Piros-Fekete fa

ADATSZERKEZETEK ÉS ALGORITMUSOK
7. GYAKORLAT

Ismétlés

Bináris keresőfa

- Minden művelet ideje a magassággal arányos. → O(h)
- Viszont kiegyensúlyozatlan esetben akár: h = n.

AVL fa

- Kiegyensúlyozott: $h = 1.44(\log_2 n)$
- Stabil teljesítmény minden esetben. → $O(\log_2 n)$

Mi jöhet még?

Piros-fekete fa

Az AVL fához hasonlóan egyfajta kiegyensúlyozott bináris keresőfa.

Szintén minden csúcsban egyensúly-információ: szín

Szabályok, amelyek garantálják a kiegyensúlyozottságot.

Algoritmusok, melyek hatékonyan fenntartják a szabályokat a módosítások során.

Eredmény: „viszonylag” kiegyensúlyozott bináris keresőfa.

Pf-fa tulajdonságai

Szabályok

- Minden csúcs színe vagy piros, vagy fekete.
  - Innen a név. ☺
- A gyökércsúcs színe fekete.
- Minden levél érték nélküli (NIL), színük fekete.
- Minden piros csúcsnak mindkét gyereke fekete.
- Bármely csúcsból bármely levélig vezető úton ugyanannyi fekete csúcs van.

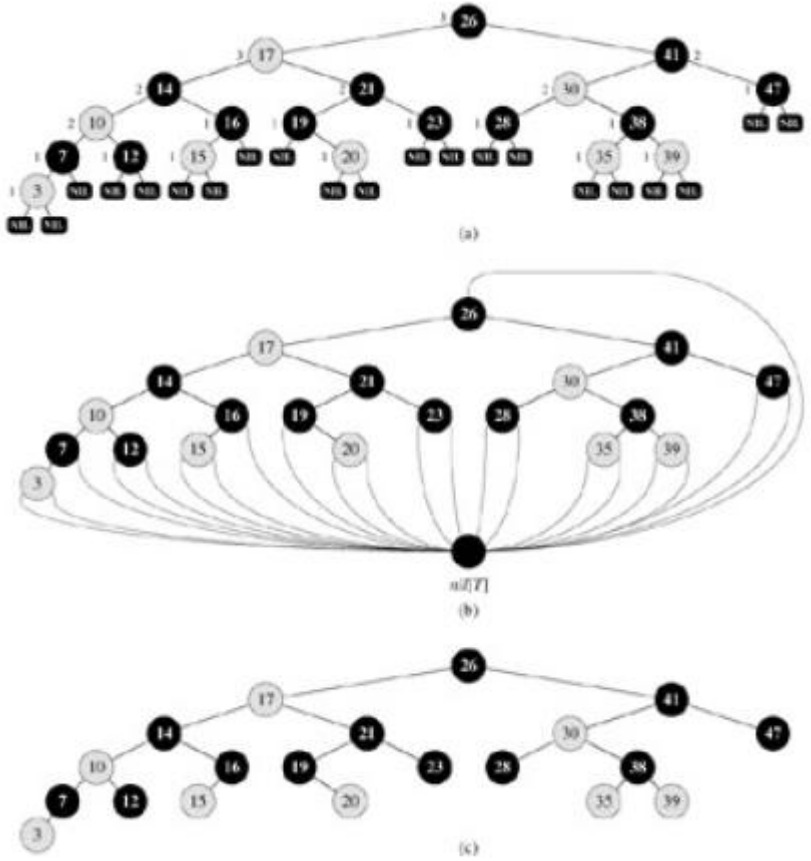
Megjegyzés: minden érték a belső csúcsokban van, azaz a levél (NIL) nem tárol értéket.

A NIL csúcs

Értékek csak a belső csúcsokban vannak.

Mivel a levelekben nincs érték, így egyetlen NIL csúcsot foglalunk le minden levél számára.

Ezeket általában elhagyjuk az ábrákon.



Szabályok következménye

A 4. (piros csomópontok nem követhetik egymást) és 5. pont (minden azonos gyökerű úton ugyan annyi fekete csúcs van) garantálja, hogy a gyökértől levélig vezető leghosszabb út legfeljebb kétszerese a legrövidebbnek.

Ez garantálja a maximum $2(\log_2(n+1))$ magasságot a PF fára.

„De hisz az AVL fa magassága maximum $1.44(\log_2 n)$!”

Piros-fekete fa vs. AVL fa

Az AVL fa „kiegyensúlyozottabb” a piros-fekete fánál.

- Emiatt a keresés gyorsabb.
- Viszont módosítás után „drágább” a karbantartás.

Törlés után szükséges forgatások száma

- AVL fánál: $O(\log_2 n)$
- Piros-fekete fánál: $O(\log_2 n)$

De, a piros-fekete fa esetén létezik úgynevezett top-down kiegyensúlyozás, vagyis egy menetben, egyszerre történik a beszúrás/törlés valamint kiegyensúlyozás.

- Mi most nem ezt fogjuk leprogramozni!

Szabályok helyreállítása

Keresés mint a bináris keresőfánál

Beszúrás és törlés

- Először mint a bináris keresőfánál
- Majd helyreállítjuk a piros-fekete fa szabályokat

Szabályok helyreállítása:

- Átszínezésekkel
- Forgatásokkal

Beszúrás

A hagyományos módon beszúrjuk a csúcsot a fába.

A beszúrt csúcs színe legyen **piros**.

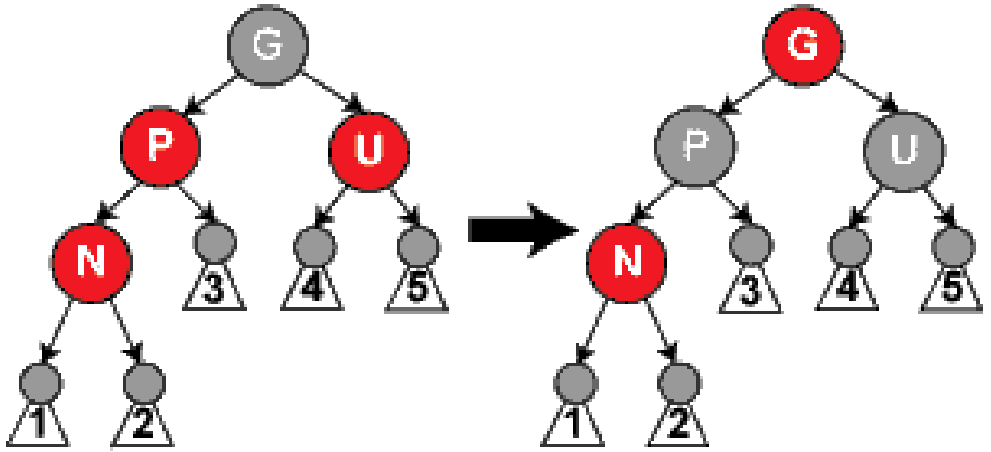
- Csak a 4. szabály sérülhet.

Amíg a **piros** csúcs szülője is **piros** – vagy el nem érjük a fa gyökerét – a későbbi diákon tárgyalt esetek szerint végezzük a kiegyensúlyozást.

A végén a fa gyökerét **feketére** állítjuk.

Beszúrás

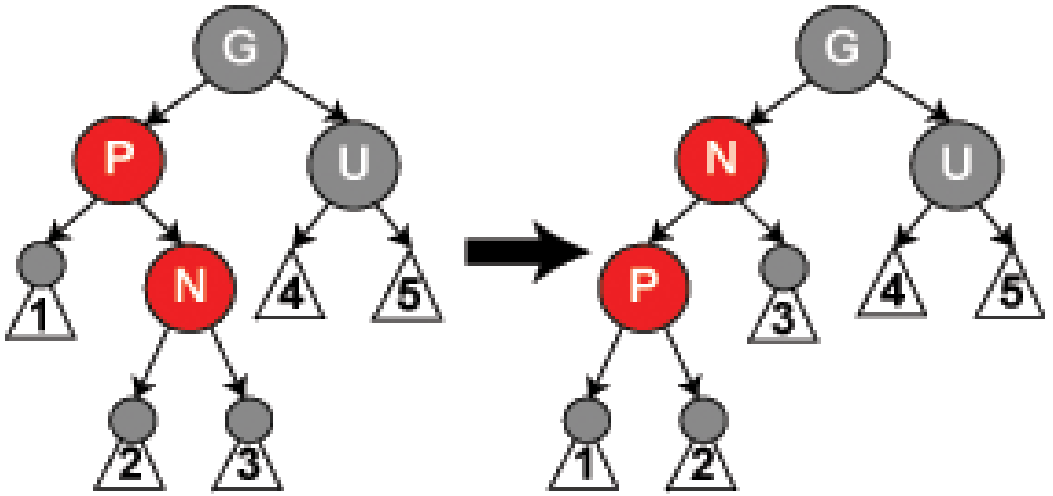
1. eset:
   - Az elem(N), a nagybácsi(U) és a szülő(P) is piros, N mindegy hogy melyik gyereke P-nek.
     - Cseréljük ki a színeket úgy, hogy a nagyszülő legyen csak piros, szülő és nagybácsi fekete!
   - Ezzel lehet, hogy a nagyszülőnek a felmenőjével lesznek konfliktusai, ezért állítsuk át az ellenőrzési mutatót a nagyszülőre, és kezdjük elölről!



Beszúrás

2. eset:

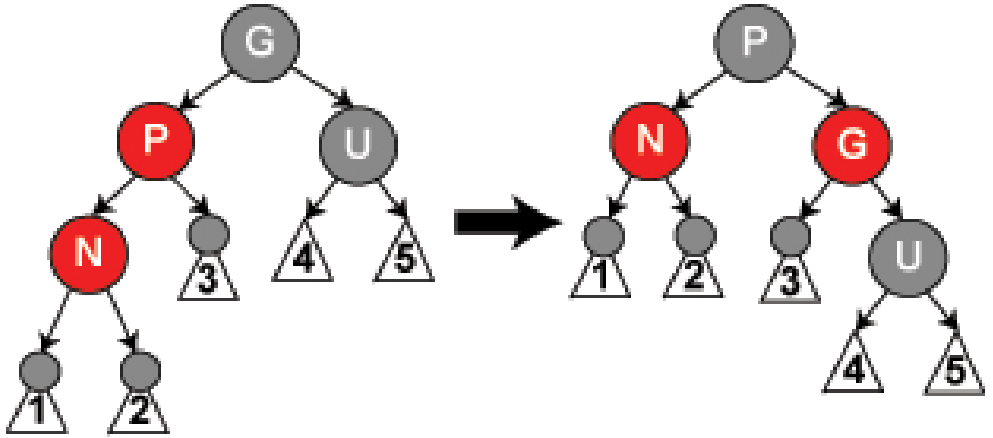
- Az aktuális elem(N) és a szülő(P) piros, de a nagybácsi(U) fekete és N ellenkező oldali gyereke P-nek, mint P, G-nek.
- Forgassunk P szerint úgy, hogy azonos oldali gyermek legyen az elem, mint a szülője (az ábra esetében balra forgatás).
- Állítsuk át az ellenőrzést a régi szülőre, és folytassuk a következő esettel:



Beszúrás

3. eset:

- P piros, U fekete(mint előbb) de N ugyanolyan oldali gyermeke P-nek, mint P G-nek.
- Cseréljük meg P és G színét,
- majd forgassunk G (nagyszülő) szerint úgy, hogy az eredeti szülő kerüljön fölé (az ábra szerinti jobbra forgatás).



Magyarázat a kiadott kódhoz

Nyissuk meg a kiadott projektet!

A szín tárolásához bevezetünk egy felsoroló (enumeration) szín típust

- `enum color_t { black, red };`

Az értéket nem tároló levelek (`NIL`) számára ténylegesen lefoglalunk egy `empty_leaf` csúcsot.

- `empty_leaf` bal és jobb gyereke, valamint szülője önmaga
- `empty_leaf` színe fekete
- `empty_leaf` kulcs mezője lényegtelen

Ez sok esetben jelentősen egyszerűsíti a dolgunkat.

Órai feladat

Az előző esetek szerint implementáljuk a kiadott kódban a `_rebalance_after_insert` függvényt!

Törlés

Ugyanúgy fogunk neki, mint a bináris keresőfánál

Ha a kidrótozott csúcs színe **piros**, akkor a piros-fekete fa szabályok nem sérülnek, nincs semmi dolgunk. ☺

Ha a kidrótozott csúcs színe **fekete**, akkor ezt a fekete színt átadjuk a gyerekének.

Kétszeresen **fekete** csúcs keletkezhet.

- Vajon melyik lesz ez közvetlenül a törlés után? ☺

Itt is esetek szerint kell a fában felfelé haladva helyreállítani a piros-fekete tulajdonságokat.

Törlés

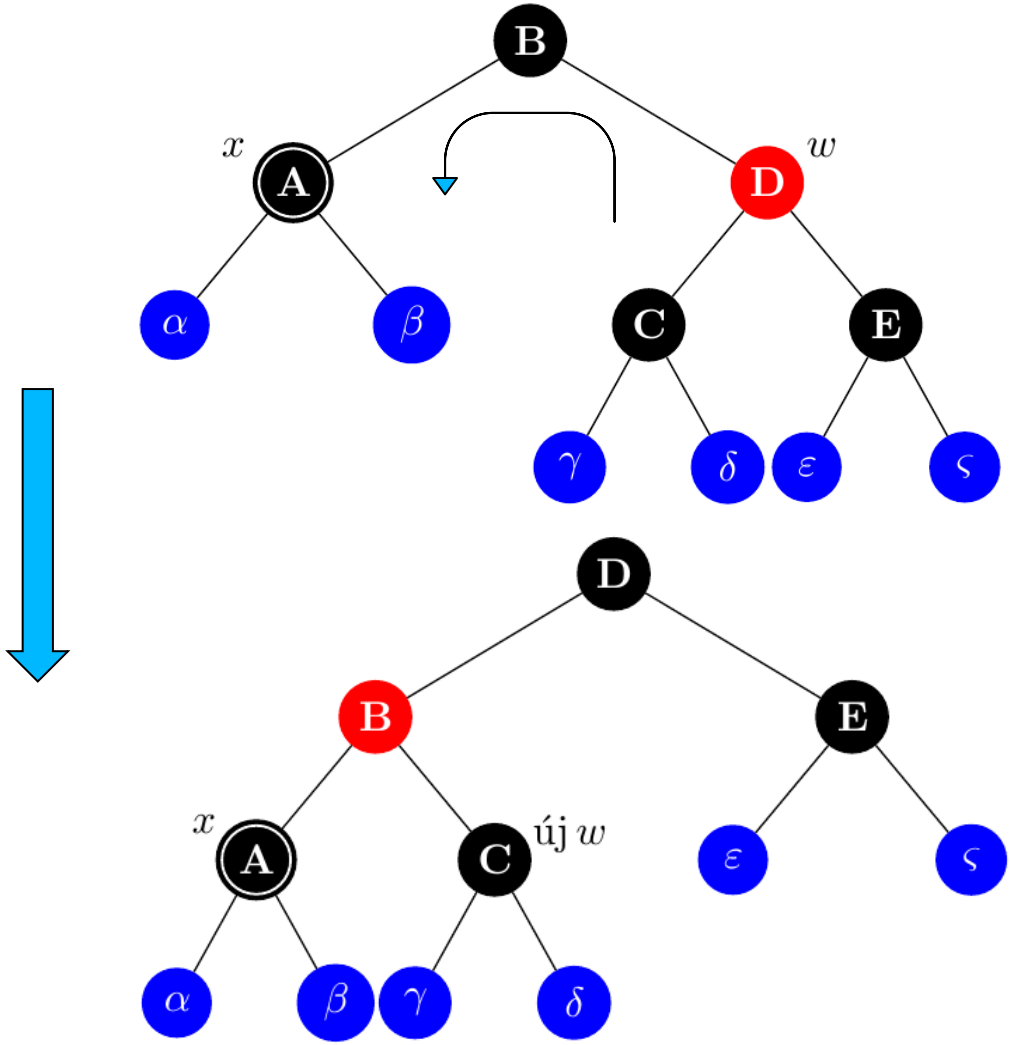
Jelölések:

x – kétszeresen fekete node

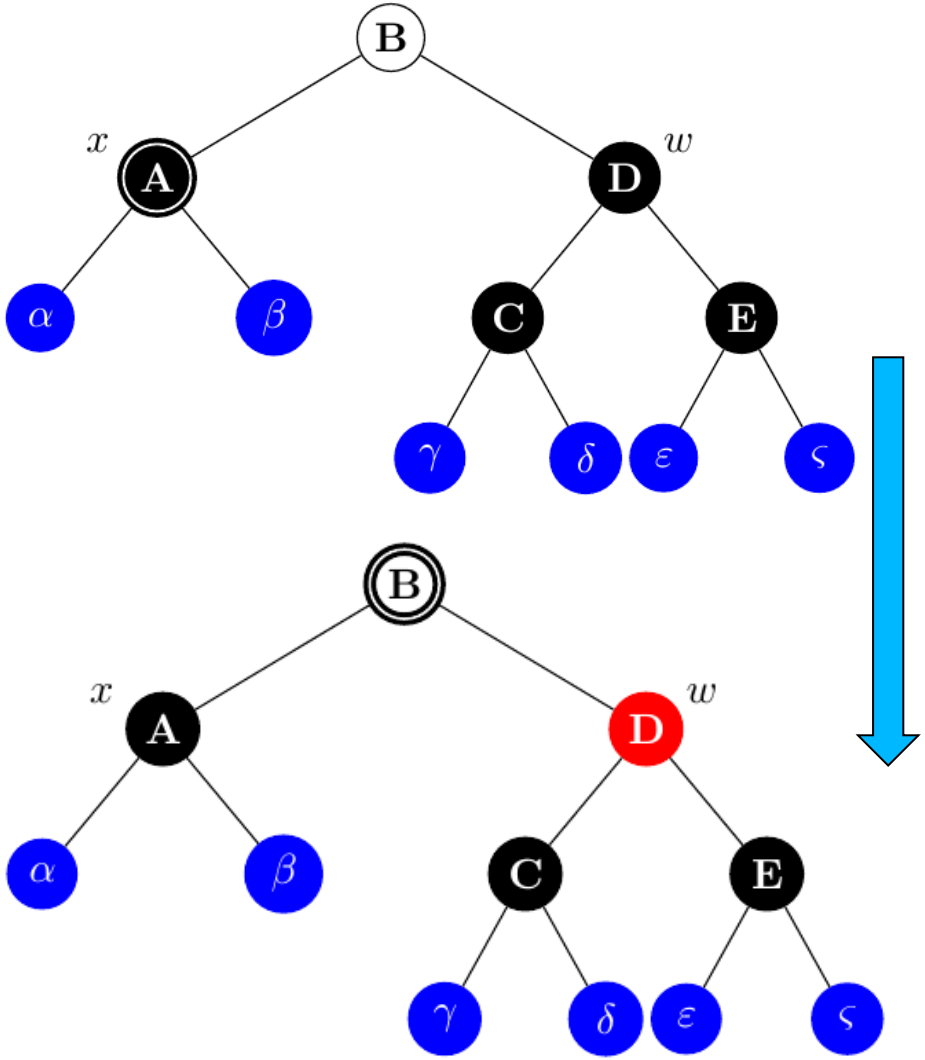
w – **x** aktuális testvére (figyelem **w** a forgatások során változhat)

Törlés

1. eset:
   - testvér piros
   - Megoldás:
     - B,D átszínezése
     - forgatás a szülő körül
   - Következmény:
     - (új) testvér fekete

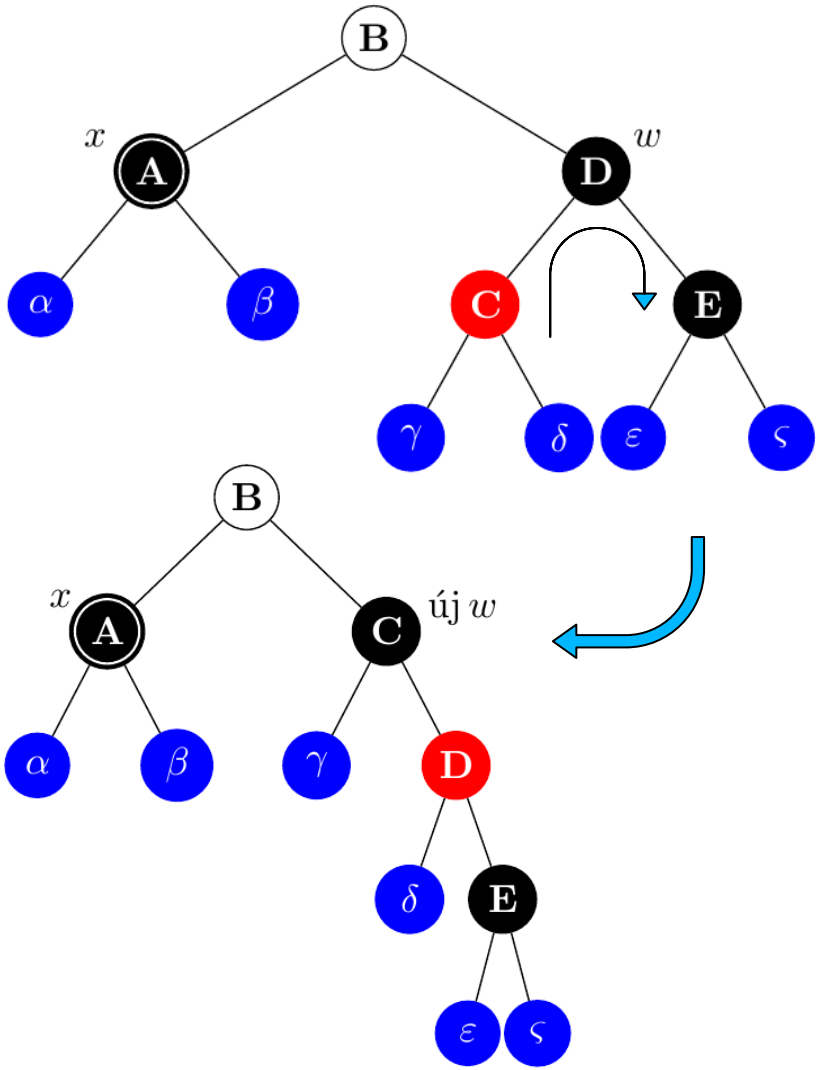


Törlés



2. eset:
- testvér fekete
- testvér gyermekei feketék
- szülő színe nincs kikötve
- Helyreállítás:
  - x legyen egyszeres fekete
  - w legyen piros
  - szülő kap egy extra feketét
- Következmény:
- a dupla fekete egy szinttel feljebb propagál

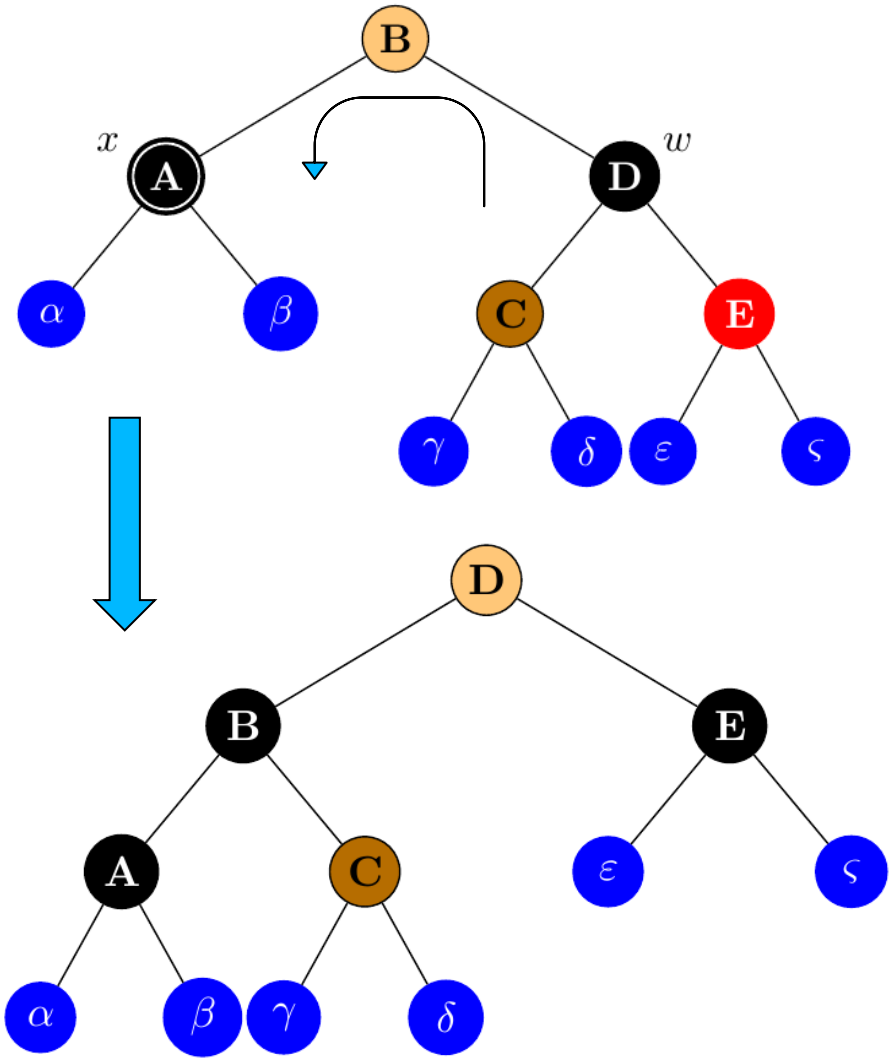
Törlés



3. eset:
- testvér fekete
- testvér távolabbi gyermeke fekete, másik piros
- Helyreállítás:
  - C, D átszínezése
  - forgatás w körül
- Következmény:
- testvér távolabbi gyermeke piros

Törlés

4. eset:
   - testvér fekete
   - testvér távolabbi gyermeke piros
   - Helyreállítás:
     - D megkapja B tetszőleges színét
     - B, E legyen fekete
     - forgatás a szülő körül
   - Következmény:
     - PF fa tulajdonságai helyreálltak.



Órai feladat

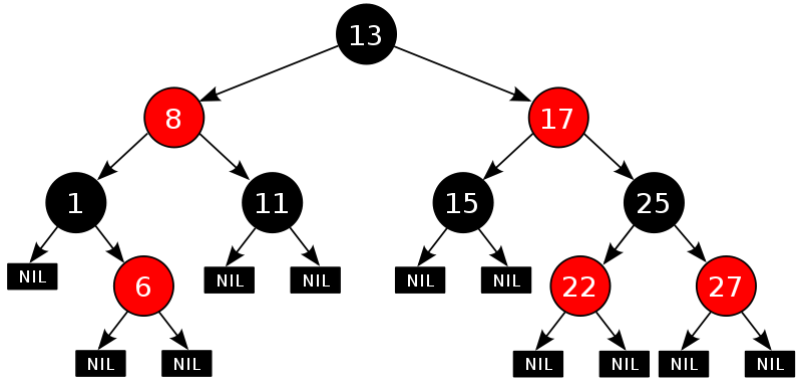
Előadáson tanult esetek szerint, illetve az előző pszeudokód alapján valósítsd meg a piros-fekete fa osztály _rebalance_after_remove függvényét!

Gyakorló feladat G07F01

Készíts vizualizációt a piros-fekete fához! Látható legyen a csúcsok kulcsa, színe, illetve a szülő-gyerek kapcsolatok a fában!

Nem muszáj grafikus megjelenítést csinálni, jó a konzolos is. Akár el is lehet forgatni a fát. Példa:

```
            (27)
        [25]-+
        |    (22)
    (17)-+
     |   [15]
[13]-+
     |   [11]
    ( 8)-+
        |    ( 6)
        [ 1]-+
```



Bal oldalt a fekete csúcsok szögletes, míg a pirosak kerek zárójelben láthatók.

Gyakorló feladat G07F02

Készíts egy teszt programot, amely összehasonlítja a Bináris Kereső, az AVL illetve PF fa használatát!

Ugyanazokat az elemeket (milliós nagyságrendű darabszám) szúrd be a BinKer, az AVL illetve a PF fába is és nézd meg, hogy hogyan változnak a fák magasságai (ezer, tízezer, százezer, millió … elem után).

A következő teszteket tudja elvégezni a program:

Elemek növekvő/csökkenő sorrendben történő beszúrása

Random elemek beszúrása

Gyakorló feladat G07F04

A **map** egy rendezett asszociatív konténer, mely kulcs-érték párokat tárol, ahol a kulcsok egyediek.

A feladat egy ilyen **map** osztály implementálása a Piros-fekete fa segítségével.

Legyen lehetőség a **map** osztálynál

- új elemet felvenni(beszúrás)
- elemet törölni
- Adott elemet(kulcs) lekérdezni
- kilistázni a tartalmát

A kulcs és az érték, lehet integer, string, és character is. (Template használata)

Házi feladat G07F06

- Egy map segítségével készítsetek statisztikát egy nagyon hosszú ékezetek nélküli (ASCII) szöveg szavainak előfordulására.
- **Bemenet**:
  - a program bemeneteként egy ”in.txt”-t fogadjon, amely a feldolgozandó szöveget tartalmazza.
  - a szövegben található írásjegyeket ne vegyétek figyelembe, továbbá kis és nagy betű sem számít.
- **Kimenet**:
  - **a)** írassátok ki a szavakat ábécé rendben (mellé az előfordulásuk számát)
  - **b)** írassátok ki a szavakat az előfordulásuk számának **csökkenő** sorrendjében, ha több szám ugyanannyiszor szerepelt, akkor ezeket soroltassátok fel a szám mellett (lásd pl.)

Házi feladat G07F06

- **Segítség**: ha az órán elkészített piros-fekete fa node osztályába egy kulcs-érték párost helyeztek és a kulcs szerint rendeztek, akkor egy map adatstruktúrát kaptok
- **Példaszöveg:**
  „That can I; At least, the whisper goes so. Our last king, Whose image even but now appear'd to us, Was, as you know, by Fortinbras of Norway, Thereto prick'd on by a most emulate pride, Dared to the combat; in which our valiant Hamlet-- For so this side of our known world esteem'd him-- Did slay this Fortinbras; who by a seal'd compact, Well ratified by law and heraldry, (...)”

Házi feladat G07F06

a) elvárt kimenet:

a ---> 5
against ---> 1
all ---> 1
and ---> 9
appear ---> 2
article ---> 1
as ---> 3
at ---> 1
been ---> 1
but ---> 2
by ---> 8
can ---> 1
carriage ---> 1
(...)

Házi feladat G07F06

b) elvárt kimenet

12 --> of, the

9 --> and

8 --> by

7 --> d, our, to

5 --> a, this, which

4 --> fortinbras, his, in

3 --> as, so

2 --> appear, but, did, for, had, hamlet, hath, he, i, is, it, king, lands, norway, now, that, those, us, was, well

1 --> against, all, article, at, been, can, carriage, chief, combat, compact, competent, compulsatory, conqueror, covenant, dared, design, diet, doth, emulate, enterprise, esteem, even, father, fell, food, foresaid, forfeit, full, gaged, goes, hand, haste, head, heraldry, here (...)