8. előadás

Speciális bináris fák

Tökéletesen kiegyensúlyozott fa, keresőfa, kiegyensúlyozott fa, AVL-fa

Adatszerkezetek és algoritmusok előadás
2018. március 27.

Kósa Márk, Pánovics János és Szathmáry László
Debreceni Egyetem
Informatikai Kar

Tökéletesen kiegyensúlyozott bináris fa

Kósa Márk
Pánovics János
Szathmáry László

Minimális magasságú fa

Azt mondjuk, hogy egy fa minimális magasságú, ha adott elemszám mellett a legalsó szint kivételével minden szintjén a lehető legtöbb adatelem helyezkedik el.

Tökéletesen kiegyensúlyozott bináris fa

Azt mondjuk, hogy egy bináris fa tökéletesen kiegyensúlyozott, ha bármely elemének bal és jobb oldali részfájában az elemek darabszáma legfeljebb 1-gyel tér el.

Megjegyzés

Minden tökéletesen kiegyensúlyozott fa minimális magasságú.

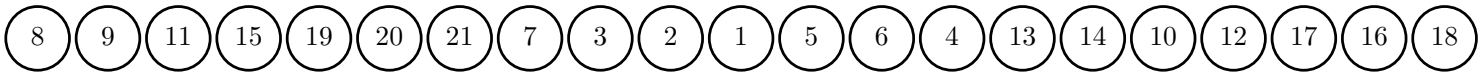
Tökéletesen kiegyensúlyozott bináris fa

Adott elemszámú (n adatelemet tartalmazó) tökéletesen kiegyensúlyozott bináris fa építésének algoritmusa

1. Ha az adatelemek száma 0, az eredmény egy üres fa, és ezzel az algoritmus véget ér.
2. Az első adatelem legyen a tökéletesen kiegyensúlyozott bináris fa gyökéreleme.
3. Osszuk két részre a megmaradt $n-1$ elemet, és
   a) az első $nb = \left\lfloor \frac{n}{2} \right\rfloor$ elemből építsük fel a gyökérelem bal oldali tökéletesen kiegyensúlyozott részfáját ugyanezzel az algoritmussal, majd
   b) a megmaradt $nj = n - 1 - nb$ elemből építsük fel a gyökérelem jobb oldali tökéletesen kiegyensúlyozott részfáját ugyanezzel az algoritmussal.

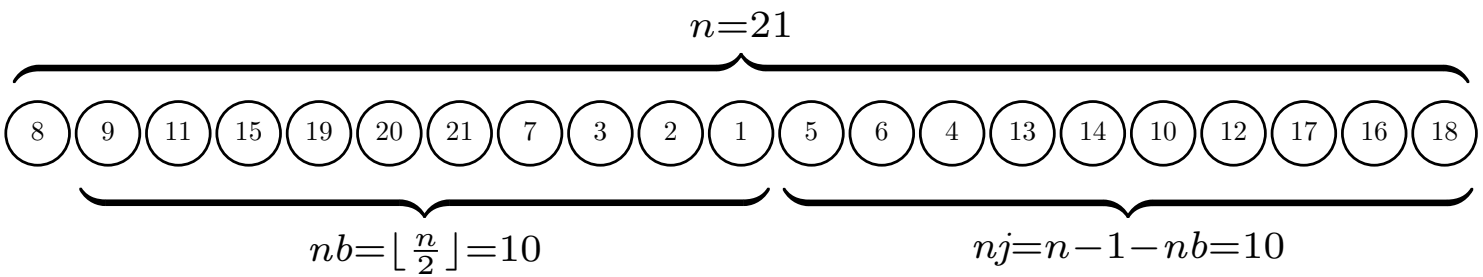
Kósa Márk
Pánovics János
Szathmáry László

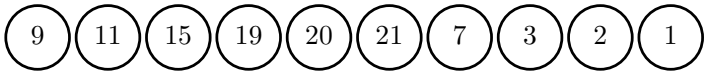
Tökéletesen kiegyensúlyozott bináris fa építése

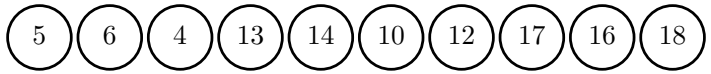


Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése

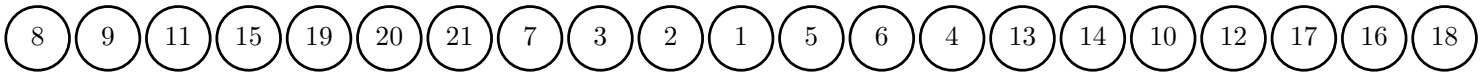


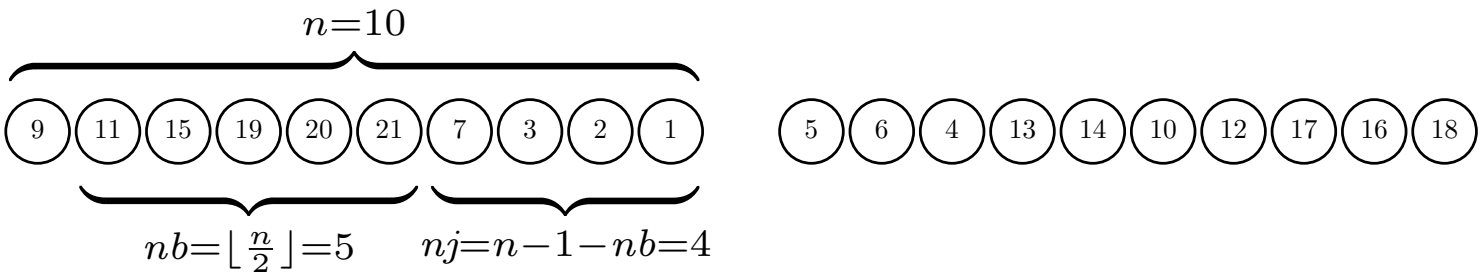




Kósa Márk
Pánovics János
Szathmáry László

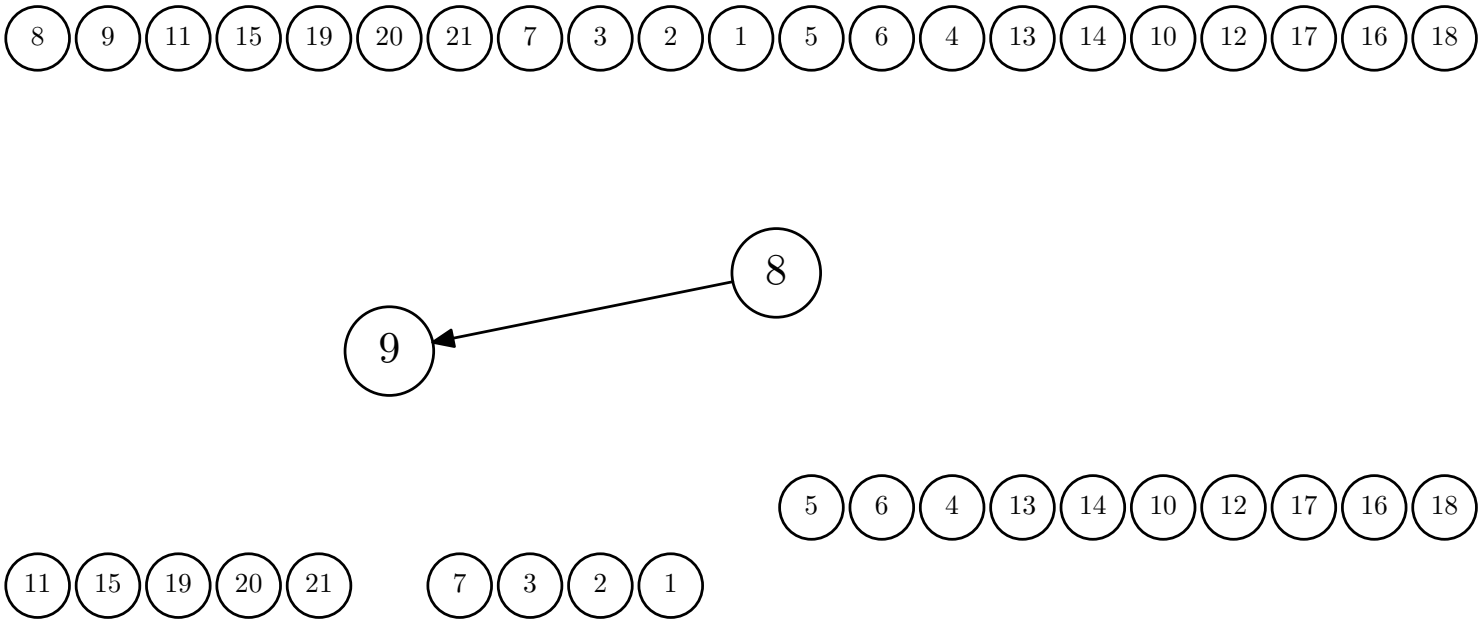
Tökéletesen kiegyensúlyozott bináris fa építése





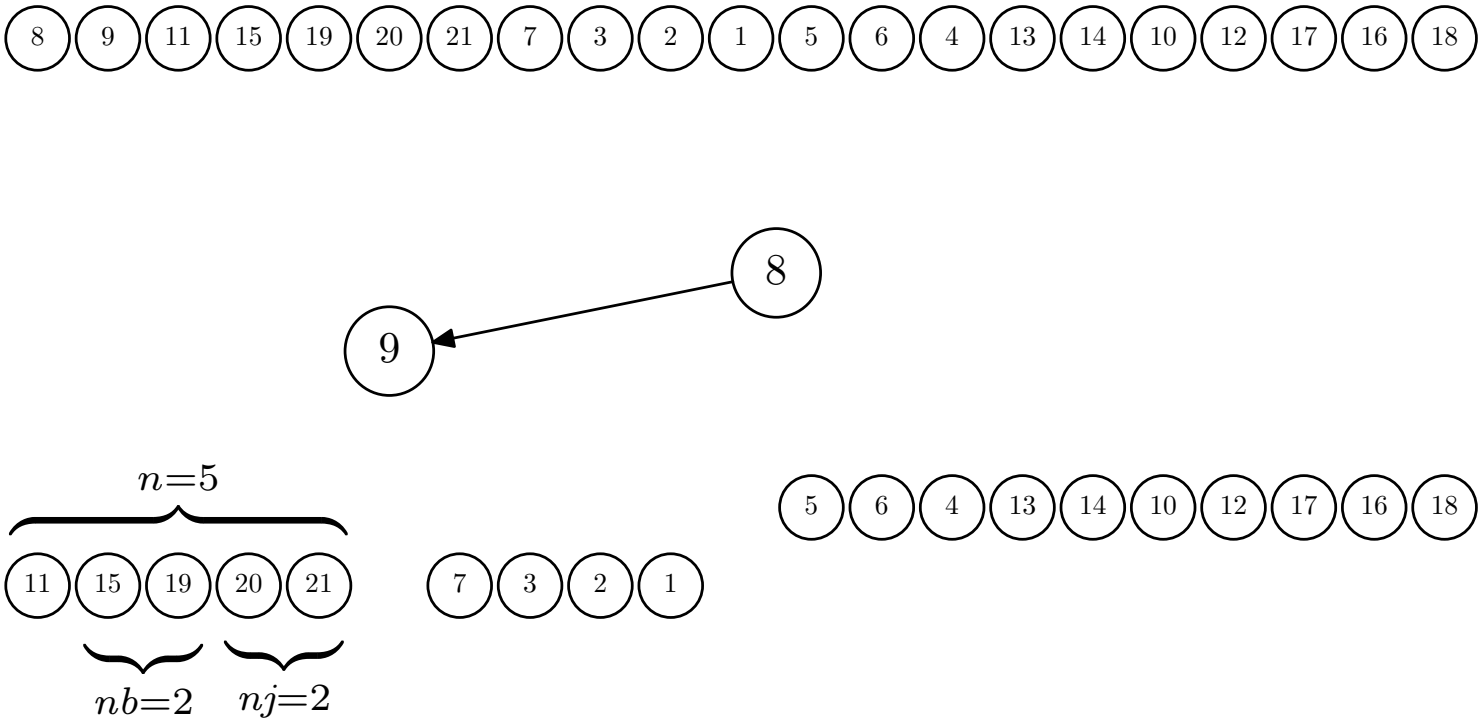
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



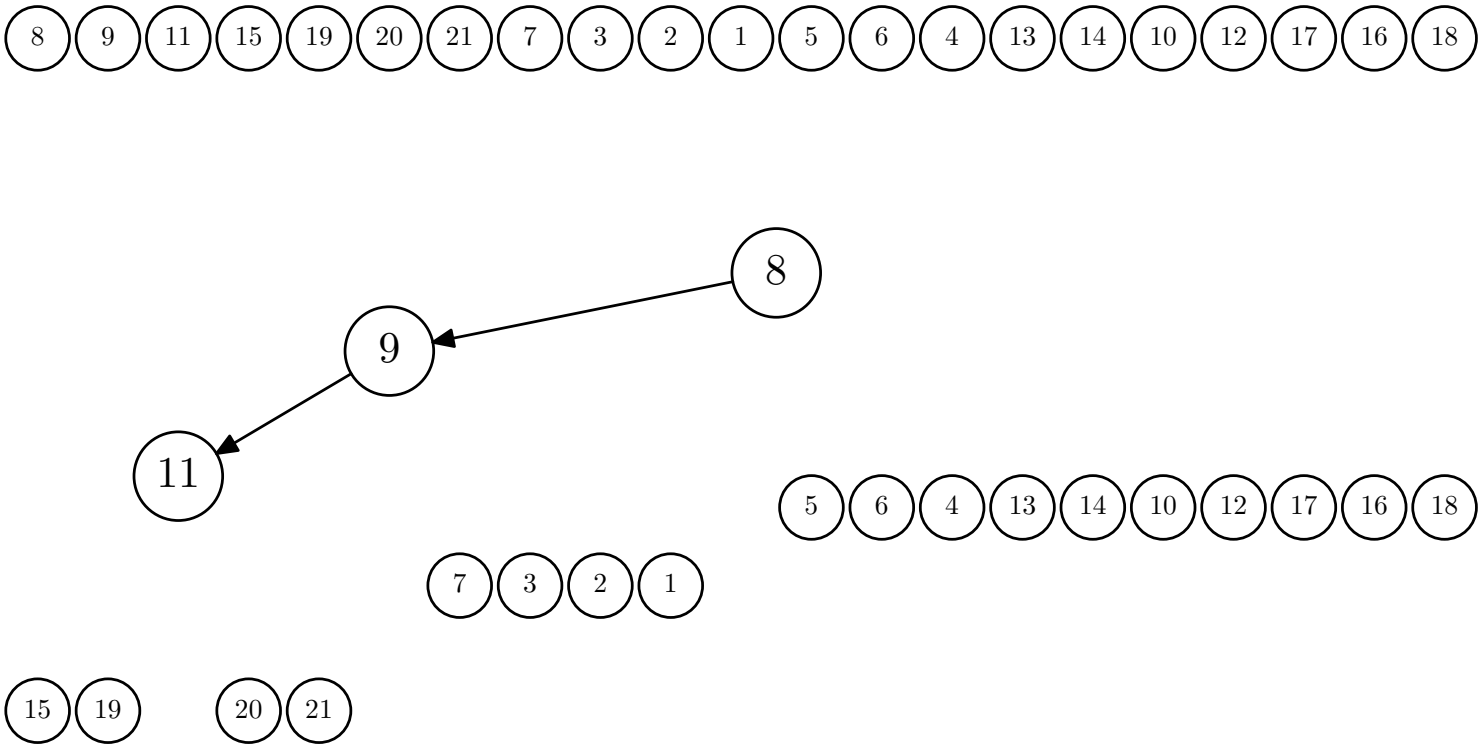
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



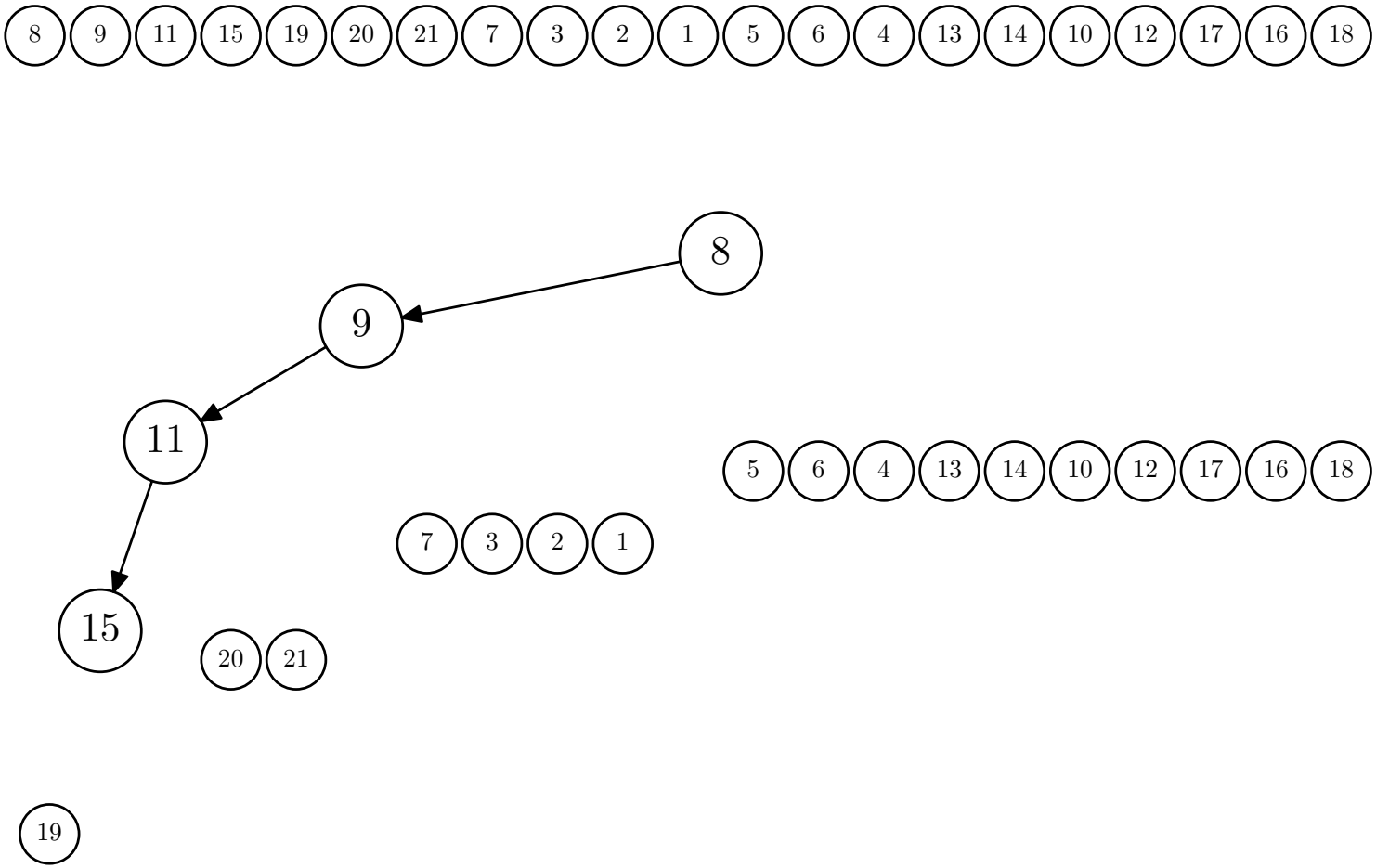
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



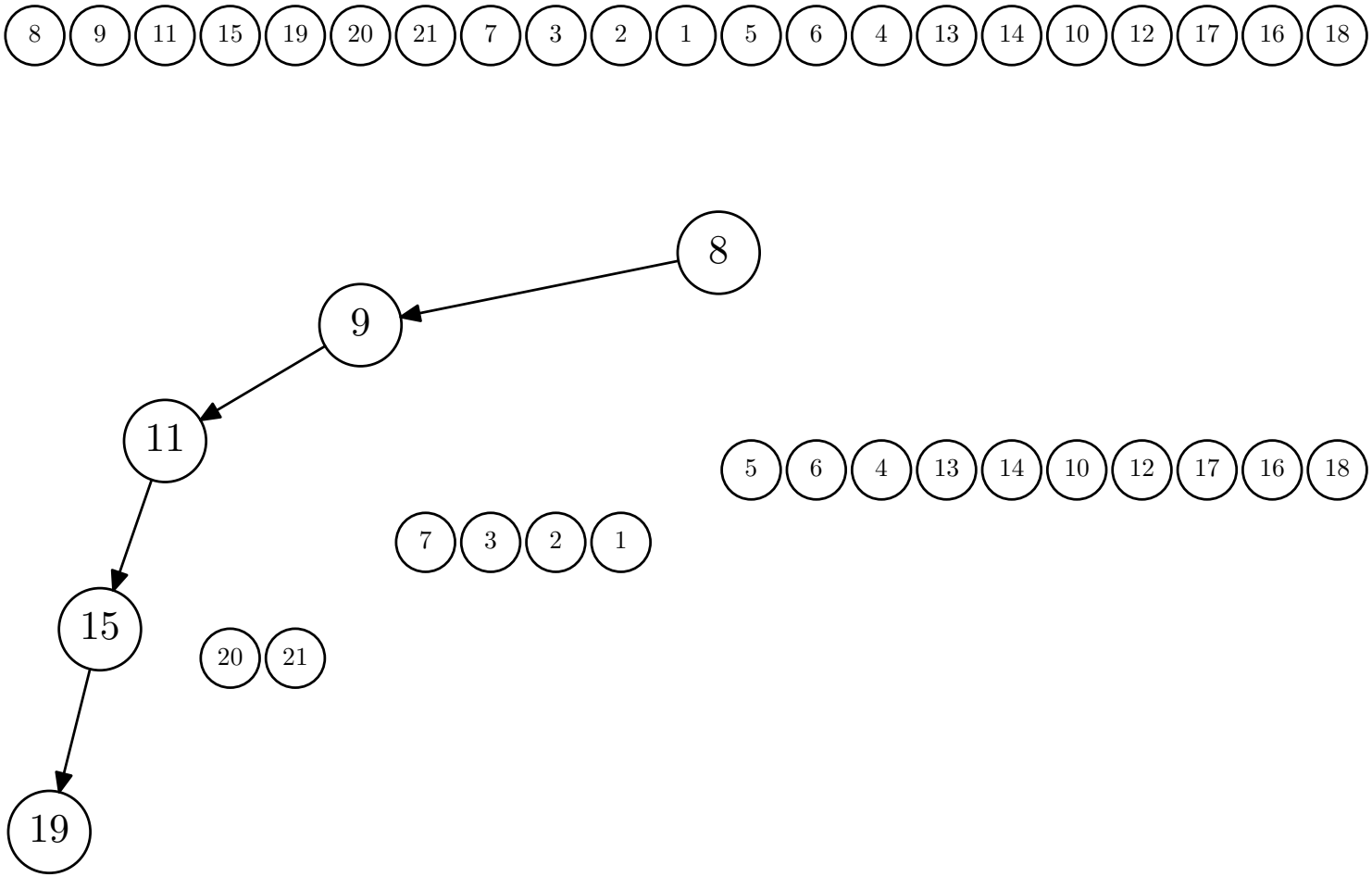
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



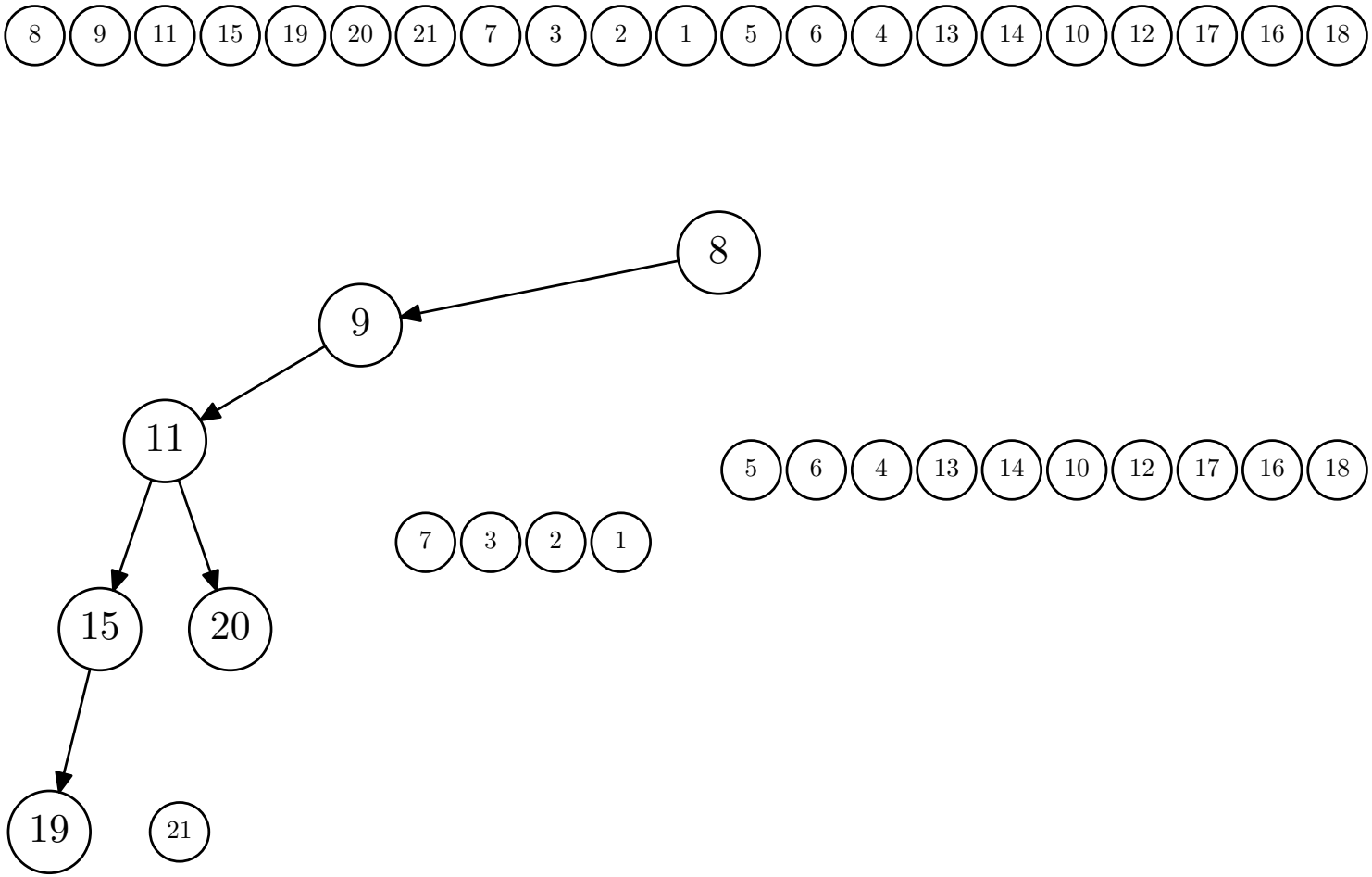
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



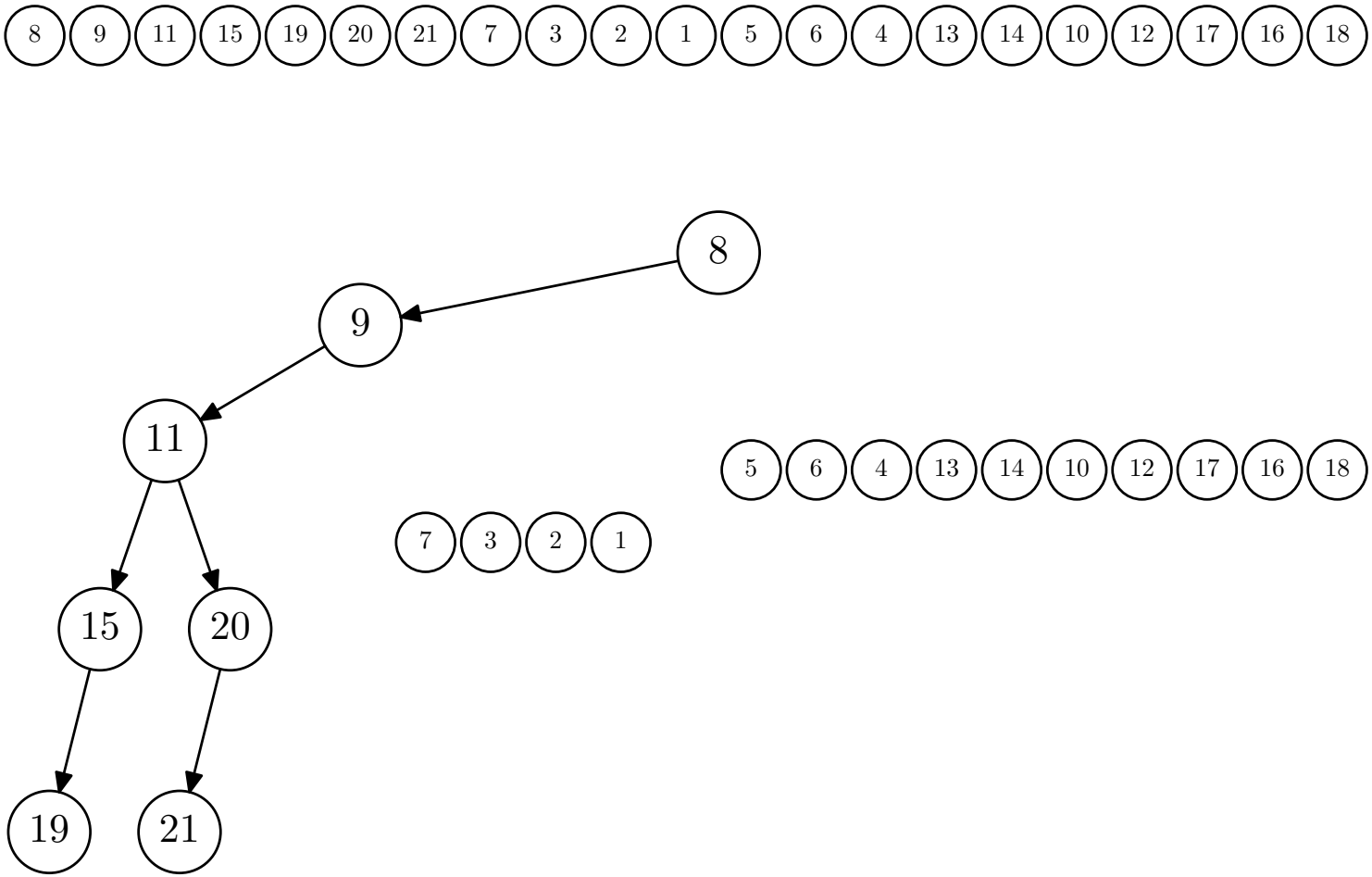
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



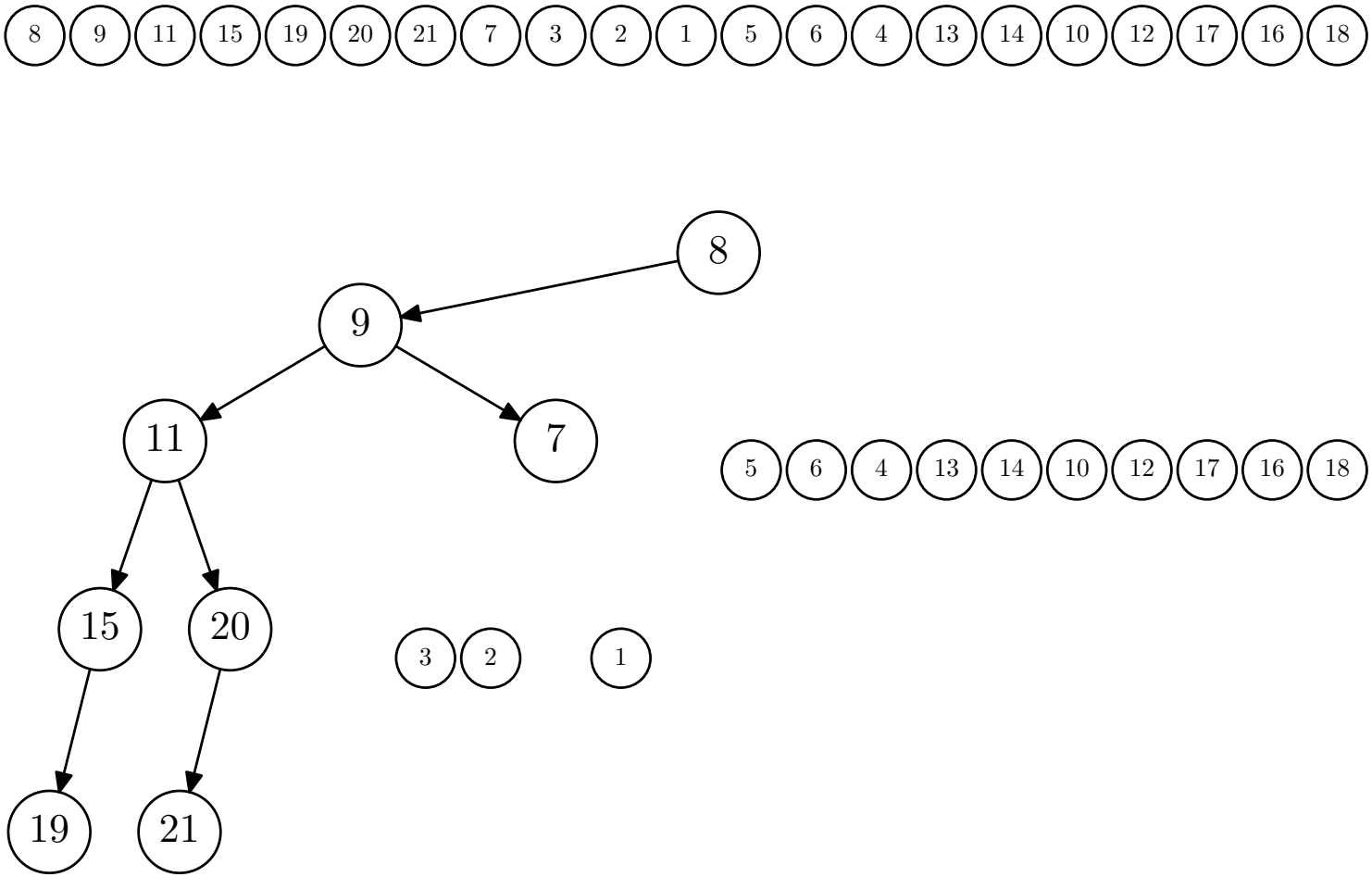
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



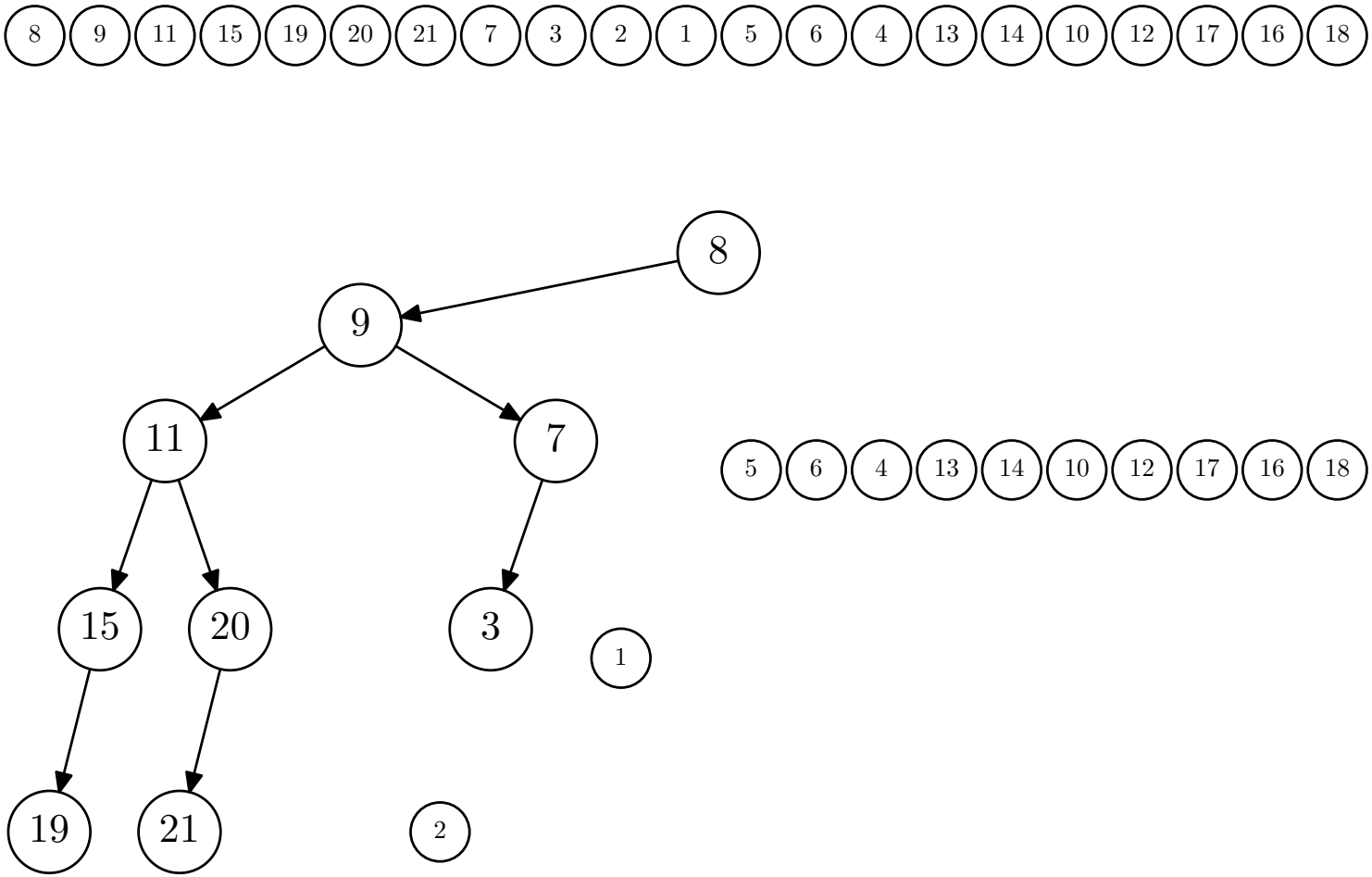
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



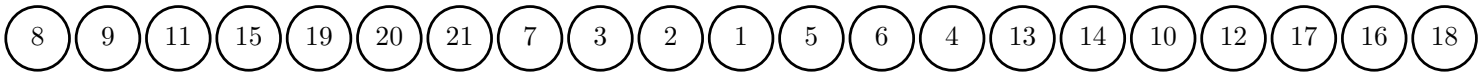
Kósa Márk
Pánovics János
Szathmáry László

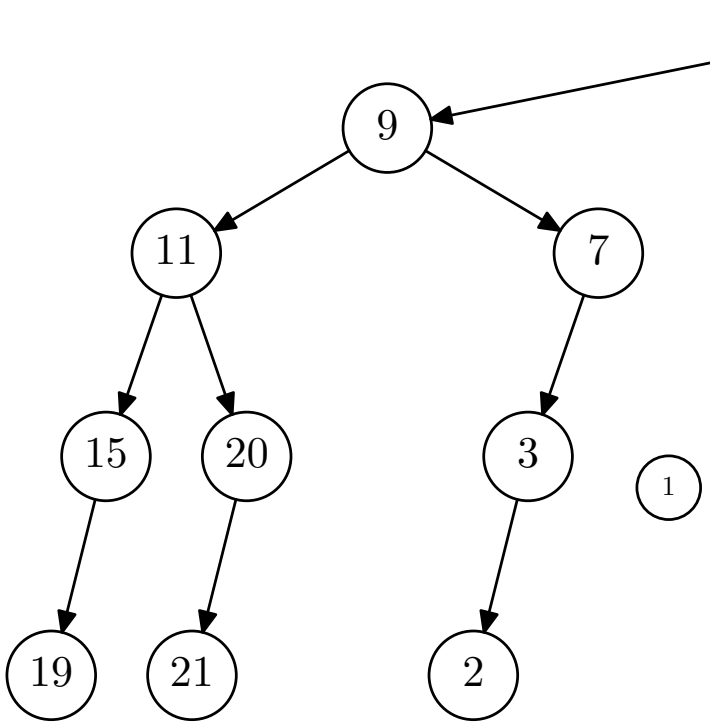
Tökéletesen kiegyensúlyozott bináris fa építése

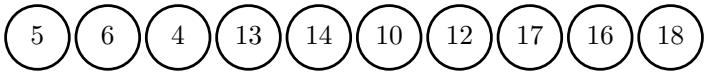


Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése

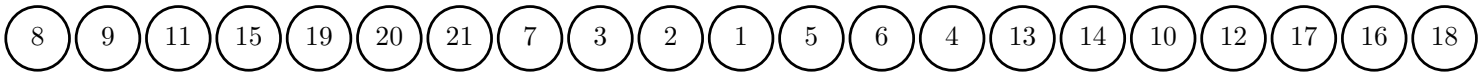


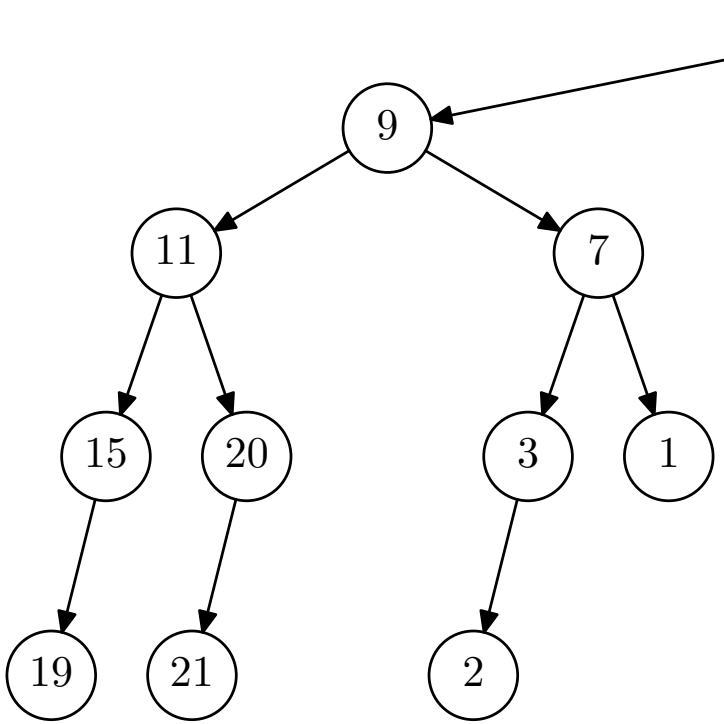


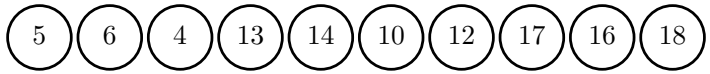


Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése

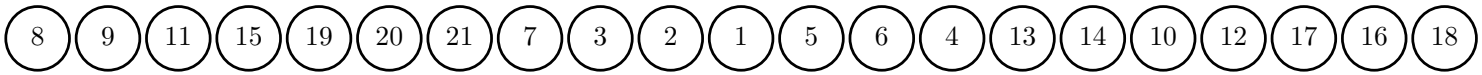


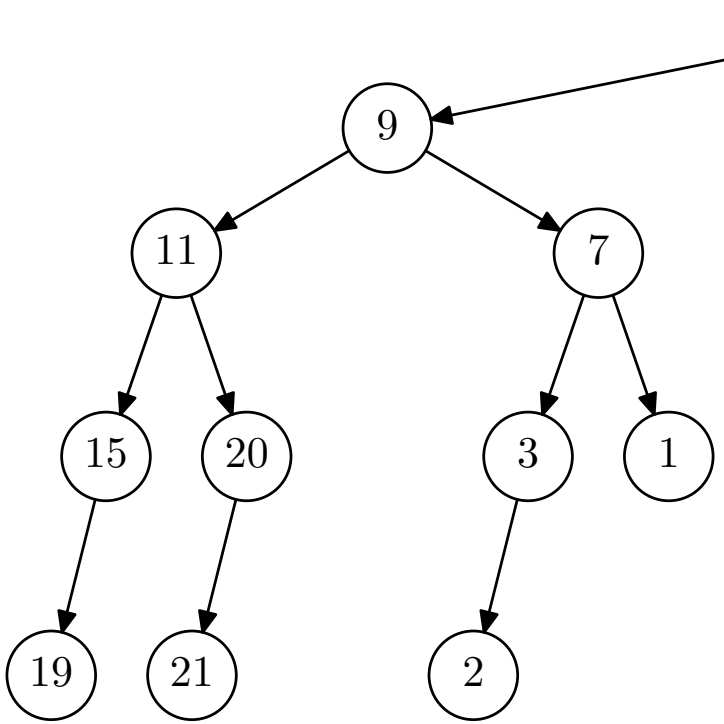


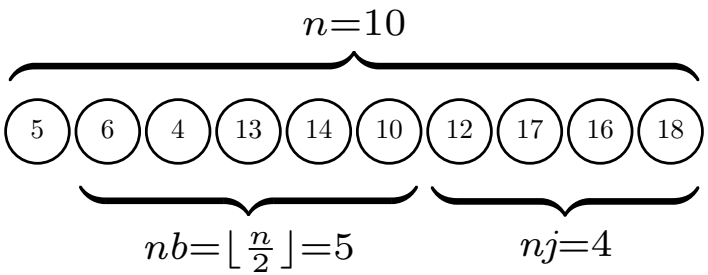


Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése

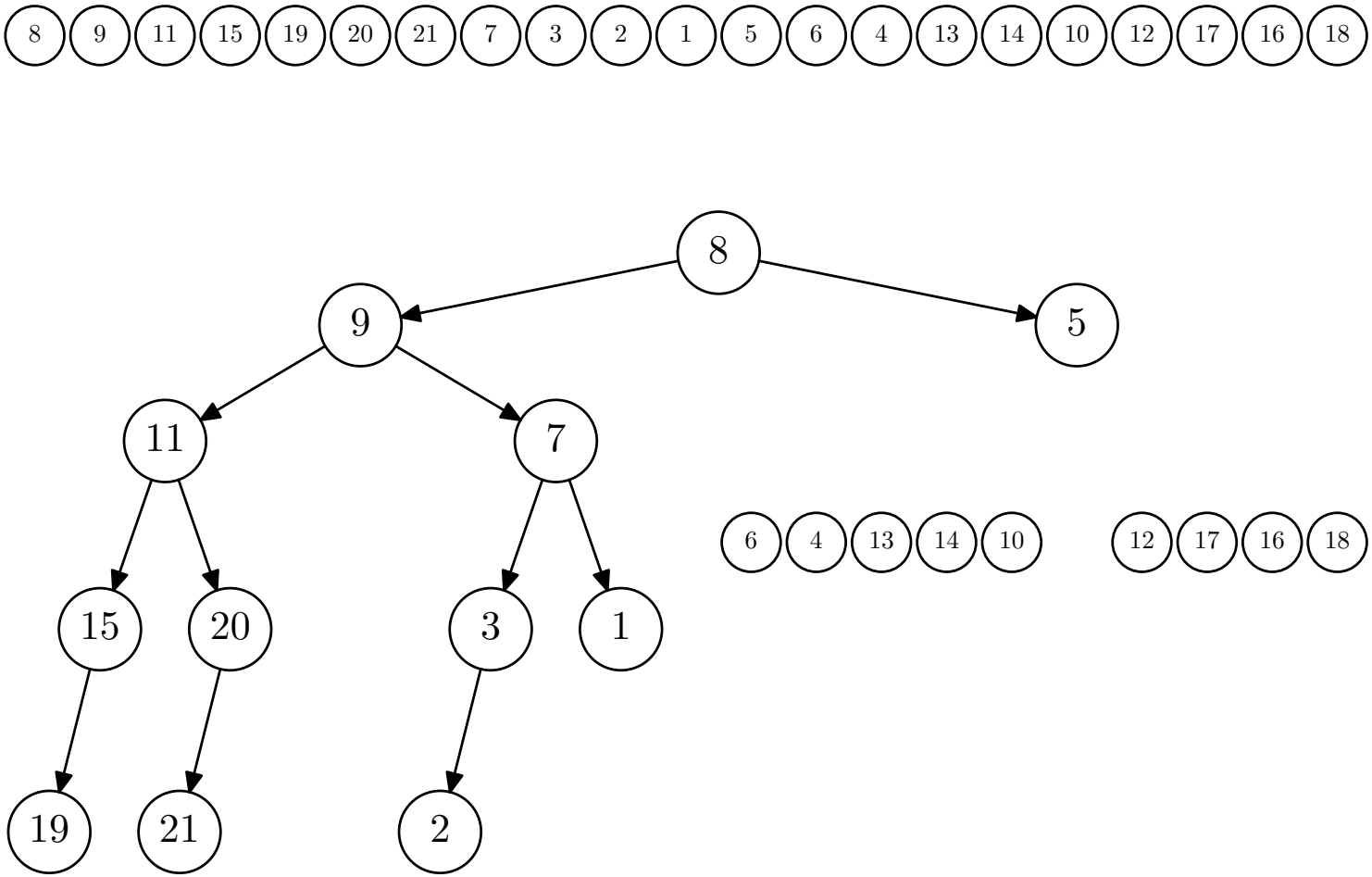






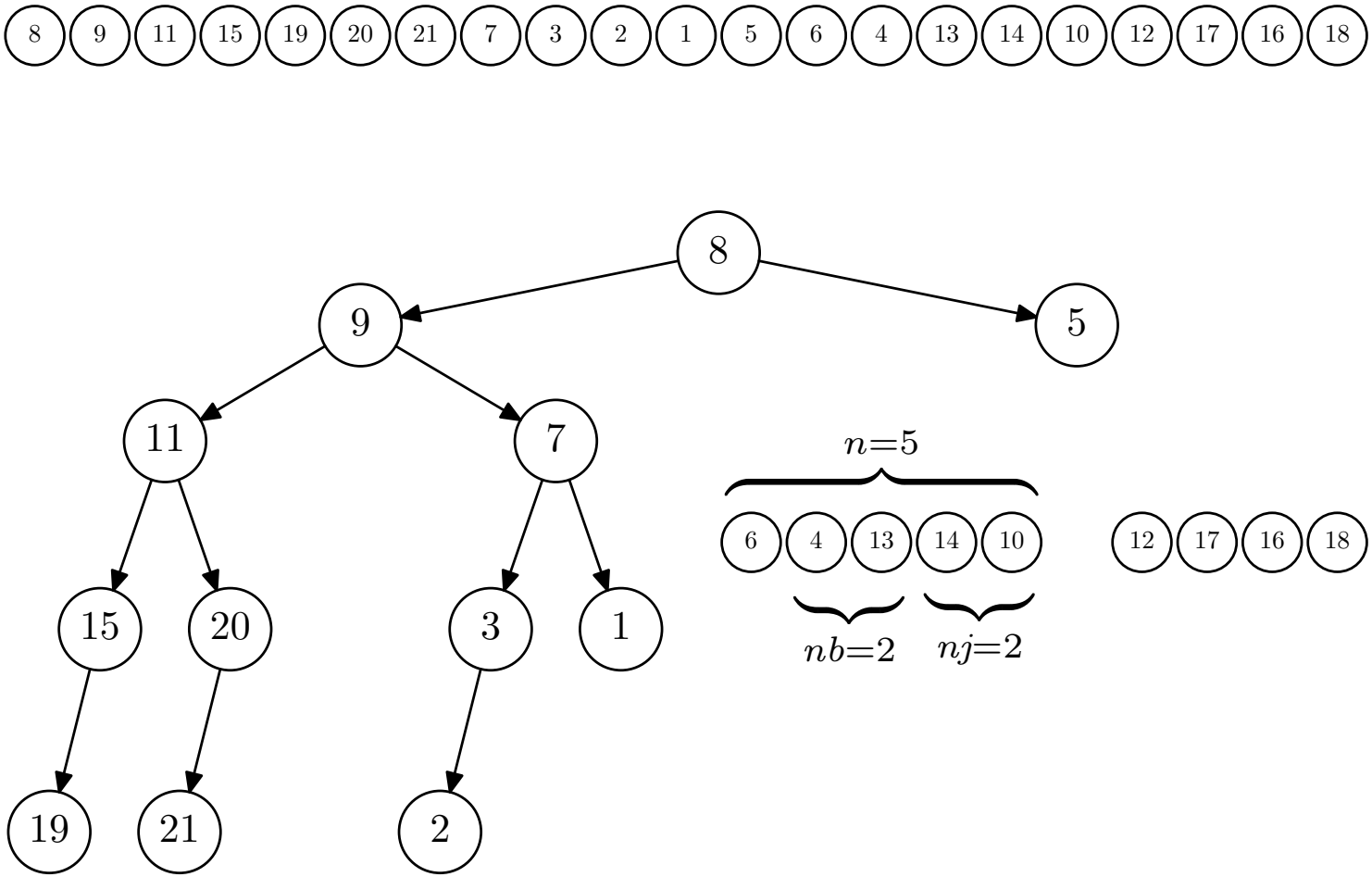
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



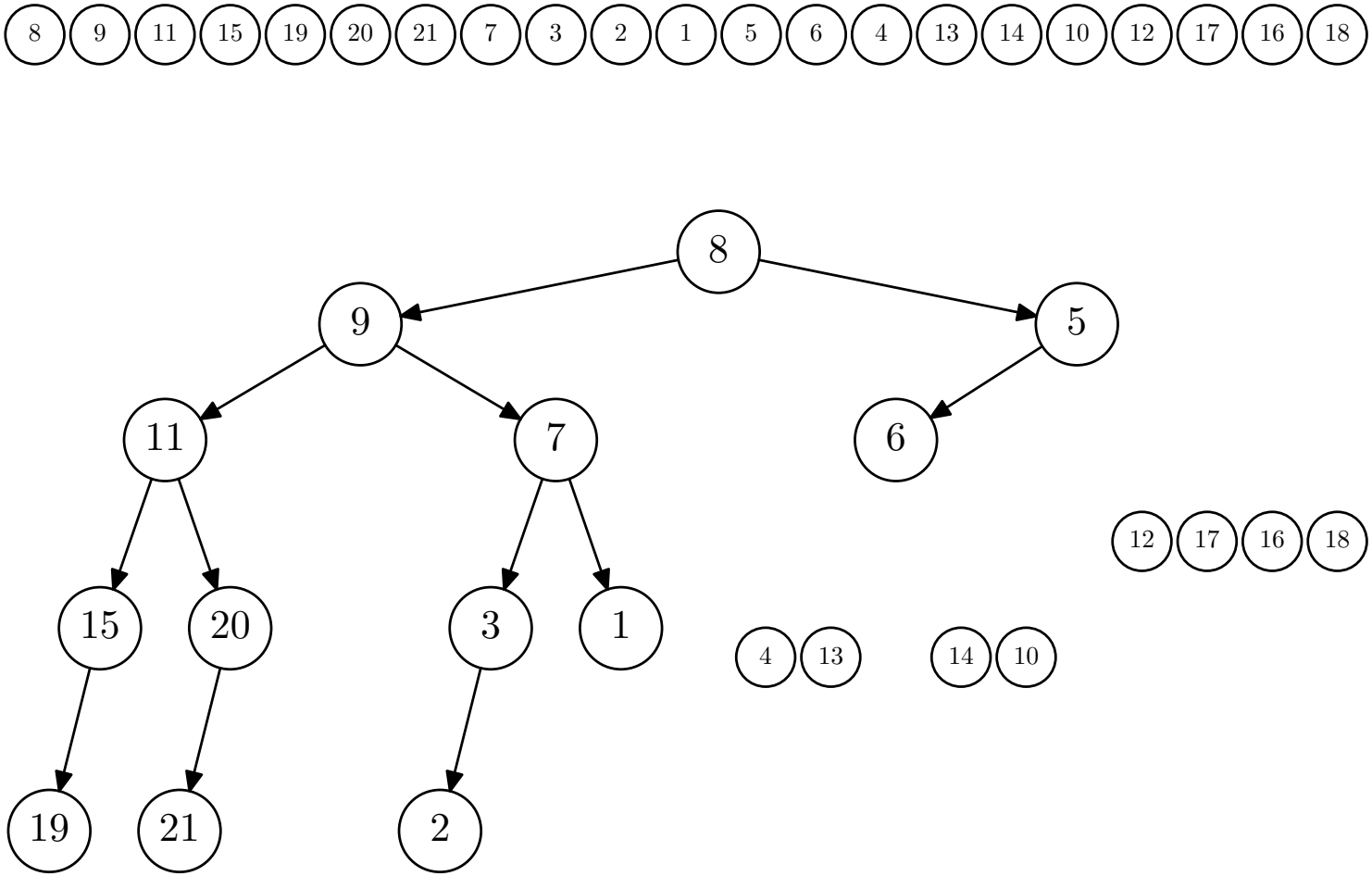
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



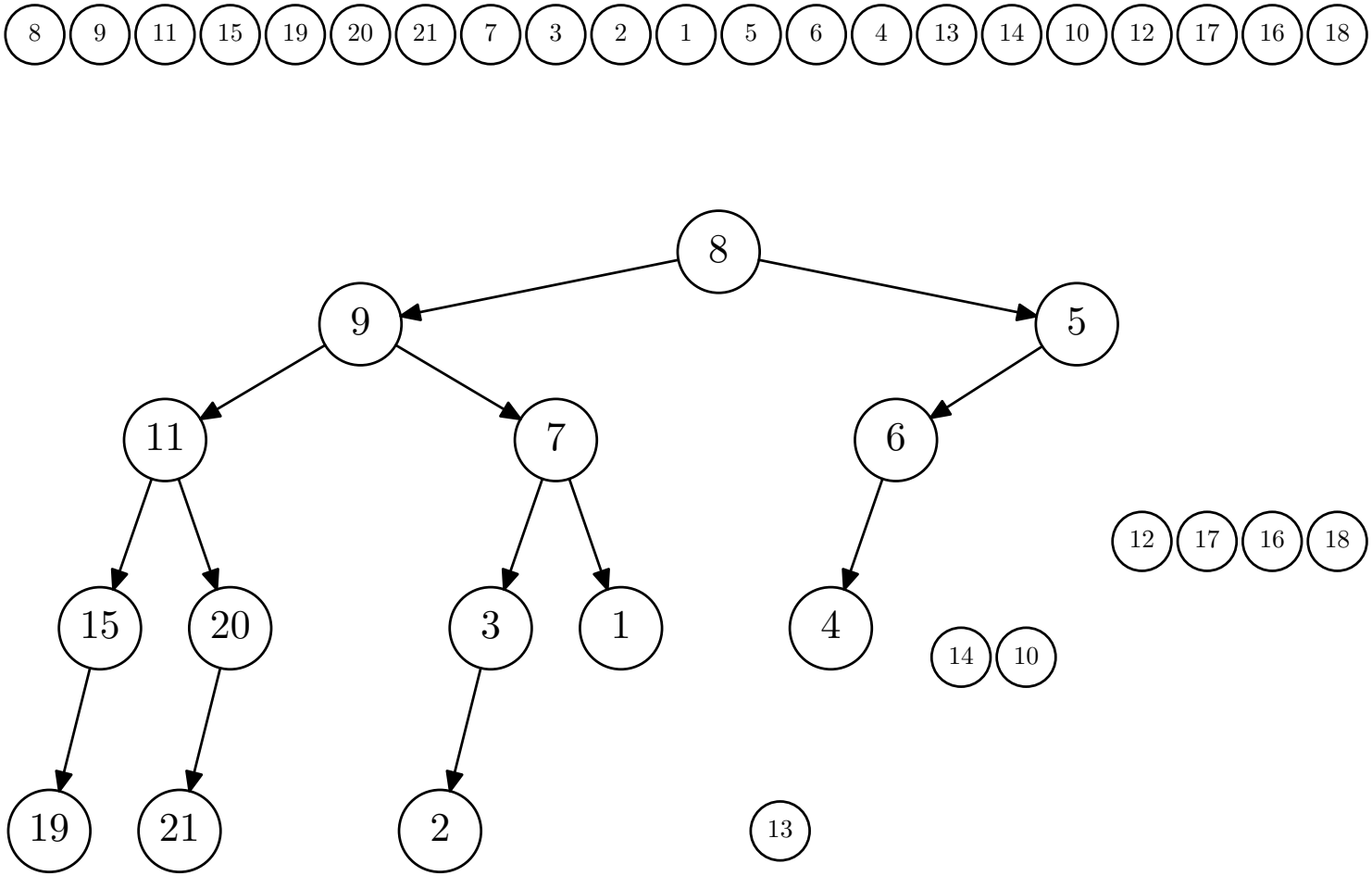
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



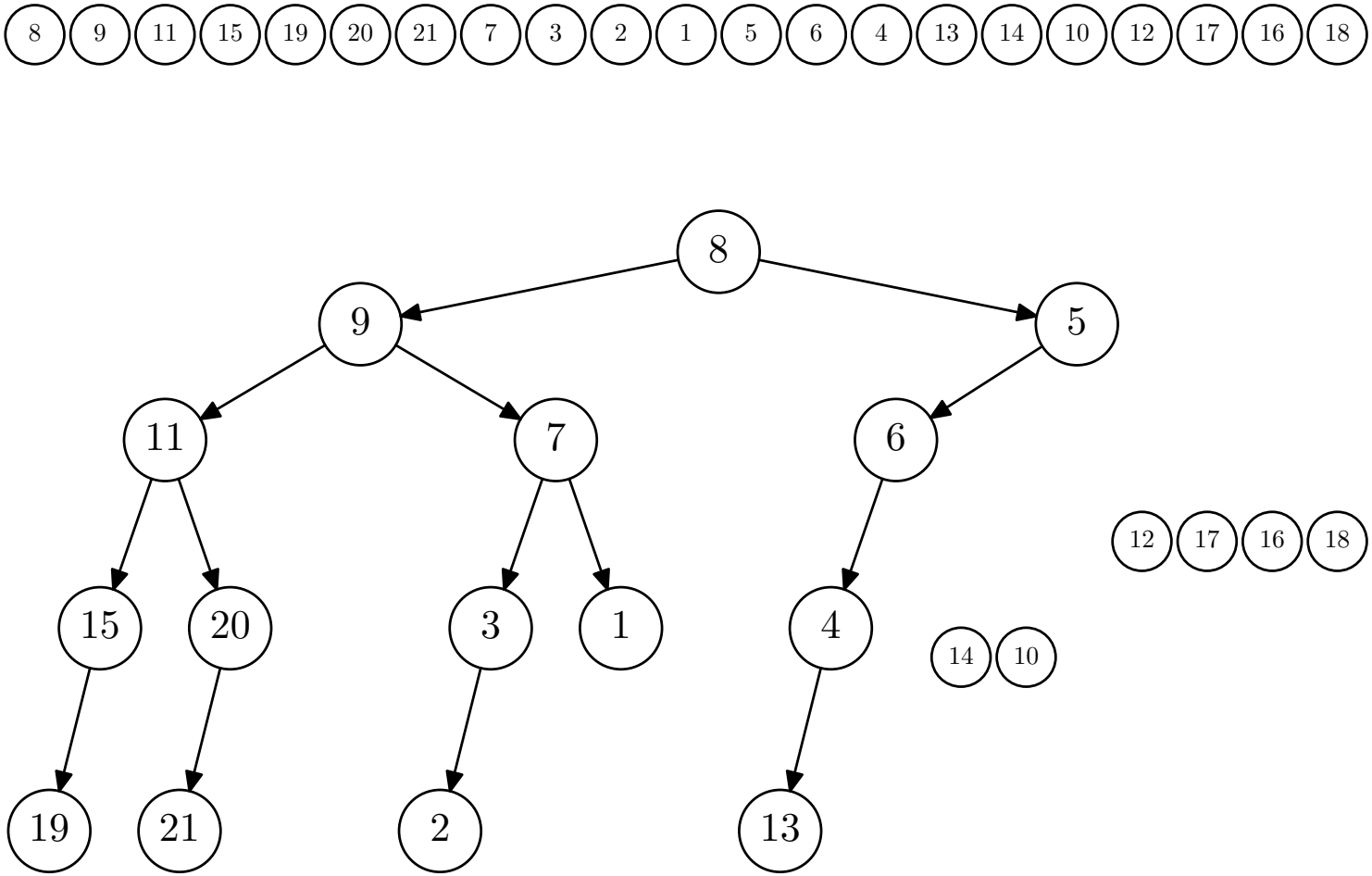
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



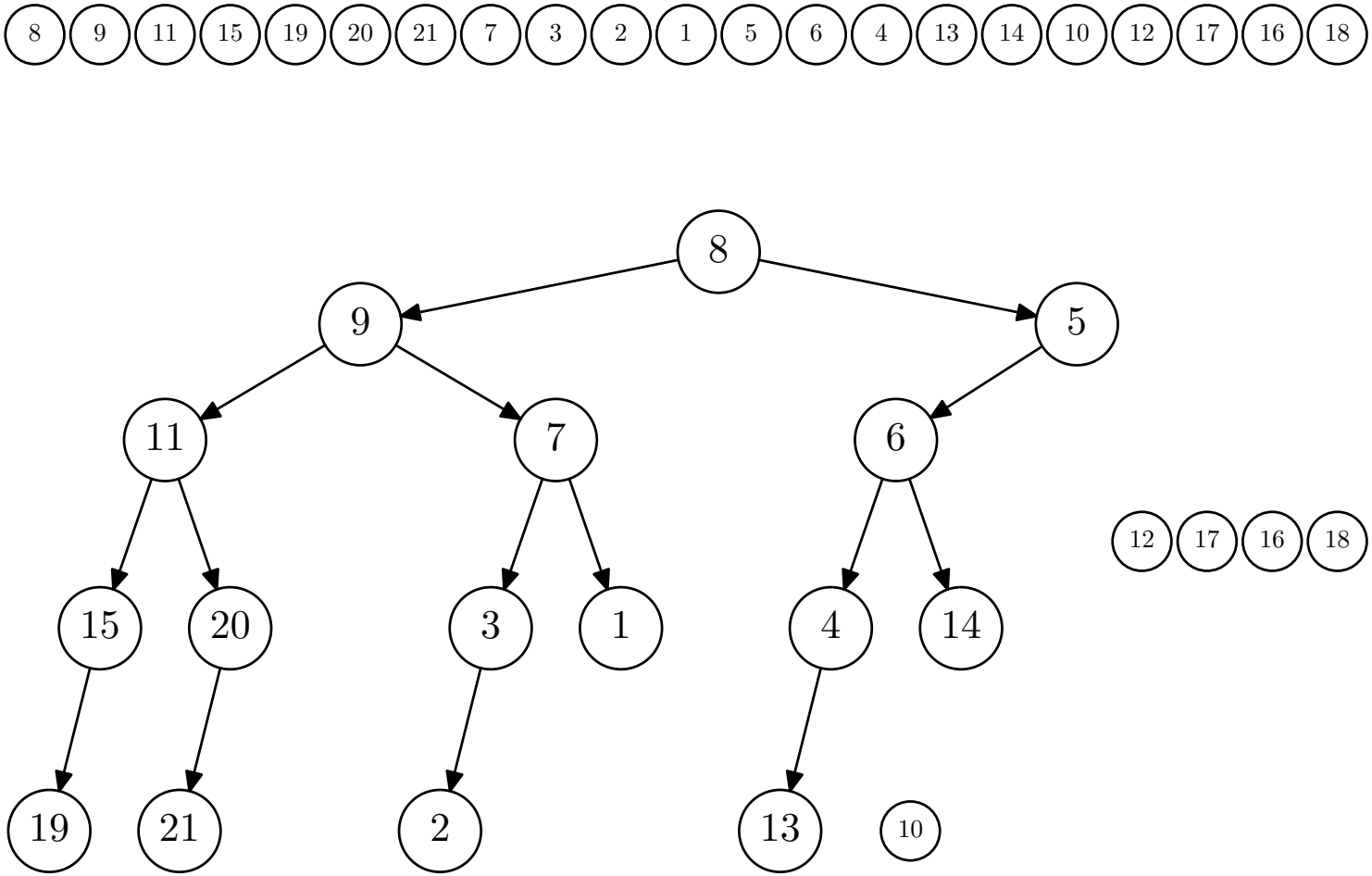
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



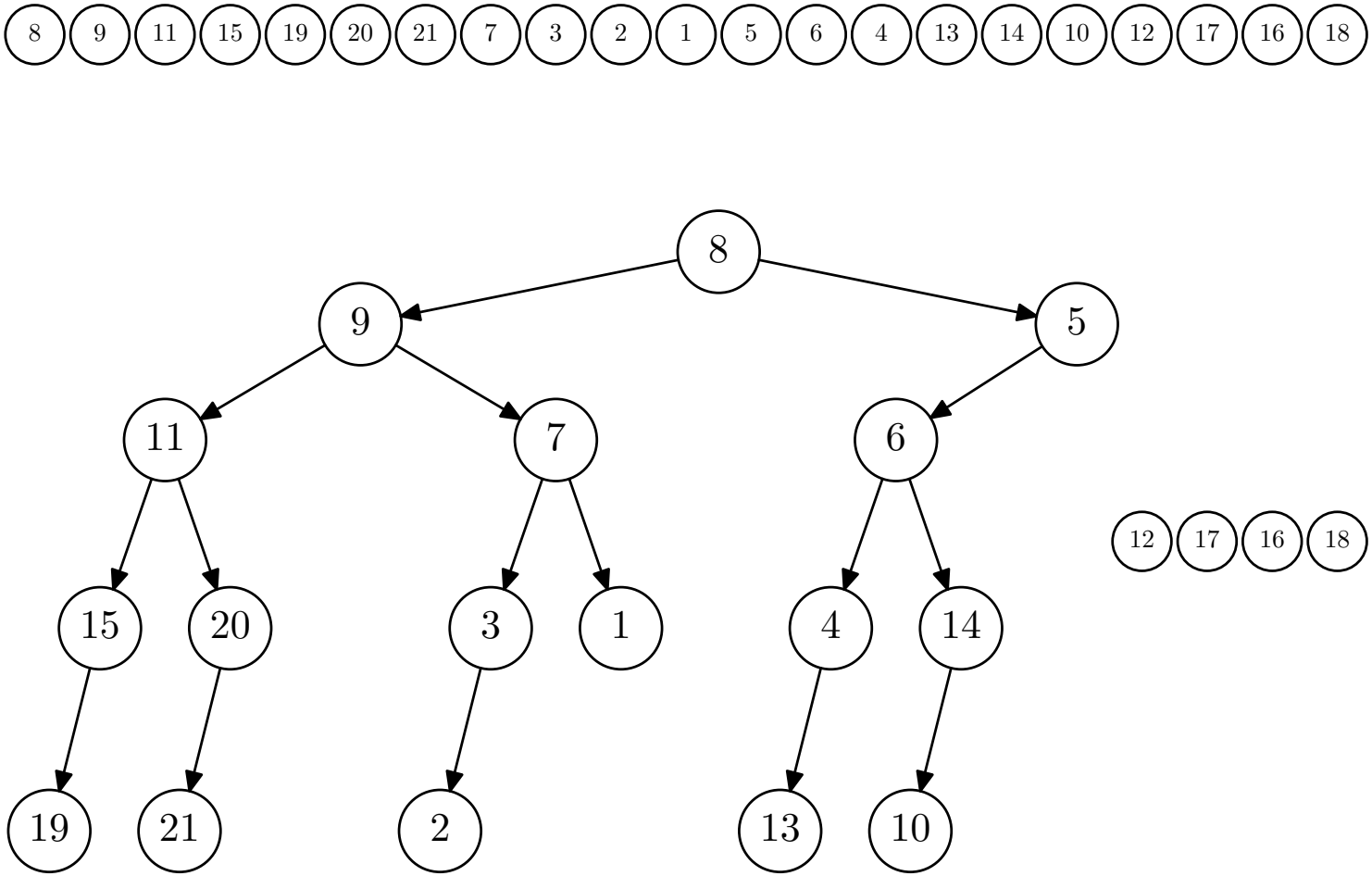
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



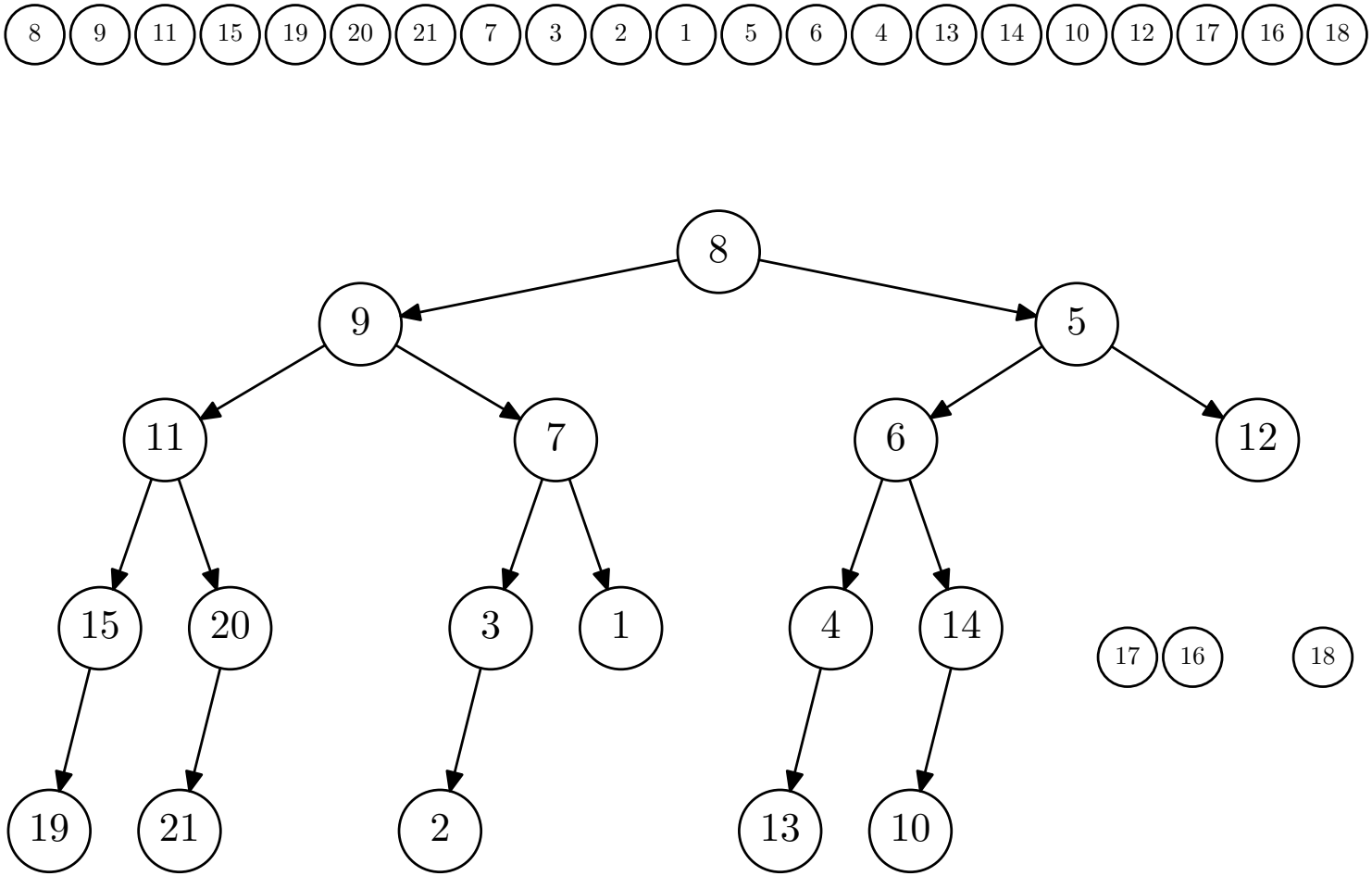
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



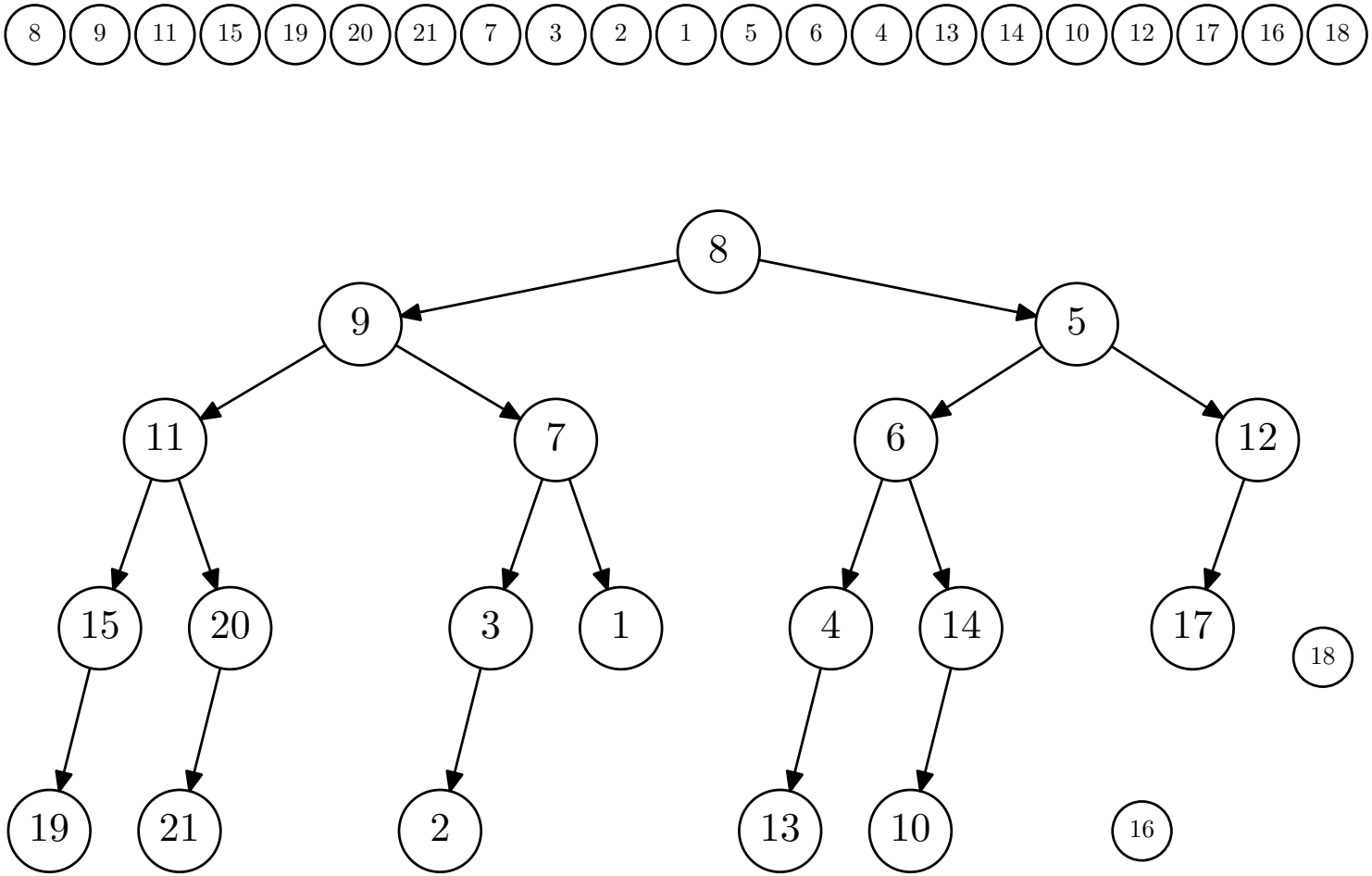
Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése



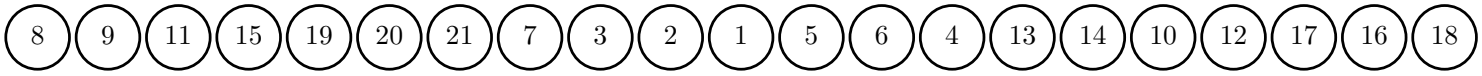
Kósa Márk
Pánovics János
Szathmáry László

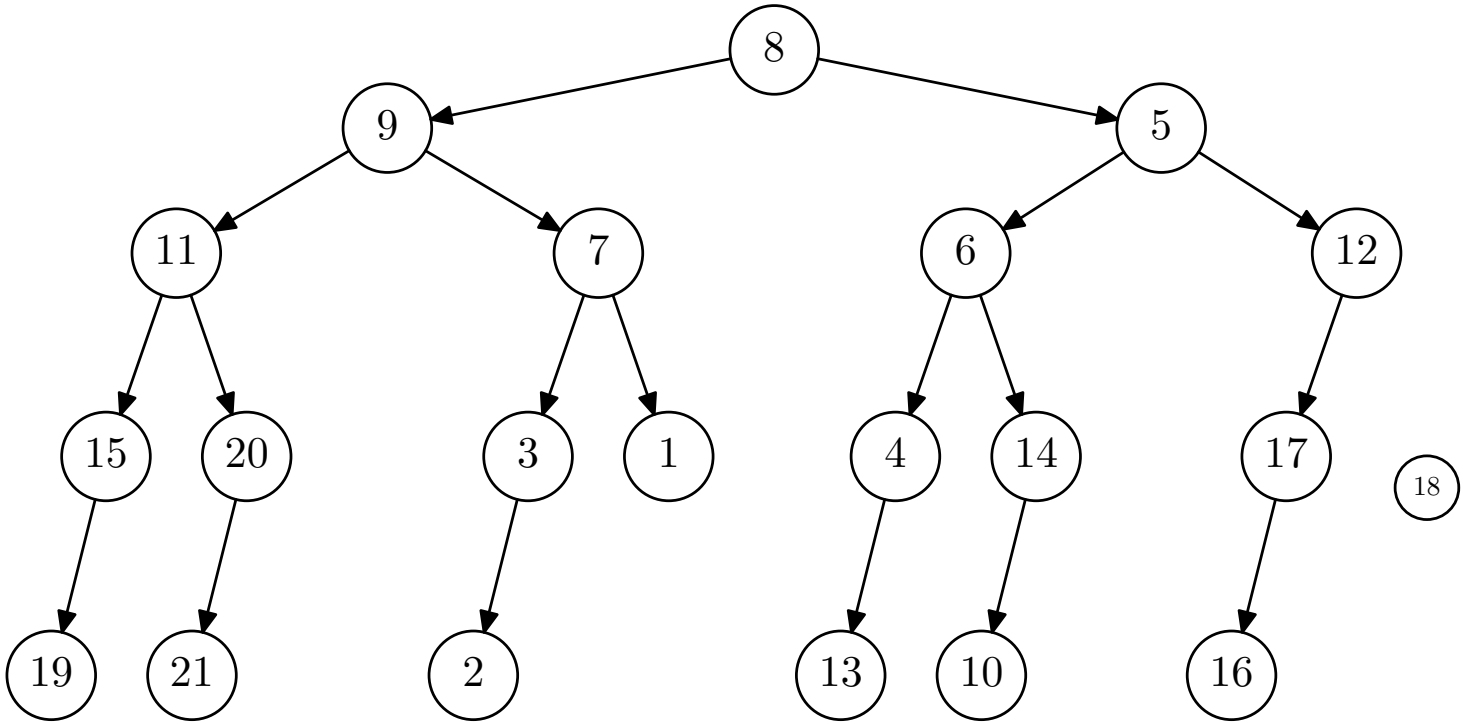
Tökéletesen kiegyensúlyozott bináris fa építése



Kósa Márk
Pánovics János
Szathmáry László

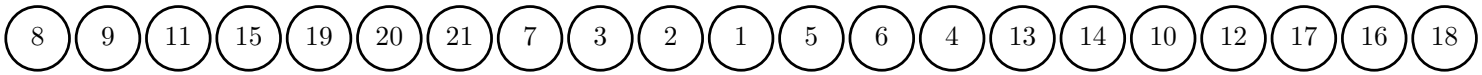
Tökéletesen kiegyensúlyozott bináris fa építése

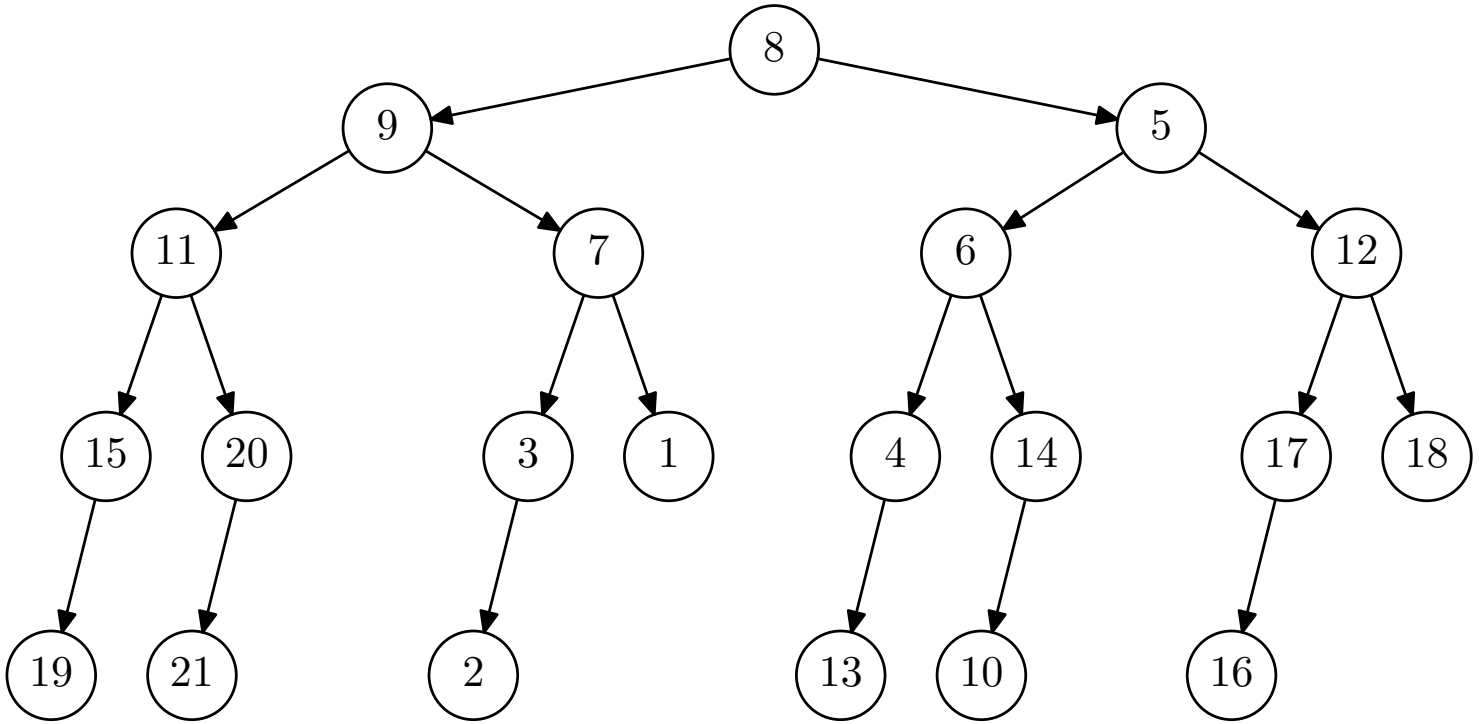




Kósa Márk
Pánovics János
Szathmáry László

Tökéletesen kiegyensúlyozott bináris fa építése





Kósa Márk
Pánovics János
Szathmáry László

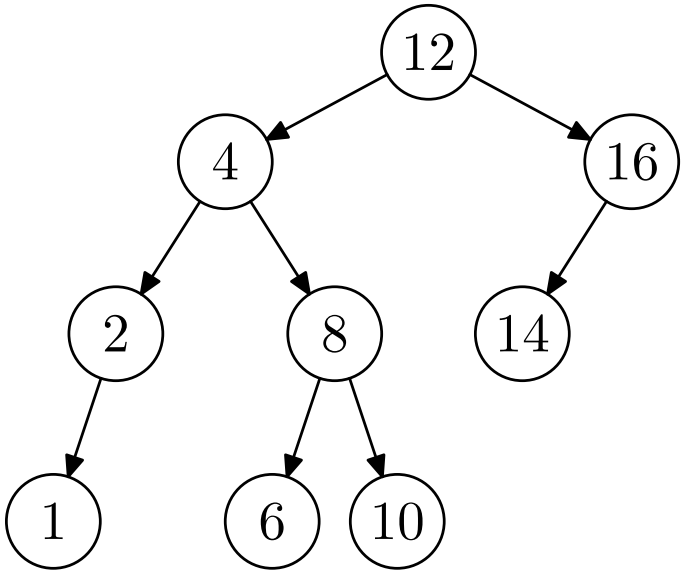
Kósa Márk
Pánovics János
Szathmáry László

Bináris keresőfa

Bináris keresőfa

A bináris keresőfa olyan rendezett bináris fa, melyben az adatelemek mindegyike rendelkezik egy kulccsal, és minden adatelemre igaz az, hogy az adatelem bal oldali részfájában lévő elemek kulcsai kisebbek, a jobb oldali részfájában lévő elemek kulcsai pedig nagyobbak az elem kulcsánál.

Példa bináris keresőfára:



Bináris keresőfa bővítése

Bináris keresőfa bővítése rekurzívan

1. Ha üres a fa, akkor a beszúrandó elem lesz a fa egyetlen eleme (levéleleme), és ezzel az algoritmus sikeresen véget ér.
2. Összehasonlítjuk a gyökérelem értékét a beszúrandó elemmel.
   a) Ha a két elem egyenlő, akkor a beszúrandó elemet nem helyezhetjük el a fában (mert nem szerepelhet benne két azonos értékű elem), és ezzel az algoritmus sikertelenül véget ér.
   b) Ha a beszúrandó elem kisebb a gyökérelemnél, akkor a gyökérelem bal oldali részfáját bővítjük a beszúrandó elemmel.
   c) Egyébként a gyökérelem jobb oldali részfáját bővítjük a beszúrandó elemmel.

Kósa Márk
Pánovics János
Szathmáry László

Bináris keresőfa bővítése

Bináris keresőfa bővítése rekurzívan (implementáció)

```
typedef struct faelem {
    int adat;
    struct faelem *bal;
    struct faelem *jobb;
} FAELEM;

FAELEM *gyoker = NULL;


void beszur(int ertek)
{
    if (gyoker == NULL)
    {
        FAELEM *uj = (FAELEM *)malloc(sizeof(FAELEM));
        uj->adat = ertek;
        uj->bal = uj->jobb = NULL;
        gyoker = uj;
    }
    else
    {
        beszur_reszfaba(gyoker, ertek);
    }
}
```


Kósa Márk
Pánovics János
Szathmáry László

Bináris keresőfa bővítése

Bináris keresőfa bővítése rekurzívan (implementáció) [folyt.]

```
void beszur_reszfaba(FAELEM *akt, int ertek)
{
    if (ertek == akt->adat) {
        /* mar letezik ilyen kulcsu elem */
        return;
    }
    else
    {
        if (ertek < akt->adat)
        {
            if (akt->bal != NULL) {
                beszur_reszfaba(akt->bal, ertek);
            }
            else {
                FAELEM *uj = (FAELEM *)malloc(sizeof(FAELEM));
                uj->adat = ertek;
                uj->bal = uj->jobb = NULL;
                akt->bal = uj;
            }
        }
        else /* if ertek > akt->adat */
        {
            if (akt->jobb != NULL) {
                beszur_reszfaba(akt->jobb, ertek);
            }
            else {
                FAELEM *uj = (FAELEM *)malloc(sizeof(FAELEM));
                uj->adat = ertek;
                uj->bal = uj->jobb = NULL;
                akt->jobb = uj;
            }
        }
    }
}
```


Kósa Márk
Pánovics János
Szathmáry László

Bináris keresőfa bővítése

Bináris keresőfa bővítése iteratívan

1. Aktuális részfának a teljes fát tekintjük.
2. Ha üres az aktuális részfa, akkor a beszúrandó elem lesz az aktuális részfa egyetlen eleme (levéleleme), és ezzel az algoritmus sikeresen véget ér.
3. Összehasonlítjuk az aktuális részfa gyökérelemének értékét a beszúrandó elemmel.
   a) Ha a két elem egyenlő, akkor a beszúrandó elemet nem helyezhetjük el a fában (mert nem szerepelhet benne két azonos értékű elem), és ezzel az algoritmus sikertelenül véget ér.
   b) Ha a beszúrandó elem kisebb az aktuális elemnél, akkor aktuális részfának tekintsük az aktuális részfa gyökérelemének bal oldali részfáját.
   c) Egyébként aktuális részfának tekintsük az aktuális részfa gyökérelemének jobb oldali részfáját.
4. Folytassuk az algoritmust a 2. lépéssel.

Kósa Márk
Pánovics János
Szathmáry László

Bináris keresőfa bővítése iteratívan (implementáció)

```
void beszur(int ertek)
{
    FAELEM *uj = (FAELEM *)malloc(sizeof(FAELEM));
    uj->adat = ertek;
    uj->bal = uj->jobb = NULL;

    if (gyoker == NULL) {
        gyoker = uj;
    }
    else
    {
        FAELEM *akt = gyoker;

        while (akt != NULL)
        {
            if (ertek < akt->adat)
            {
                if (akt->bal == NULL) {
                    akt->bal = uj;
                    return;
                }
                akt = akt->bal;
            }
            else if (ertek > akt->adat)
            {
                if (akt->jobb == NULL) {
                    akt->jobb = uj;
                    return;
                }
                akt = akt->jobb;
            }
            else {
                /* mar letezik ilyen kulcsu elem */
                return;
            }
        }
    }
}
```


Kósa Márk
Pánovics János
Szathmáry László

Törlés bináris keresőfából

Törlés bináris keresőfából rekurzívan

1. Ha üres a fa, akkor nem tudunk törölni, és ezzel az algoritmus sikertelenül véget ér.
2. Összehasonlítjuk a gyökérelem értékét a törlendő elemmel.
   - a) Ha a törlendő elem kisebb a gyökérelemnél, akkor a gyökérelem bal oldali részfájából töröljük a törlendő elemet.
   - b) Ha a törlendő elem nagyobb a gyökérelemnél, akkor a gyökérelem jobb oldali részfájából töröljük a törlendő elemet.
   - c) Ha a két elem egyenlő, akkor megnézzük, hogy a gyökérelemnek hány rákövetkezője van.
     - i) Ha a gyökérelemnek egy rákövetkezője sincs (azaz levélelem), akkor egyszerűen törölhető.
     - ii) Ha a gyökérelemnek egy rákövetkezője van, akkor felülírjuk a gyökérelemet azzal a rákövetkező elemmel (azaz egy szinttel feljebb csúsztatjuk a gyökérelem nem üres részfáját).
     - iii) Ha a gyökérelemnek két rákövetkezője van, akkor a gyökérelem értékét felülírjuk a gyökérelem bal oldali részfája legjobboldalibb elemének az értékével, majd a gyökérelem bal oldali részfájából töröljük ezt a legjobboldalibb elemet.

   Ezzel az algoritmus sikeresen véget ér.

Kósa Márk
Pánovics János
Szathmáry László

Kiegyensúlyozott fa

Kiegyensúlyozott bináris fa

Azt mondjuk, hogy egy bináris fa kiegyensúlyozott, ha bármely elemére igaz, hogy az elem bal oldali és jobb oldali részfájának magasságkülönbsége legfeljebb 1.

Megjegyzés

Minden tökéletesen kiegyensúlyozott fa egyben kiegyensúlyozott is.

Kiegyensúlyozott keresőfa (AVL-fa)

Akkor nevezünk egy bináris fát kiegyensúlyozott keresőfának vagy AVL-fának, ha kiegyensúlyozott is és keresőfa is egyben.

Megjegyzés

Az AVL-fa elnevezés Georgij Makszimovics **A**delszon-**V**elszkij és Jevgenyij Mihajlovics **L**andisz nevéből származik. Az AVL-fáról először egy 1962-es cikkükben írtak.

Kósa Márk
Pánovics János
Szathmáry László

AVL-fák bővítése

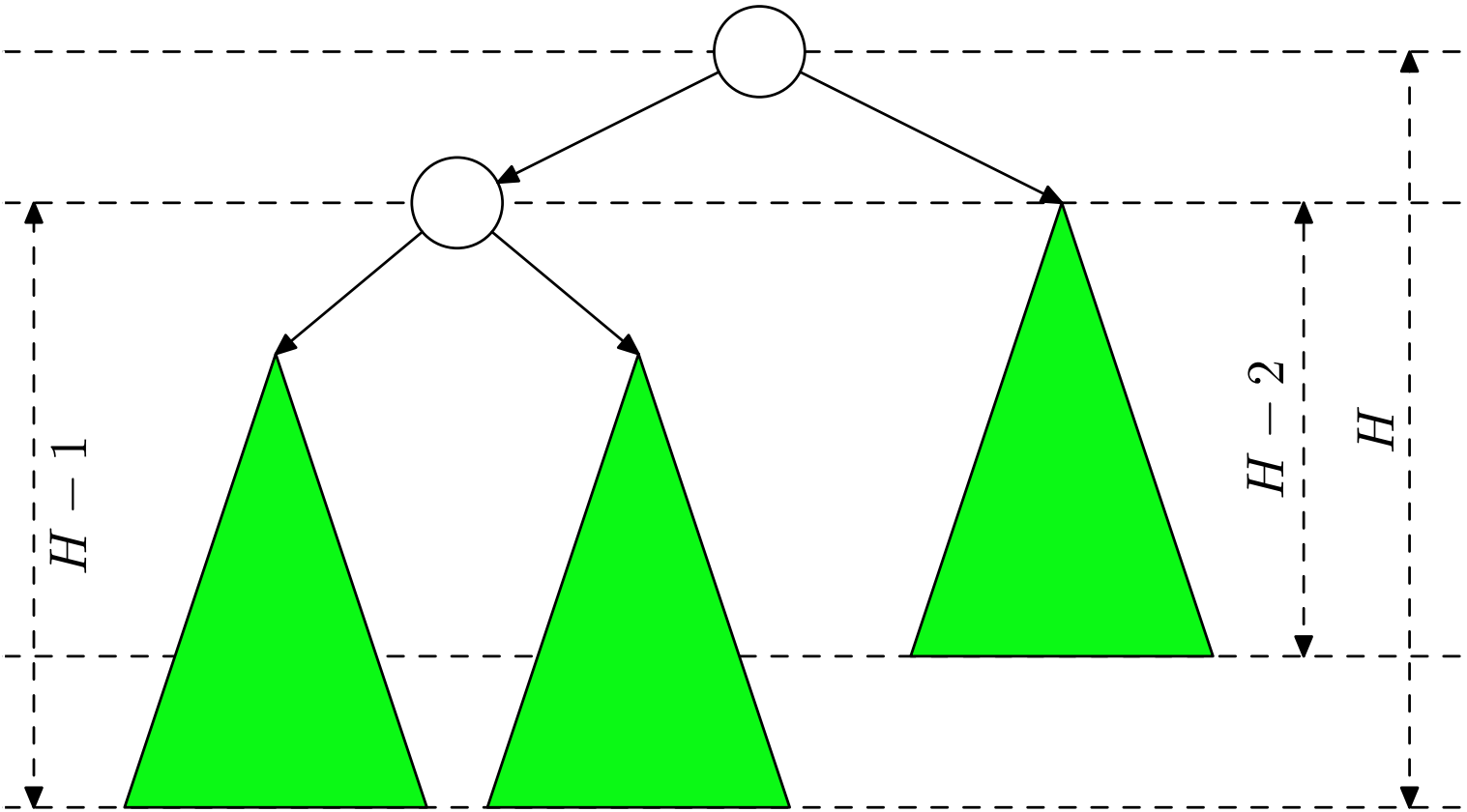
Egy AVL-fát úgy bővítünk, mint egy keresőfát: mindig levélelemmel. A levélelemmel történő bővítést követően a következő esetek fordulhatnak elő:

1. A fa továbbra is kiegyensúlyozott. Ekkor nincs teendőnk, készen vagyunk.
2. A fa elveszti kiegyensúlyozottságát. Ekkor egy vagy két forgatással újra kiegyensúlyozottá kell tennünk a fát.

Kósa Márk
Pánovics János
Szathmáry László

Amikor elromlik a kiegyensúlyozottság: LL és LR bővítés

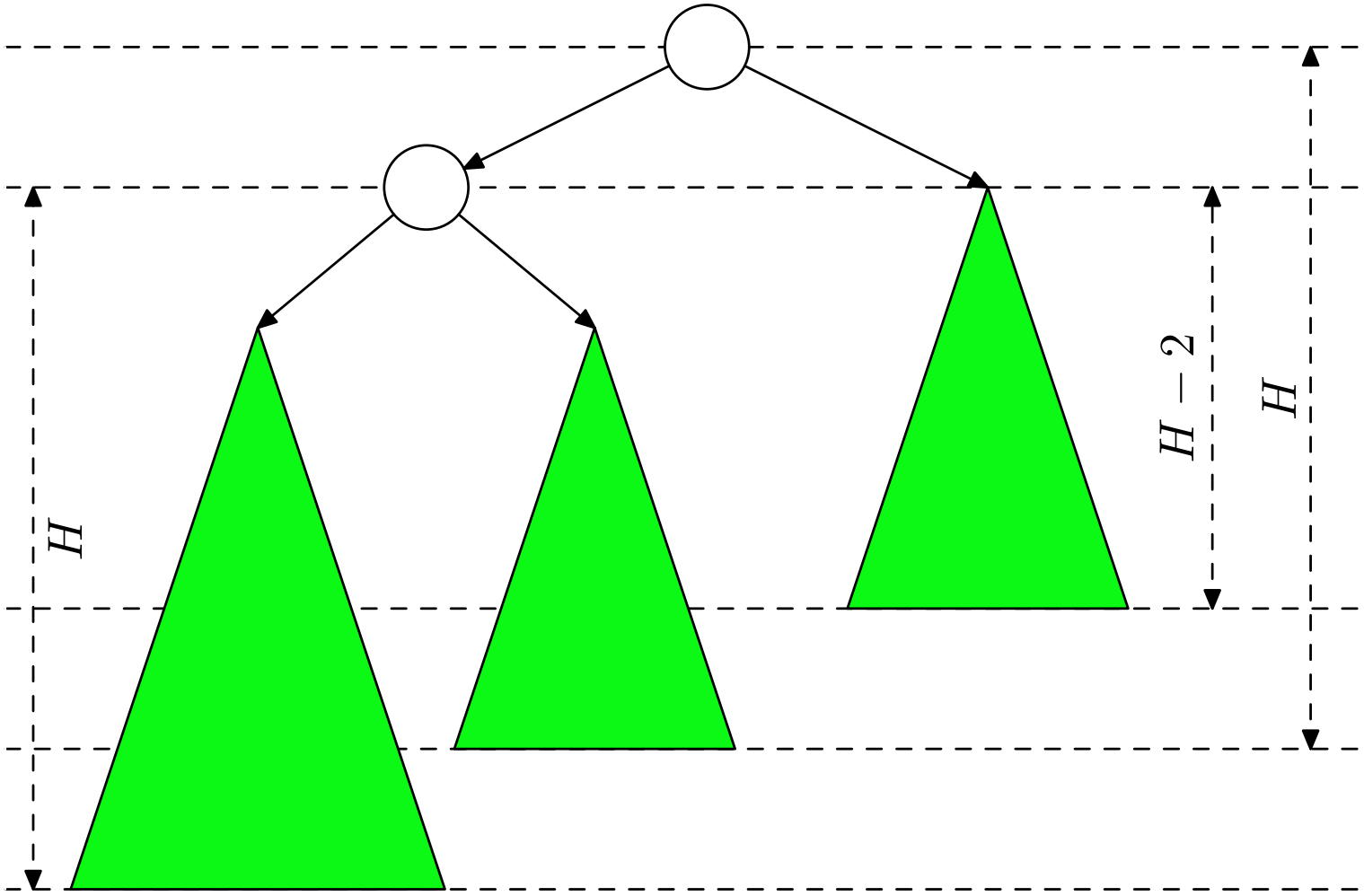
Alaphelyzet: a gyökérelem bal oldali részfája egy szinttel magasabb, mint a jobb oldali részfája - a fa még kiegyensúlyozott.



Kósa Márk
Pánovics János
Szathmáry László

Amikor elromlik a kiegyensúlyozottság: LL és LR bővítés

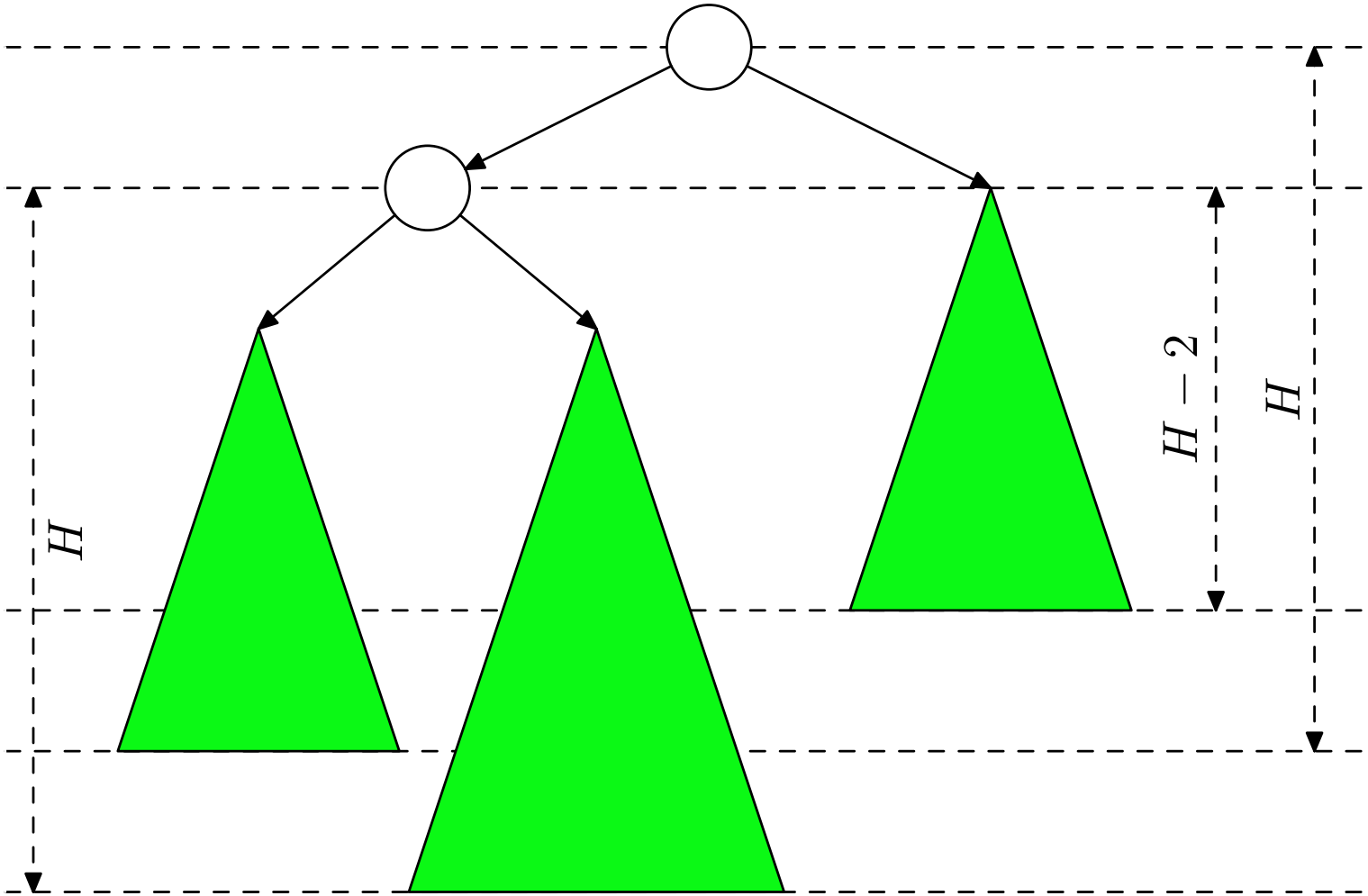
LL bővítés: a gyökérelem bal oldali részfájának bal oldali részfájába kerül az új elem - a fa elveszti kiegyensúlyozottságát.



Kósa Márk
Pánovics János
Szathmáry László

Amikor elromlik a kiegyensúlyozottság: LL és LR bővítés

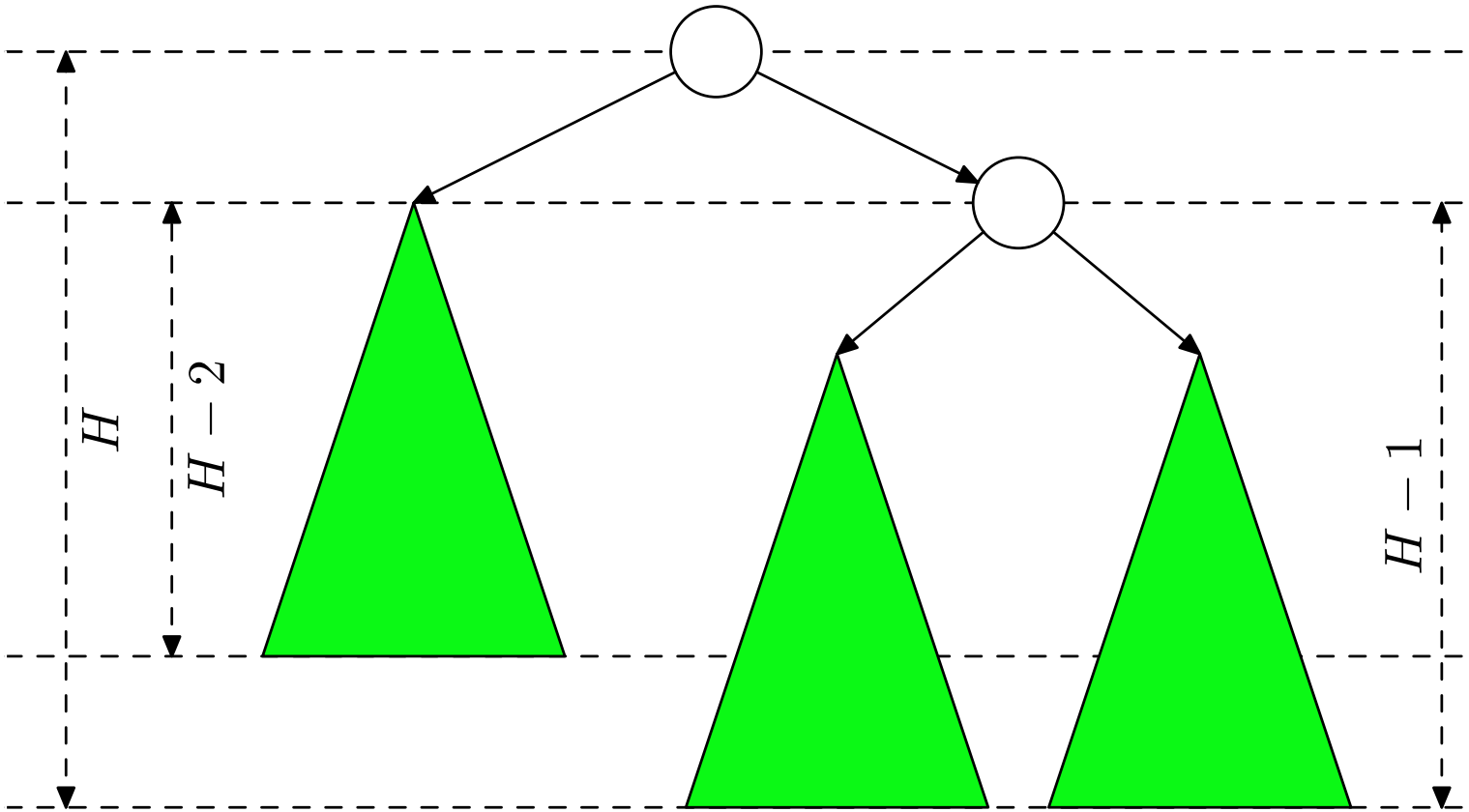
LR bővítés: a gyökérelem bal oldali részfájának jobb oldali részfájába kerül az új elem - a fa elveszti kiegyensúlyozottságát.



Kósa Márk
Pánovics János
Szathmáry László

Amikor elromlik a kiegyensúlyozottság: RL és RR bővítés

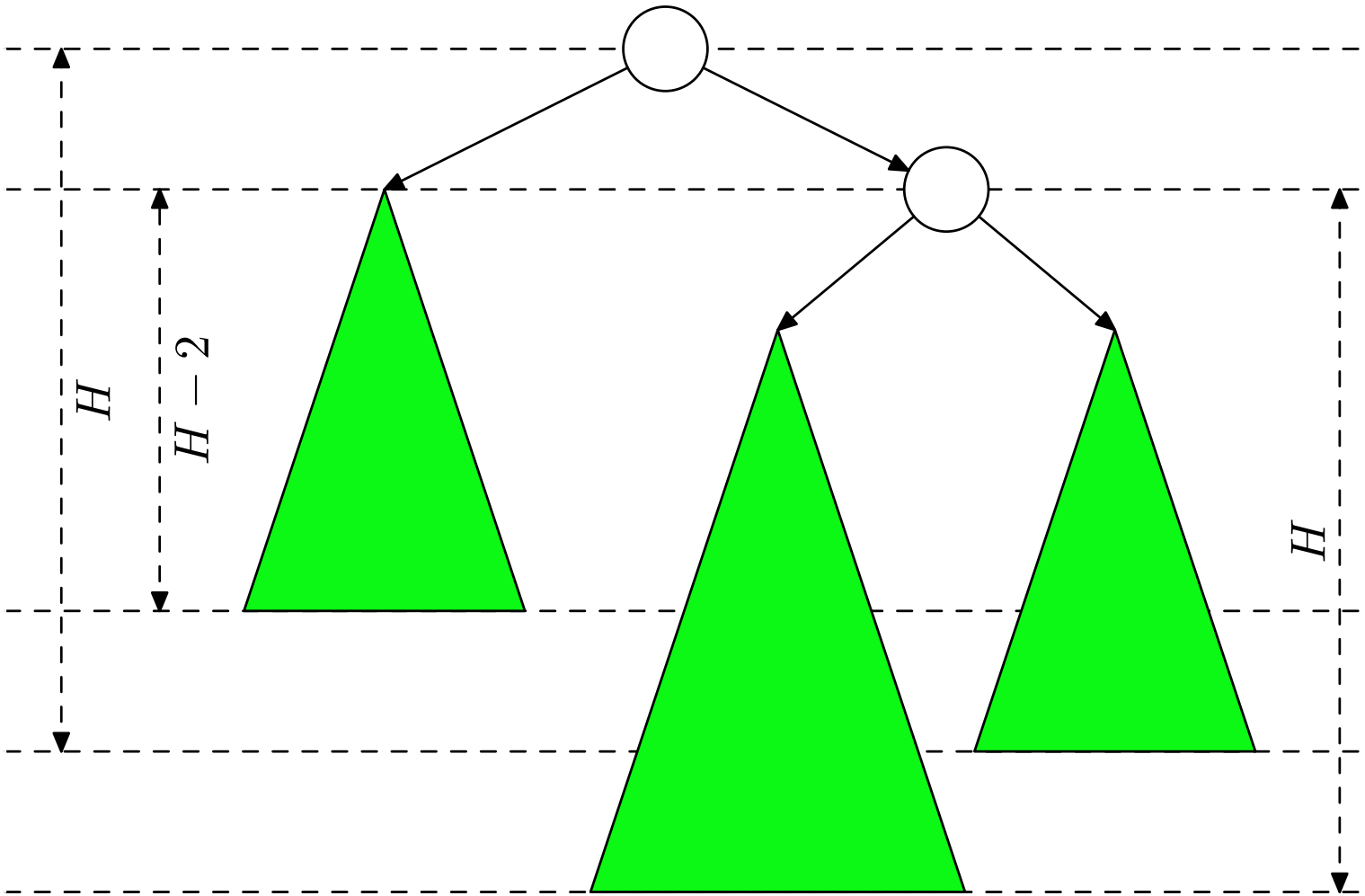
Alaphelyzet: a gyökérelem bal oldali részfája egy szinttel alacsonyabb, mint a jobb oldali részfája - a fa kiegyensúlyozott.



Kósa Márk
Pánovics János
Szathmáry László

Amikor elromlik a kiegyensúlyozottság: RL és RR bővítés

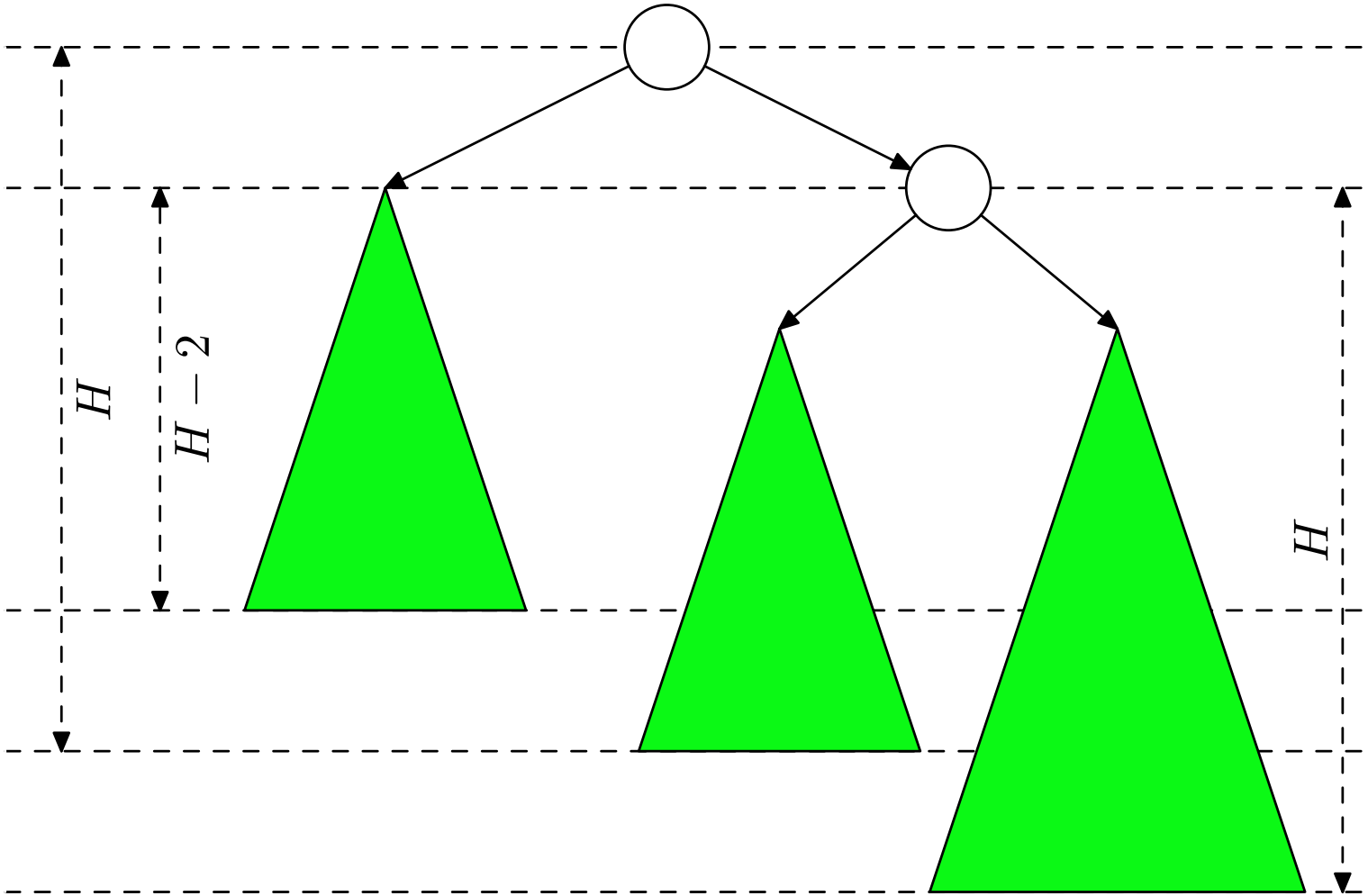
RL bővítés: a gyökérelem jobb oldali részfájának bal oldali részfájába kerül az új elem - a fa elveszti kiegyensúlyozottságát.



Kósa Márk
Pánovics János
Szathmáry László

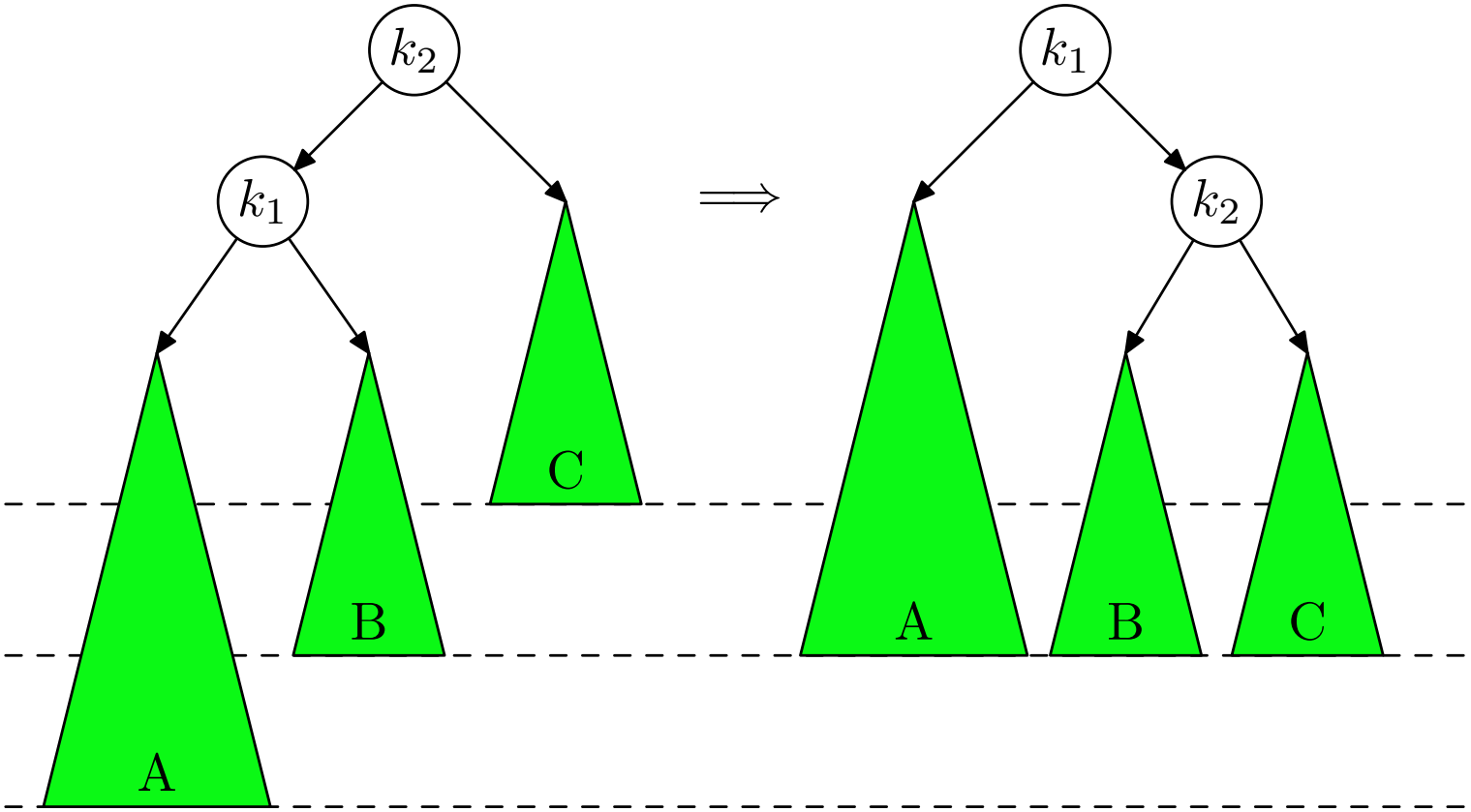
Amikor elromlik a kiegyensúlyozottság: RL és RR bővítés

RR bővítés: a gyökérelem jobb oldali részfájának jobb oldali részfájába kerül az új elem - a fa elveszti kiegyensúlyozottságát.



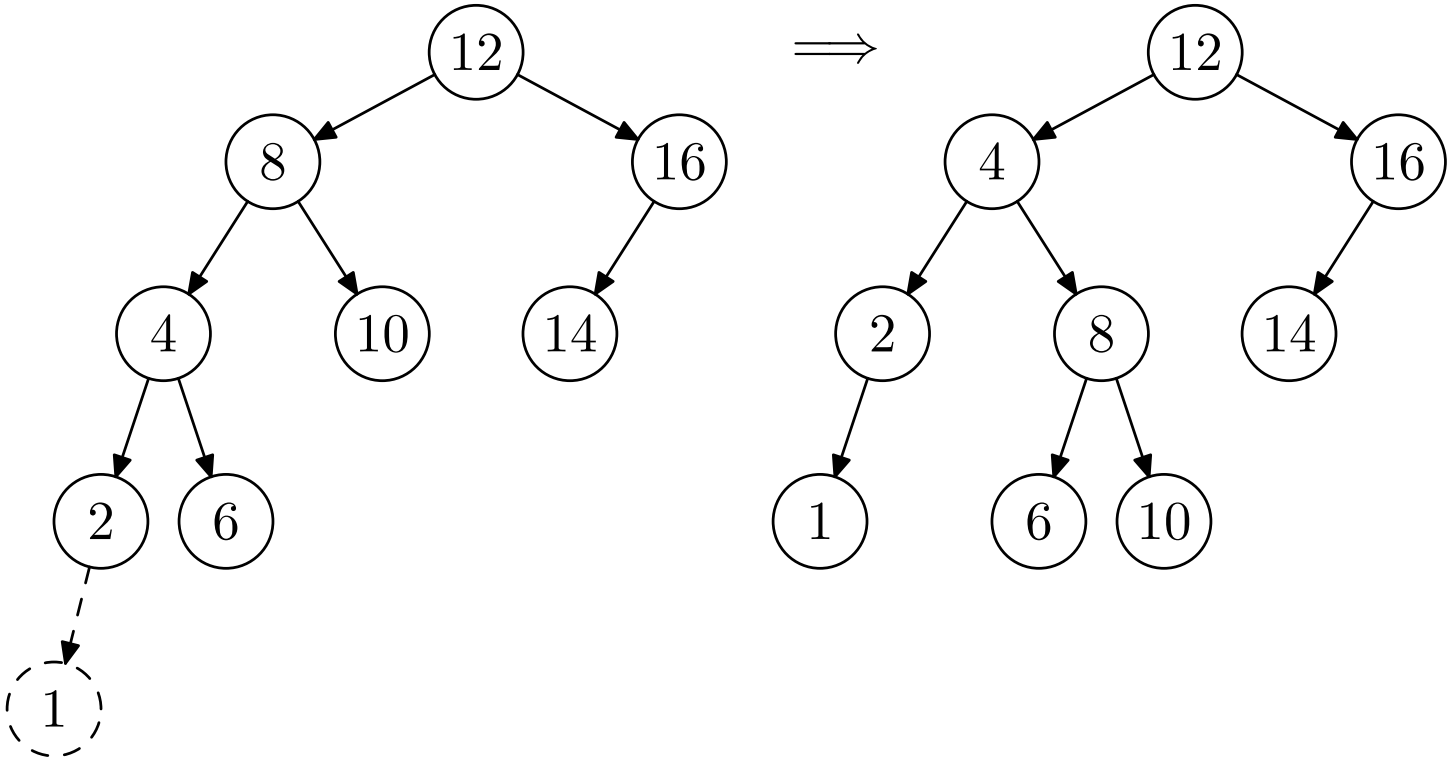
Kósa Márk
Pánovics János
Szathmáry László

Az LL bővítés megoldása



Kósa Márk
Pánovics János
Szathmáry László

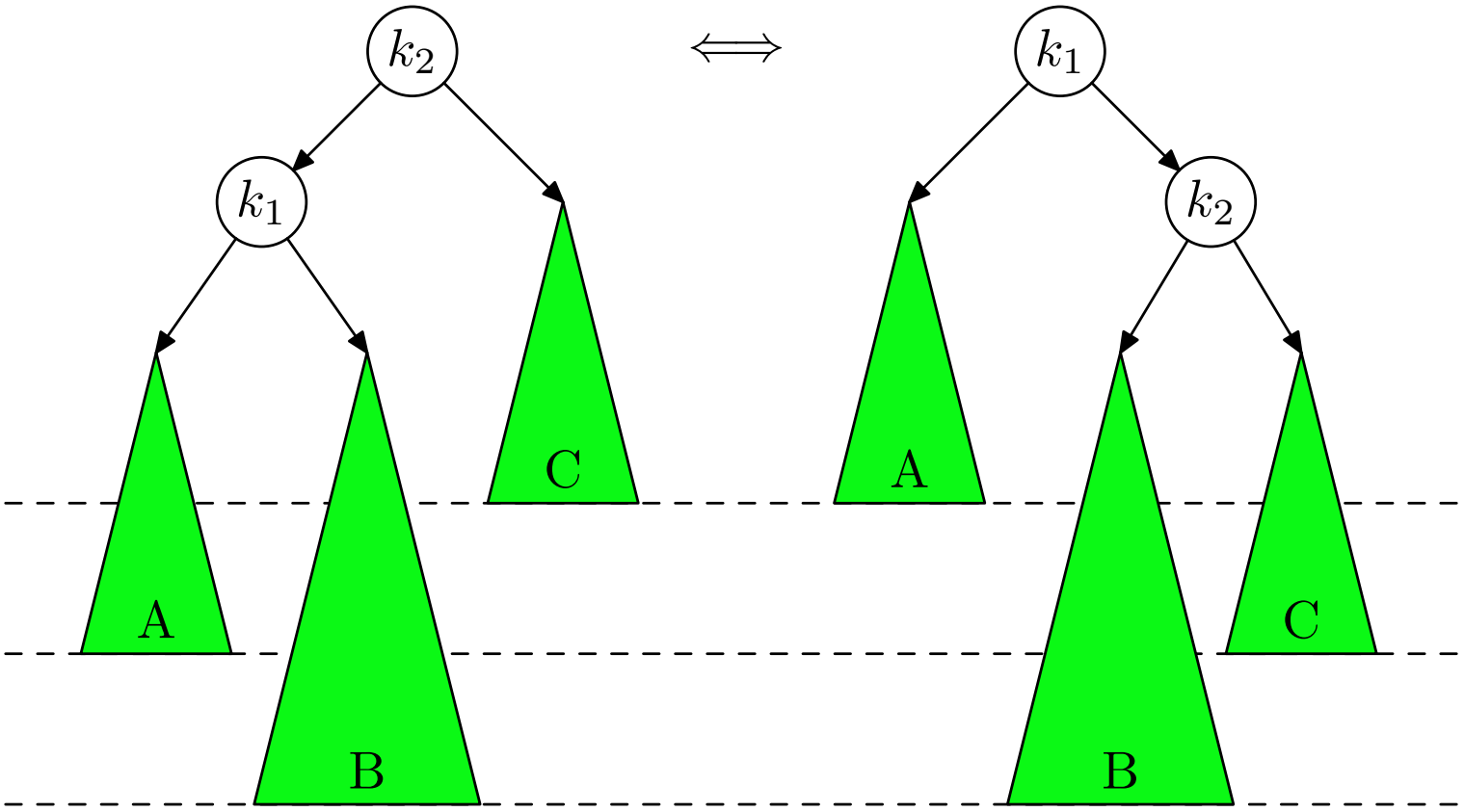
Példa LL bővítésre



Kósa Márk
Pánovics János
Szathmáry László

Az LR bővítés megoldása

Az LR bővítés nem oldható meg egyetlen forgatással: sem a kiinduló, sem az egyszer elforgatott fa nem kiegyensúlyozott.

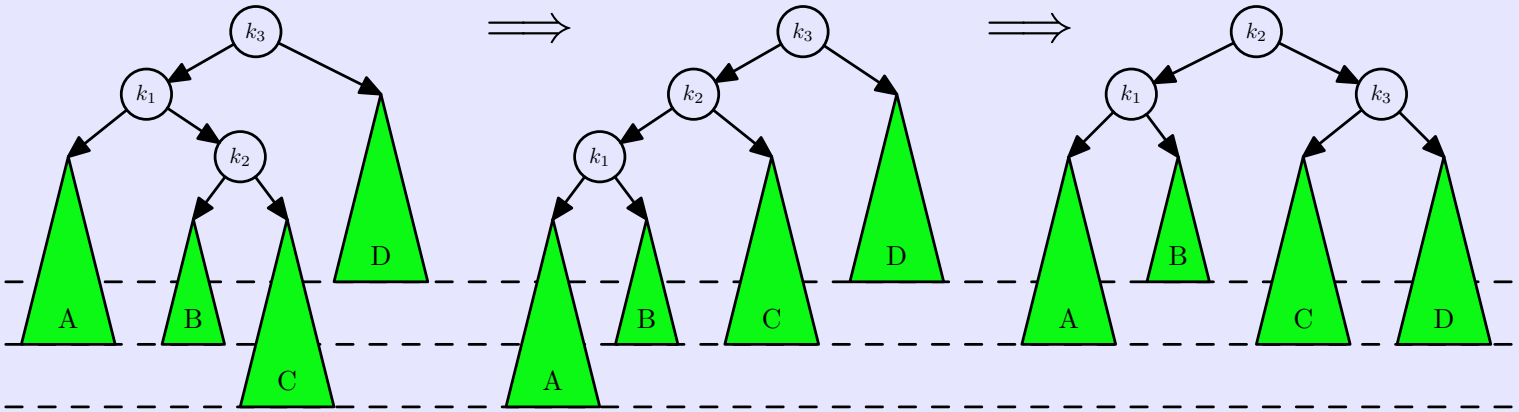


Kósa Márk
Pánovics János
Szathmáry László

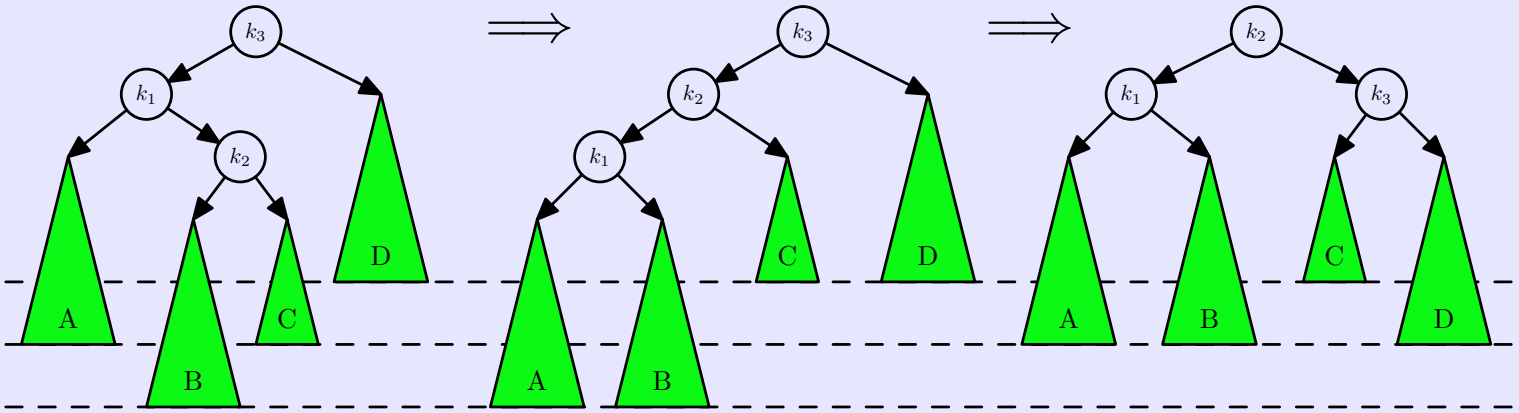
Az LR bővítés megoldása

A megoldás: két forgatás!

Az egyik eset

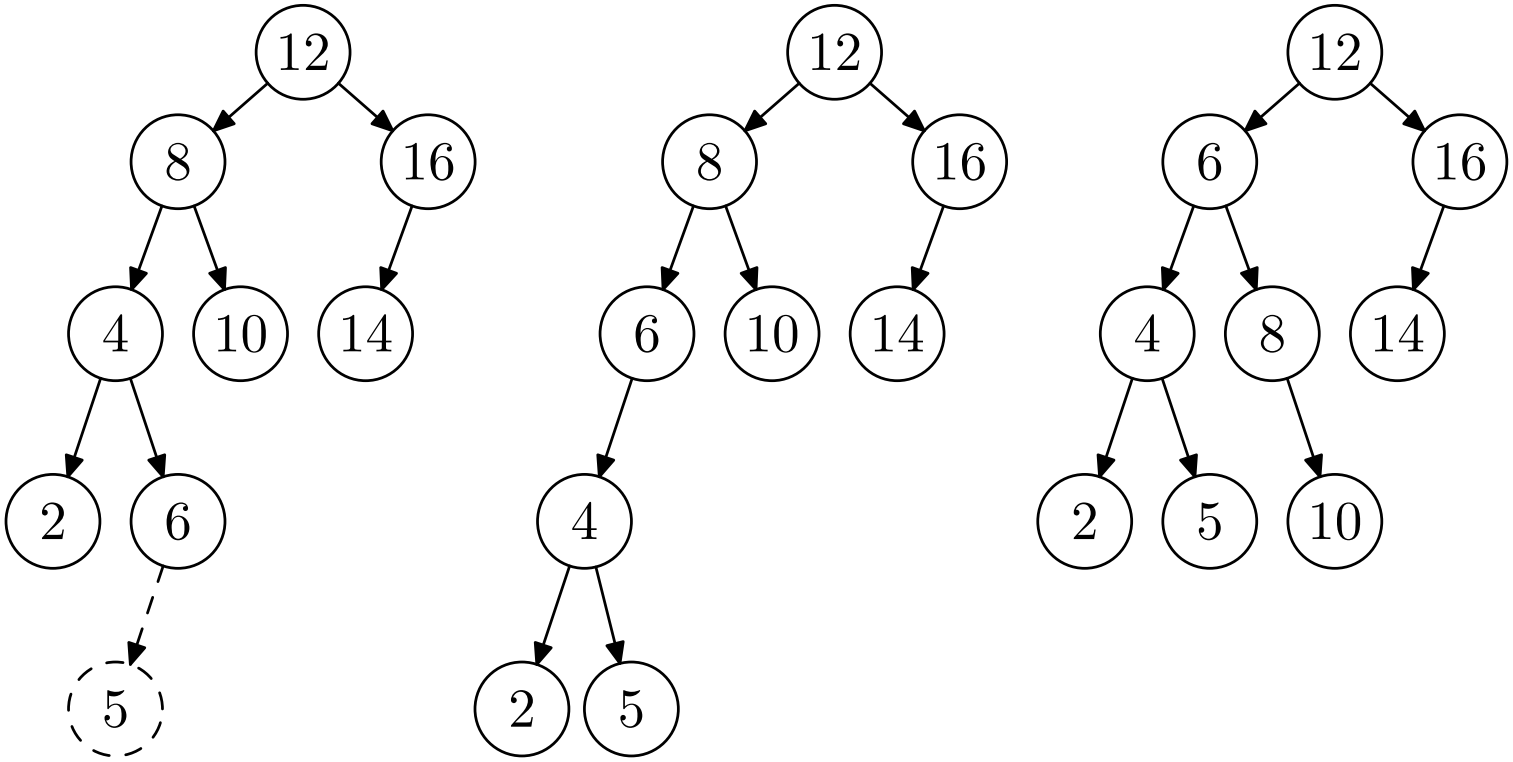


A másik eset



Kósa Márk
Pánovics János
Szathmáry László

Példa LR bővítésre



Kósa Márk
Pánovics János
Szathmáry László

Törlés AVL-fából

Egy AVL-fából ugyanúgy törlünk, mint egy keresőfából. A törlést követően a következő esetek fordulhatnak elő:

1. A fa továbbra is kiegyensúlyozott. Ekkor nincs teendőnk, készen vagyunk.
2. A fa elveszti kiegyensúlyozottságát. Ekkor a törlés helyétől a gyökér felé haladva megkeressük az első olyan elemet, amelyhez mint gyökérelemhez tartozó részfa már nem kiegyensúlyozott, és a bővítésnél ismertetett négy eset közül a megfelelőt alkalmazva kiegyensúlyozzuk ezt a részfát. Ezt a lépést a gyökér felé haladva addig ismételjük, amíg a teljes fa kiegyensúlyozott nem lesz.

Kósa Márk
Pánovics János
Szathmáry László