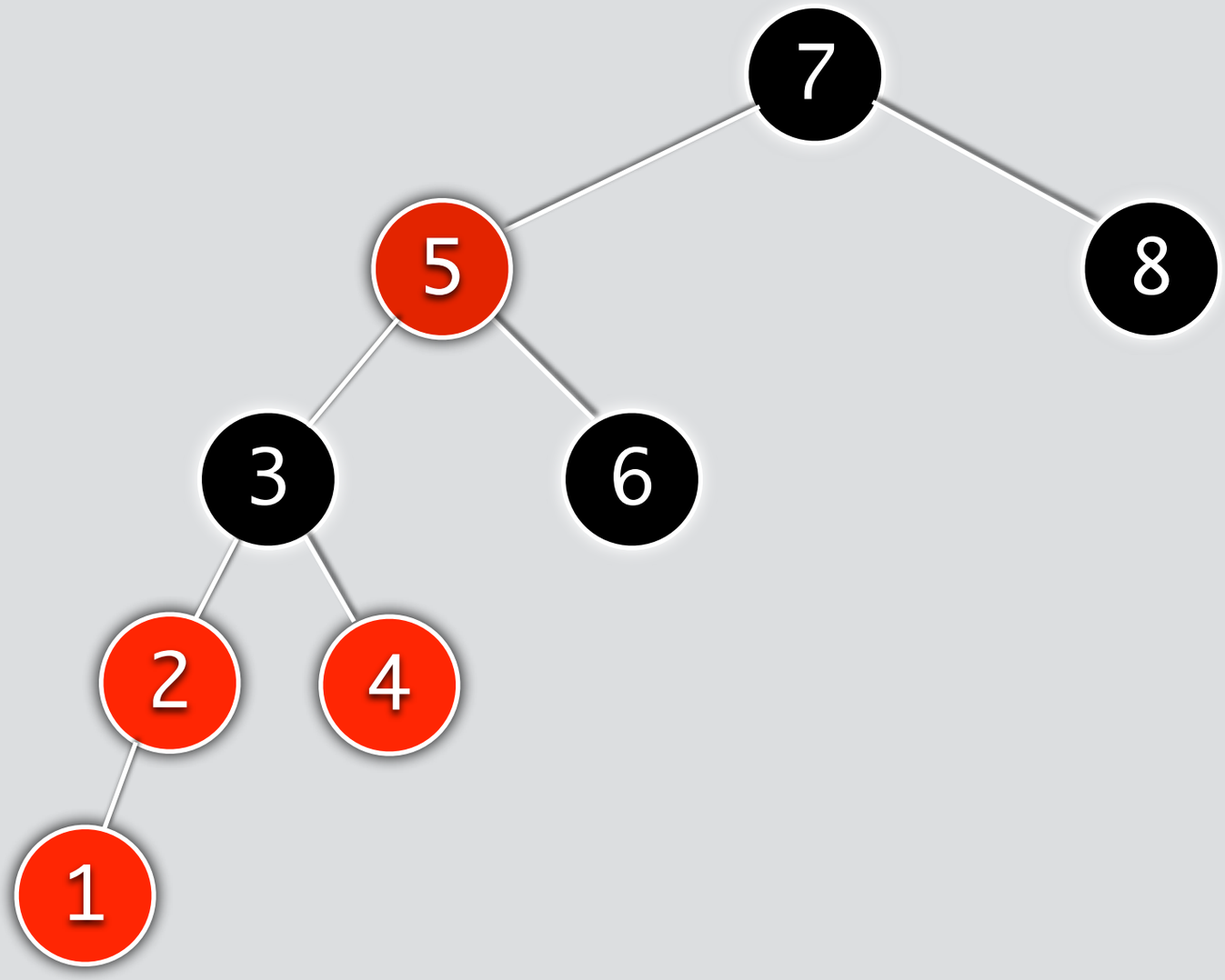


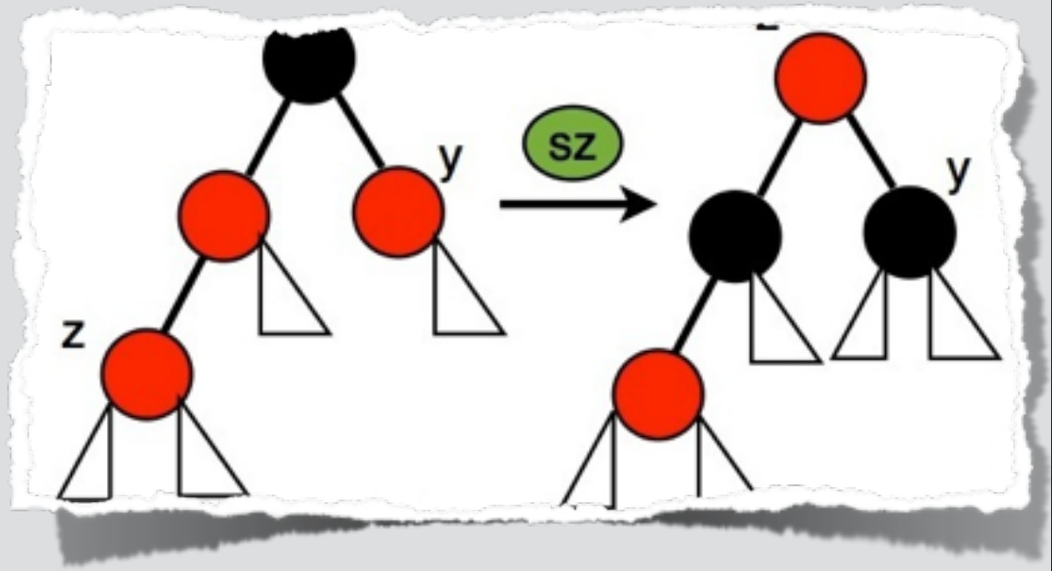


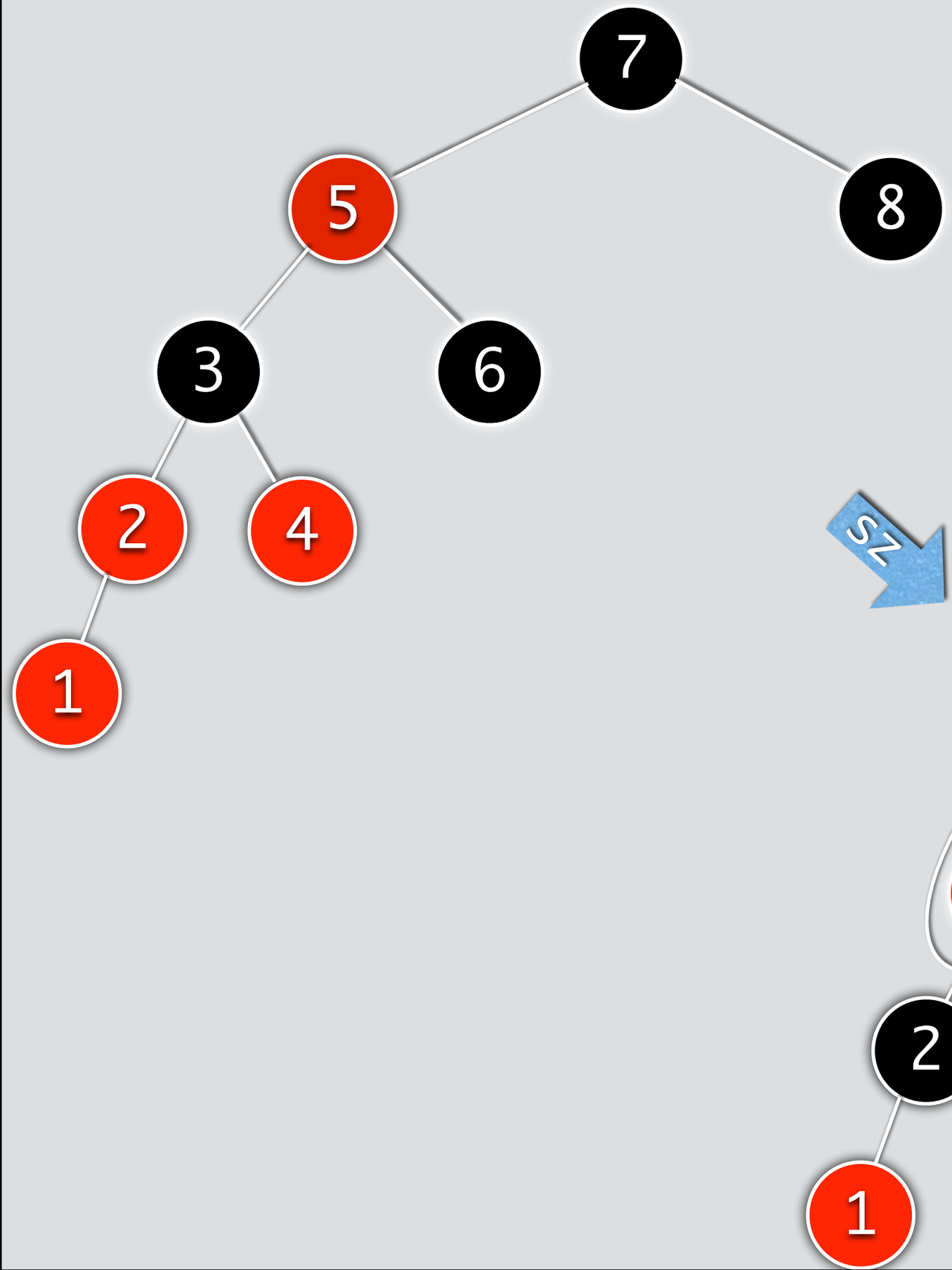
példa: 7, 8, 5, 6, 3, 2, 4, 1



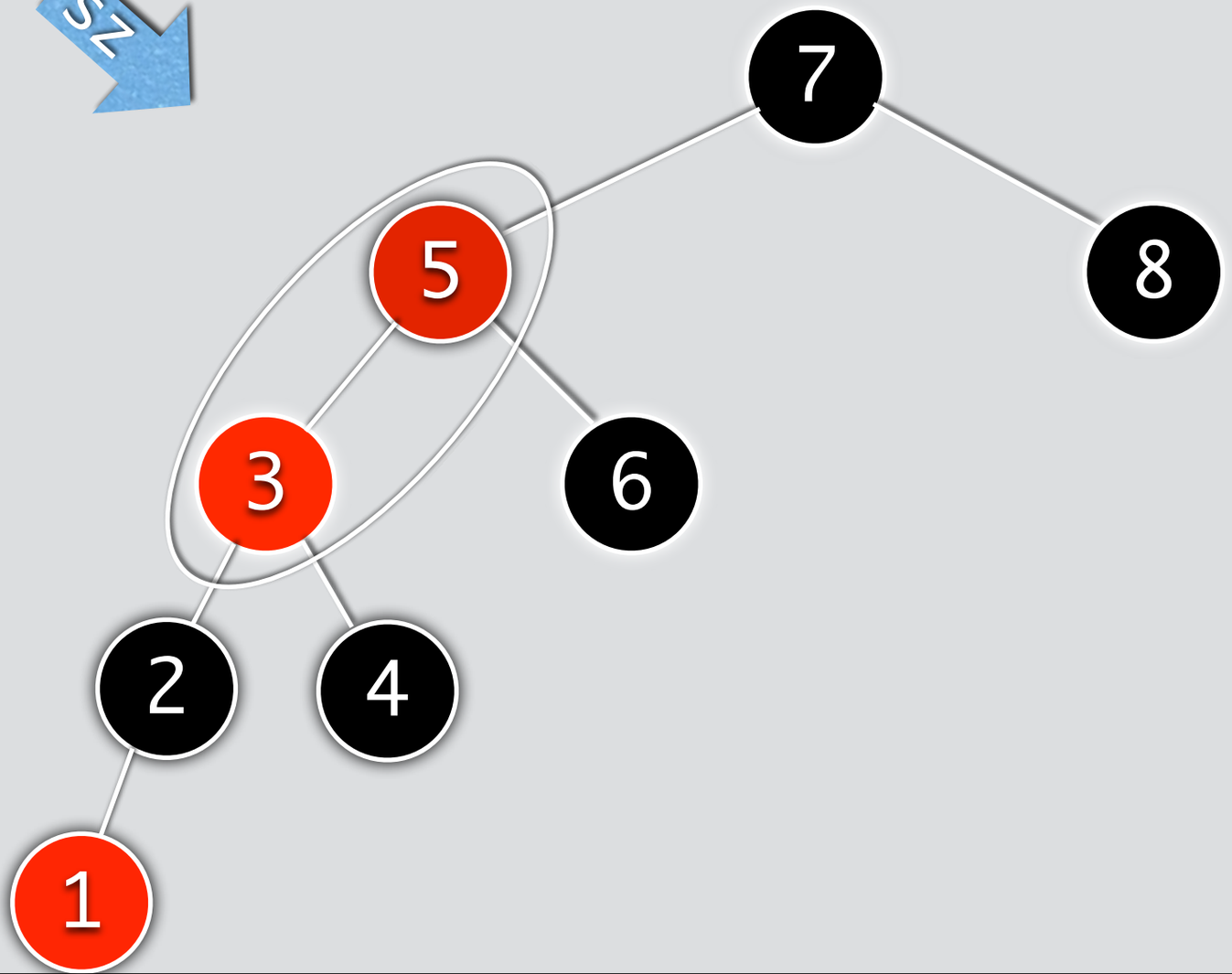
példa: 7, 8, 5, 6, 3, 2, 4, 1

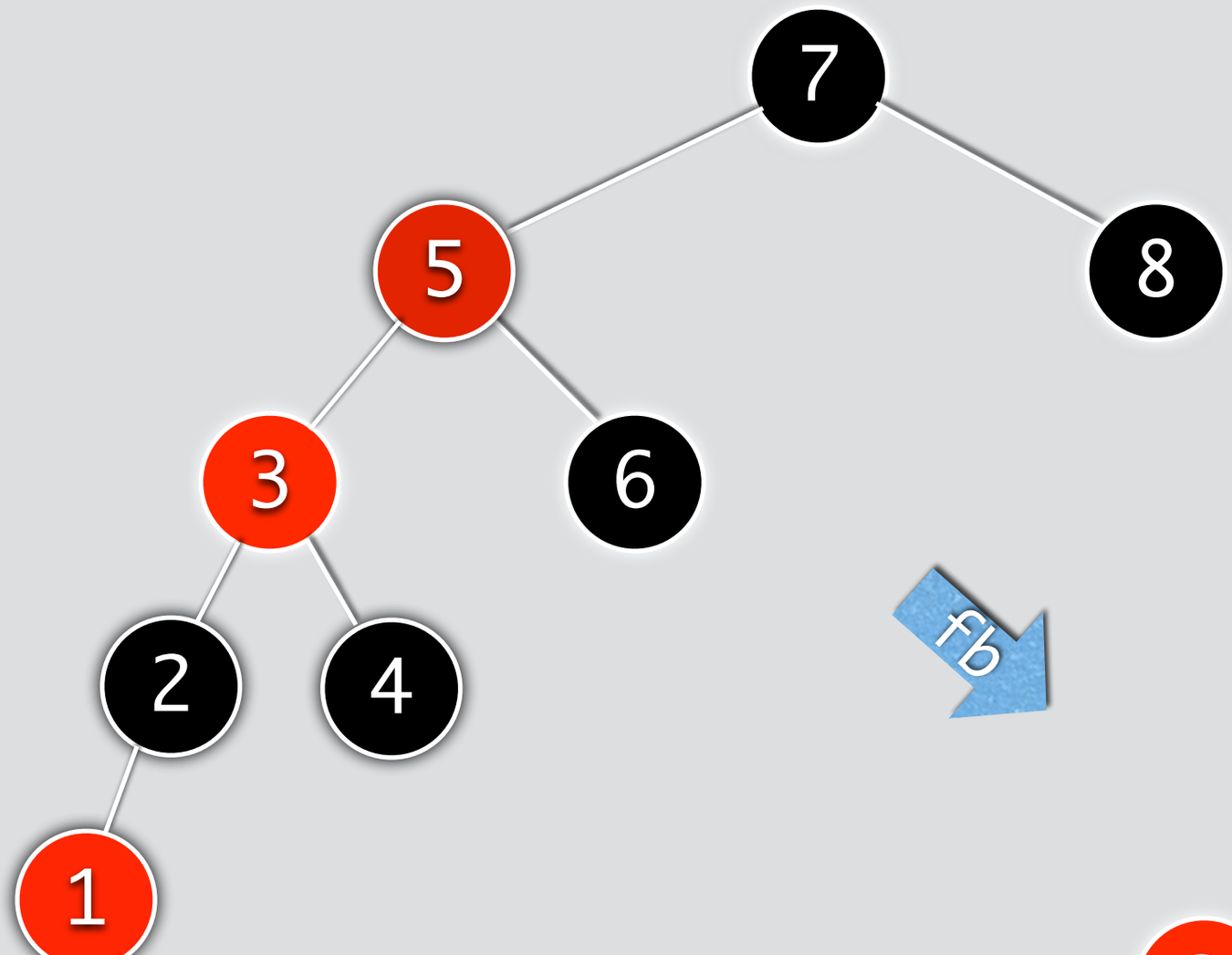


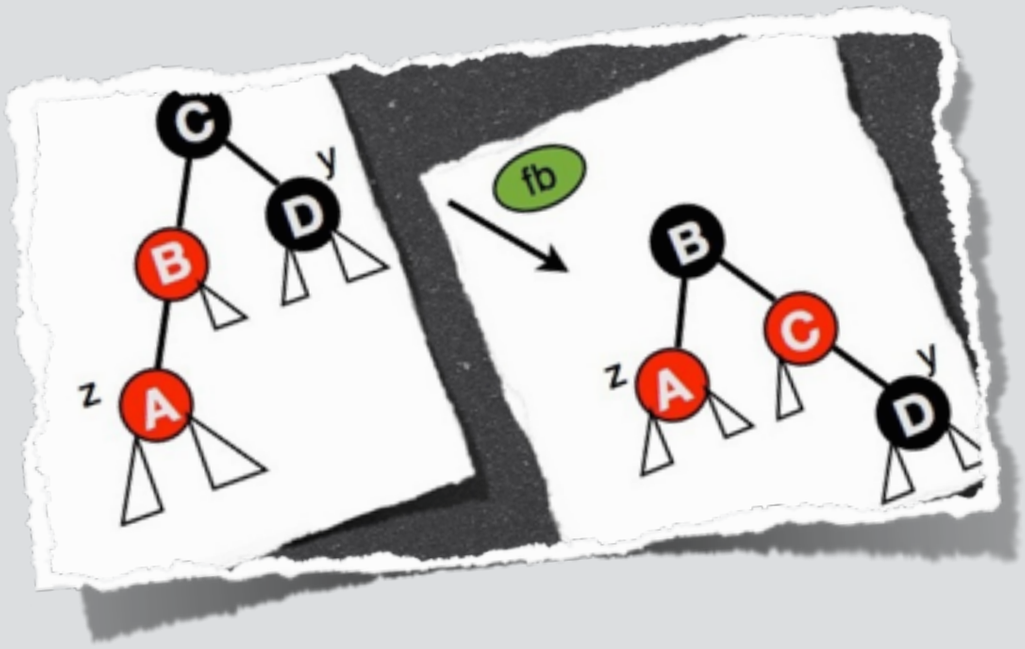




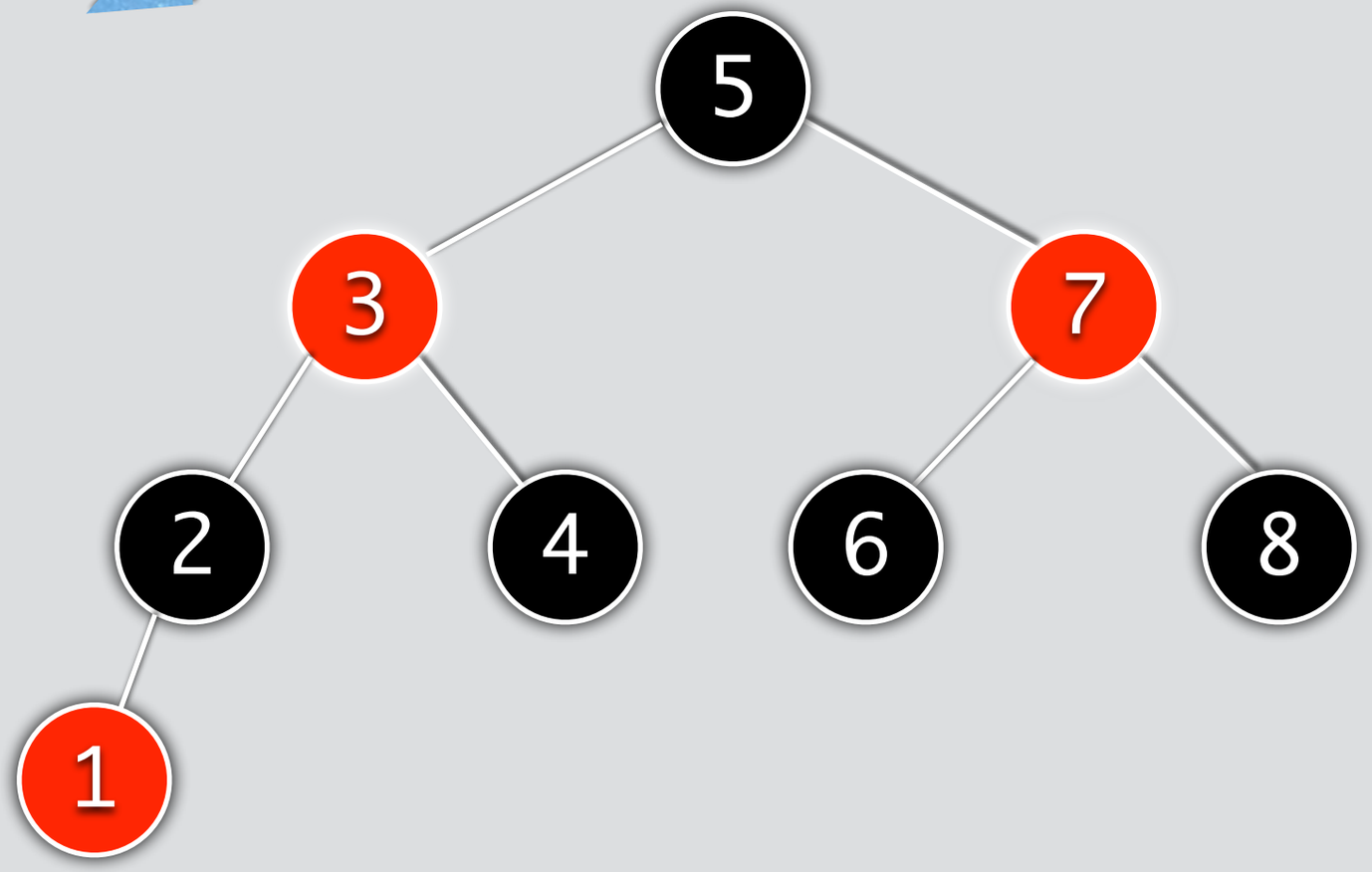
SZ



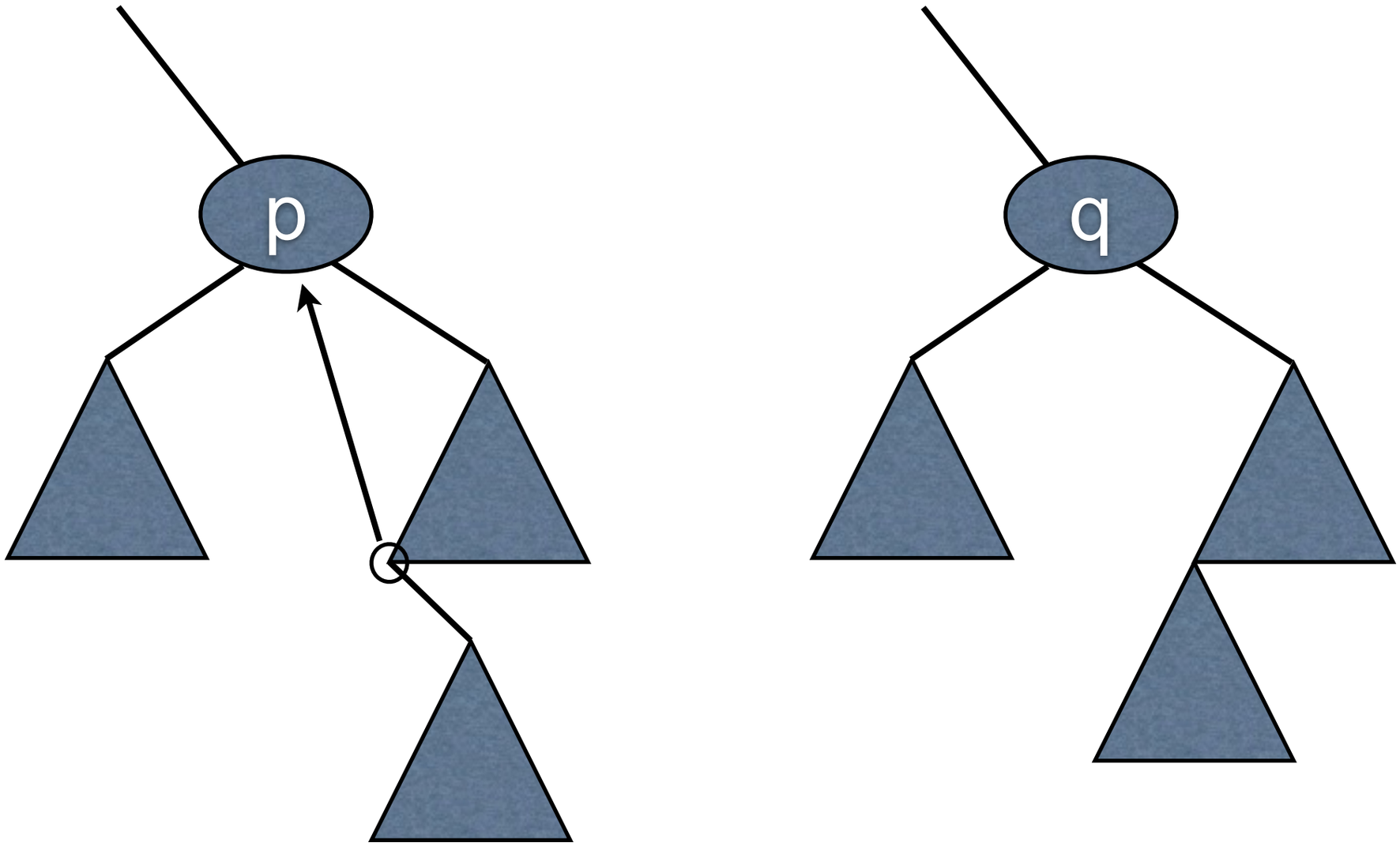




fb



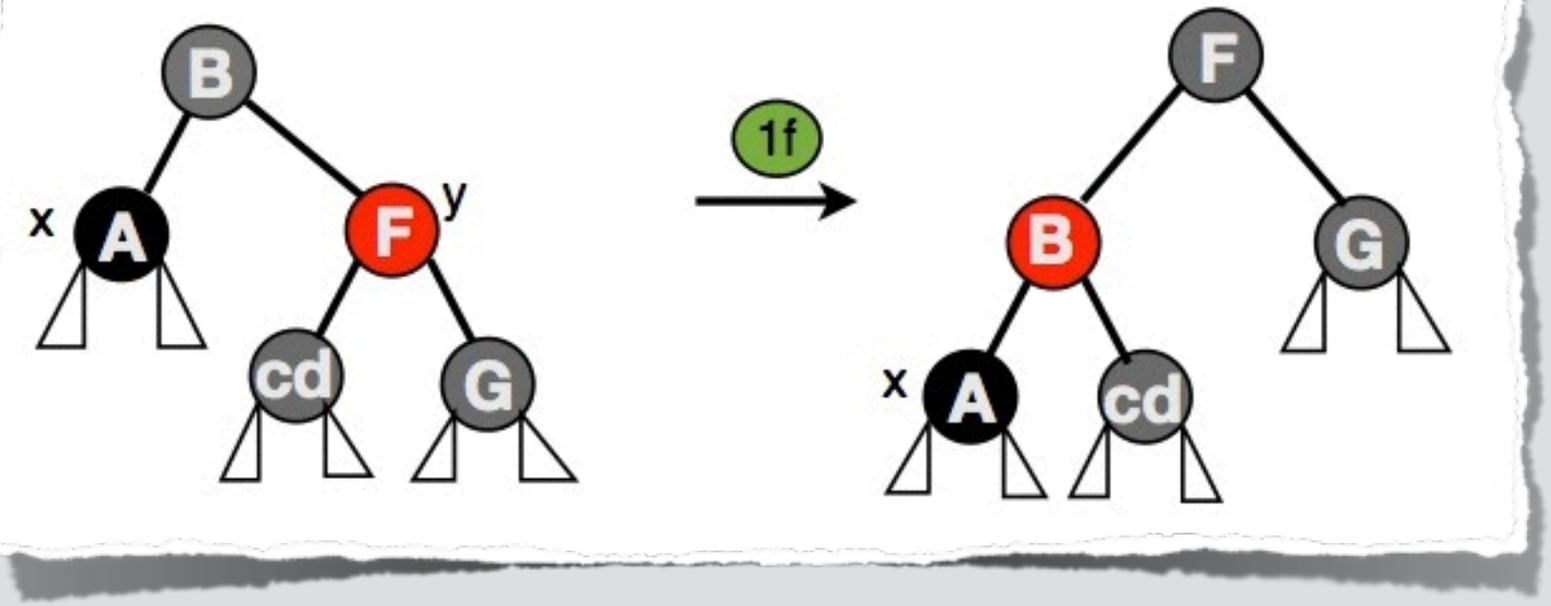
Törlés piros-fekete fából

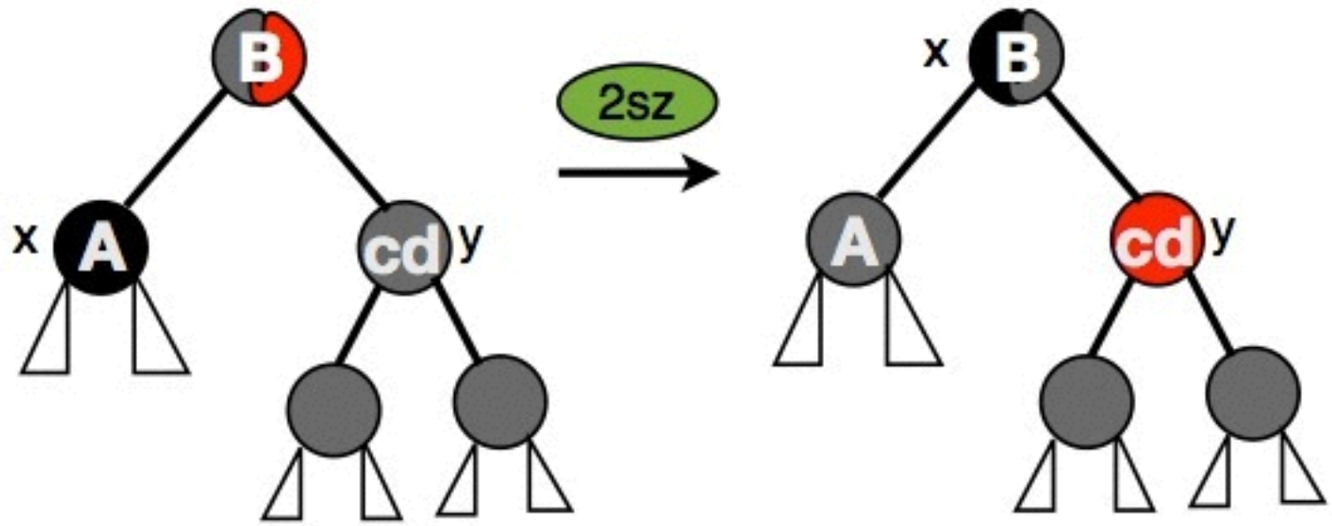


Törlés Piros-Fekete fából

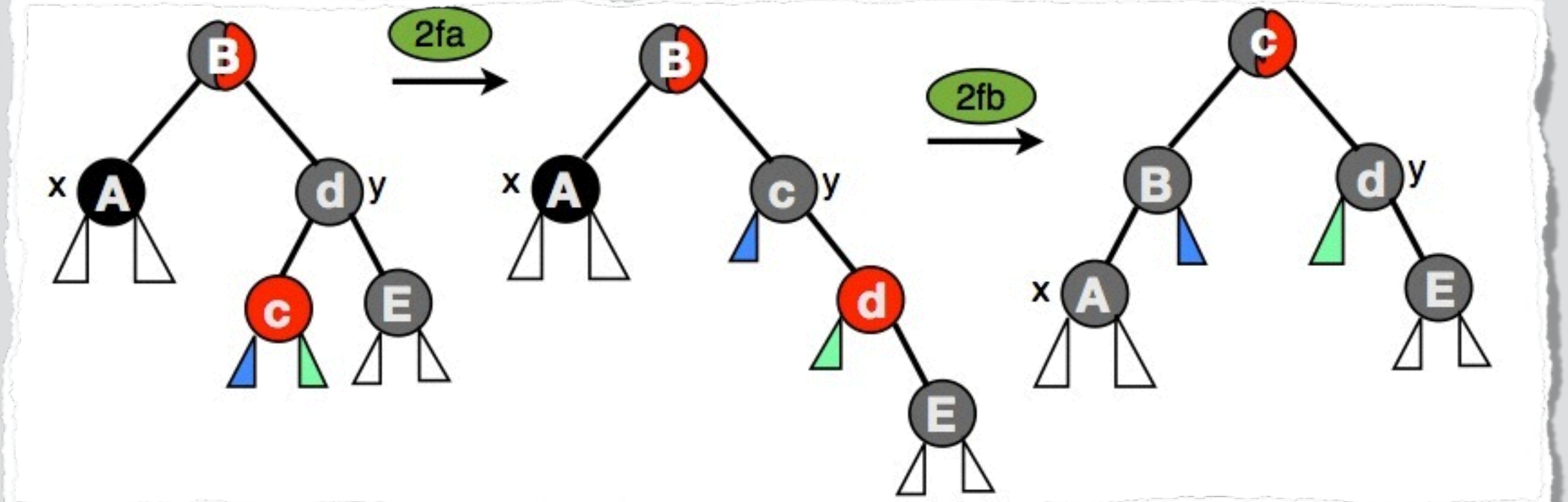
A fából a keresofa Töröl műveletének megfelelően töröljük az adott pontot. Ha a pont vagy a fia piros akkor nincs másteendő, egyébként sérül az 5. tulajdonság és ezt helyre kell állítani. Ezt úgy oldjuk meg, hogy a ténylegesen törölt pont fia kap egy extra fekete értéket, és duplán fekete lesz, majd forgatásokkal és átszínezéssel ezt megszüntetjük.

Törlés utáni javítás I.



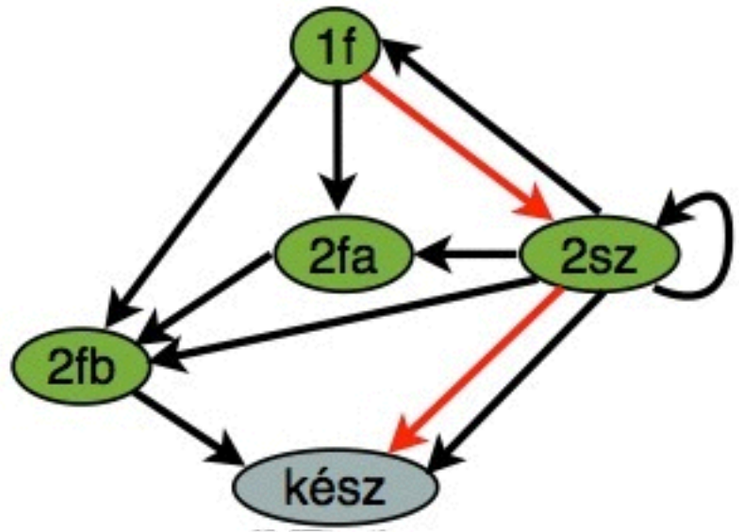


Törlés utáni javítás II.

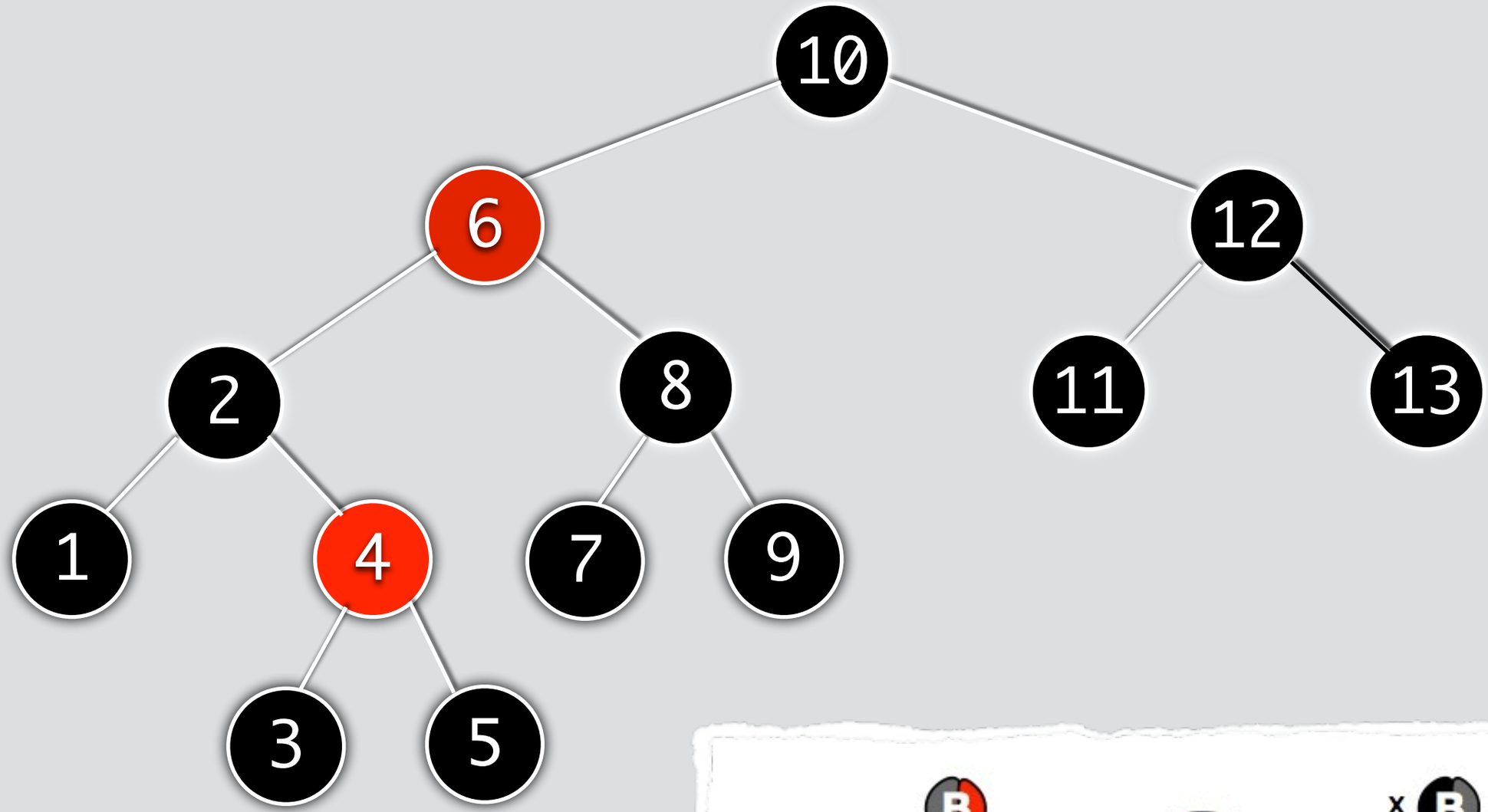


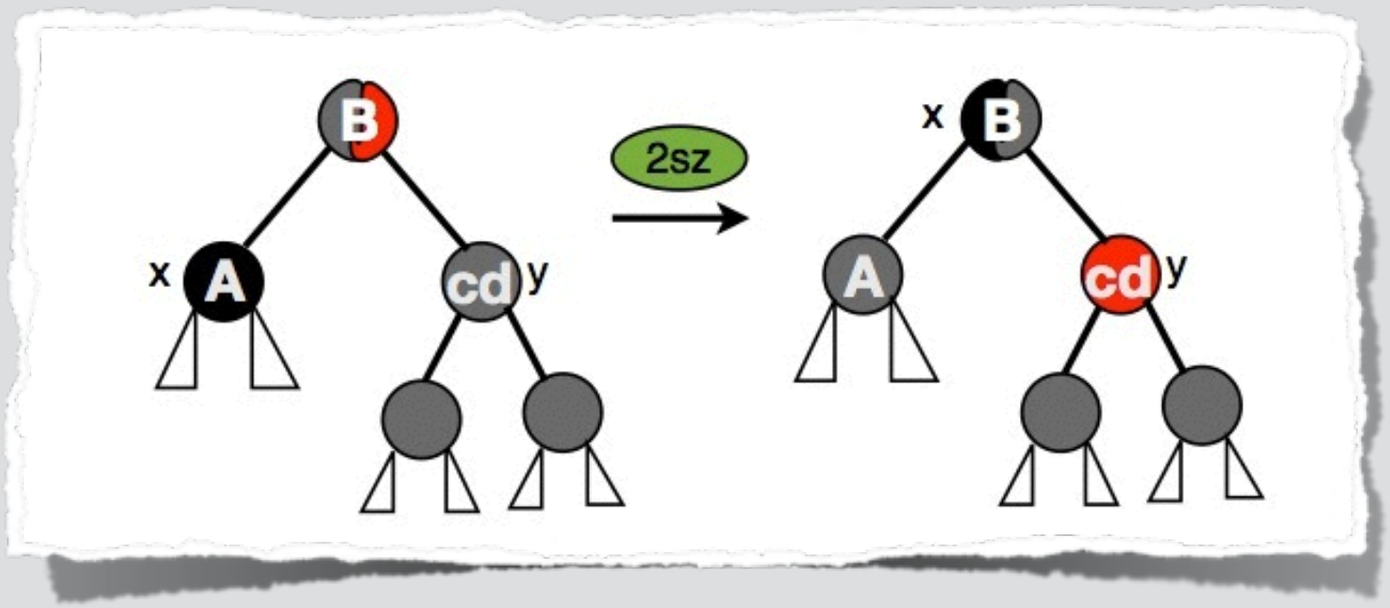
Törlés utáni javítás

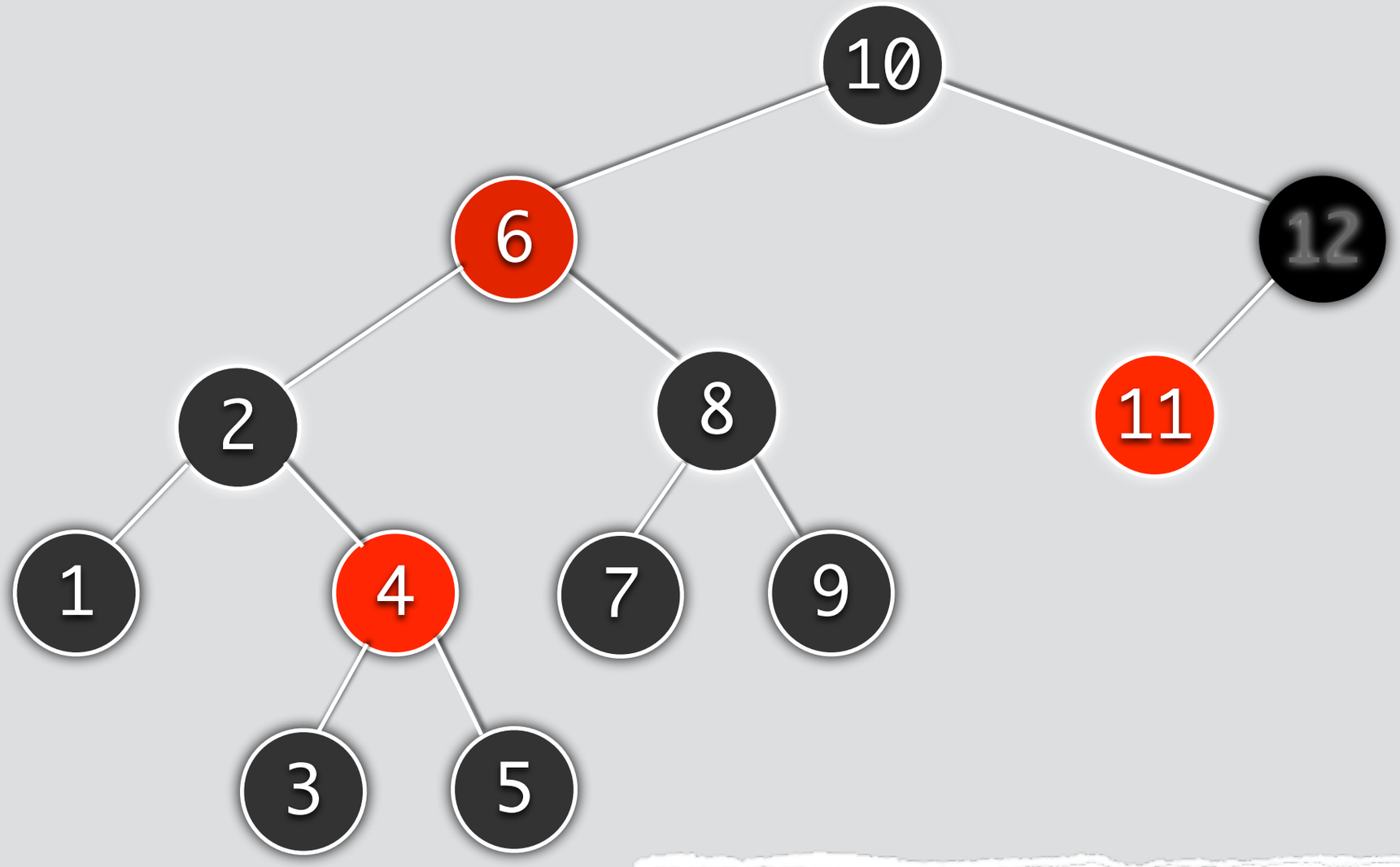
1. eset: testvér piros -> nagybácsi és testvér fekete, apa piros (forgat is)
2. eset: testvér fekete
   (sz) mindkét unokaöccs fekete -> kész vagy probléma 1 szinntel feljebb (csak szinez)
   (fa) csak a jobb unokaöccs fekete -> jobb unokaöccs piros (forgat is)
   (fb) jobb unokaöccs piros -> kész (forgat is)

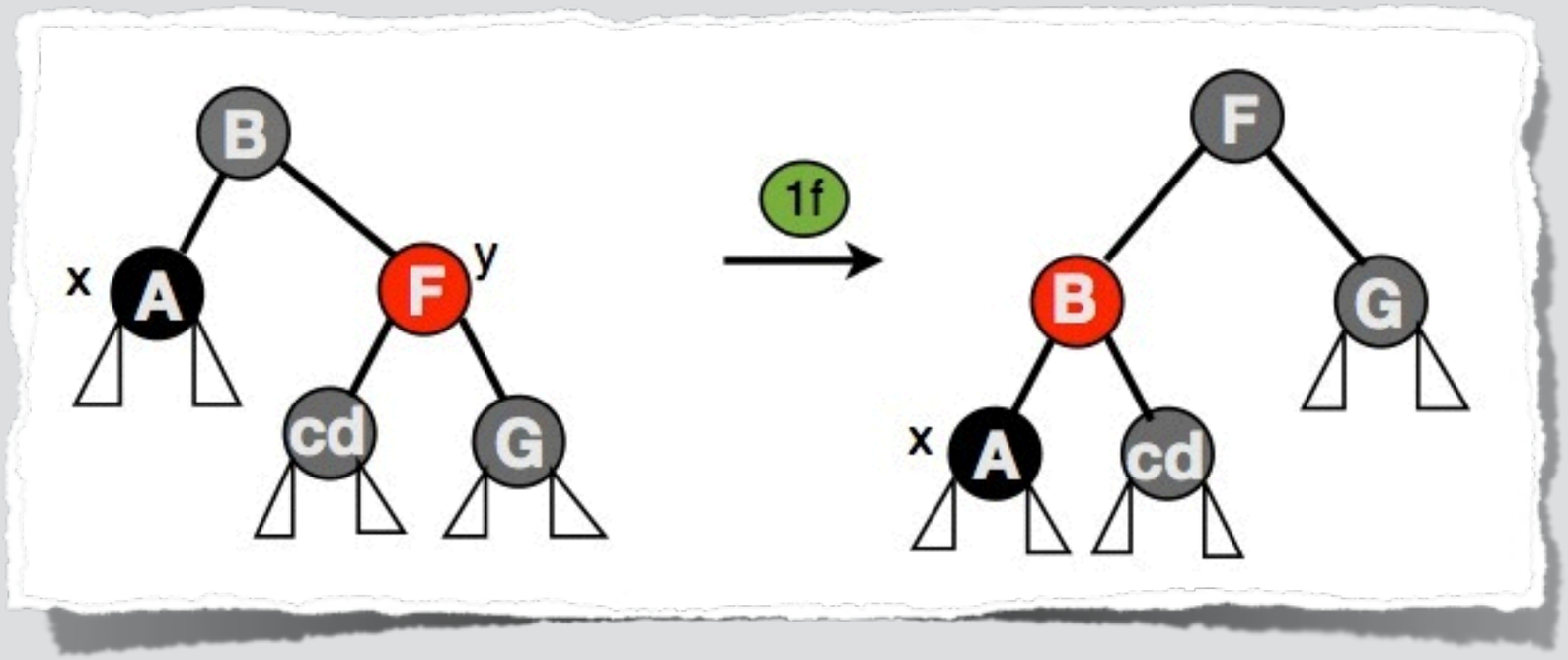


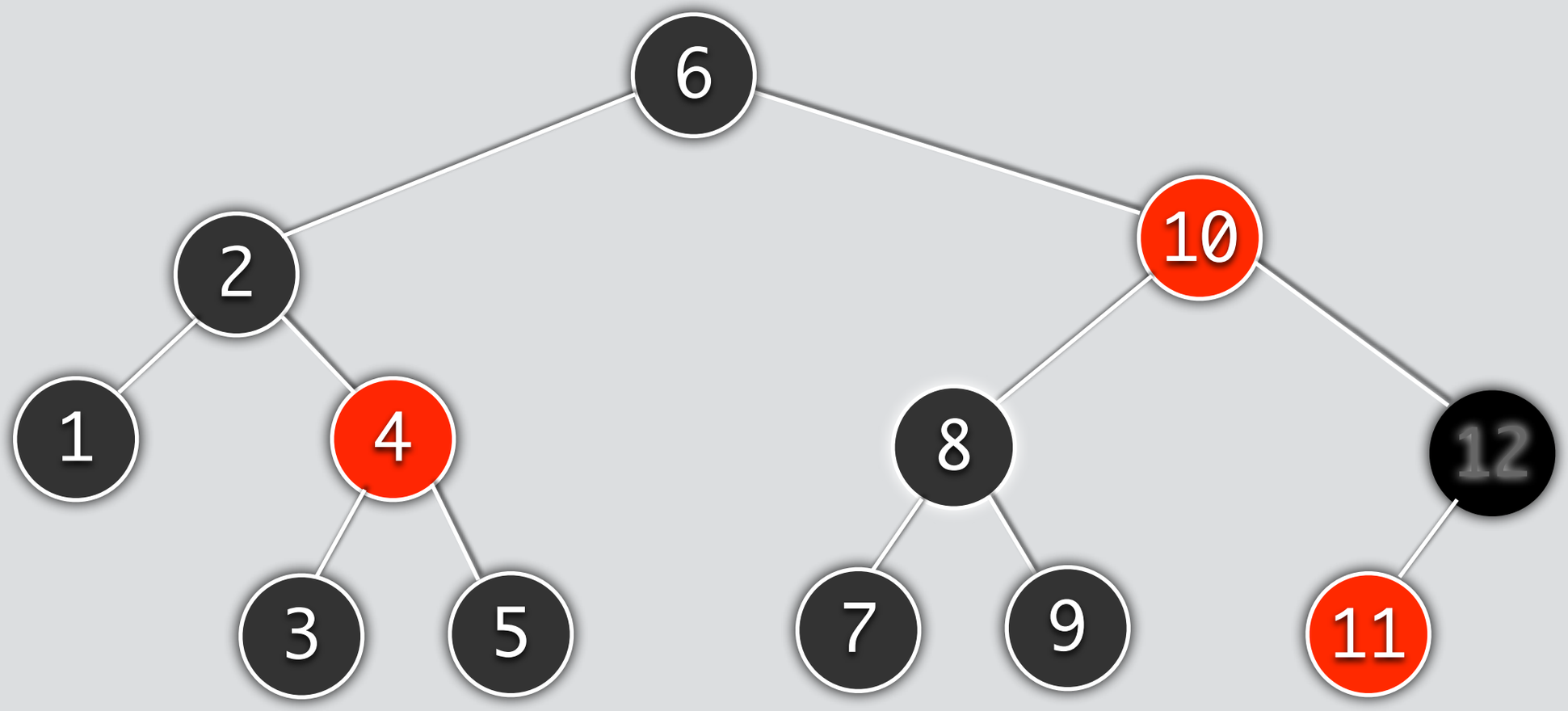
O(logn)

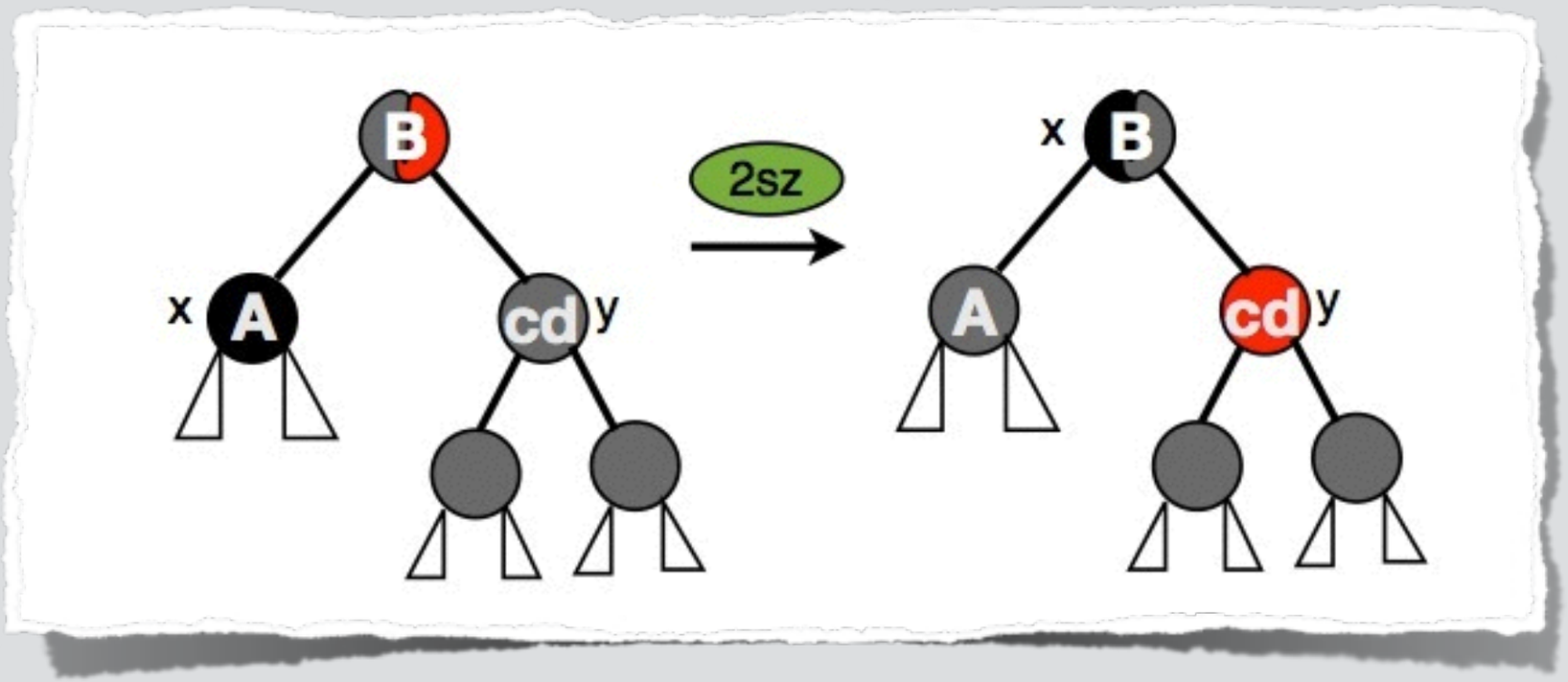

10
6
12
2
8
11
13
1
4
7
9
3
5

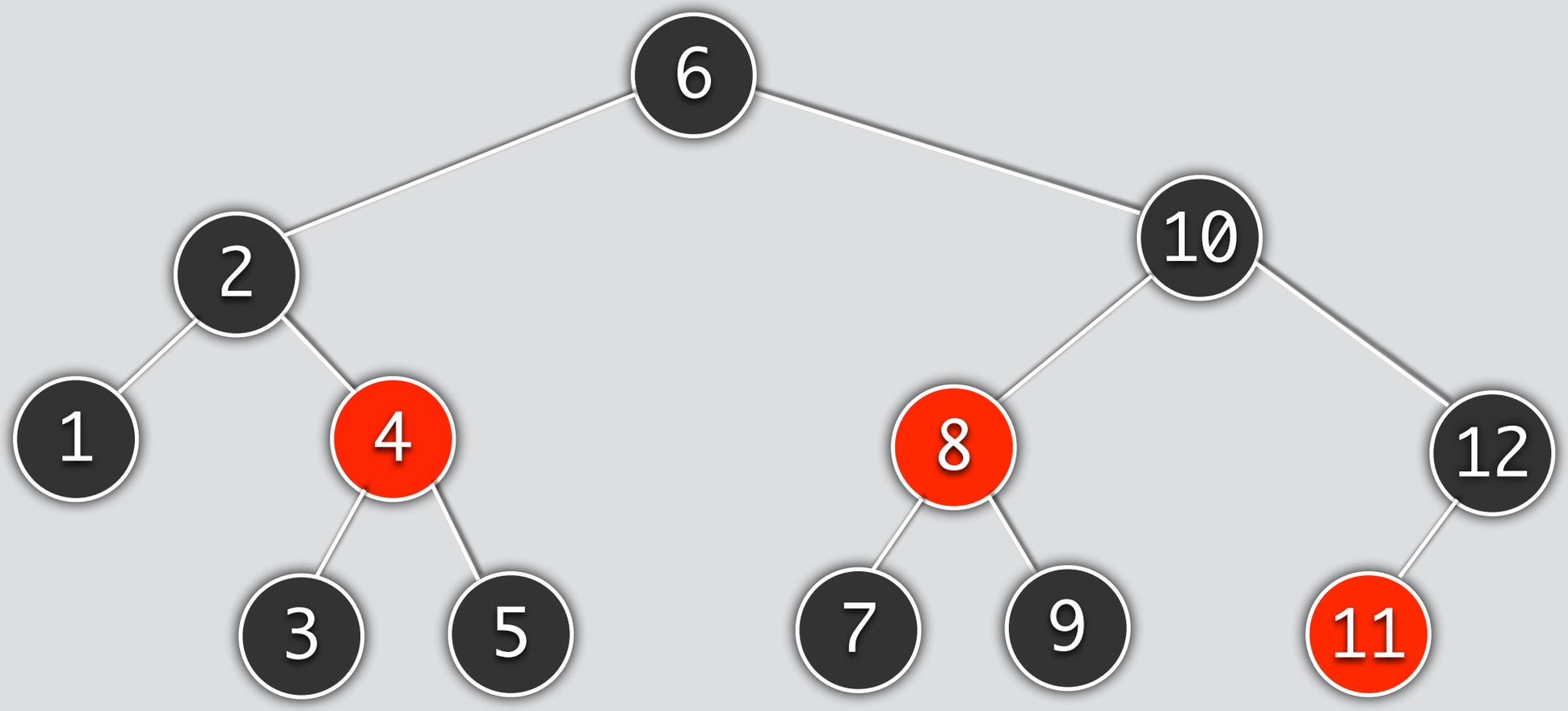

B
x
A
cd
y
2sz
x
B
A
cd
y

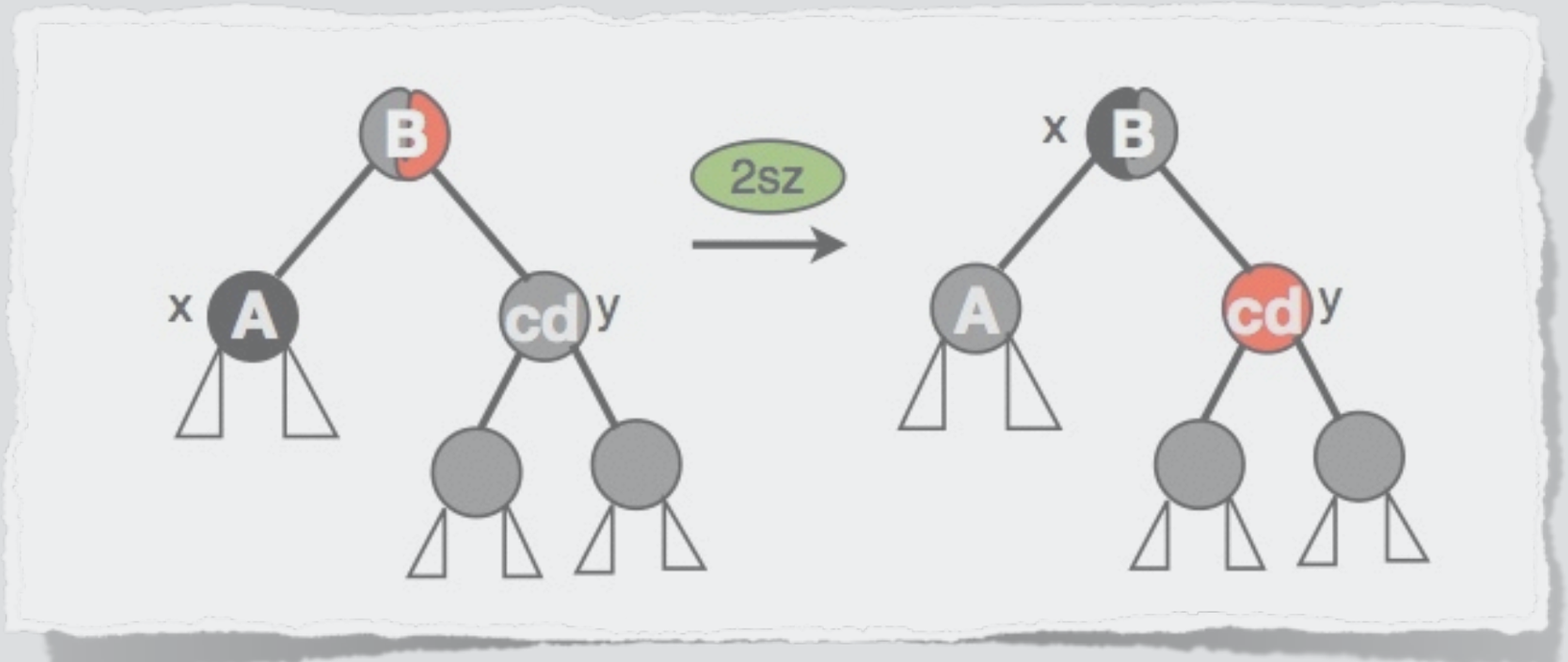

10
6
12
2
8
11
1
4
7
9
3
5

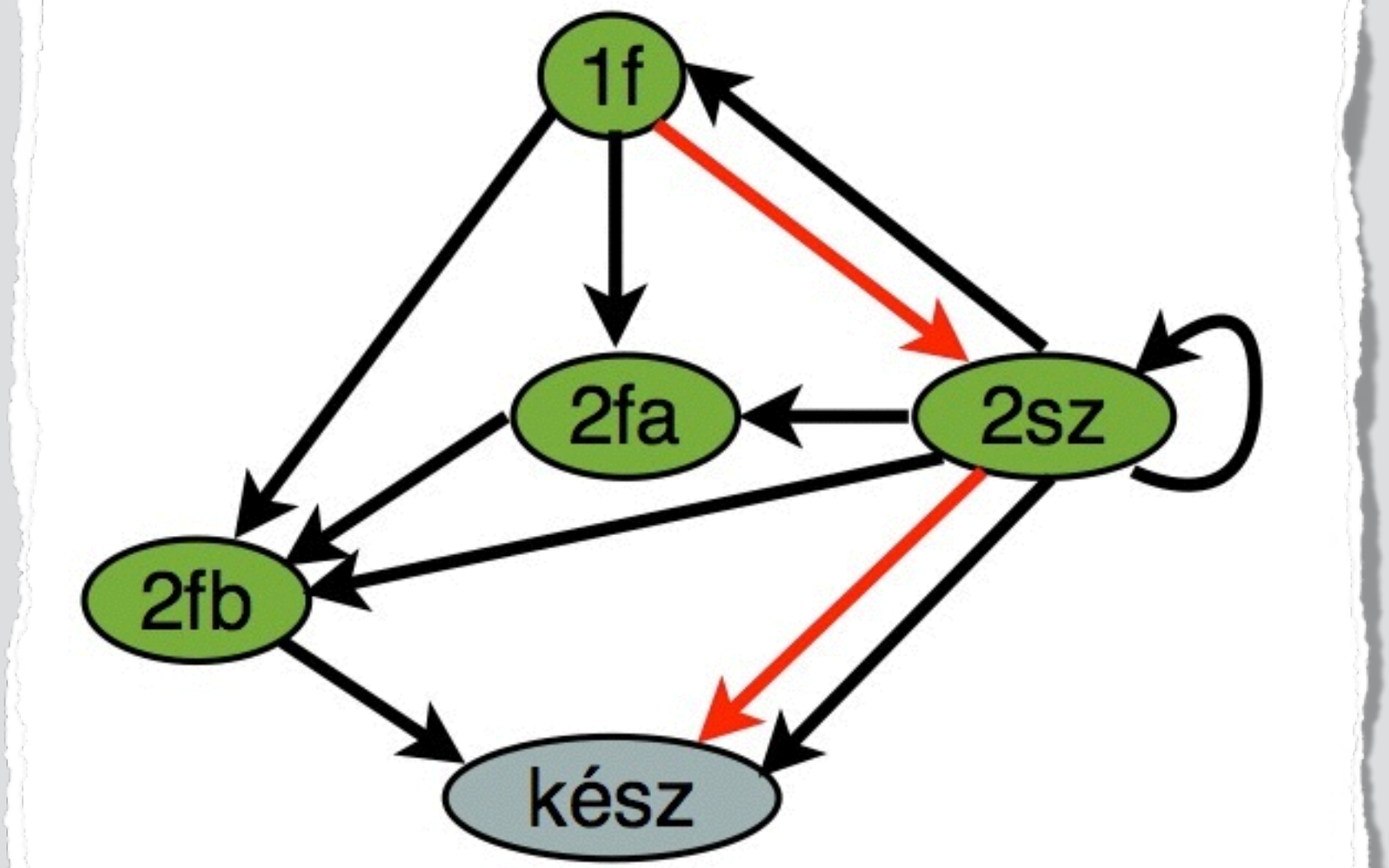

B
x
A
F
y
cd
G
1f
F
B
G
x
A
cd

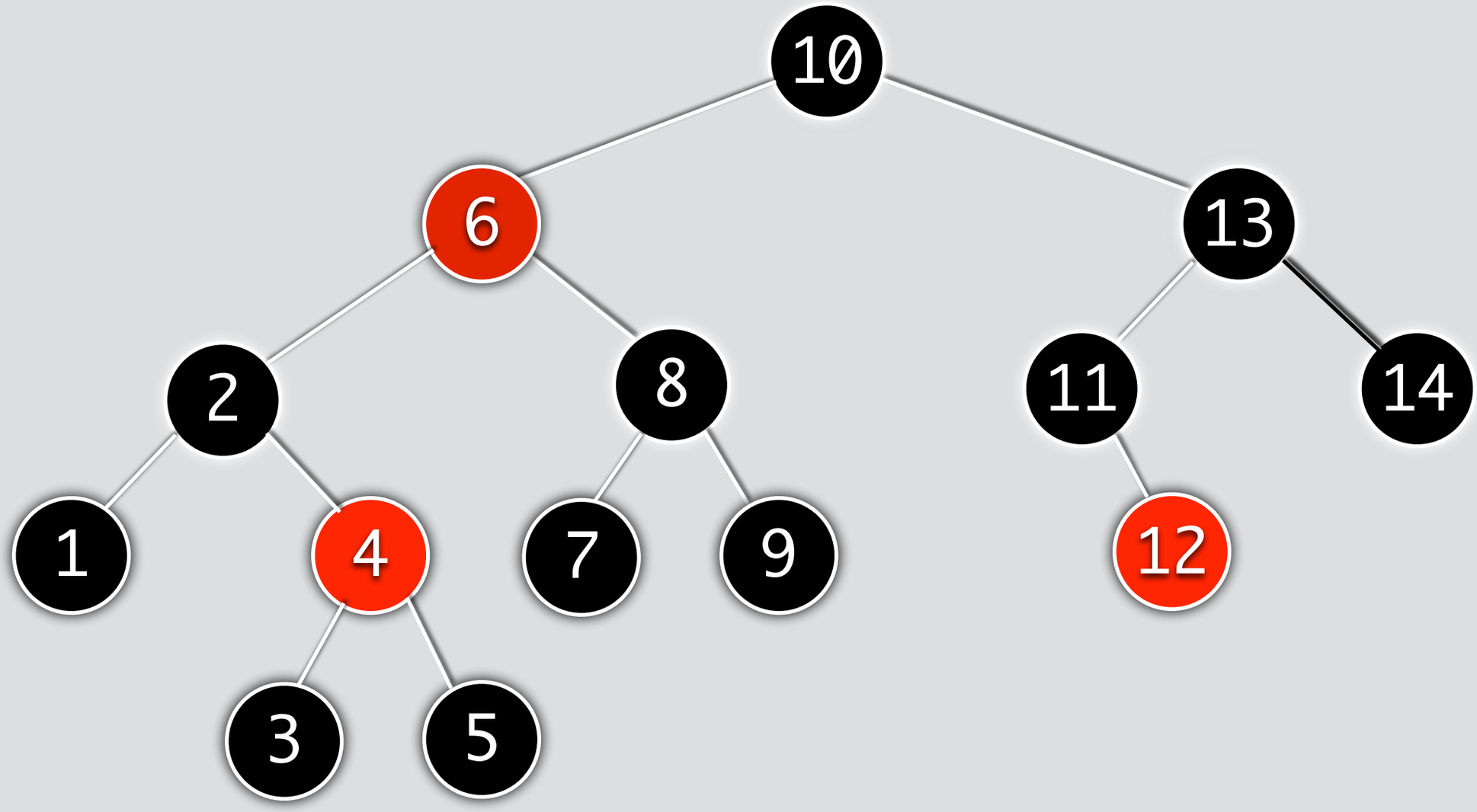

6
2
10
1
4
8
12
3
5
7
9
11

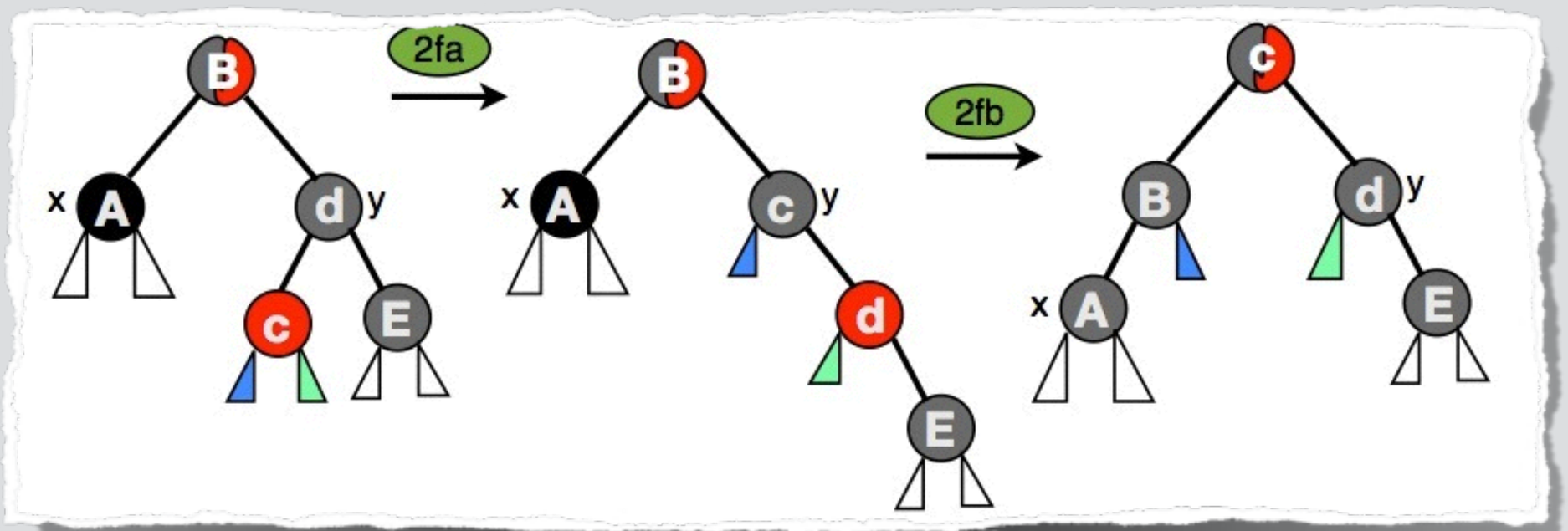

B
x
A
cd
y
2sz
x
B
A
cd
y

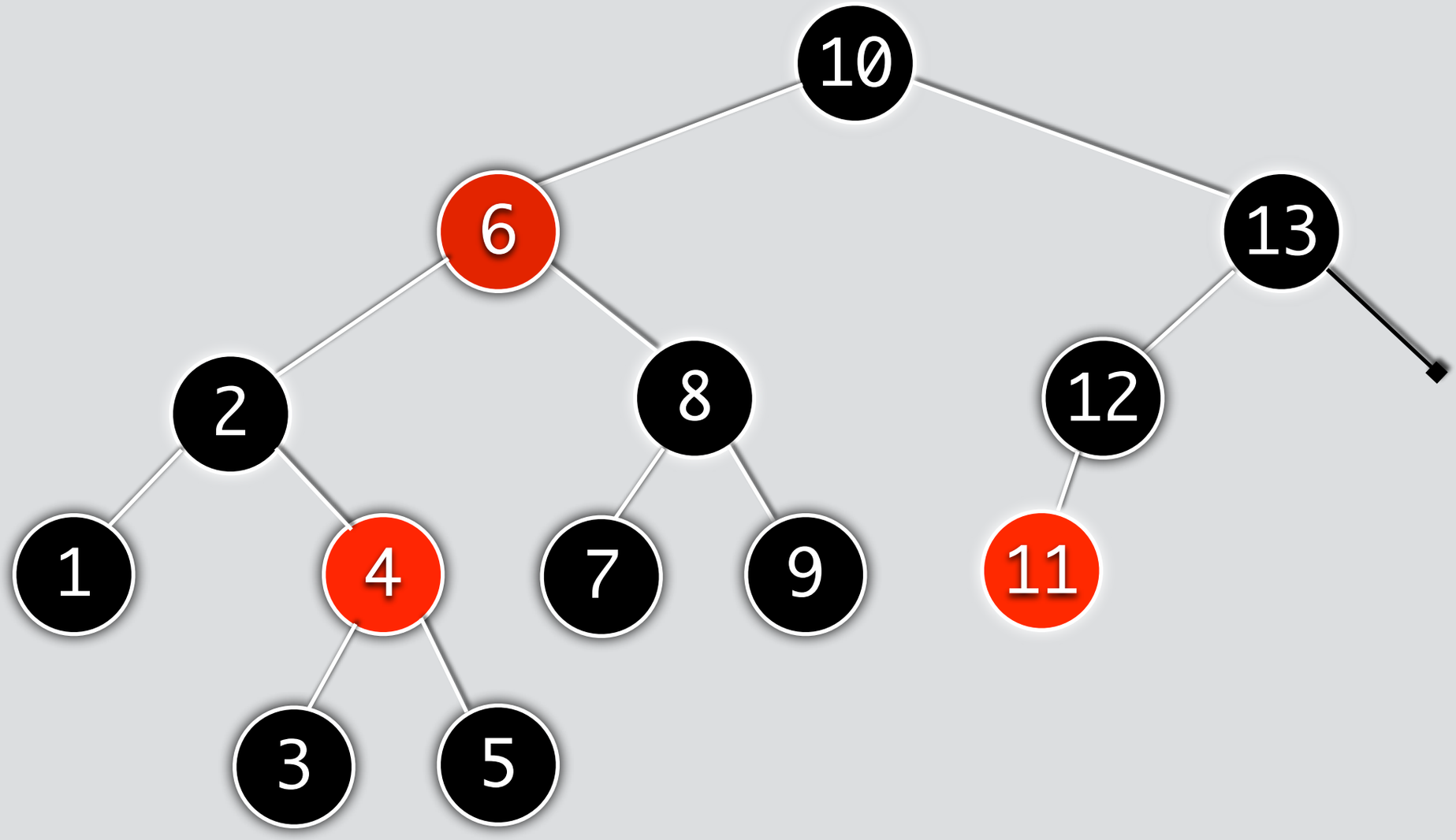

6
2
10
1
4
8
12
3
5
7
9
11

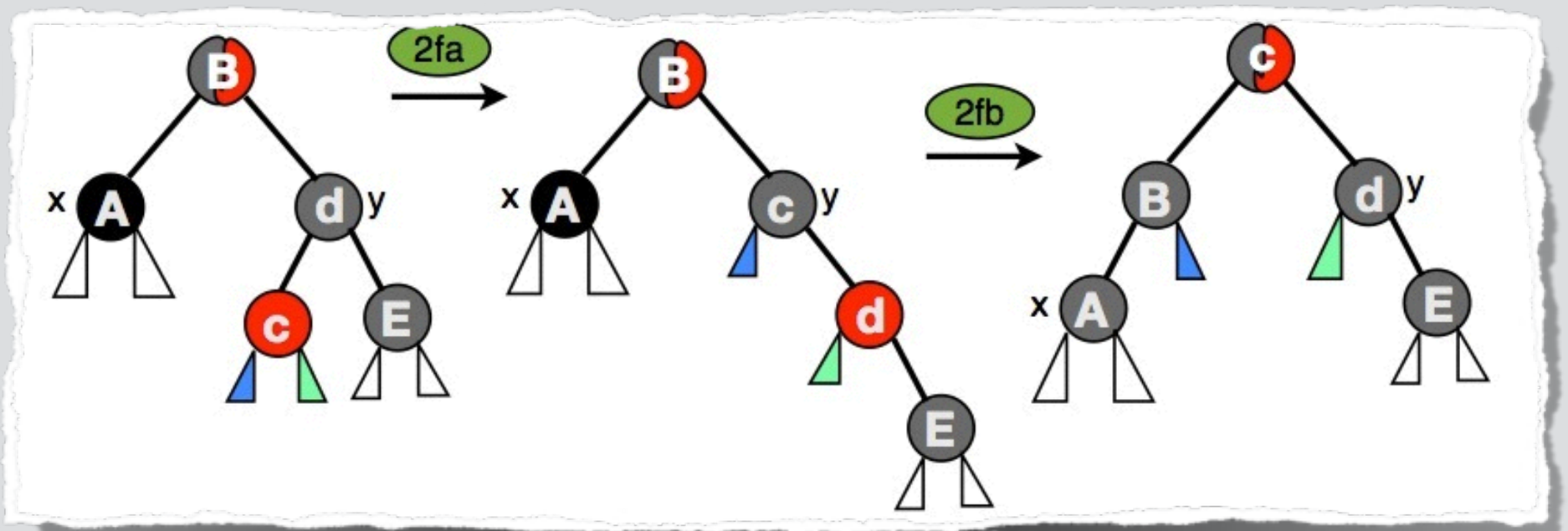

B
x
A
cd
y
2sz
x
B
A
cd
y

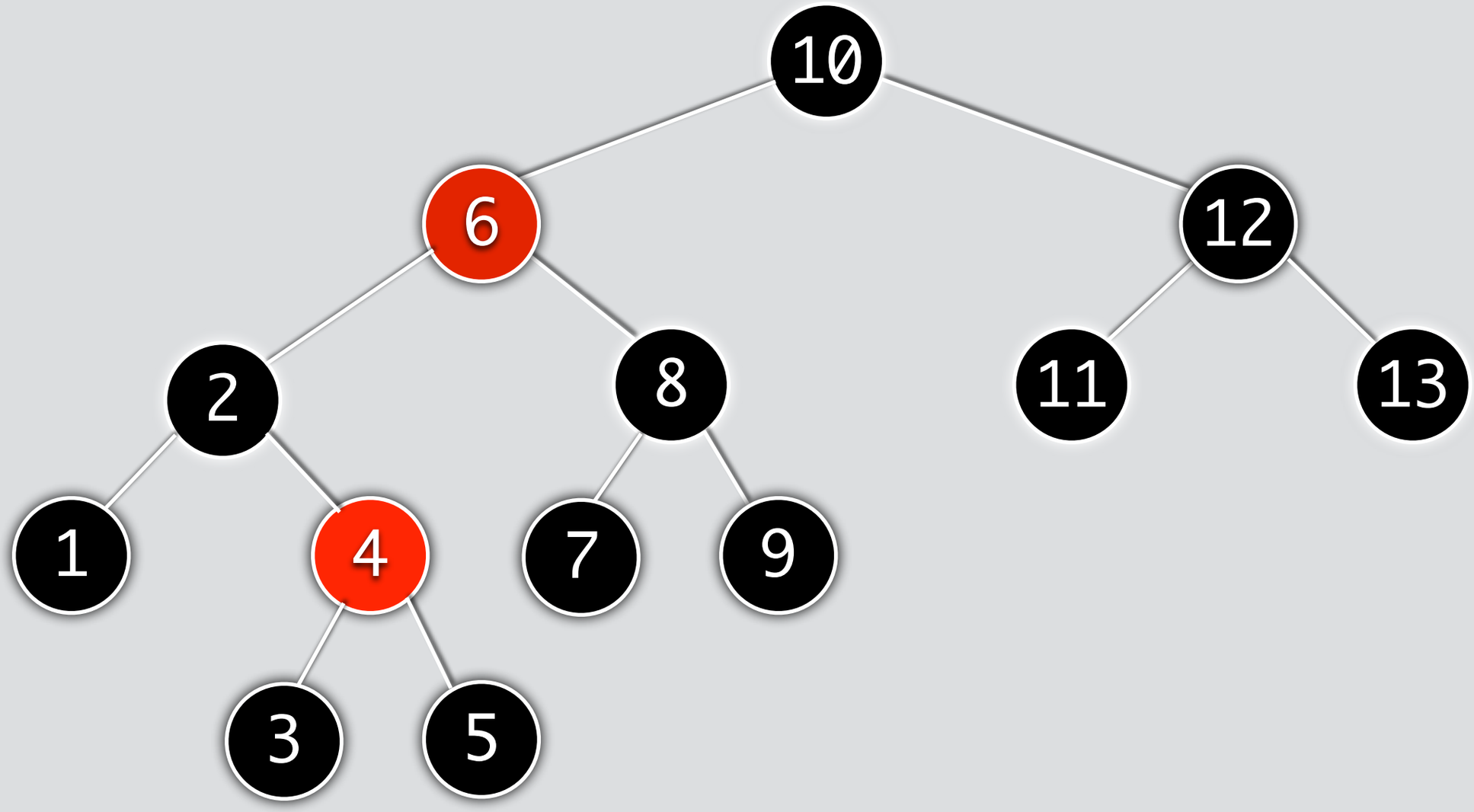

1f
2fa
2sz
2fb
kész

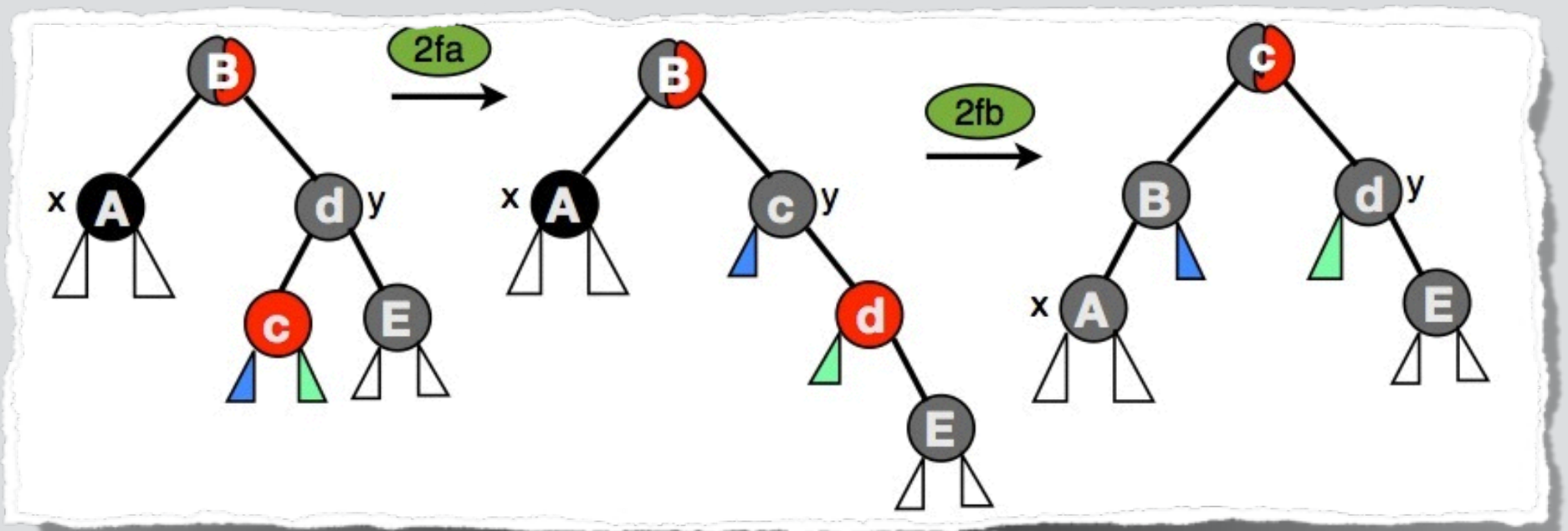

10
6
13
2
8
11
14
1
4
7
9
12
3
5

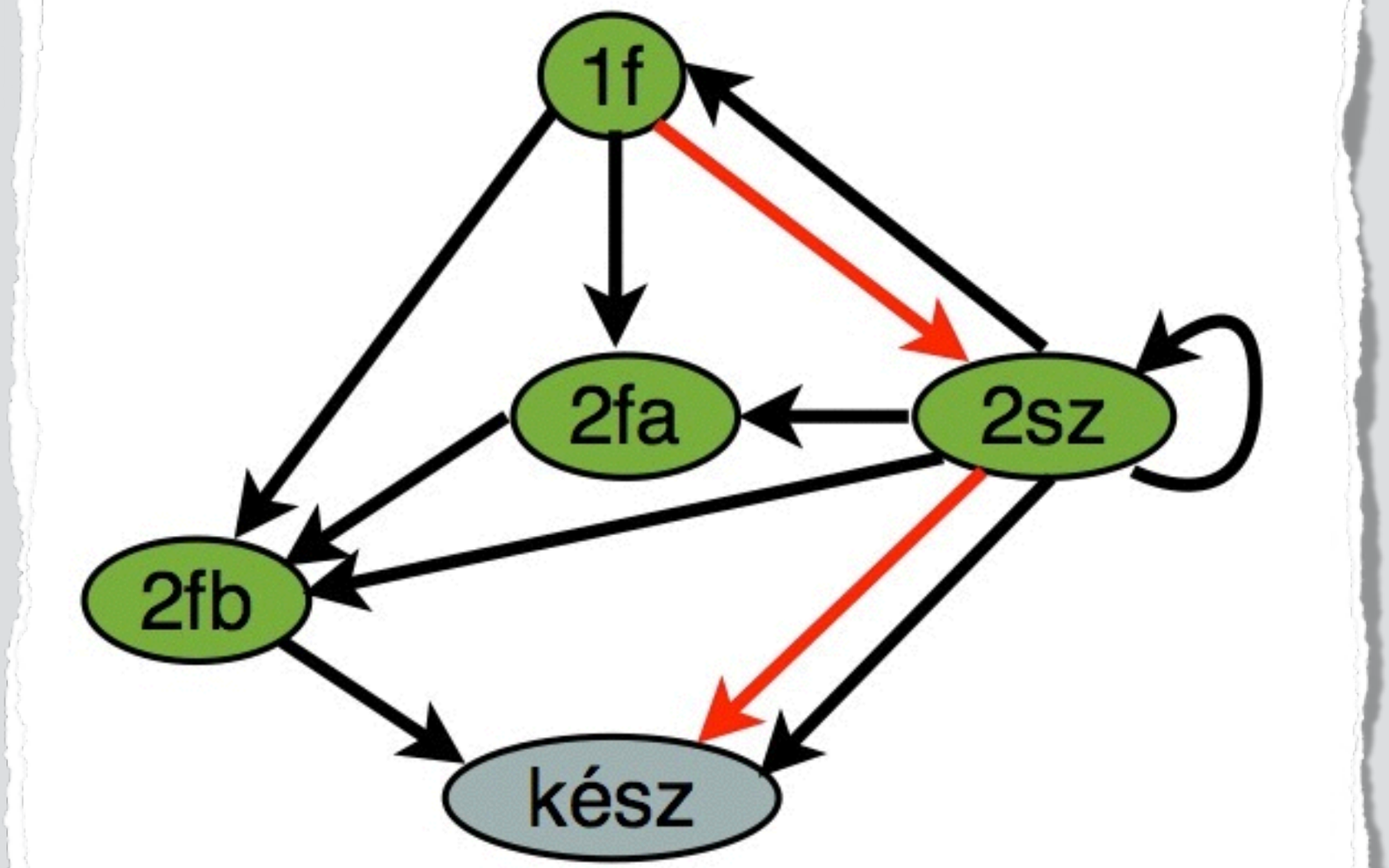

2fa
2fb
x
A
B
d
y
c
E
x
A
B
c
y
d
E
x
A
c
B
d
y
E

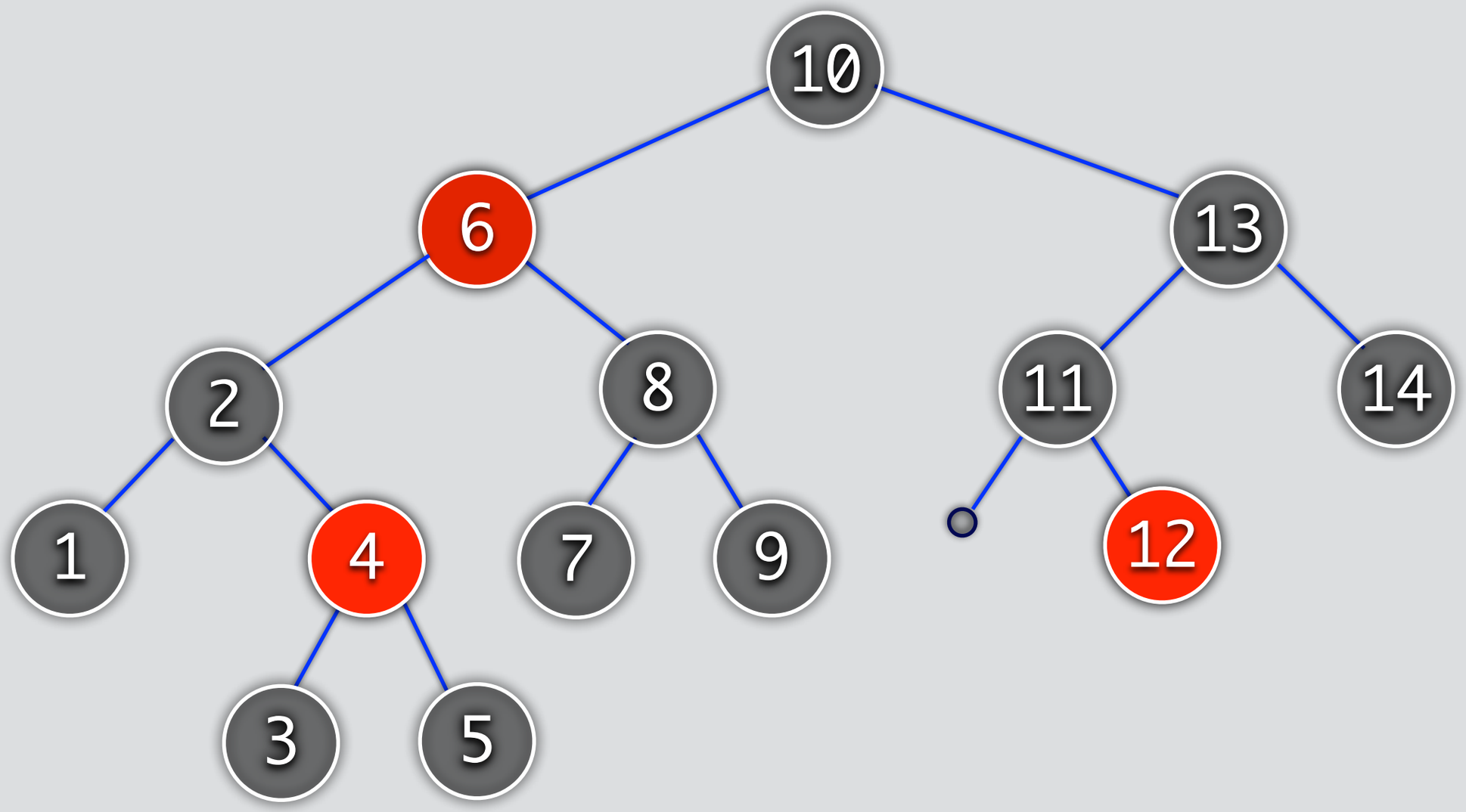

10
6
13
2
8
12
1
4
7
9
11
3
5

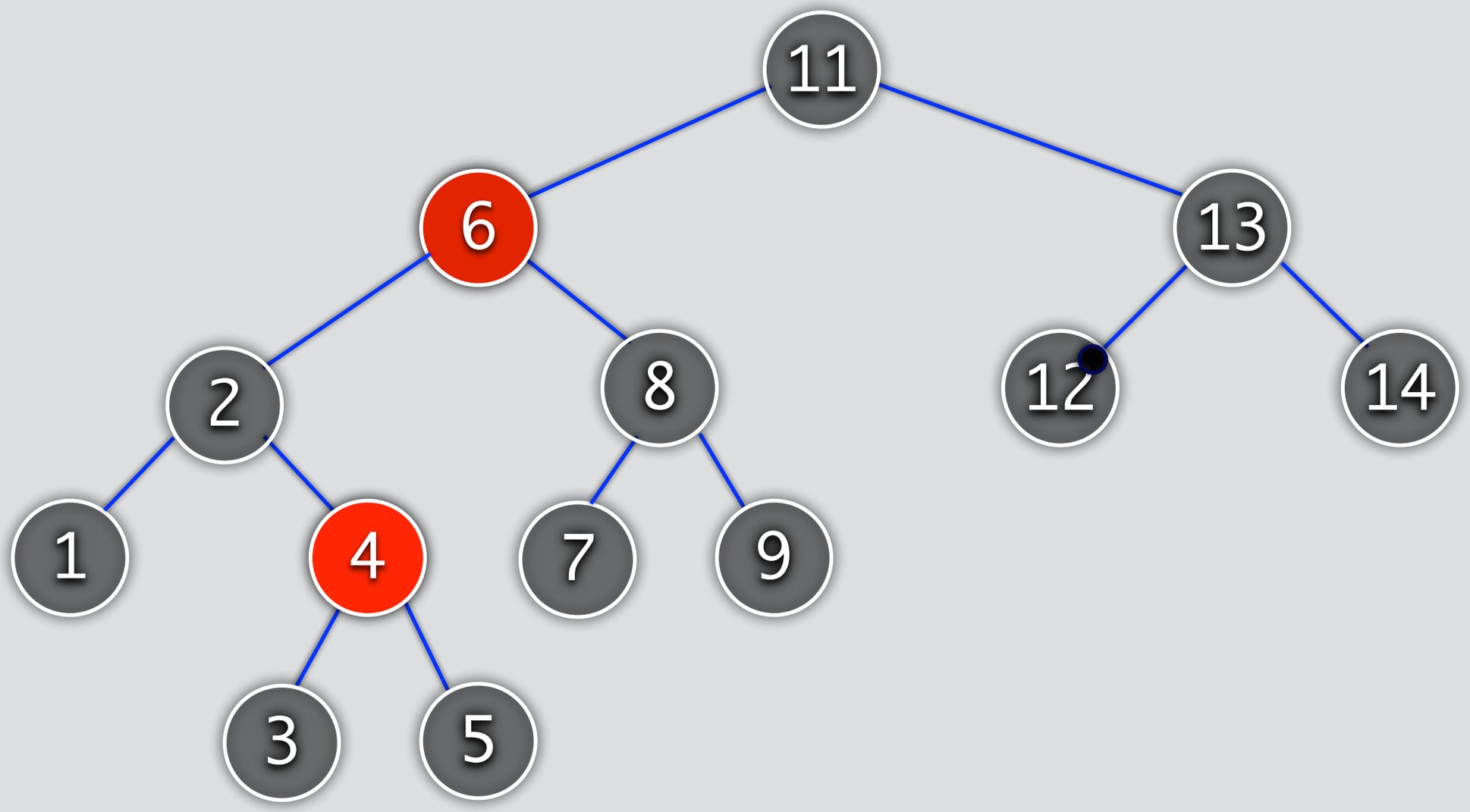

B
x
A
d
y
c
E
2fa
B
x
A
c
y
d
E
2fb
c
B
d
y
x
A
E

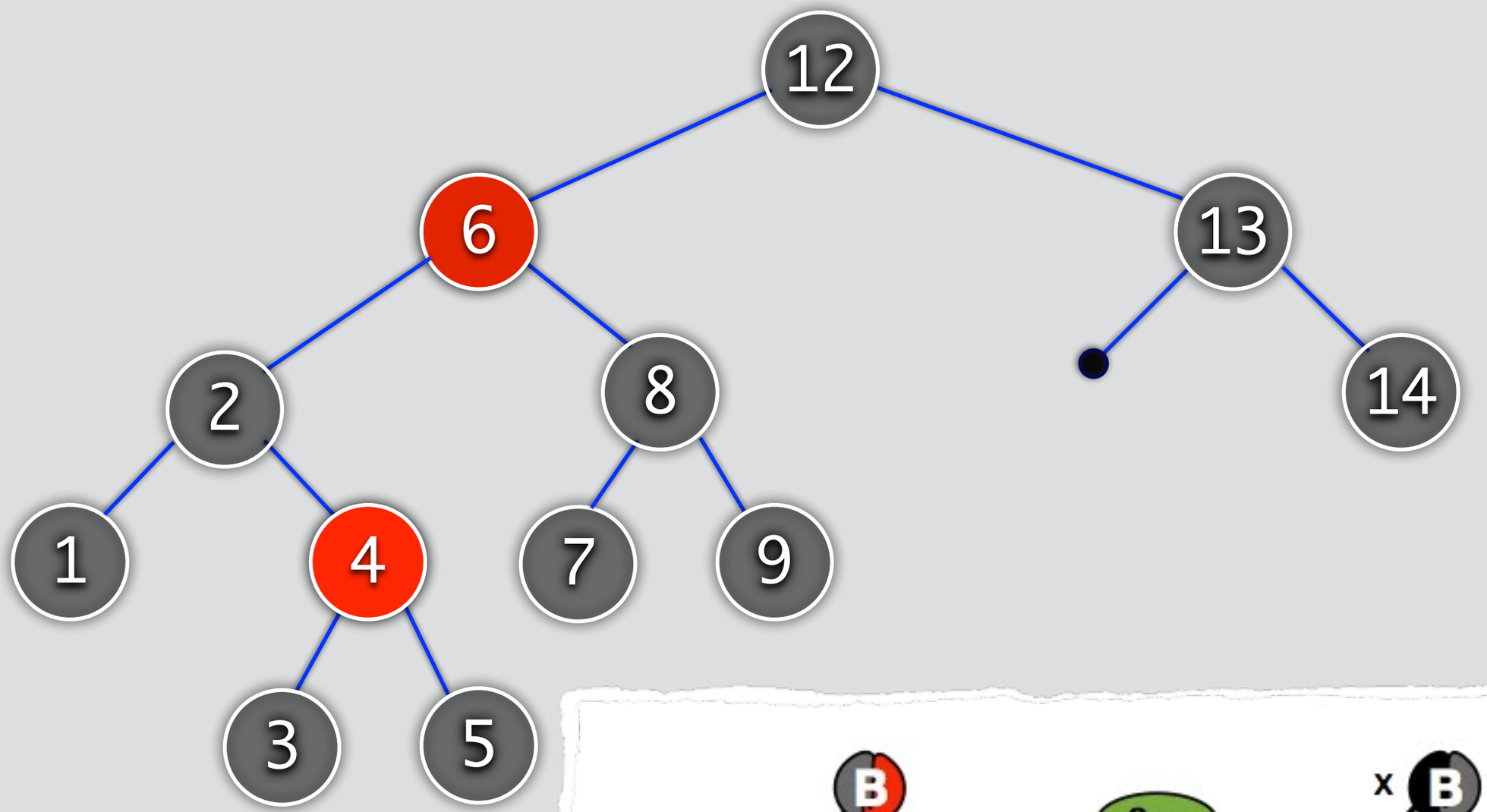

10
6
12
2
8
11
13
1
4
7
9
3
5

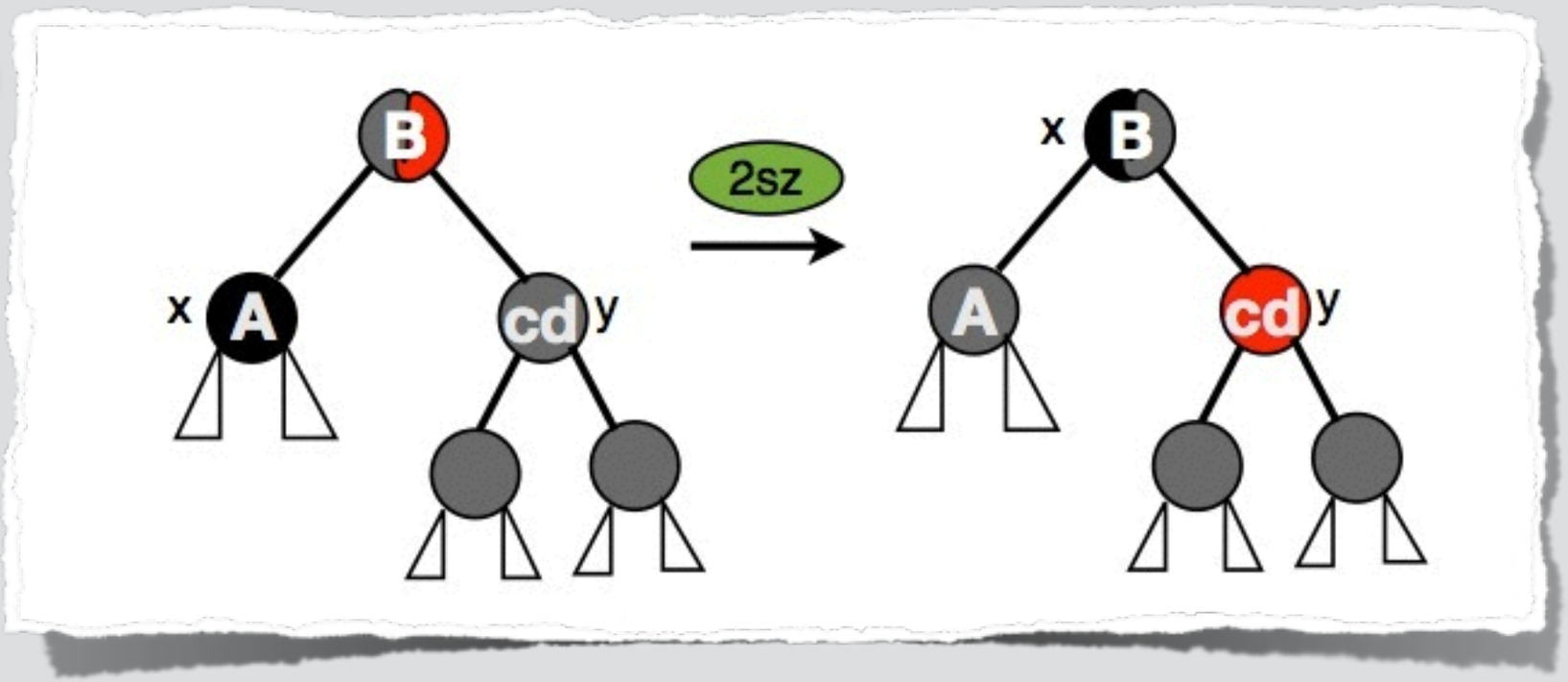

B
x
A
d
y
c
E
2fa
B
x
A
c
y
d
E
2fb
c
B
d
y
x
A
E

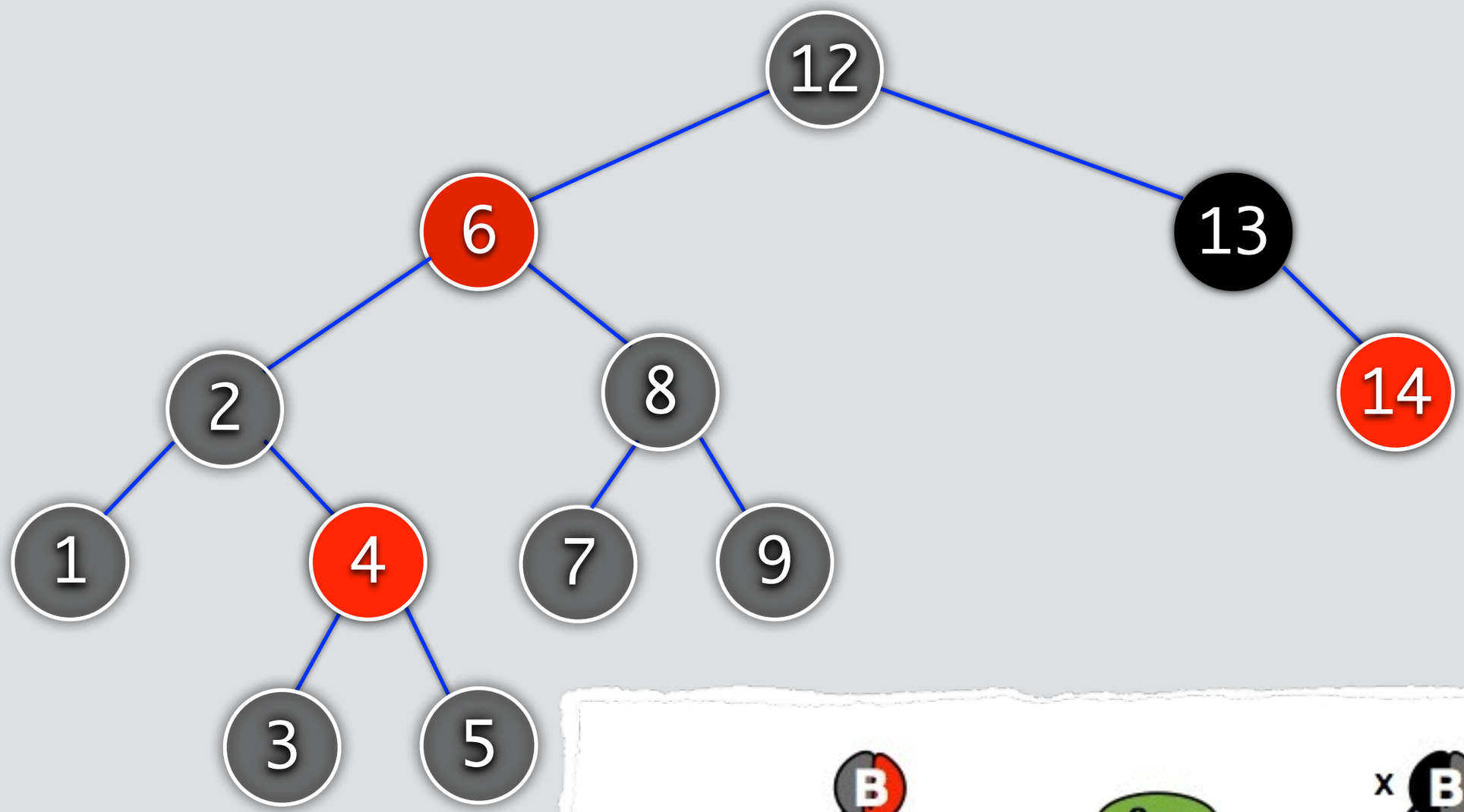

1f
2fa
2sz
2fb
kész

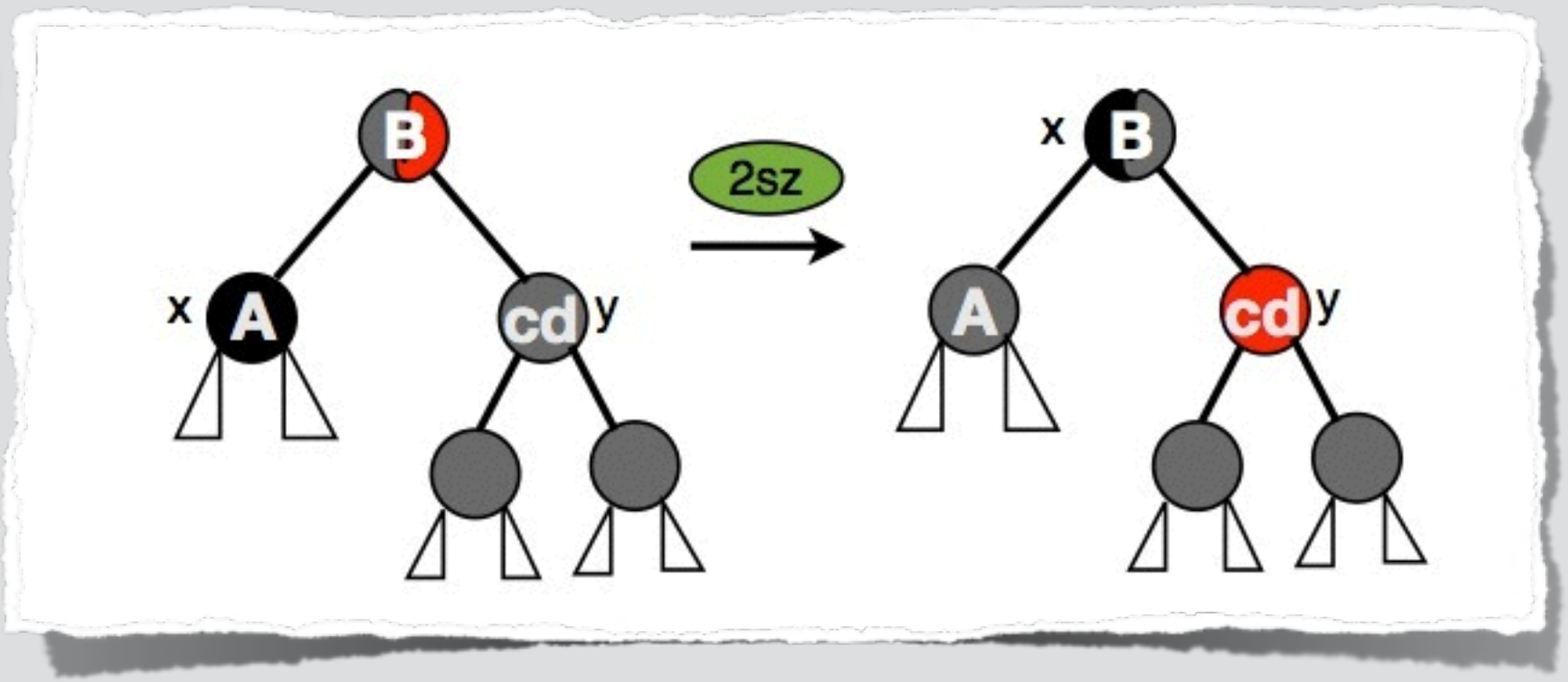

10
6
13
2
8
11
14
1
4
7
9
12
3
5

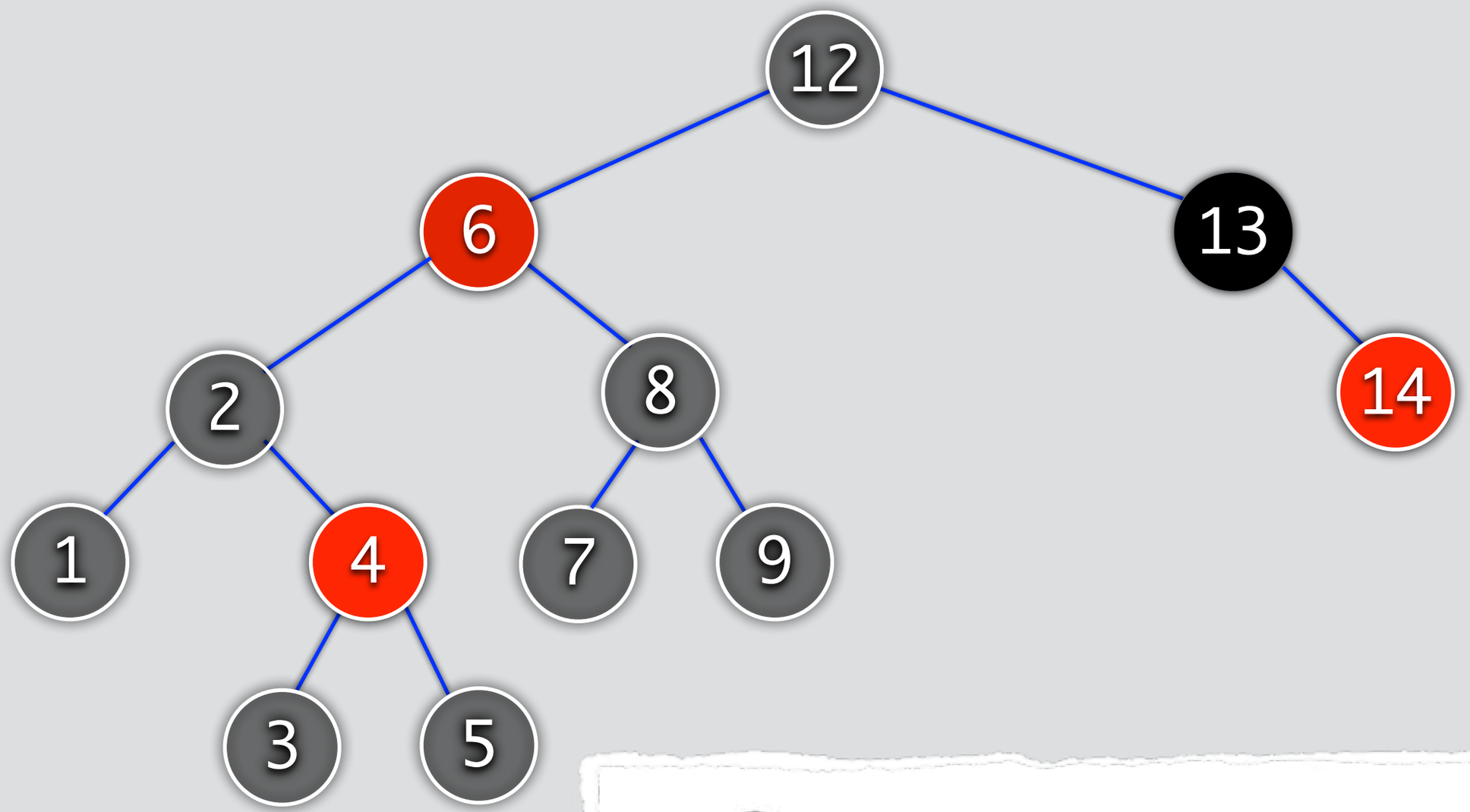

11
6
13
2
8
12
14
1
4
7
9
3
5

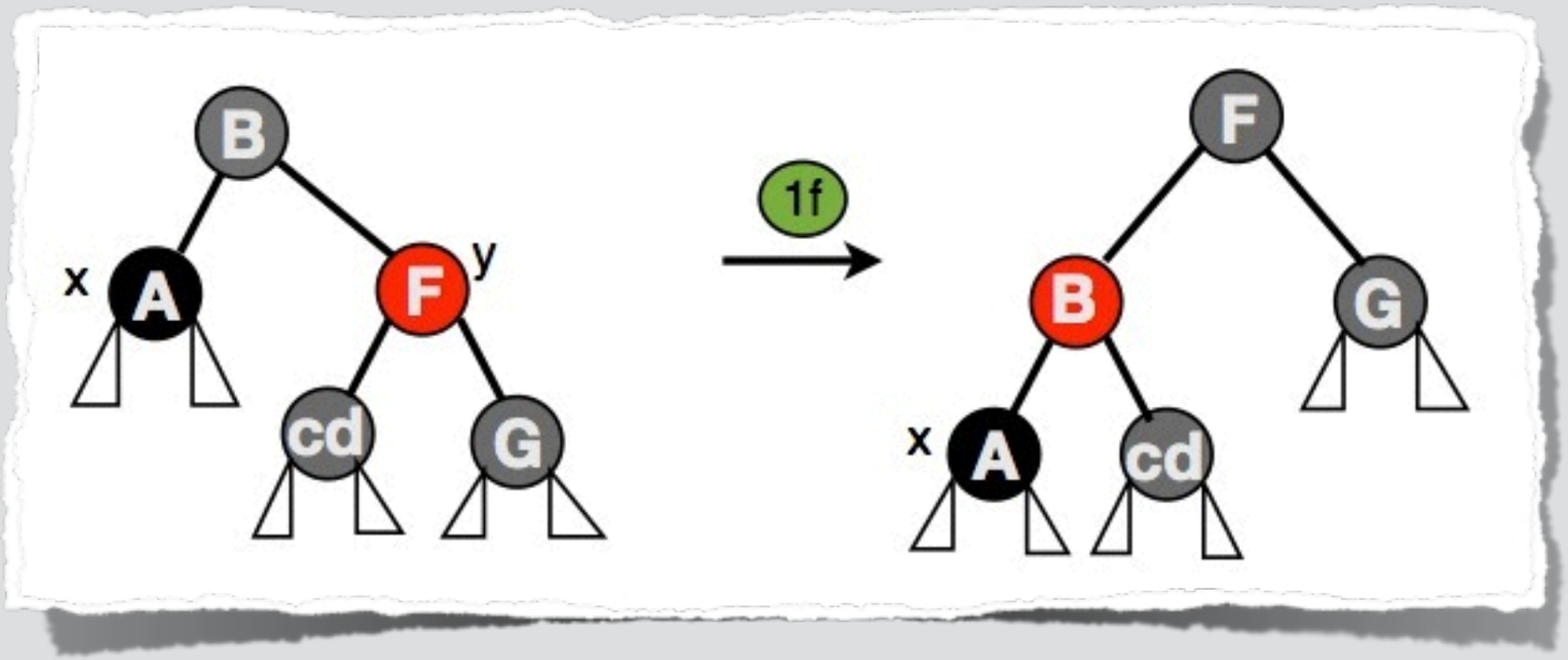

12
6
13
2
8
14
1
4
7
9
3
5

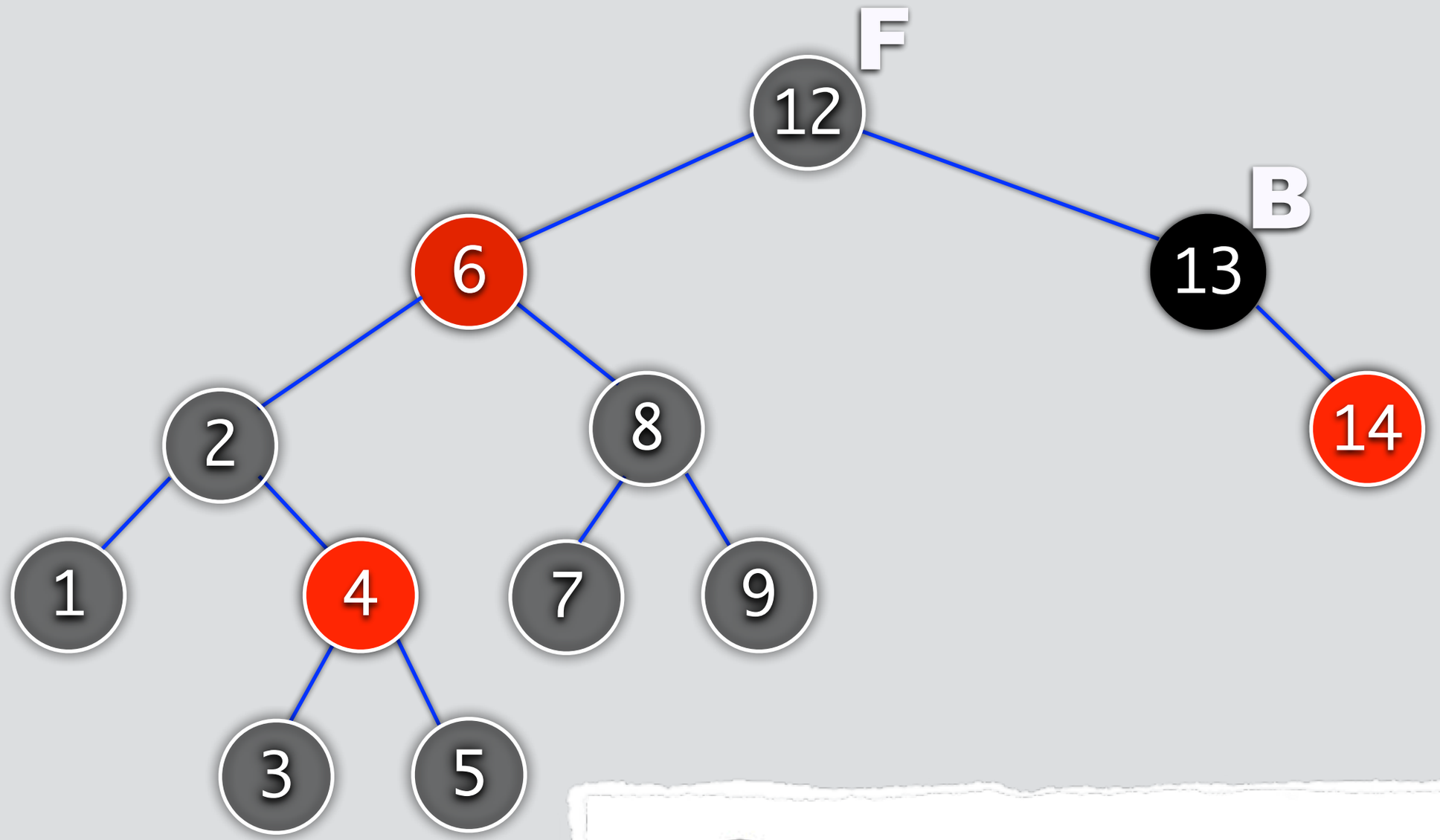

B
x
A
cd
y
2sz
x
B
A
cd
y

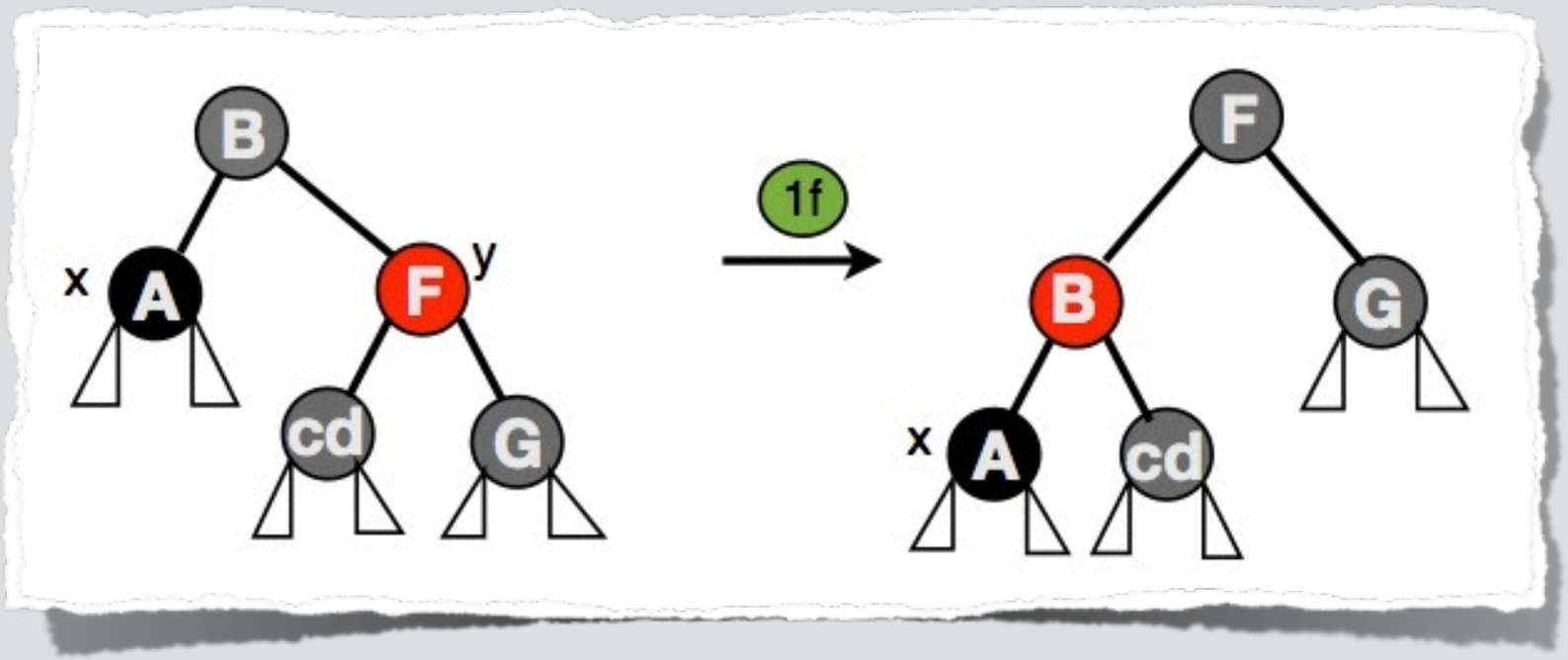

12
6
13
2
8
14
1
4
7
9
3
5

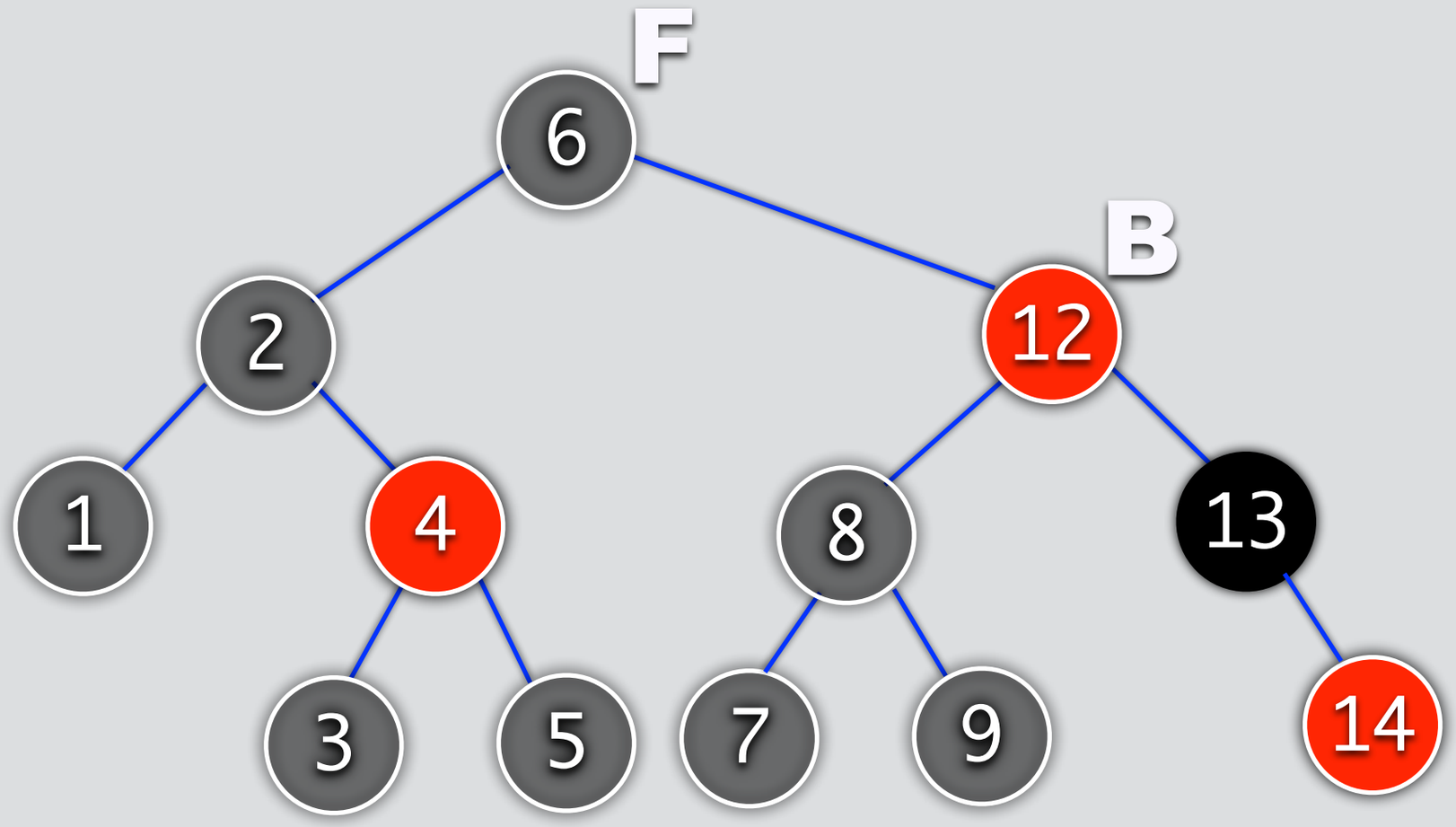

B
x
A
cd
y
2sz
x
B
A
cd
y

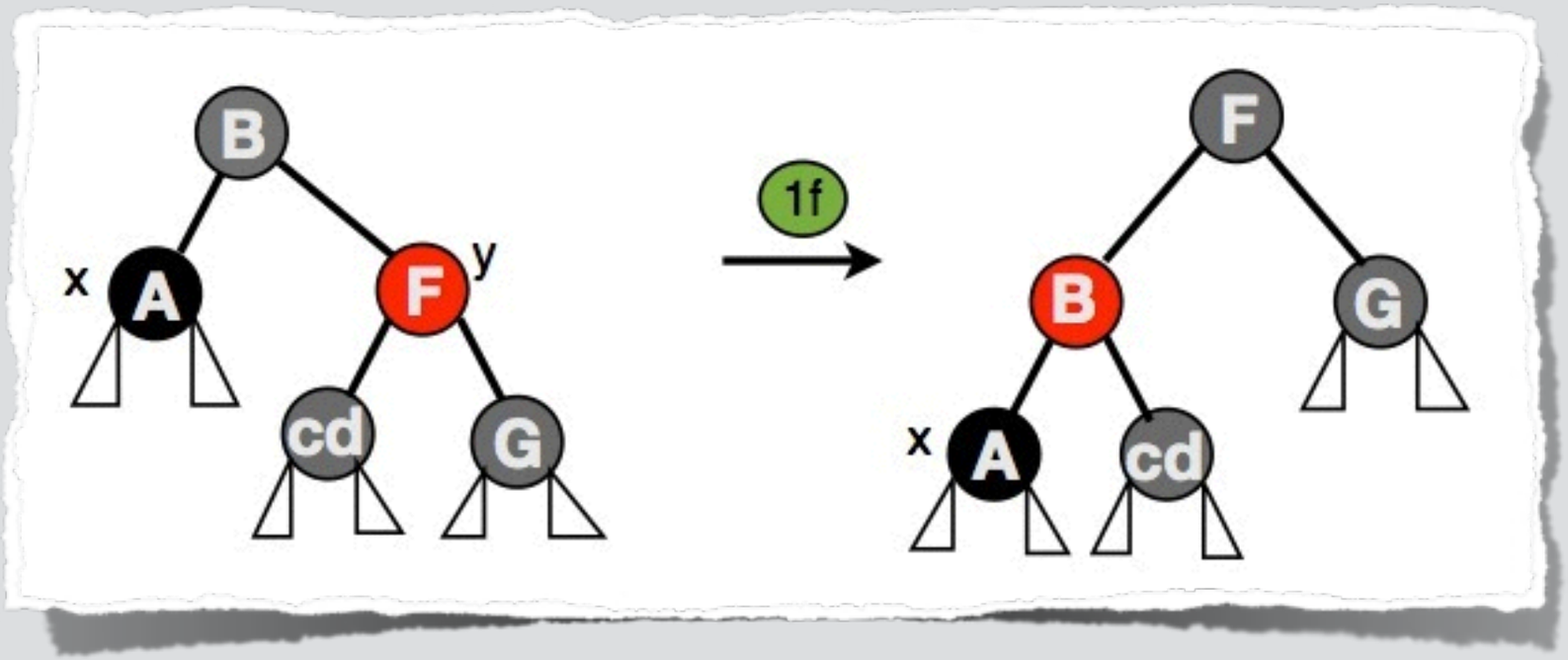

12
6
13
2
8
14
1
4
7
9
3
5

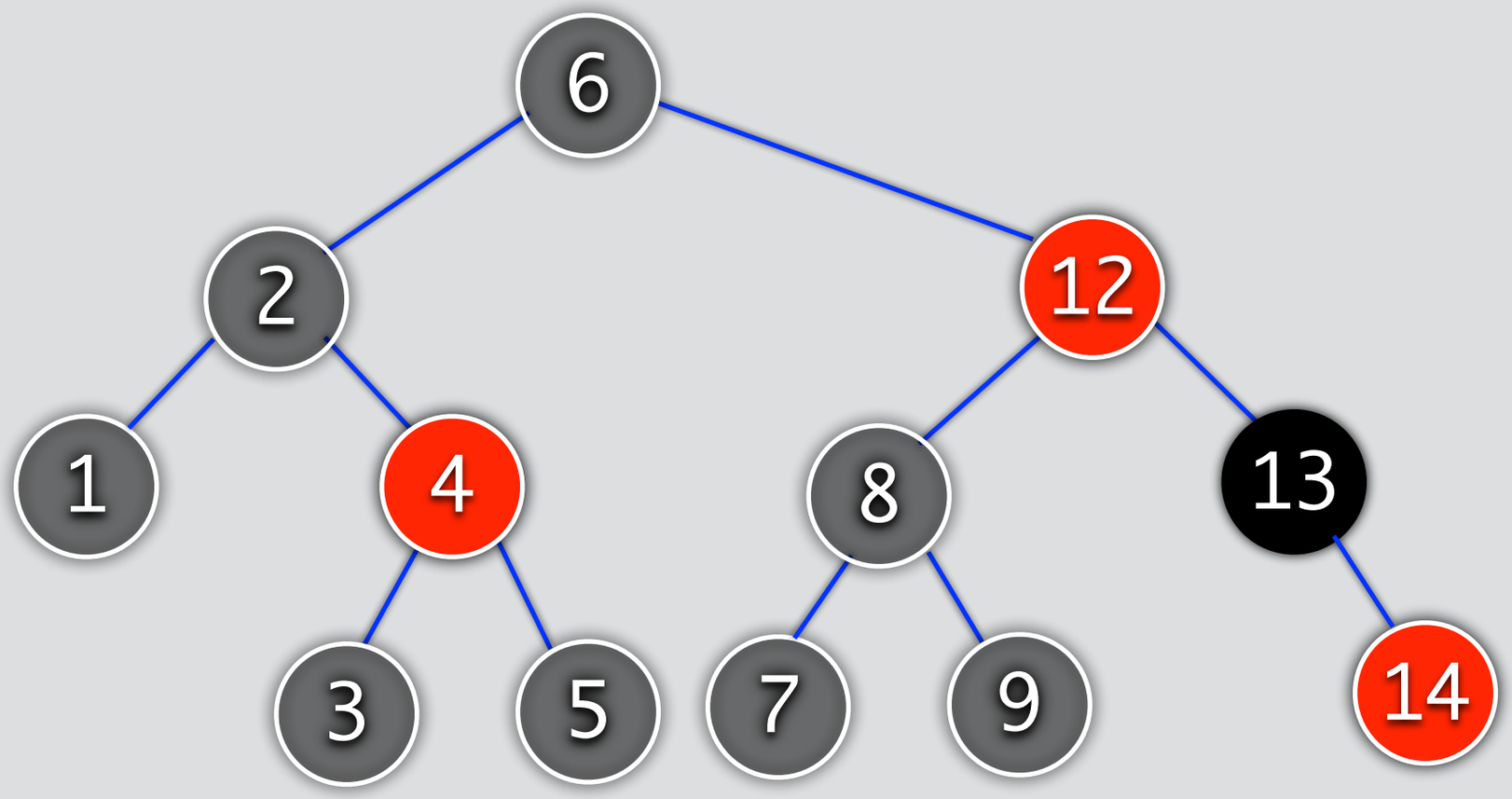

B
x
A
F
y
cd
G
1f
F
B
G
x
A
cd

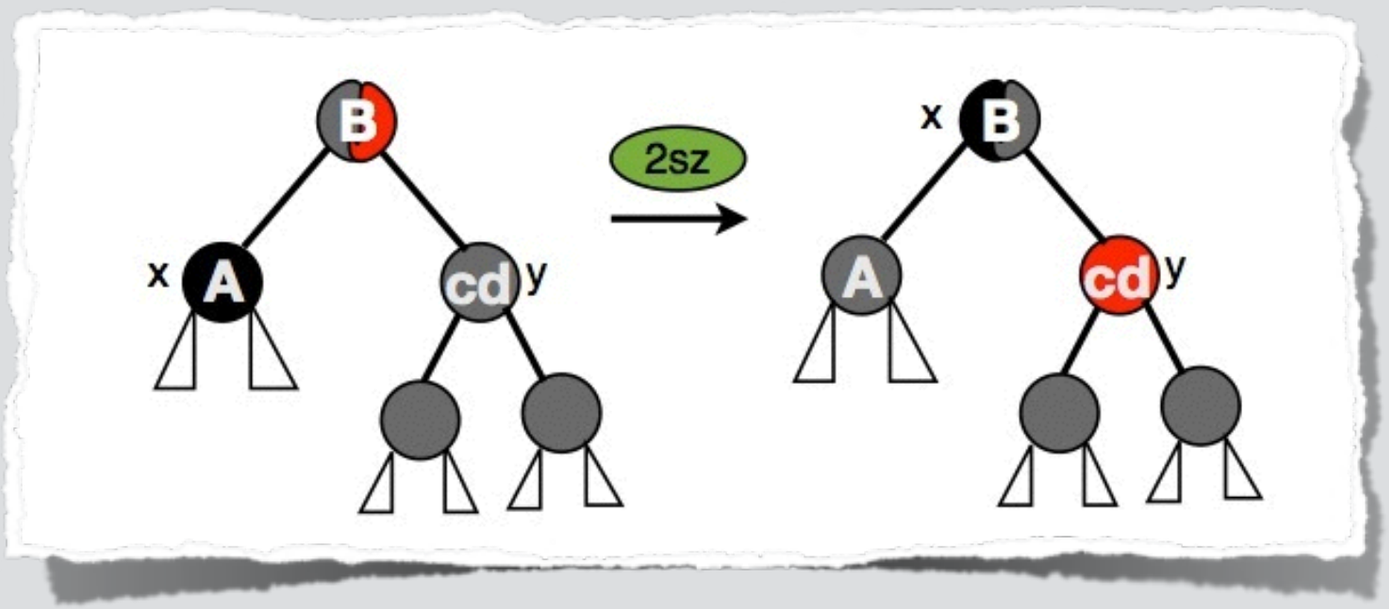

F
12
B
6
13
2
8
14
1
4
7
9
3
5

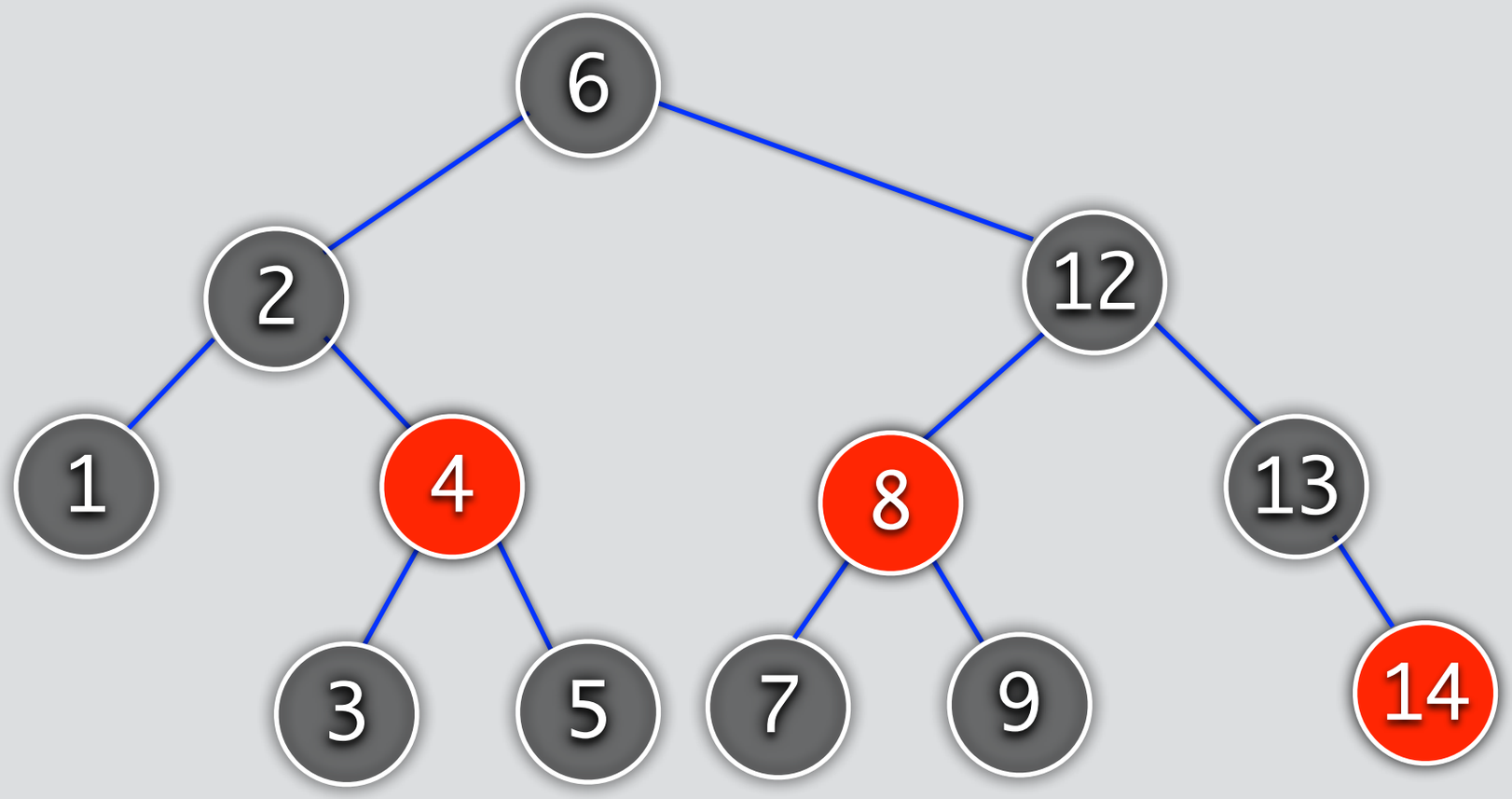

B
x
A
F
y
cd
G
1f
F
B
G
x
A
cd

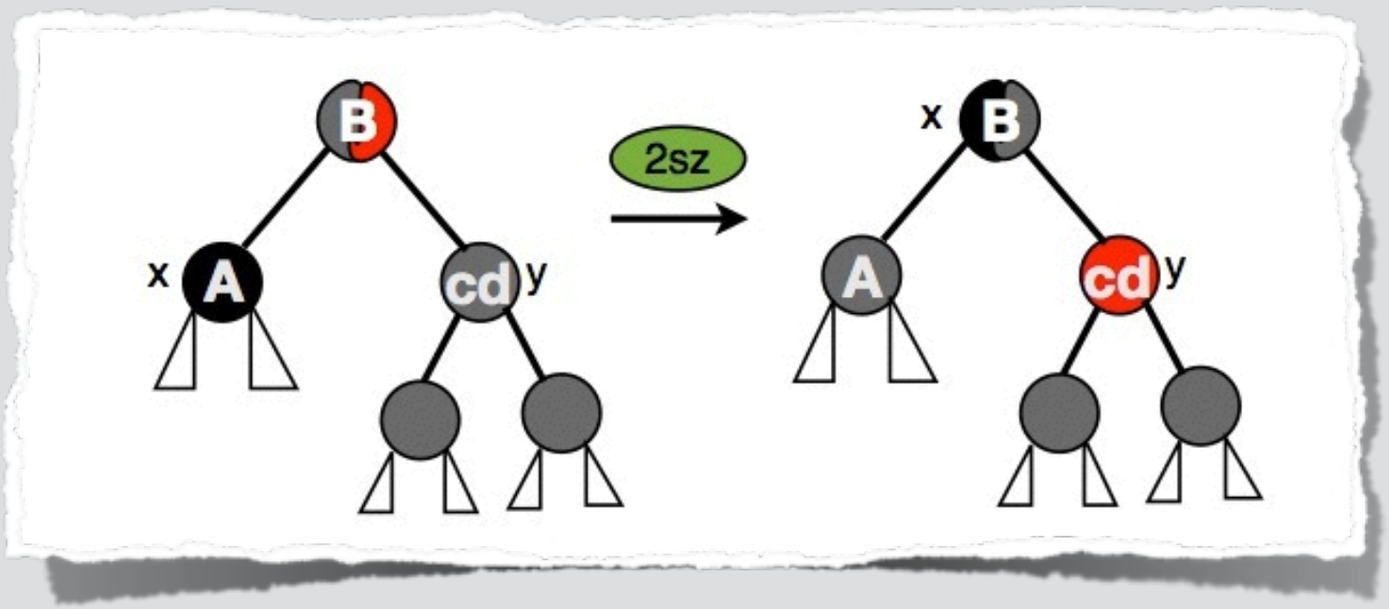

F
6
B
2
12
1
4
8
13
3
5
7
9
14


B
x
A
F
y
cd
G
1f
F
B
G
x
A
cd


6
2
12
1
4
8
13
3
5
7
9
14


B
x
A
cd
y
2sz
x
B
A
cd
y


6
2
12
1
4
8
13
3
5
7
9
14


B
x
A
cd
y
2sz
x
B
A
cd
y