Piros-fekete fák

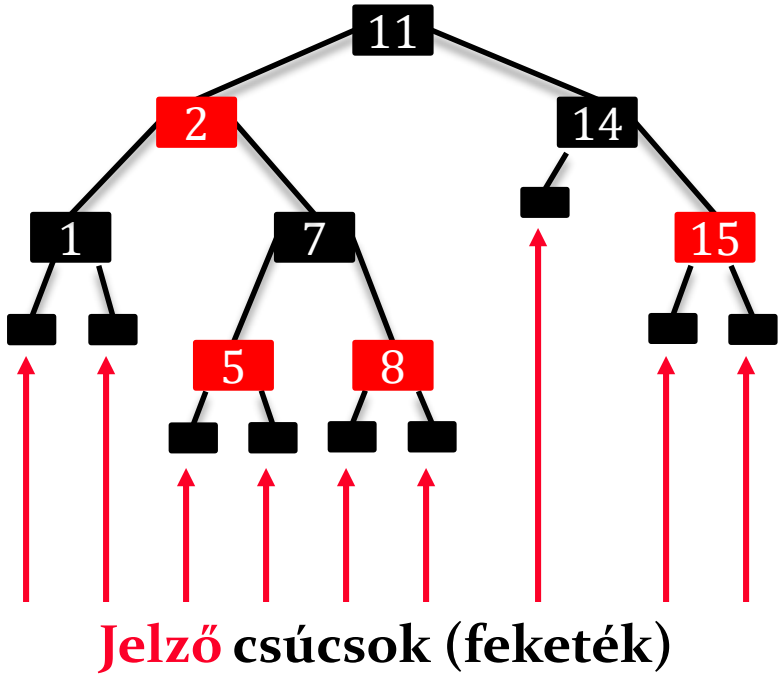
7. előadás

Piros-Fekete fák

- A piros-fekete fa olyan bináris keresőfa, melynek minden pontja egy extra bit információt hordoz: ez a pont színe, amelynek értékei: PIROS vagy FEKETE.
- A pontok színezésének korlátozásával biztosítható, hogy piros-fekete fában bármely, a gyökértől levélig vezető út hossza nem lehet nagyobb, mint a legrövidebb ilyen út hosszának kétszerese. Tehát az ilyen fák megközelítőleg kiegyensúlyozottak.
- A fa minden pontja tartalmazza a *szín*, *kulcs*, *bal*, *jobb* és *szülő* mezőket. A levelek speciális, NIL-t tartalmazó elemek lesznek, a fa kulcsot tartalmazó pontjai a belső pontok.

Piros-Fekete fák

- Egy Piros-Fekete fa
  - Minden csúcs PIROS vagy FEKETE
  - A gyökér, és minden levél FEKETE

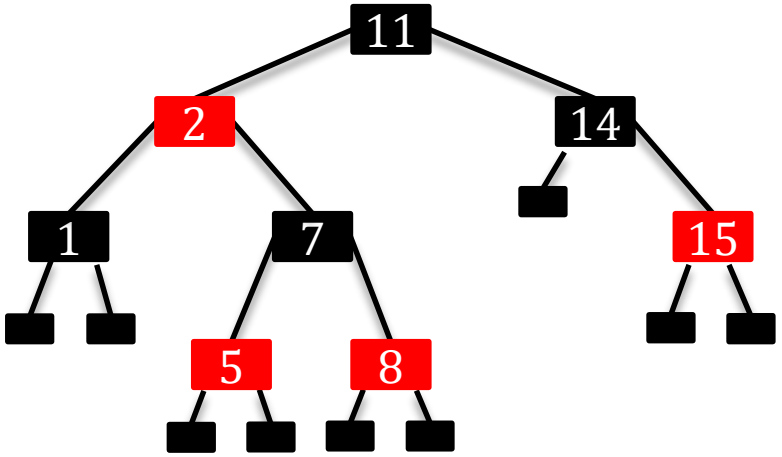


- A levelek FEKETE „jelző” csúcsok, nem tartalmaznak adatot

Piros-Fekete fák

- Egy Piros-Fekete fa
  - Minden csúcs PIROS vagy FEKETE
  - A gyökér, és minden levél FEKETE
  - Ha egy csúcs PIROS, akkor mindkét gyereke FEKETE

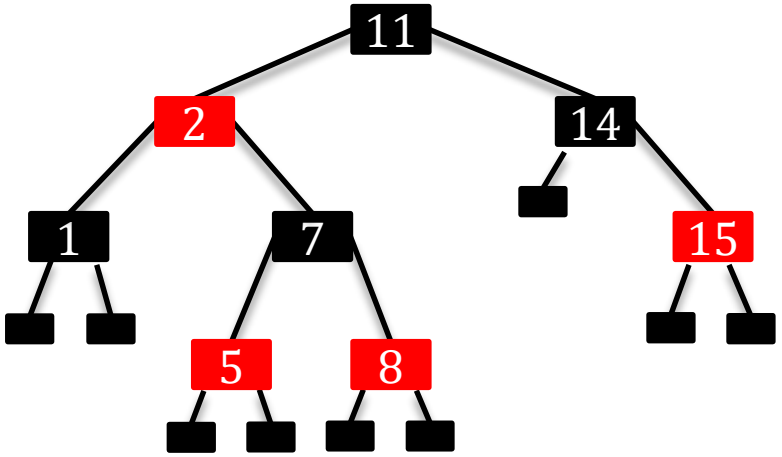
- Ebből következik, hogy egyik úton sem lehet két egymást követő PIROS csúcs.
  - De akárhány egymást követő FEKETE csúcs lehet.



Piros-Fekete fa definíciója

- Egy Piros-Fekete fa
  - Minden csúcs PIROS vagy FEKETE
  - A gyökér, és minden levél FEKETE
  - Ha egy csúcs PIROS, akkor mindkét gyereke FEKETE
  - Minden csúcsból minden út egy levélig ugyanannyi FEKETE csúcsot tartalmaz

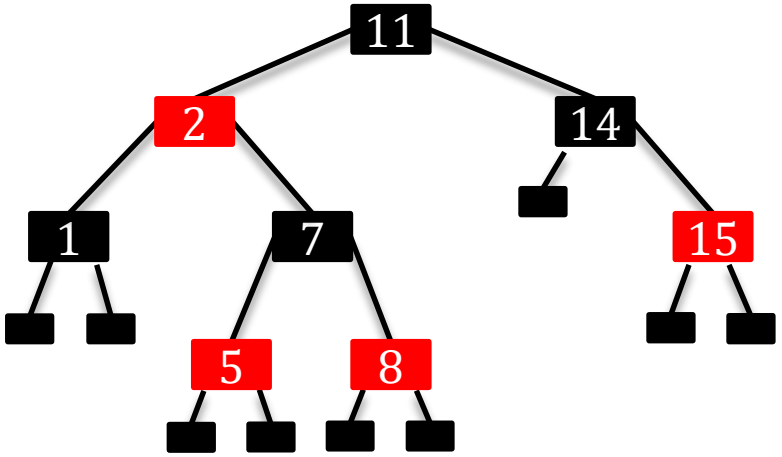
- A gyökértől minden úton 3 FEKETE csúcs van



Piros-Fekete fák

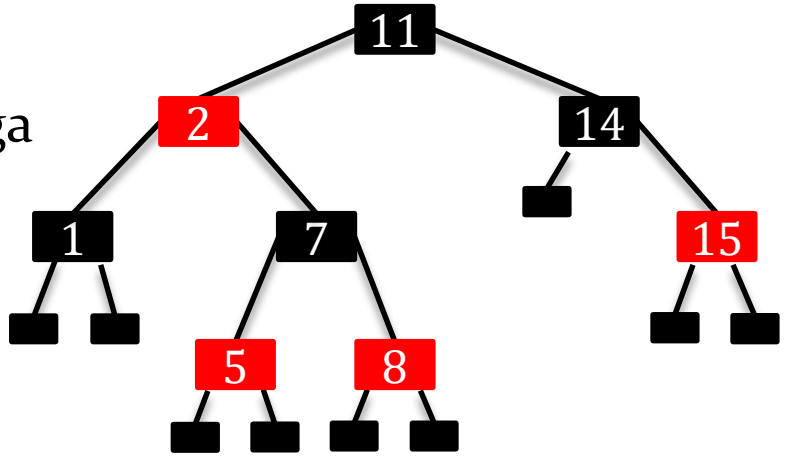
- Egy Piros-Fekete fa
  - Minden csúcs PIROS vagy FEKETE
  - A gyökér, és minden levél FEKETE
  - Ha egy csúcs PIROS, akkor mindkét gyereke FEKETE
  - Minden csúcsból minden út egy levélig ugyanannyi FEKETE csúcsot tartalmaz

- Az x pont fekete-magassága
  - fm(x) – az x pontból induló, levélig vezető úton található, x-t nem tartalmazó fekete pontok száma.
  - A fa fekete-magassága: fm(a fa gyökere)



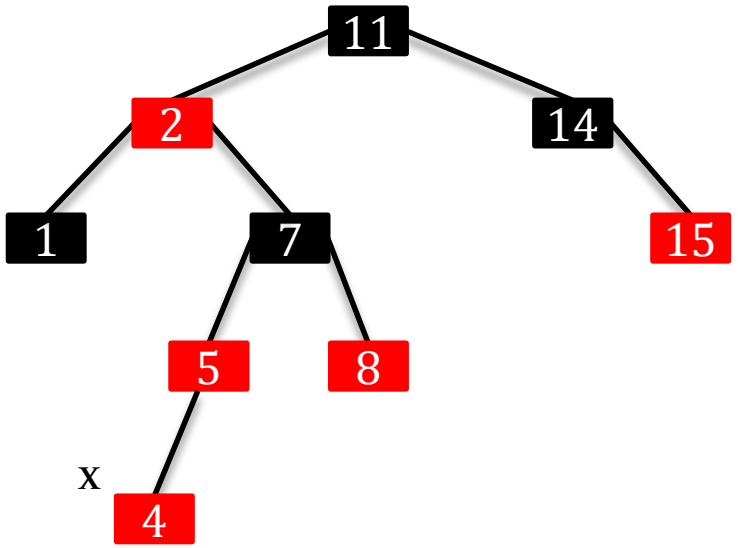
Piros-Fekete fák

- Lemma
  - Bármely n belső pontot tartalmazó piros-fekete fa magassága $\leq 2\log_2(n+1)$
  - A bizonyítás alapja, hogy a magasság $\leq 2 *$ fekete-magasság
    - A részletek a Cormen könyven megtalálhatók
- Keresési idő
  - $\mathcal{O}(\log_2 n)$
    - Látható ebből a PF fák hatékonyága



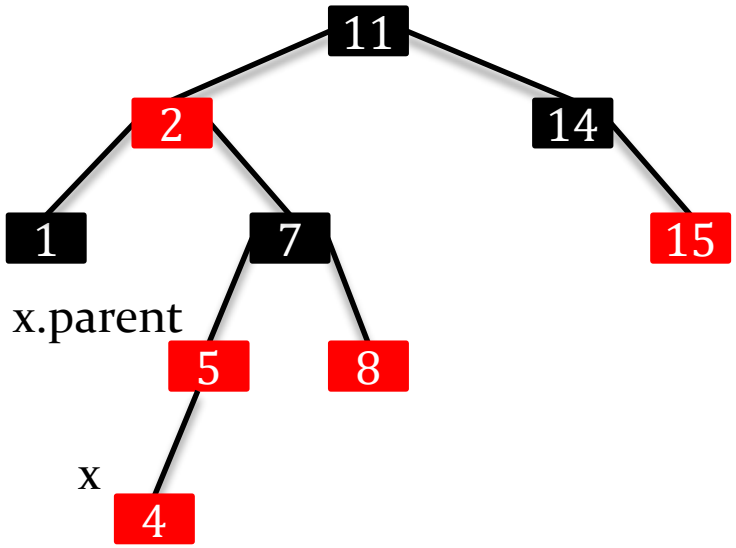
Piros-Fekete fák – beszúrás

- Egy új csúcs beszúrása
  - Szükség van a PF fa tulajdonság helyreállítására
  - Szúrjuk be a 4-et
    - Keressük meg a helyét a bináris keresőfában
    - Színezzük pirosra – Miért?



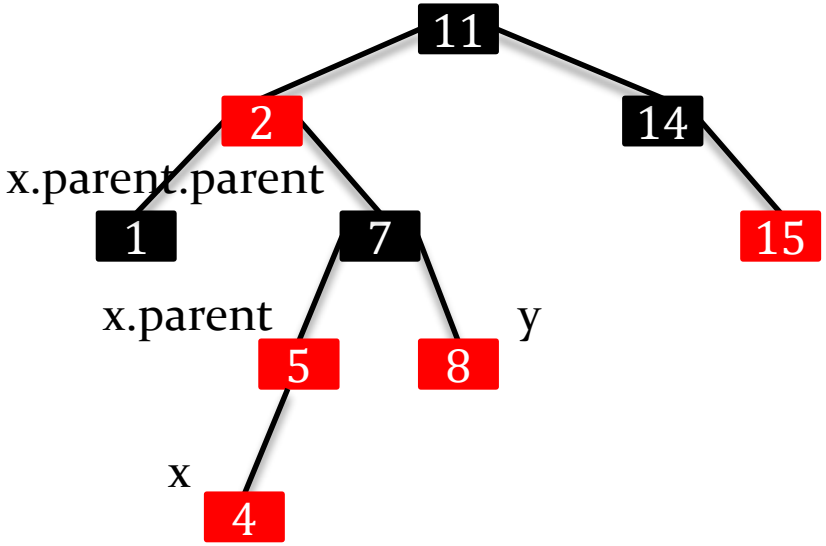
Piros-Fekete fák – beszúrás

- Állítsuk helyre a PF tulajdonságot
  - Amíg az x és szülője egyaránt piros, vagy el nem értük gyökeret helyreállítási lépéseket kell tenni



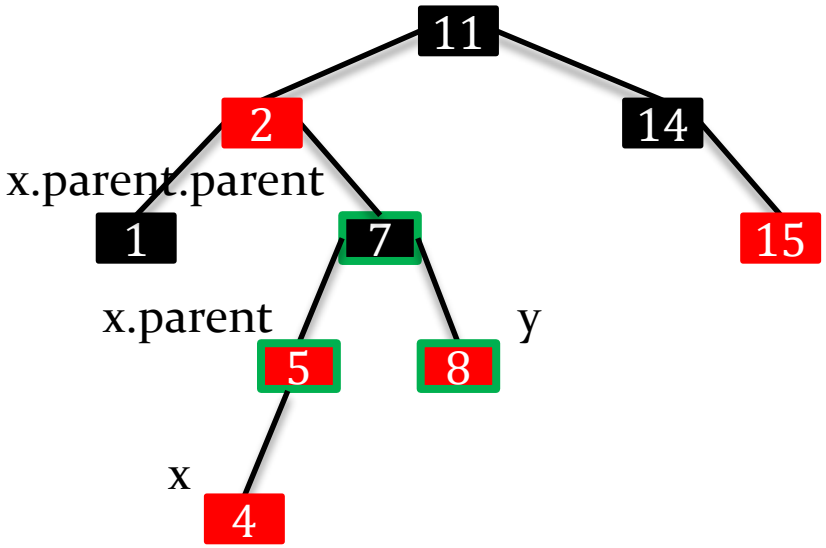
Piros-Fekete fák – beszúrás

- Állítsuk helyre a PF tulajdonságot
  - Jelöljük meg a csúcsokat
    - A szülő balra van a nagyszülőtől, és az x balra van a szülőtől
      - Itt a szülő azonos oldali gyereke a nagyszülőnek, de lehetne ellenkező oldali is
    - A nagyszülő jobbgyereke (nagybácsi) legyen az y



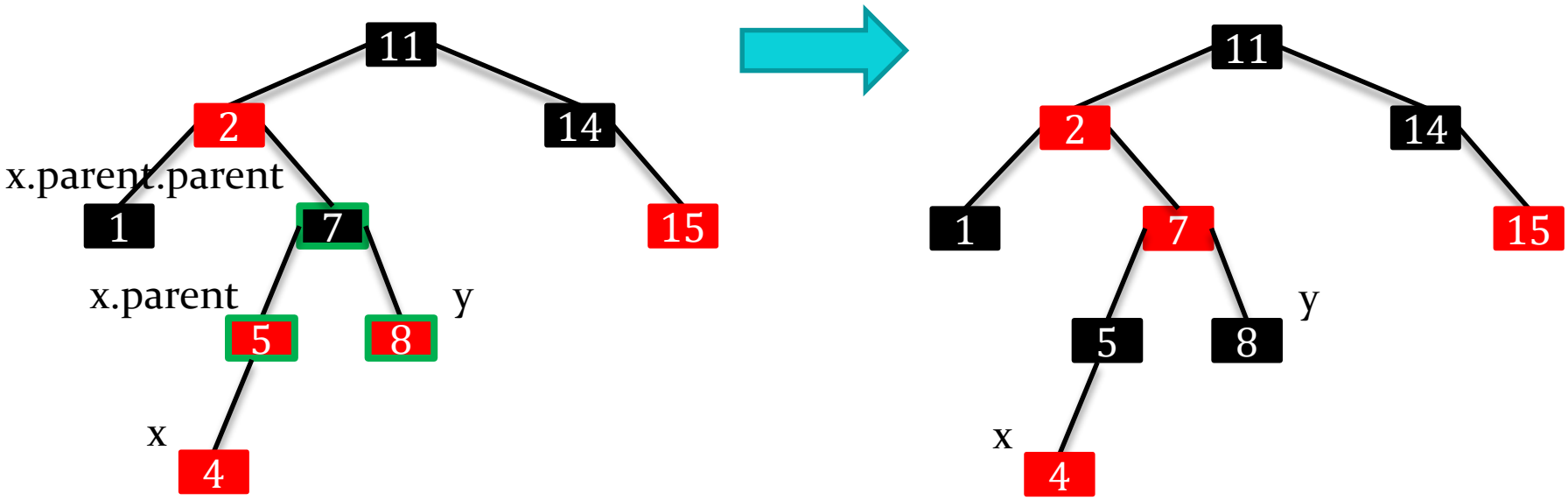
Piros-Fekete fák – beszúrás

- 1. eset
  - Ha a nagybácsi színe piros, cseréljük meg y, a nagyszülő és a szülő színét



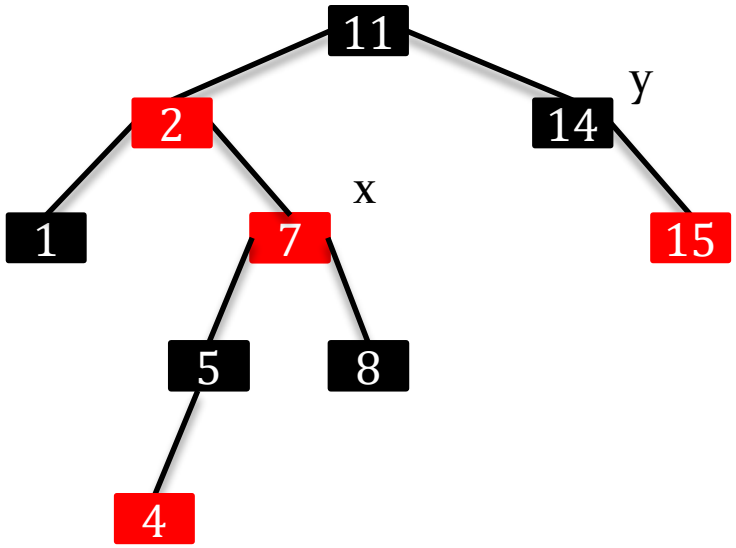
Piros-Fekete fák – beszúrás

- 1. eset
  - Ha a nagybácsi színe piros, cseréljük meg y, a nagyszülő és a szülő színét



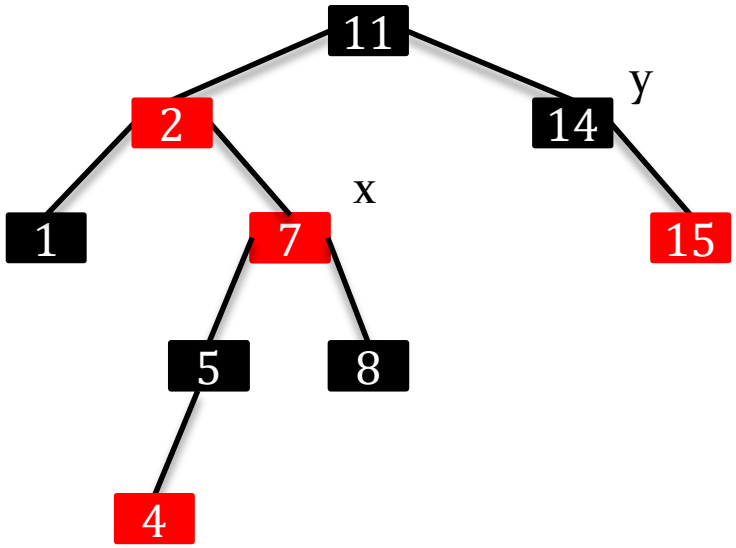
Piros-Fekete fák – beszúrás

- Az előzőt folytatva
  - x jelölje az eddigi x nagyszülőjét
  - Ebben a helyzetben az x szülője megint balgyerek
  - Az y pedig jelölve továbbra is az x nagybácsiját
    - Ez most nem piros, hanem fekete, ami egy másik, új eset



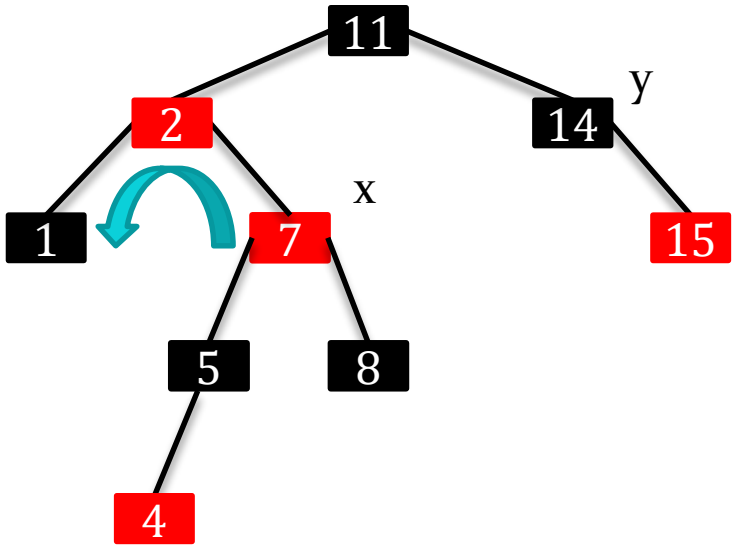
Piros-Fekete fák – beszúrás

- 2. eset
  - Ha az x nagybácsija fekete, az x és szülője is piros és a szülő ellenkező oldali gyereke a nagyszülőnek, mint az x a szülőnek …
    - Megjegyzés: ez az eset függetlenül is előfordulhat az előzőtől



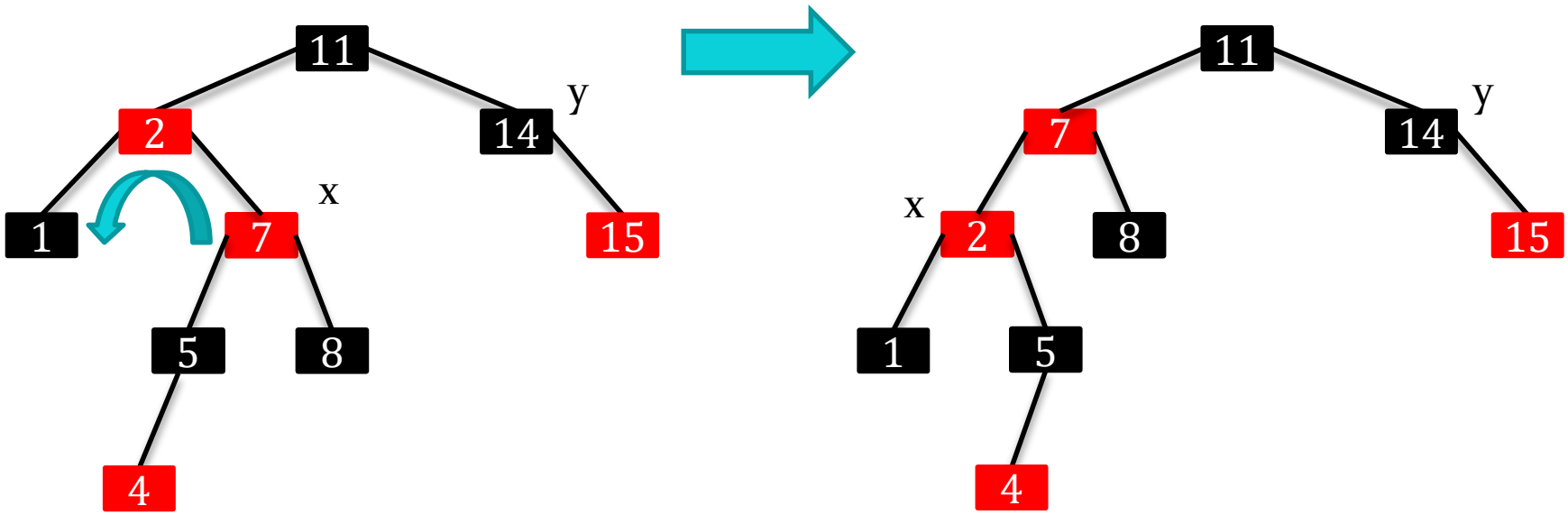
Piros-Fekete fák – beszúrás

- 2. eset
  - … akkor forgassunk x és szülője körül balra …



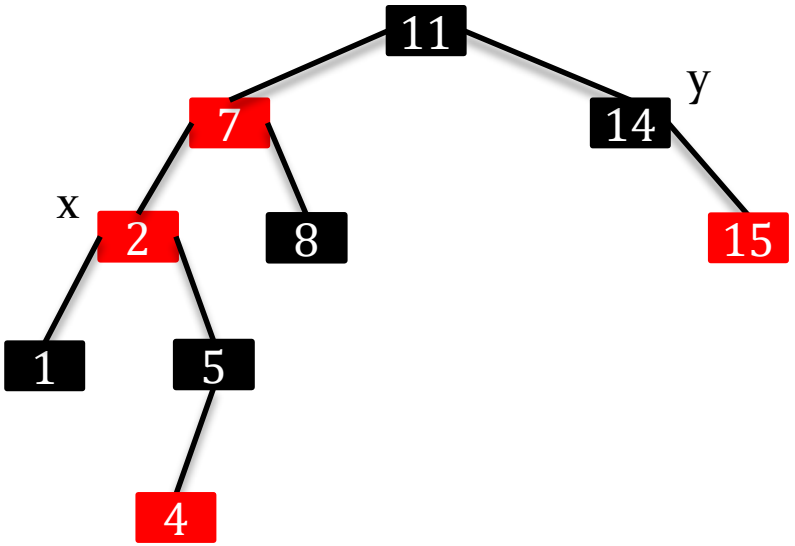
Piros-Fekete fák – beszúrás

- 2. eset
  - … jelöljük át, legyen x az eddigi x szülője és
  - forgassunk x és gyereke körül balra …
    - Ezzel az x és szülője, valamint a szülő és nagyszülő közötti oldaliságot azonossá tettük, mást még nem oldottunk meg.



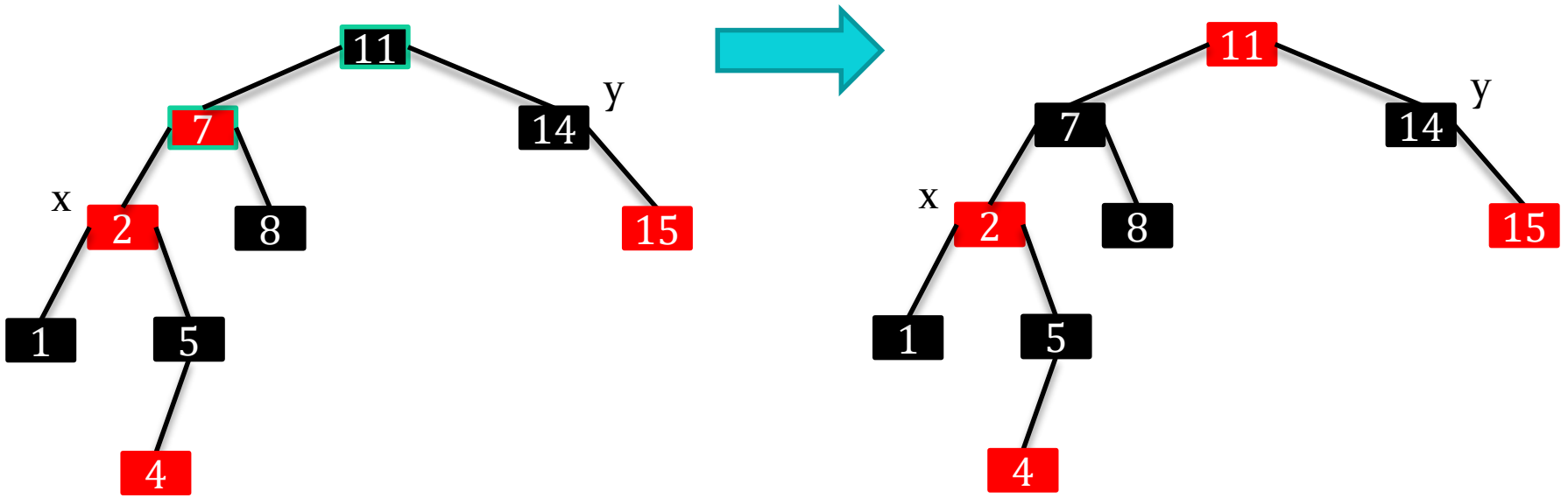
Piros-Fekete fák – beszúrás

- 3. eset
  - Ha az x nagybácsija fekete, az x és szülője is piros és a szülő azonos oldali gyereke a nagyszülőnek, mint az x a szülőnek …
    - Megjegyzés: ez az eset függetlenül is előfordulhat az előzőtől



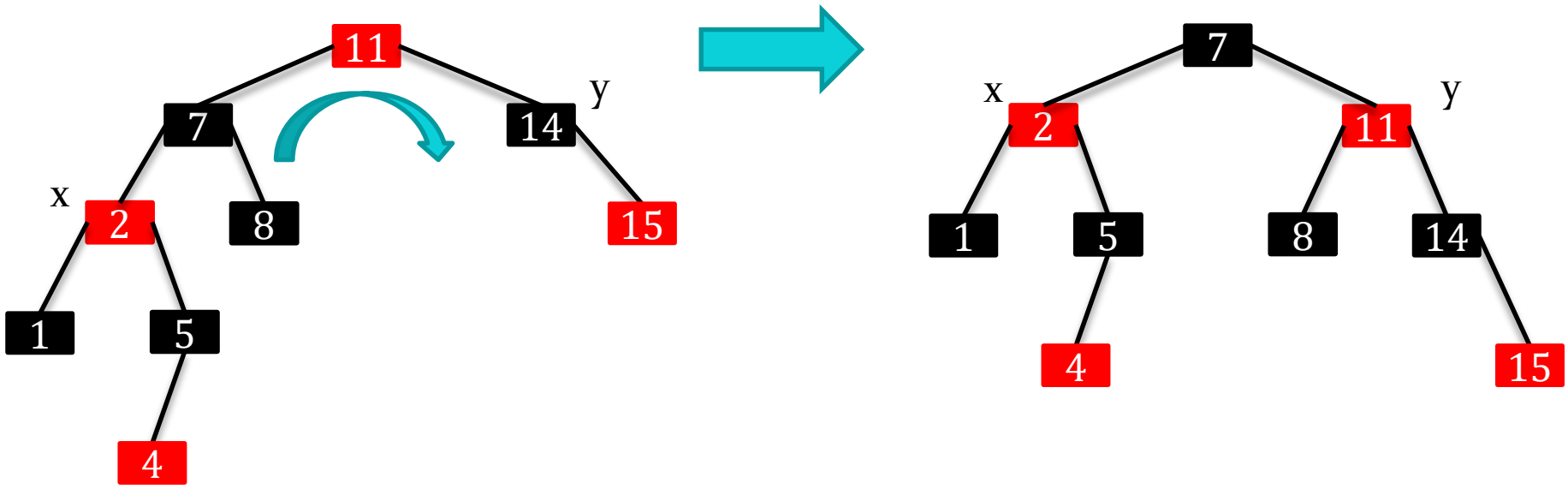
Piros-Fekete fák – beszúrás

- 3. eset
  - … akkor kicseréljük a színeket a szülő és a nagyszülő között …



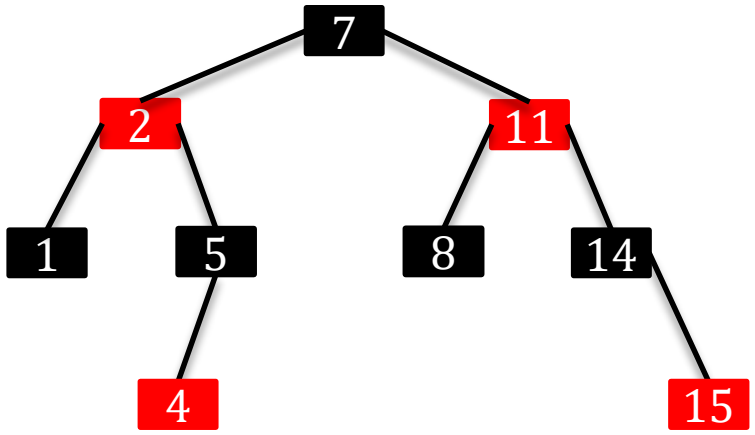
Piros-Fekete fák – beszúrás

- 3. eset
  - … akkor kicseréljük a színeket a szülő és a nagyszülő között
  - … és forgatunk a szülő és a nagyszülő mentén jobbra



Piros-Fekete fák – beszúrás

- 3. eset
  - … akkor kicseréljük a színeket a szülő és a nagyszülő között
  - … és forgatunk a szülő és a nagyszülő mentén jobbra
  - Ez most már PF fa



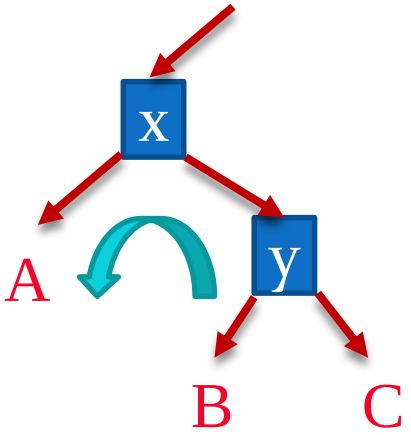
Piros-Fekete fák – beszúrás

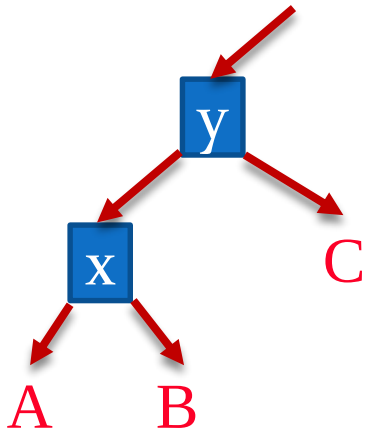
- Van egy ekvivalens esetszétválasztás, amikor a szülő a nagyszülő jobbján van!
  - Természetesen akkor az irányok megváltoznak, megfordul
    - Az azonos oldaliság vizsgálata megmarad
- Mindig addig megyünk felfelé, amíg az x szülője is piros

Piros-Fekete fák – algoritmusok

- Forgatások:
  - BALRA-FORGAT, JOBBRA-FORGAT
  - Mindkettő megőrzi a kulcsok inorder sorrendjét
  - A művelet időigénye $\mathcal{O}(1)$
  - A BALRA-FORGAT feltételezi, hogy jobb[x] ≠ NIL

```
BALRA-FORGAT(T, x)
y ←jobb[x]; jobb[x] ← bal[y]
if bal[y]  ≠ NIL
  then szülő[bal[y]] ← x
szülő[y] ← szülő[x]
if szülő[x]= NIL
  then gyökér[T] ← y
  else if x=bal[szülő[x]]
         then bal[szülő[x]] ← y
         else jobb[szülő[x]] ← y
bal[y] ← x
szülő[x] ← y
```





Piros-Fekete fák – algoritmusok

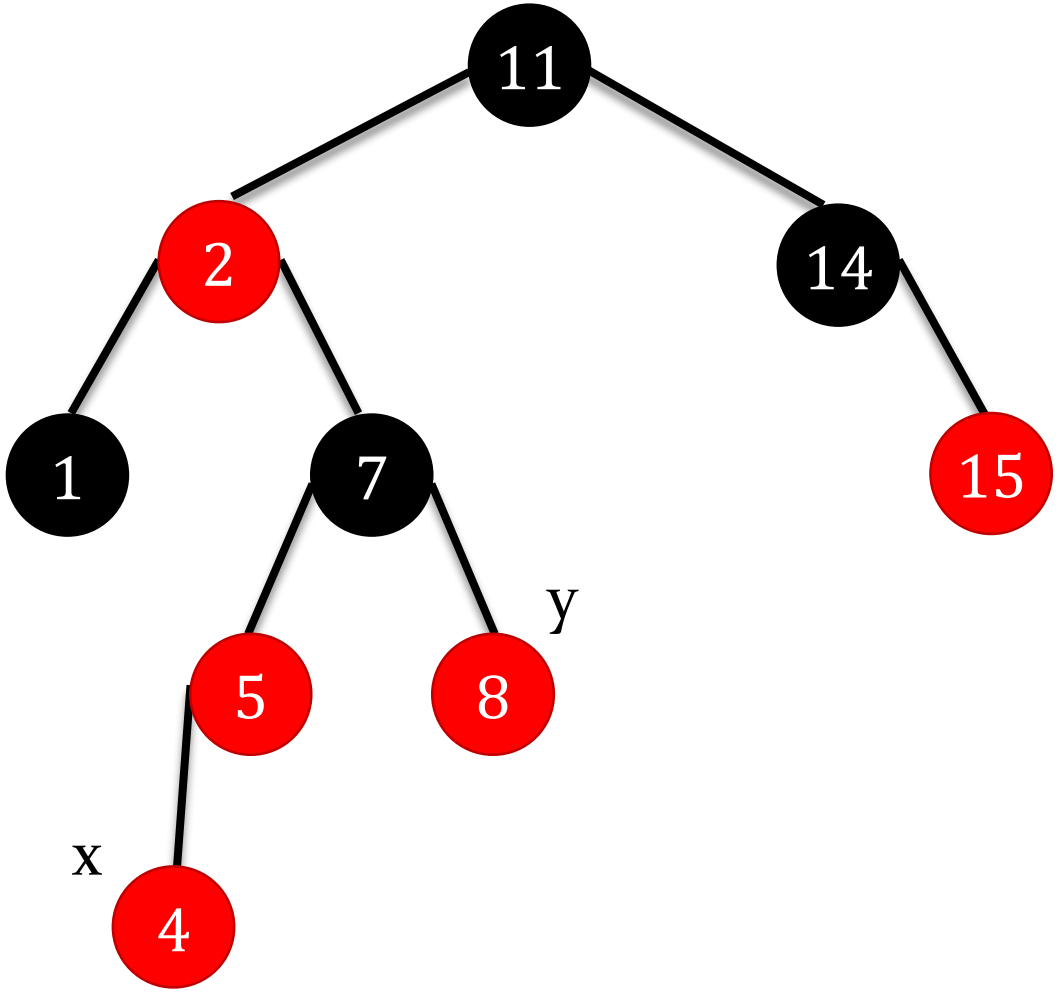
- Lehetséges esetek annak megfelelően, hogy az x, illetve x szülője bal- vagy jobbgyerek-e, és hogy milyen színű a nagybácsi, illetve a nagyszülő:
  1. eset: szülő és nagybácsi piros, nagyszülő fekete (oldaltól nem függ)
  2. eset: szülő piros, nagybácsi fekete, x jobbgyerek (ellenkező oldali gyerek) (balra forgatok, így 3.eset)
  3. eset: szülő piros, nagybácsi fekete, x balgyerek (azonos oldali gyerek)

Piros-Fekete fák – algoritmusok

```
PF-Fába-beszúr(T,x)
Fába-beszúr(T,x)  -- beszúrom a bináris keresőfába
szín[x]←PIROS
while x ≠ gyökér[T] and szín[szülő[x]]=PIROS do
  if szülő[x] = bal[szülő[szülő[x]]] then
    y ← jobb[szülő[szülő[x]]]
    if szín[y] = PIROS then
      szín[szülő[x]]← FEKETE                    --1. eset
      szín[y] ← FEKETE                          --1. eset
      szín[szülő[szülő[x]]]←PIROS               --1. eset
      x←szülő[szülő[x]]                         --1. eset
    else if x = jobb[szülő[x]] then
        x←szülő[x]                              --2. eset
        BALRA-FORGAT(T,x)                       --2. eset
      szín[szülő[x]]← FEKETE                    --3. eset
      szín[szülő[szülő[x]]]←PIROS               --3. eset
      JOBBRA-FORGAT(T, szülő[szülő[x]])         --3. eset
  else
    -- Az előzőeknek szerint, csak az oldalak fordítva
szín[gyökér[T]]← FEKETE
```

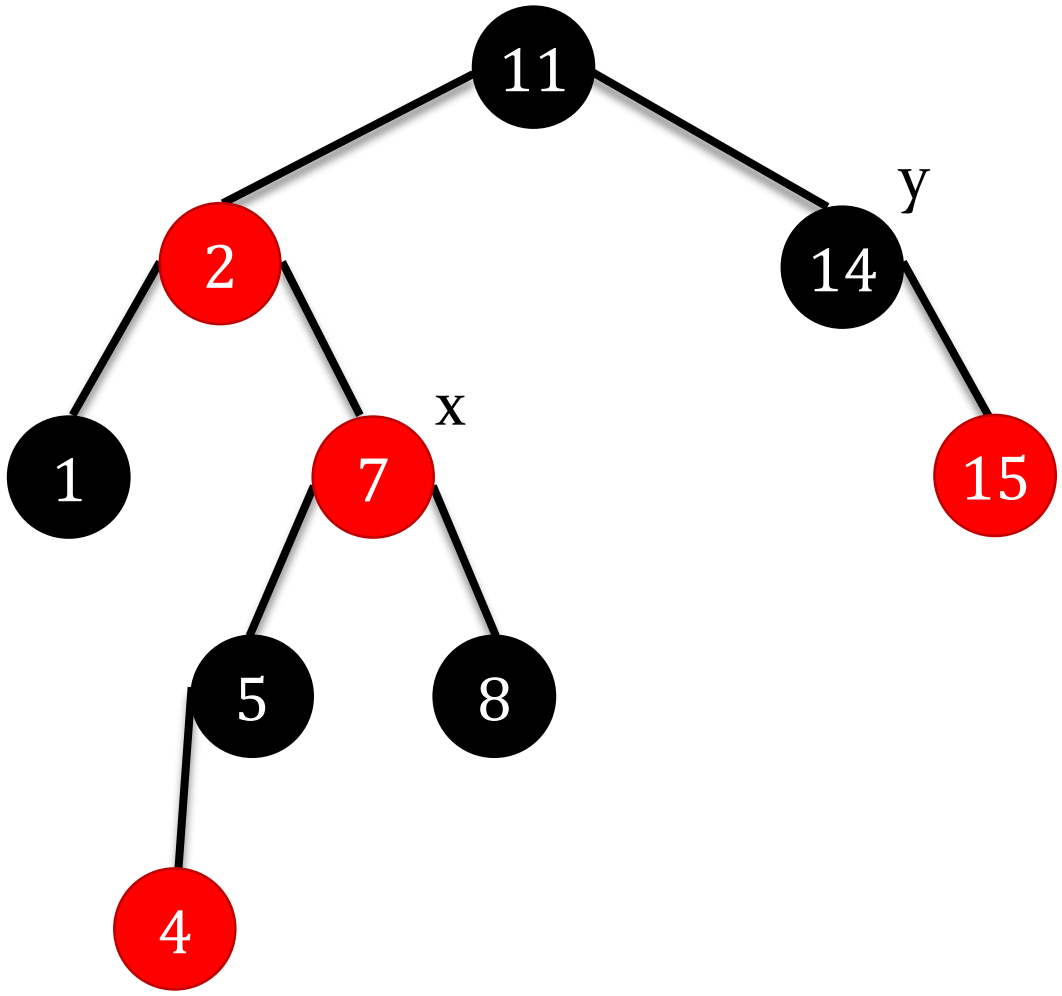
Példa

- Fába-beszúrral betettük x-et, és pirosra színeztük.
- Mivel a szülője is piros, így a 3. tulajdonság nem teljesül.
- y – a jobb nagybácsi is piros, tehát 1. eset áll fenn …



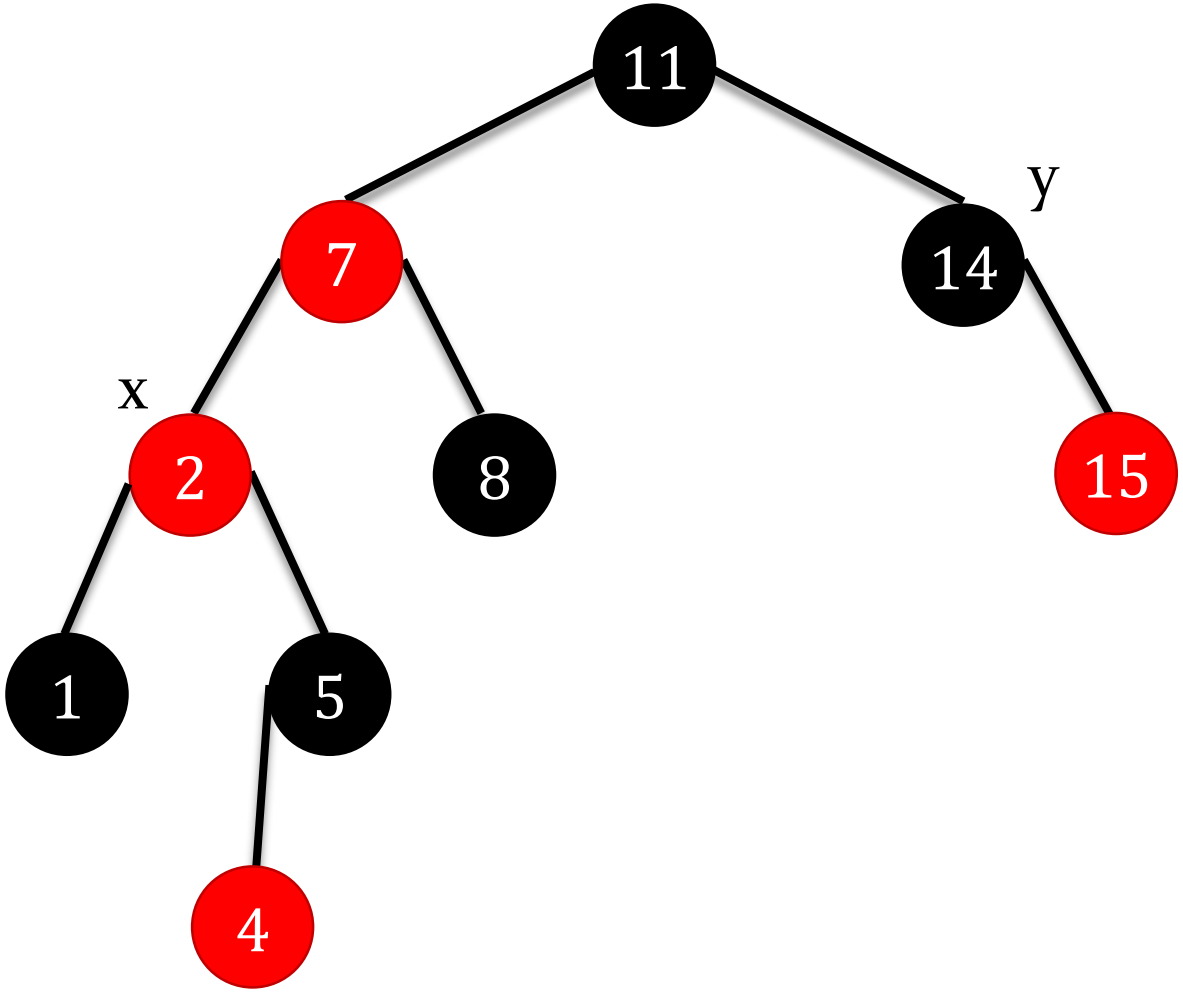
Példa

- x feljebb kerül, x és az apja szintén piros, de most a jobb nagybácsi fekete, és x jobb fia szülőjének
- 2.eset, balraforgatás …



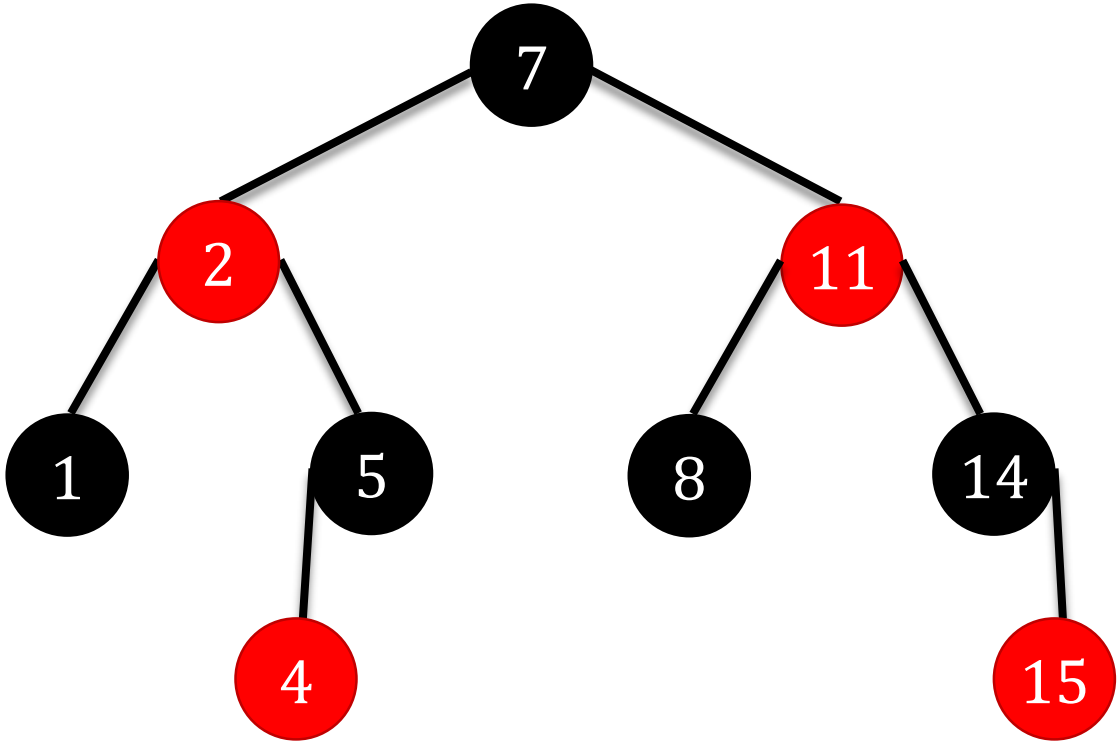
Példa

- most már x bal fia szülőjének
- ez a 3. eset, így x nagyszülőjének jobbraforgatása és a színek cseréje jön …



Példa

- … és kész a piros-fekete fa



Piros-Fekete fák – törlés

Egy z csúcs törlése egy bináris keresőfából:

1. eset: z-nek nincs nem-nil gyereke
   - töröljük a csúcsot (a szülőt a nil-elemmel kötjük össze)
2. eset: z-nek egy nem-nil gyereke van
   - töröljük a csúcsot (a szülőt a gyerek-elemmel kötjük össze)
3. eset: z-nek két nem-nil gyereke van
   - megkeressük a közvetlen rákövetkezőjét
   - átmásoljuk belőle az adatot z-be (z színe változatlan marad)
   - töröljük a rákövetkezőt (ennek csak 0 v. 1 gyereke lehet)

- Jelöljük y-nal a ténylegesen kitörölt elemet.

Piros-Fekete fák – törlés

Ha a kitörölt elem (y) piros volt

Tulajdonság és leírása	Igaz?
1. Minden csúcs piros vagy fekete	igen
2. A gyökér fekete	igen
3. Minden levél (nil[T]) fekete	igen
4. Ha egy csúcs piros, mindkét gyerek fekete	igen
5. Minden út egy csúcstól a leszármazott levelekig ugyanannyi fekete csúcsot tartalmaz	igen

Ebben az esetben nincs tennivaló.

Piros-Fekete fák – törlés

Ha a kitörölt elem (y) fekete volt

Tulajdonság és leírása	Igaz?
1. Minden csúcs piros vagy fekete	igen
2. A gyökér fekete	???
3. Minden levél (nil[T]) fekete	igen
4. Ha egy csúcs piros, mindkét gyerek fekete	???
5. Minden út egy csúcstól a leszármazott levelekig ugyanannyi fekete csúcsot tartalmaz	???

Az „elvesztett” fekete csúcsot kell „pótolni”.

Piros-Fekete fák – törlés

- Ahhoz, hogy helyreállítsuk a fekete csúcsok számát minden úton, a gyerekének adjuk a kitörölt csúcs fekete színét
- Úgy képzeljük, hogy a gyerekének most van egy extra feketéje
  - Ha a gyerek színe fekete, akkor most dupla-fekete,
  - Ha a gyerek színe piros, akkor most piros-fekete
    - Ténylegesen nem változtatjuk a csúcsot a kódban, csak úgy vesszük, mintha…
- Ezzel a legnehezebb, 5. tulajdonságot teljesítjük
  - Cserébe az 1., 2. tulajdonságot nem
  - A gyerek se nem piros, se nem fekete.

Piros-Fekete fák – törlés

- Az 1. tulajdonság helyreállítására ezt az „extra” feketét visszük felfelé a fában, míg a következők egyike igaz nem lesz:
- x egy piros-fekete csúcsra mutat. Ekkor feketére színezzük, és kész
- x a fa gyökerére mutat. Ha dupla-fekete, eltávolítjuk az egyik feketéjét, és kész, mert ha a gyökérből viszünk el egy extra feketét, akkor egyforma marad a levelekhez vezető utakon a feketék száma
- Olyan ponthoz érünk, ahol forgatásokkal és átszínezésekkel el tudjuk távolítani az extra feketét úgy, hogy nem sértjük meg a piros-fekete tulajdonságokat többé

Piros-Fekete fák – törlés

```
PF-FÁBÓL-TÖRÖL(T,z) -- törlés, mint bináris keresőfából,
                       majd p-f helyreállítás
  if bal[z]=nil[T] or jobb[z]=nil[T]          -- jelölő strázsa
    then y←z                                  -- 0 vagy 1 gyerek
    else y←FÁBAN-KÖVETKEZŐ(T,z)               -- 2 gyerek
  if bal[y]≠nil[T]
    then x←bal[y]                 -- x az y 0 vagy 1 gyerekére mutat
    else x←jobb[y]
  szülő[x] ←szülő[y]              -- ha volt (egy) gyereke, befűzzük
  if szülő[y]= nil[T]
    then gyökér[T]←x
    else if y = bal[szülő[y]]
      then bal[szülő[y]]←x
      else jobb[szülő[y]]←x
  if y≠z
  then kulcs[z]←kulcs[y]          -- ha két gyerek volt
                                     (ha van egyéb mező, azt is)
```

Piros-Fekete fák – törlés

```
PF-FÁBÓL-TÖRÖL(T,z)   -- folytatás
if szín[y]=FEKETE     -- ha y PIROS, akkor PF
                         tulajdonság OK
                         (fm nem változott sehol,
                         piros-piros szülő-gyerek
                         kapcsolat nem keletkezett.)
then PF-FÁBÓL-TÖRÖL-JAVÍT(T,x)
                      -- az x ténylegesen a
                         törlendőnek a gyereke!
return y
```

Piros-Fekete fák – törlés

- Mi sérül?
  - Ha az y fekete volt, akkor most minden út, ami az y-n keresztül haladt, eggyel kevesebb fekete pontot fog tartalmazni
    - y őseire nem teljesül a 5. tulajdonság
- Amikor y-t eltávolítjuk, fekete értékét „továbbadjuk” a gyerekének
  - Ha x is fekete volt, akkor most „kétszeresen fekete” lesz
    - Ez sérti az 1. tulajdonságot

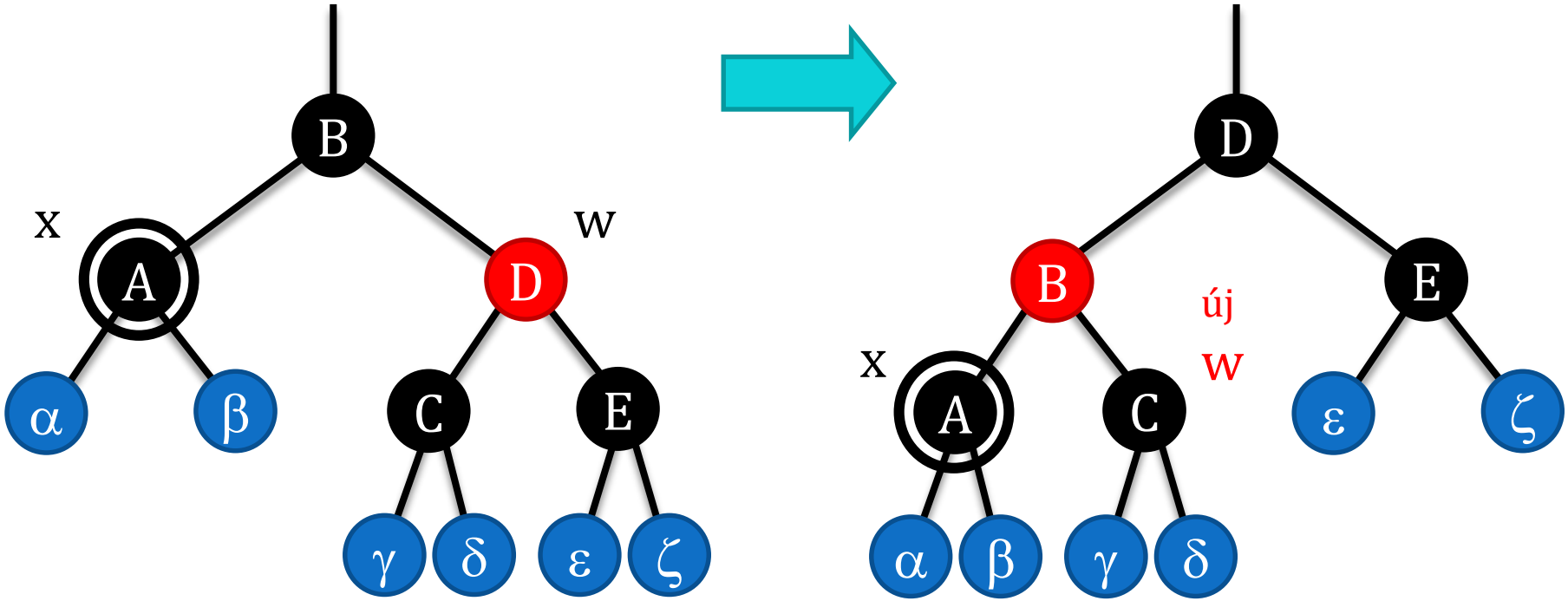
Piros-Fekete fák – törlés

- Lehetséges esetek:
  - Csak azokat vesszük figyelembe, ahol x balgyerek, ahol jobb, az szimmetrikusan adódik.
  - w jelöli x testvérét (w <- jobb[szülő[x]])

1. eset: x testvére w piros
2. eset: x testvére w fekete és w mindkét gyereke fekete
3. eset: x testvére w fekete, w balgyereke piros, jobbgyereke fekete
4. eset: x testvére w fekete, w jobbgyereke piros

1. eset

- x duplán fekete, és a testvére (w) piros
  - w-nek van fekete fia
  - így w és szülő[x] színét felcserélve és balra forgatva a szülő[x]-t, az x új testvére fekete lesz
  - 2.,3., vagy 4. eset



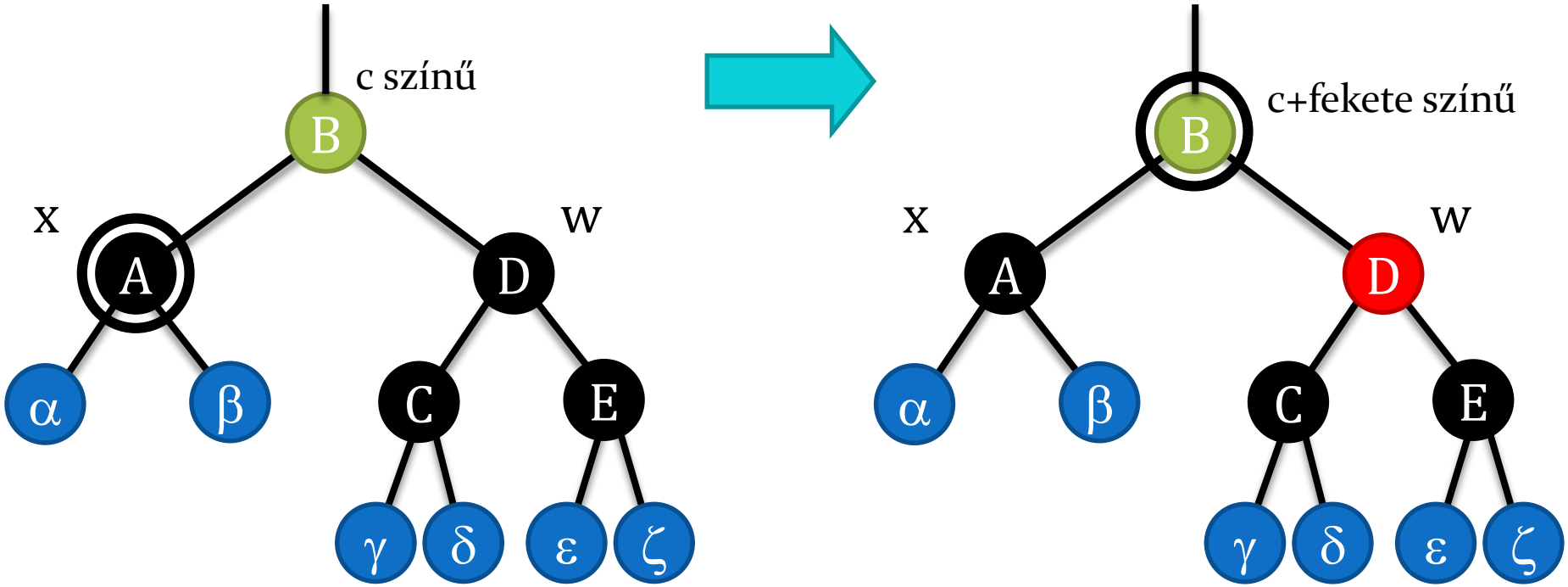
Piros-Fekete fák - törlés

- 1. eset: x testvére w piros
  - Pszeudokód:

```
if szín[w]=PIROS
then szín[w]←FEKETE
     szín[szülő[x]]←PIROS
     BALRA-FORGAT (T, szülő[x])
     w←jobb[szülő[x]]
```

2. eset

- A w is fekete, így nézzük az ő gyerekeinek a színét
  - ha mindkét fia fekete: elvehetünk egy feketét x-től és w-től így x egyszer fekete, w piros lesz, és szülő[x] kap extra feketét,
  - Ezután folytatjuk a helyreállítási ciklust a szülő[x]-re



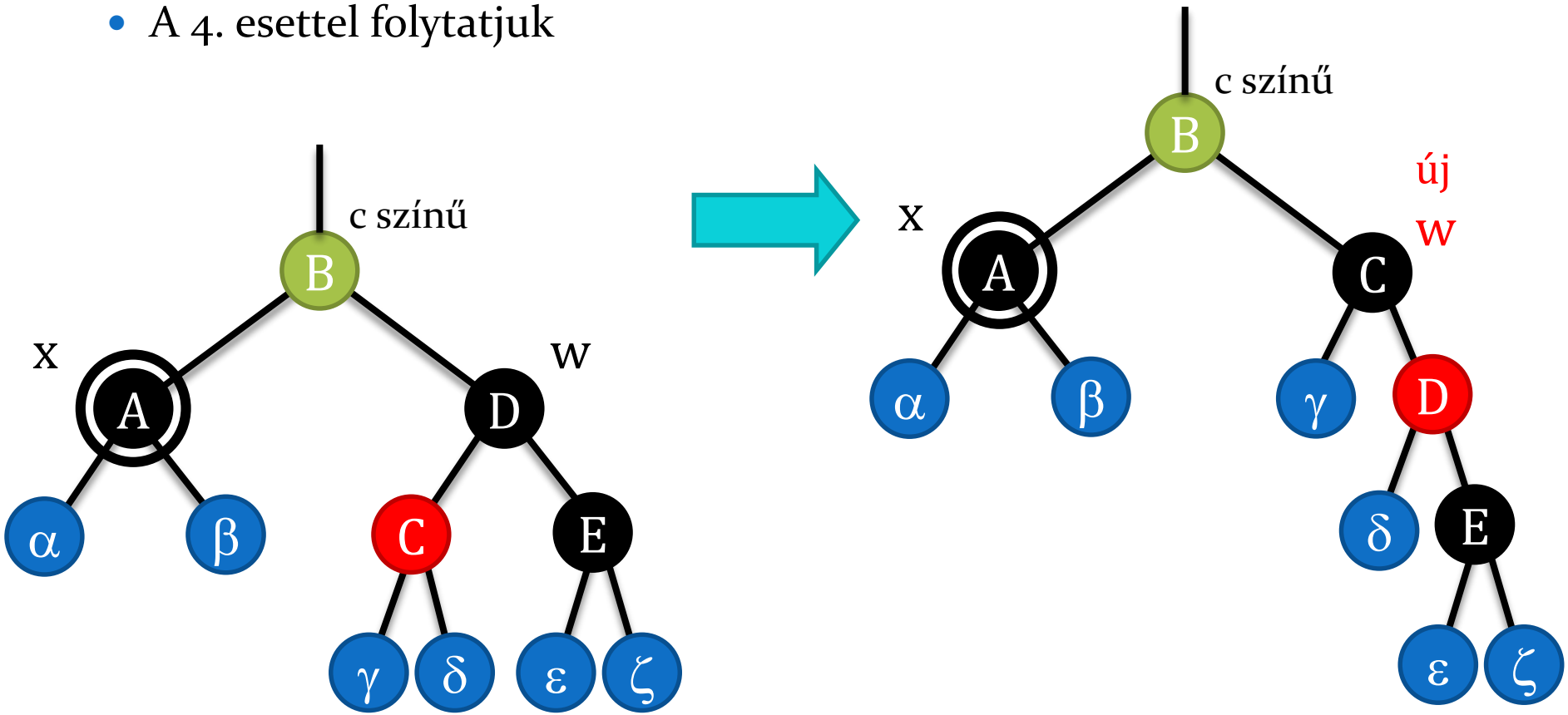
Piros-Fekete fák - törlés

- 2. eset: x testvére w fekete és w mindkét gyereke fekete
  - Pszeudokód:

```
if szín[bal[w]]=FEKETE and
   szín[jobb[w]]=FEKETE
   then szín[w]←PIROS
        x←szülő[x]
```

3. eset

- w fekete, bal fia piros, jobb fia fekete
  - w és bal[w] színét cseréljük, majd jobbforgatás w-re, az új w fekete, és jobb fia piros
  - A 4. esettel folytatjuk



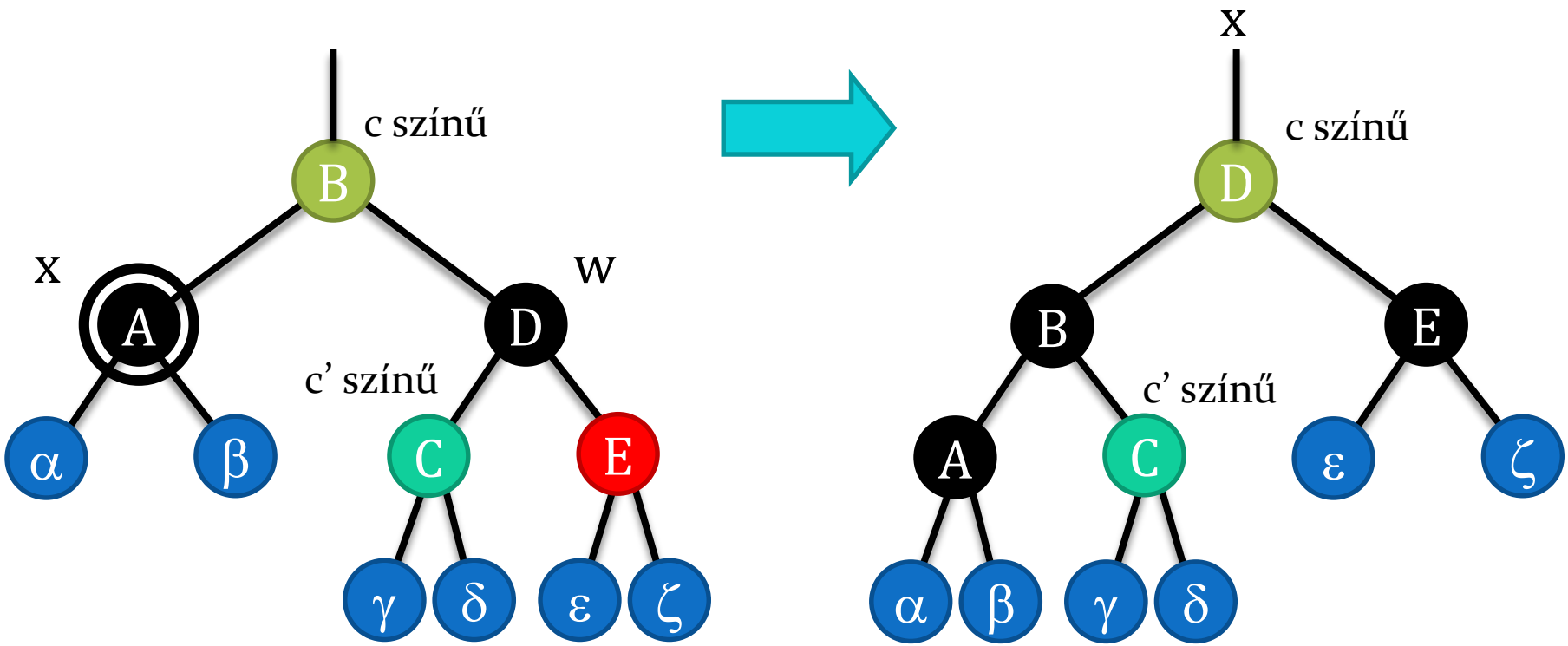
Piros-Fekete fák - törlés

- 3. eset: x testvére w fekete, w balgyereke piros, jobbgyereke fekete
  - Pszeudokód:

```
else if szín[jobb[w]]=FEKETE
     then szín[bal[w]]←FEKETE;
          szín[w]←PIROS,
          JOBBRA-FORGAT(T,w);
          w←jobb[szülő[x]]
```

4. eset

- w fekete és jobb fia piros
  - w, szülő[x] és jobb[w] színét váltjuk, majd szülő[x] körül balraforgatva úgy törölhetjük le az x extra feketéjét, hogy a PF tulajdonság marad
  - Ezután x legyen a részfa gyökere és kész



Piros-Fekete fák - törlés

- 4. eset: x testvére w fekete, w jobbgyereke piros
  - Pszeudokód:

```
szín[w]←szín[szülő[x]]
szín[szülő[x]]←FEKETE;
szín[jobb[w]]←FEKETE
BALRA-FORGAT(T, szülő[x])
x ← gyökér[T]
```

Piros-Fekete fák – algoritmusok

```
PF-FÁBÓL-TÖRÖL-JAVÍT(T,x)
while x ≠ gyökér[T] and szín[x]= FEKETE
   do if x = bal[szülő[x]]
      then w←jobb[szülő[x]]
           if szín[w]=PIROS
           then szín[w]←FEKETE; szín[szülő[x]]←PIROS
                BALRA-FORGAT(T,szülő[x]); w←jobb[szülő[x]]
           if szín[bal[w]] = FEKETE and szín[jobb[w]] = FEKETE
              then szín[w]← PIROS; x←szülő[x]
              else if szín[jobb[w]]= FEKETE
                   then szín[bal[w]]←FEKETE; szín[w]←PIROS
                        JOBBRA-FORGAT(T,w); w←jobb[szülő[x]]
                   szín[w]← szín[szülő[x]]
                   szín[szülő[x]]←FEKETE; szín[jobb[w]]←FEKETE
                   BALRA-FORGAT(T, szülő[x])
                   x←gyökér[T]
       else ua., mint a then, csak a bal és jobb felcserélve
szín[gyökér[T]]←FEKETE
```

Piros-Fekete fák – Elemzés

- Hozzáadás
  - Beszúrás
    - Összehasonlítások $\mathcal{O}(\log_2 n)$
    - PF-tulajdonsághoz $\mathcal{O}(\log_2 n)$
      - Minden lépésnél x felfelé mozog a fában legalább egy szintet
  - Összesen $\mathcal{O}(\log_2 n)$
- Törlés
  - Összesen szintén $\mathcal{O}(\log_2 n)$
- Bonyolultabb, de $\mathcal{O}(\log_2 n)$ viselkedést ad dinamikus esetekben is!

Dinamikus fák: PF vagy AVL?

- Beszúrás
  - AVL: két menet a fán keresztül
    - lefelé a csúcs beszúrásához
    - felfelé az újrakiegyensúlyozáshoz
  - Piros-Fekete: két menet a fán keresztül
    - lefelé a csúcs beszúrásához
    - felfelé az újrakiegyensúlyozáshoz
- A Piros-Fekete népszerűbb?

Dinamikus fák: PF vagy AVL?

- Beszúrás
  - Ha a Cormen et al. könyvet olvassuk,
    - nem indokolja, hogy miért részesíti előnyben a Piros-Fekete fákat
  - Weiss könyvében szerepel, hogy egy Piros-Fekete fát *ki lehet egyensúlyozni egy menetben*
    - M A Weiss, Algorithms, Data Structures and Problem Solving with C++, Addison-Wesley, 1996
    - Így a Piros-Fekete fák hatékonyabbak lesznek, mint az AVL fák!
- Fontos olvasni a szakirodalmat

Dinamikus fák

- Beszúrás egy menetben
  - Ahogy megyünk lefelé a fán, ha találunk egy csúcsot két piros gyerekkel, változtassuk a színét pirosra és a gyerekekét feketére
  - Ez nem változtatja a fekete csúcsok számát egyetlen úton sem
  - Ha ennek a csúcsnak a szülője piros volt, akkor forgatás kell …
    - A forgatás lehet sima, vagy dupla